Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského
Ústav Informatiky

Počítač ako učiteľ písania

Diplomová práca

Juliana Lipková
autor

RNDr. Marek Nagy
školiteľ

Bratislava Máj 2008

Týmto prehlasujem, že som diplomovú prácu vypracovala samostatne s použitím uvedenej literatúry, elektronických zdrojov a s odbornou pomocou diplomového vedúceho.

Bratislava, máj 2008

Juliana Lipková

Poďakovanie

Rada by som poďakovala svojmu školiteľovi RNDr. Marekovi Nagyovi za jeho cenné rady k tejto práci, ochotu spolupracovať a jeho neustály úsmev. Tiež musím poďakovať svojim rodičom, za výchovu k radosti zo vzdelania, pretože bez nich by som nebola tým, čím som. Vďačím aj svojmu priateľovi Jozefovi Šiškovi hlavne za jeho nekonečnú psychickú podporu a svojim kamarátom Mišovi Forišekovi a Monike Steinovej. Taktiež by som rada poďakovala mojej 5-ročnej neteri Saške Kunovičovej, na ktorej som mohla pozorovať jej vývoj pri použití aplikácie.

Obsah

Kapitola 1

Úvod a motivácia

1.1 Motivácia

Na základnej škole sa žiaci v prvom ročníku učia písať. Nielen preto, aby sa naučili písať, ale aj preto, aby rukou hýbali a boli schopní sledovať nejaký ťah. Tým sa vyvarujú zdravotným problémom, ktoré by mohli nastať z dôvodu nehybnosti ruky. Predmet písanie je teda veľmi dôležitým pre zdravý vývoj dieťaťa.

Na hodinách sa však väčšinou používa metóda Drill&kill. Žiaci bývajú znechutení, takže nepracujú na svojom vývine. Aj keď všetci poznajú heslo J.A. Komenského "škola hrou", len málokde sa naozaj využíva. Počítač má veľa možností na motiváciu. Vďaka animáciam a zvukom robí proces učenia zaujímavejším a hravejším.

1.1.1 Využitie počítača na hodine písania

Na hodine písania je bežný počítač využiť ťažké. Existujú však zariadenia, ktoré sa priam núkajú. Sú nimi tabletové počítače, interaktívne tabule a dotykové displeje.

V niektorých prácach ľudia odporúčajú aj obyčajné tabletové zariadenie. Ja som však silne proti tomuto názoru. Problém je, že po pripojení tabletového zariadenia k počítaču sa ťah zobrazuje na monitore, avšak dieťa píše na tablet. Pritom žiak máva problémy už len s obyčajným sledovaním čiary. Týmto by písal na iné miesto, ako vidí, čím by sa jeho vizuálny vnem nespájal s kinetickým.

> V skutočnosti žiačik 1. ročníka postupuje dosť náhodne, "hľadá sa". Odhadom musí vo svojej predstave nájsť bod a začiatočný ťah písmena, odhadnúť predpokladanú šírku a výšku písmen, mať v pamäti proporcie písmena. V predstavách porovnávať rozličné výšky, šírky

> a dĺžky časti písmen. Zároveň fixovať ich tvar, kontrolovať momentálne vytváraný tvar písmen a stopu pera, ktorá je ešte stále vedená neistým pohybom ruky a nedostatočne koordinovaná so zrakovou predstavou písmena.([Sup98])

Takýto počítač na hodinu písania neprináša len motiváciu hrou. Je schopný robiť úkony, ktoré človek robiť nedokáže alebo dokáže s veľkou ťažkosťou.

Jedna z najdôležitejších schopností počítača je animácia. Smer, ktorým dieťa píše, je dôležitý. V písankách ho znázorňujú šípkami, avšak je to len schéma, ktorej veľa detí v tomto veku nerozumie. Animácia nielen, že ukazuje smer písania písmena, žiak vie pohyb sledovať, takže vidí, ako môže písmeno napísať.

U každého človeka prevažuje určitý druh vnemu. Ak je dieťa kinetický typ, animácia je presne to, čo mu pomôže. Ak je auditívnym typom, pomôžu mu zvuky popisujúce pohyb. Iba ak je vizuálnym typom, pomôžu mu bežné formy písanky.

1.1.2 Aplikácia na výučbu písania

Ako sme si práve uvedomili, počítač je dobrý nástroj, ktorý sa dá využiť na hodine písania. V mnohých prácach sa už o to ľudia usilovali. My sa však v tejto práci vyberieme iným smerom ako sa vydali v ostatných prácach.

Kedže rozpoznávanie písma je oblasť, ktorá sa už dlhé roky vyvýja, objavené algoritmy sa silou-mocou používali v didaktickom softvéri. Nebral sa ohľad na didaktiku, vedci chceli ukázať ako pekne fungujú algoritmy.

Poďme sa však vybrať opačným smerom. Začnime štúdiom didaktiky písania a hľadajme algoritmy, ktoré budú pomáhať skutočnej výučbe písania. Aj didaktika písania má svoju históriu a objavy. Dokonca história je dlhšia. Nezabúdajme na jej objavy!

1.2 Ciele práce

V tejto práci bude našim cieľom rozobrať možnosti aplikácie, vhodnosť použitia rôznych algoritmov a popísať, ako by mala celá aplikácia vyzerať. Budeme sa sústrediť na deti vo veku $5-7$ rokov alebo ekvivalentný vek pri deťoch s poruchami. Táto práca je nielen prácou informatickou, má za cieľ využiť znalosti z oblasti didaktiky písania.

Výsledky budú neskôr použité na programovanie open-source aplikácie v štipendijnom programe Google Summer of Code 2008, v ktorom už žiadosť na

tento projekt bola prijatá a akceptovaná ako jedna z mála žiadostí na svete. Mentorskou organizáciou je One Laptop per Child.

1.3 Štruktúra práce

V kapitole 2 sa pozrieme na vývoj v oblasti rozpoznávania písma a to na takzvané online metódy. V druhej časti si ozrejmíme niektoré znalosti z oblasti didaktiky písania.

Kapitola 3 sa venuje hlavne obsahovej stránke aplikácie, čo by mal ideálny softvér na výučbu písania obsahovať.

V kapitole 4 si rozoberieme fungovanie algoritmov na detekciu chýb a ich vhodnosť použitia do pedagogickej aplikácie.

Kapitola 2

Úvod do problematiky

V tejto kapitole si povieme niečo o výskume v oblasti rozpoznávania písma a v oblasti didaktiky písania. Neskôr tieto dva smery prepojíme.

2.1 Rozpoznávanie písma

Výskum v rozpoznávaní písma je už veľmi rozsiahly. Popíšeme základné algoritmy pre rozpoznávanie písma (Skryté Markovské modely, DTW algoritmus) a potom si povieme niečo o potrebných poznatkoch z didaktiky písania.

Rozpoznávače písma delíme na dva druhy: offline a online. Rozdiel je v tom, že kým v online rozpoznávaní dostávame na vstup dvojrozmerné súradnice bodov ako funkcia času, pri offline rozpoznávaní dostaneme iba statický obrázok bez akýchkoľvek dodatočných informácií[Cech]. Na offline rozpoznávanie sa používajú napríklad neurónové siete a na online DTW algoritmus alebo Skryté Markovovské modely.

Kedže každé písmeno má svoj smer, ktorým by sa malo písať, offline rozpoznávanie nie je na proces učenia vhodné, pretože smer písania úplne zanedbáva. Preto sa v tejto práci sústredíme na aplikáciu online metód.

2.1.1 Skrytý Markovovský model

Podľa [Sra07] je Skrytý Markovovský model definovaný takto.

Definícia Skrytý Markovovský model (HMM) je pätica $M = (S, \Sigma, T, E, \Pi)$, kde $S = \{S_1, S_2, \ldots, S_m\}$ je konečná množina skrytých stavov, Σ je výstupná abeceda, T, $\{\{t_i(j)\}_{i=1}^{m}\}_{j=1}^{m}$ je matica pravdepodobností prechodov medzi stavmi, E, $\{\{e_i(j)\}_{i=1}^{m}\}_{j=1}^{|\Sigma|}$ je matica pravdepodobností výstupov, $\Pi = \{\pi_1, \pi_2, \ldots, \pi_m\}$ je počiatočná distribúcia stavov.

Skrytý Markovovský model modeluje Markovovský proces. Generuje postupnosť pozorovaní nad abecedou Σ. V závislosti od počiatočnej distribúcie Π začína proces v niektorom z m stavov. V každom kroku sa vyberie nový stav podľa pravdepodobnostnej distribúcie $t_k(1), \ldots, t_k(m)$, kde k je aktuálny stav. Výstupom každého kroku je tiež symbol podľa pravdepodobnostnej distribúcie $e_l(1), \ldots, e_l(m)$, kde l je stav v predchádzajúcom kroku. Postupnosť týchto stavov nazývame *možnou postupnosťou stavov.*

Definícia Postupnosť stavov HMM $s_1, \ldots, s_n$ nazveme možnou postupnosťou stavov (state path) vtedy a len vtedy, ak

$$(\forall i)(1 \leq i < n)(t_{s_i}(s_{i+1}) > 0).$$

Definícia Skrytý Markovovský model generuje postupnosť $X_1, \ldots, X_n$ vtedy a len vtedy, ak existuje možná postupnosť stavov $s_1, \ldots, s_n$ taká, že

$$(\forall i)(1 \leq i < n)(e_{s_i}(X_i) > 0).$$

Pravdepodobnosť vygenerovania postupnosti X je

$$e_{s_n}(X_n)\pi_{s_1}\Pi_{i=1}^{n-1}t_{s_i}(s_{i+1})e_{s_i}(X_i),$$

čo je súčin počiatočných pravdepodobností všetkých pravdepodobností prechodov a výstupov.

V súvislosti so Skrytými Markovovskými modelmi riešime tieto problémy ([Rab89]):

(1) pre dané pozorovanie vypočítať, s akou pravdepodbnosťou ho mohol model so zadanými parametrami vygenerovať

(2) pre dané pozorovanie vypočítať, aká je najpravdepodobnejšia postupnosť stavov v modeli so zadanými parametrami

(3) pre zadanú množinu pozorovaní, nájsť najpravdepodobnejšie parametre modelu

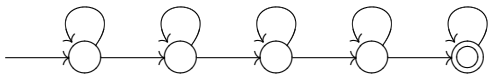
(1) sa rieši forward-backward algoritmom, (2) sa rieši Viterbi algoritmom a (3) sa rieši Baum-Welch algoritmom. Všetky tieto algoritmy sú založené na princípe dynamického programovania (Baum-Welch algoritmus využíva forward-backward algoritmus). Viac informácií o týchto algoritmoch si môžete nájsť v [Sra07].

HMM a problematika rozpoznávania písma

Zadefinovali sme Skryté Markovovské modely. Otázkou však ostáva, ako ich môžeme použiť pri rozpoznávaní písania.

V našom prípade je pozorovanie X postupnosť zmien súradníc. Zmeny stavov pravdepodobne nastávajú vo význačných bodoch písmena, a to tam, kde sa mení zakrivenie. Model vyzerá takto:

Obr. 2.1: Skryté Markovovské modely pre problém rozpoznávania písma



Povedzme, že písmeno má 5 význačných bodov. V tých budeme prechádzať do nasledujúcich stavov. S nejakou pravdepodobnosťou prejdeme do ďalšieho stavu (podľa hodnôt T) a s nejakou pravdepodobnosťou budeme dávať na výstup zmeny súradníc (podľa hodnôt E). Jednoduchšie povedané, v jednom stave sa posúvame po jednej krivke, prejdeme do iného stavu a tam meníme smer krivky.

Pre každé písmeno majme natrénovaný jeden model, t.j. máme jeho parametre.

Dôležité je uvedomiť si, že to, čo sú stavy, sme si iba my sami povedali. Závisí to od natrénovania modelu. Model trénujeme iba na pozorovaniach. Avšak toto sú pre model pravdepodobne najzaujímavejšie miesta.

Ak chceme zistiť, ku ktorému písmenu sa najviac podobá napísane písmeno, pozrime sa na jeho zmeny súradníc, ktoré sú našim novým pozorovaním. Následne zistime forward-backward algoritmom, aká je pravdepodobnosť vygenerovania daného pozorovania každým z modelov. Model, ktorého pravdepodobnosť je najväčšia, pravdepodobne prislúcha nášmu písmenu.

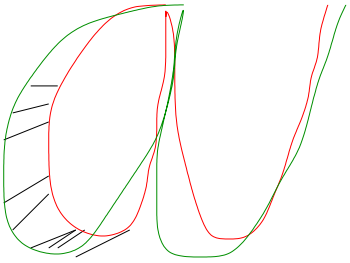
Tejto problematike sa venoval Marek Tomacha v [Tom07]. Pre viac informácií môže čitateľ nahliadnuť do jeho práce.

2.1.2 Dynamic time warping algoritmus

Dynamic time warping (DTW) je algoritmus, ktorého výstupom je miera podobnosti dvoch postupností. To, čo potrebujeme určiť, je najvhodnejšie priradenie bodov z jednej postupnosti k bodom druhej postupnosti. Najvhodnejšie znamená také, že spolu bude miera čo najmenšia.

Pozrime sa na príklad. Máme písmeno, ktoré je určené postupnosťou bodov. Dostaneme druhé písmeno. Zoberme si priradenie bodov v jednej postupnosti k bodom druhej postupnosti. Čierne čiary znázorňujú priradenie bodov.

Obr. 2.2: Priradenie bodov v DTW algoritme



Naša miera môže byť napríklad súčet druhých mocnín vzdialenosti bodov každej dvojice. DTW algoritmus nám pomôže nájsť to priradenie, kde je táto miera najmenšia.

DTW algoritmus funguje na princípe dynamického programovania. Aby sme našli najvhodnejšie priradenie pre určitú vybranú prefixovú podpostupnosť[1], stačí sa pozrieť na už vypočitanú o 1 menšiu podpostupnosť a jedno z troch možných priradení. Podľa [MR81] platí

$$D(t,i) = min[D(t,i-1), D(t-1,i-1), D(t-1,i)] + d(t,i)$$

D je celková miera postupnosti a d je miera vzdialenosti dvoch bodov.

Algoritmus vyzerá nasledovne:

1. Inicializácia

$$D(1,i) = \begin{cases} d(1,i) & \text{pre } i = 1 \\ d(1,i) + D(1,i-1) & \text{pre } i = 2,\ldots,N \end{cases}$$

2. Rekurzia pre $t = 2,\ldots,T$

$$D(t,i) = \begin{cases} d(t,i) + D(t-1,i) & \text{pre } i = 1 \\ d(t,i) + \min\Big(D(t,i-1), & \\ \qquad D(t-1,i-1), & \text{pre } i = 2,\ldots,N \\ \qquad D(t-1,i)\Big) & \end{cases}$$

3. Výsledok

$$\Delta = D(T,N)$$

Týmto sme našli iba najlepšiu mieru vzdialeností. Nezistili sme, ako majú byť body popárované. To doplníme do algoritmu jednoducho. Vždy si zapíšeme

[1] prvých niekoľko členov postupnosti

dvojicu $[t, i]$, ktorej miera nám zväčšila sumu. Sú to presne argumenty funkcie D.

1. Inicializácia

$$D(1,i) = \begin{cases} d(1,i) & \text{pre } i = 1 \\ d(1,i) + D(1,i-1) & \text{pre } i = 2,\ldots,N \end{cases}$$

$$\phi(1,i) = \begin{cases} [0,0] & \text{pre } i = 1 \\ [1,i-1] & \text{pre } i = 2,\ldots,N \end{cases}$$

2. Rekurzia pre $t = 2,\ldots,T$

$$D(t,i) = \begin{cases} d(t,i) + D(t-1,i) & \text{pre } i = 1 \\ d(t,i) + \min\Big(D(t,i-1), & \\ \qquad D(t-1,i-1), & \text{pre } i = 2,\ldots,N \\ \qquad D(t-1,i)\Big) & \end{cases}$$

$$\phi(t,i) = \begin{cases} [t-1,i] & \text{pre } i = 1 \\ \arg\min\Big(D(t,i-1), & \\ \qquad D(t-1,i-1), & \text{pre } i = 2,\ldots,N \\ \qquad D(t-1,i)\Big) & \end{cases}$$

3. Výsledok

$$\Delta = D(T,N)$$

$$z_K = [T,N]$$

4. Určenie priradení

$$z_k = \phi(z_{k+1}), \text{ pre } k = K-1,\ldots,1$$

$$Z = \{z_1,\ldots,z_K\}$$

V množine Z budú uchované jednotlivé páry. Toto nám pômôže pre danú mieru vzdialenosti bodov určiť, aké je najlepšie popárovanie bodov. V mojej diplomovej práci to bude jeden z kľúčových algoritmov.

2.2 Didaktické pozadie

Väčšina ľudí, ktorí už vedia písať, berie písanie ako samozrejmosť. Avšak aby sme do tohto štádia dospeli, museli sme prekonať náročnú cestu. Podľa [Sup98] sú vo vývine dieťaťa nasledovné etapy:

- **prvá etapa, tzv. predkaligrafická**

  Charakterizuje ju úsilie o zvládnutie kaligrafických noriem písania. Písmo prezrádza ešte motorickú neschopnosť zvládnuť stanovenú normu písania a má všetky typické známky písma začiatocníckeho. Táto fáza trvá 2-4 roky (od 5. až 6. roka do 8. až 9. roka) a predpokladá osvojenie techniky písania.

- **druhá etapa, tzv. kaligrafická fáza infantilná**

  Ide v nej o zdokonaľovanie písania a prispôsobovanie sa požiadavkám kladených na písmo (obdobie od 8. až 9. roka do 11. až 12. roka). Žiak píše dosť rýchlo a ľahko. Technika písania prestáva byť veľkým problémom a začína sa prejavovať problém ortografie. Dôležitá etapa vo vývine písanej reči končí približne v 12. roku veku dieťaťa.

- **tretia etapa, tzv. postkaligrafická fáza**

  Táto etapa je charakterizovaná plynulosťou techniky písania a rýchlosťou procesu písania. Písmo sa zjednodušuje a nadobúda individuálny, svojský rukopis.

V rôznych krajinách (druhoch písma) sa tieto etapy líšia. Vo väčšine prípadov je to však iba vekovou hranicou. Veková hranica sa tiež posúva ak ide o dieťa s určitými poruchami.

Dieťa pri napísaní slova postupuje nasledovne ([Sup98]):

1. slovo musí správne počuť
2. snaží sa ho správne vysloviť
3. rozkladá ho na hlásky (fonémy), určuje prvú a ďalšie nasledovné
4. hlásku spája s príslušným písmenom (grafémou) a predstavuje si ho
5. písmeno musí správne napísať
6. písmená navzájom spája do slova

V tejto práci sa budeme venovať bodom (4) a (5). Ostatné body sa budú môcť prepracovať v neskorších prácach. Budeme sa zaoberať deťmi v predkaligrafickej etape.

2.2.1 Ciele výučby písania

Pozrime sa na túto etapu, čo sa s dieťaťom deje v škole, čo je cieľom dieťa naučiť.

> Cieľom na 1. stupni ZŠ je naučiť dieťa písať písanou latinkou, osvojiť a upevniť si písmo. Položiť základy čitateľného, primerane rýchleho a úhľadného rukopisu a vypestovať návyk vkusnej úpravy grafického prejavu. Predpokladom na splnenie tohto cieľa je automatizácia písacieho pohybu t.j. schopnosť správne písať písmená a ich spoje za pomoci zrakovej a pohybovej kontroly. ([Sup98])

My v práci nesplníme celý tento cieľ. Takýto projekt by bol príliš rozsiahly a dá sa mu venovať v neskoršej práci. Veď aj samotné učenie písania je dlhý proces.

> Aby sme v školskej praxi predišli uvedeným oslabeniam výkonu v grafickom prejave, mali by sme si uvedomiť, že písanie nie je iba nácvik písmen. Ako dlhodobý, komplexný poznávací proces sa skladá z mnohých, čiastkových procesov, ktoré sa musia deti najprv naučiť, pričom výsledok sa hneď nedostaví. Až v priebehu učenia, teda postupne, sa upevní, osvojí a zautomatizuje. ([Sup98])

Náš cieľ bude zúžený. Bude pripomínať cieľ v prvom polroku hodiny písania. Budeme sa snažiť dosiahnúť, aby si dieťa **osvojilo základné tvary písmen a tzv. hygienu písania, vyvinulo jemnú motoriku, naučilo sa prepájať si vizuálny a kinetický, prípadne auditívny vnem, pomocou sledovania ťahu**.

Taktiež sa budeme snažiť splniť základné požiadavky písma ([Sup98]):

- čitateľnosť
- úhľadnosť
- plynulosť

Kapitola 3

Návrh aplikácie na výučbu písania

Výučba písania sa od rozpoznávania písma veľmi líši. Zatiaľ čo pri rozpoznávaní stačí povedať, s akou pravdepodobnosťou je napísané písmeno istým písmenom, pri výučbe sa potrebuje dieťa nielen trénovať písanie písmen, ale tiež odstraňovať chyby, ktoré pri písaní robí. Rozpoznávacie algoritmy sa snažia spravené chyby ignorovať, aby dosiahli rozpoznanie. Mojím cieľom je hlavne poukazovať na chyby, ktoré dieťa spravilo.

Toto znie ako negativistický prístup k výučbe, ktorý je tak typický pre naše školstvo. Avšak ak sa správne použije, dieťa poukazovanie na chyby nepocíti. Stačí, ak aplikácia zistí, aké chyby dieťa robí a následne bude vyberať cvičenia, ktoré vedú k odstraňovaniu týchto chýb.

3.1 Aplikácia po obsahovej stránke

Najprv je dôležité uvedomiť si nasledovné ([Tes07])

> Posuzujeme-li vhodnost využití počítače při reedukaci specifických poruch učení, musíme si uvědomit, že počítač nemůže nahradit všechny nebo alespoň většinu činností člověka. Vyjdeme proto z obvyklých činností, které terapeut při nápravě poruch provádí, a posoudíme, které můžeme svěřit počítači. Jde zejména o následující úkoly :
>
> - vyhledávání archivovaných dat a posouzení následujícího postupu nápravy, výběr metody,

- příprava pomůcek pro příslušnou metodu (kontrola správnosti a úplnosti dat), instruktáž a zacvičení dítěte,
- hodnocení činnosti dítěte při plnění jednotlivých úkolů,
- třídění výsledků podle zvolených kritérií,
- interpretace výsledků,
- navržení dalšího postupu nápravy,
- archivace získaných informací,
- periodické zpracování přehledů a hodnocení.

Aplikácia by teda nemala nahradiť úplne učiteľa. Mala by mu však pomôcť poukázať na chyby, ktoré on prehliadol, chyby, ktoré sú pre človeka ťažko nájditeľné (smer písania, tlak) a archivovať a porovnávať zmeny v písaní každého žiaka. Týmto vieme sledovať aj vývoj žiakovho písma.

Súčasťou aplikácie by nemal byť iba samotný nástroj pre deti na písanie, ale aj nástroj pre učiteľa na analýzu žiakovho písma.

Výukový program musí zaistiť tri nutné podmienky ([Tes07]):

1. predanie informacií (učiva) žiakovi,
2. kontrolu získanej úrovne znalostí,
3. následnú reakciu podľa výsledkov spätnoväzobnej informácie.

Náša aplikácia môže spĺňať kritéria nasledovne:

1. animovanie procesu písania písmena (sprevádzanú zvukom),
2. kontrolovanie správnosti napísaného písmena,
3. výber úloh v závislosti od (2).

Vhodné je, aby každý znak mal svoju kategóriu, do ktorej patrí. Tým vieme zaistiť vyberanie znaku podľa chyby resp. podľa nastavenej náročnosti úlohy. Napríklad možme začať lastovičkovými písmenami (l, e, b, j), pokračovať oblúkovými (a, o), až prídeme k najťažším (f, g, q).

V aplikácii by sa nemali cvičiť iba ozajstné písmena. Mali by sme cvičiť aj tvarové prvky ([Sup98]). Zaistiť sa to dá tým, že v zozname znakov budú aj samotné prvky, ktoré budú mať príslušnú kategóriu.

> Po uvoľňovacích cvikoch ruky (gymnastické a grafické cviky) nasledujú prípravné cviky budúcich písmen. Zvyčajne sú to tvarové prvky – priamky, hadovky, vlnovky, oblúky, ovály a podobne. Volíme také

cviky, ktoré sa bez zmien môžu používať pri písaní písmen a číslic. Účelné grafické cviky v prípravnom období sa patrične prejavia v celkových požiadavkách na písmo a písanie. Žiak si lepšie uvedomuje stopu (ťah) písmena.

3.2 Odstraňovanie chýb

V aplikácii je vhodné sa zameriavať na chyby a na ich odstraňovanie. Pozrime sa teraz na jednolivé chyby a možnosti ich odstraňovania. Navrhované úlohy by mali byť súčasťou aplikácie, nielen kvôli rozmanitosti, aby sa dieťa neunavilo, ale aj kvôli tomu, aby aplikácia dieťaťu naozaj pomohla.

- **Vysoký tlak**

  Vysoký tlak vieme na vhodných zariadeniach (väčšina tabletových zariadení) ľahko zistiť. Keď prekročí dieťa istý prah tlaku, pošleme ho na cvičenia na rozhýbanie ruky akými sú hlavne krúživé cvičenia. Tie môžeme buď sledovať aplikáciou (cvičenia sa dajú nájsť v [Virg06]) alebo ponechať na dieťa, aby robilo pre ňho zábavné cvičenia a neoverovať programom.

- **Zlý tvar písmena**

  Dieťa si pravdepodobne nezapamätalo písmeno. Preto mu musíme znova vysvetliť postup, ako písmeno napísať. A to nielen typickým vizuálnym vnemom. Treba využívať aj pohyb a sluch. To potvrdzuje aj [Sla06]:

  > Při nácviku vlastního psaní se snažíme zapojit co nejvíce smyslů žáka. Podporujeme vytváření vizuálně-akustických spojů: dítě ohýbá písmena z drátu, sestavuje ze sirek apod.

  Možné cvičenie môže vyzerať nasledovne. Písmeno bude aplikácia postupne animovať a pritom bude pustený zvuk popisujúci písmeno. Napríklad pri písanom písmene *a* sa bude animovať oblúčik a bude pustený zvuk *oblúčik*, pri animovaní dolného zátrhu bude pustený zvuk *jamka*. Realizovať to môžeme tak, že každé písmeno bude mať označkované body, v ktorých sa mení tvarový prvok. Tvarových prvkov je málo (okolo 30), takže ich vieme ľahko nahovoriť.

- **Sklon väčší ako 90 stupňov**

  Podľa [Sup98] takýto sklon spôsobuje nesprávne držanie pera alebo zlé držanie tela pri písaní. Správne sedenie sa dá dosiahnuť jednoduchými upozorneniami a inštrukciami ako si sadnúť. Držanie pera sa dá napraviť

riekankami o sypaní piesku alebo solení. Pri týchto úkonoch má dieťa ruku presne v správnej polohe na uchytenie pera ([Sup98]).

- **Nerovnaký sklon písma**

  Nerovnaký sklon písma je spôsobený nehybnosťou ruky. Väčšinou je to položením ruky na písanku, takže dieťa hýbe iba zápästím ([Sup98]). Dôvod je väčšinou ten, že žiak je už z písania unavený. Dá sa to napraviť napríklad cvičením ruky podľa riekanky, keď dieťa od písanky vstane a začne rukou krúžiť opisujúc rôzne tvary. V prípade, že dieťa nie je unavené, ruku pokladá na písanku preto, že nevie o tom, že na písanke má mať iba jemne položený malíček a ruka sa má voľne pohybovať po písanke. Toto sa dá riešiť vysvetlením a následným trénovaním.

- **Natlačené písmo**

  Väčšinou je toto písmo úzke a nízke. Podľa [Sup98] je problém v malej pohyblivosti ruky v smere písma. To sa dá riešiť rovnako ako pri vysokom tlaku. Dieťa môžeme navádzať na rozcvičovanie ruky vo vzduchu alebo robiť kresliť krúžky a elipsy.

- **Roztiahnuté písmo**

  Podľa [Sup98] je dôsledkom zdĺhavého premýšľania pri písaní, pomalého vybavovania predstavy o písmenách, vrátane motorických predstáv.

  Pri takomto písme je vhodné písmeno viackrát animovať a nechať dieťa písať priamo po animácii. Tým si upevní vizuálnu a aj motorickú predstavu o písmene.

Ako si môžeme všimnúť, dobrá aplikácia na učenie písania musí obsahovať veľa drobnejších úloh, aby mohla naprávať chyby, ktoré dieťa robí. Nielen z dôvodu, že každá chyba má svoje špecifické riešenie. Viac úloh robí aplikáciu pre dieťa zaujímavejšou. Dieťa tým vydrží dlhšie vstrebávať informácie, pretože má z práce radosť.

> Terapeut, který použije řadu zajímavých pomůcek, jistě získá větší nadšení žáků než ten, který se stroze omezí na jednu činnost. Počítač je pomůckou, která vhodně doplní tradiční nápravné postupy, protože přináší velké možnosti i v době mimo vyučování (školní družina, domov apod.) a většina žáků je vítá se zájmem.([Tes07])

3.3 Žiaci s poruchami

Aplikácia, ktorú navrhujeme, je vhodná aj pre žiakov s rôznymi psychickými alebo fyzickými poruchami. Takáto aplikácia môže byť pre nich dobrá pomôcka v čase, keď pri sebe nemajú odborníka na ich poruchu, čo väčšinou býva doma. Vtedy môžu trénovať a zábavnym spôsobom si zlepšovať svoje schopnosti.

- **hyperaktivita**

  Tým, že sa v aplikácii bude striedať veľa úloh rôzneho charakteru, pričom úlohy budú krátke, dieťa nebude strácať záujem, pretože vždy príde niečo nové.

- **dysgrafia**

  Počítač je trpezlivý. Dieťa budeme kvôli dlhej neschopnosti napísať správne tvar písmena často posielať na cvičenie, kde mu bude znak animovaný. Čas na naučenie každého písmenka nebude obmedzený inými žiakmi, ktorí problémy nemajú.

- **mentálna zaostalosť**

  Tak ako v špeciálnych školách, aj v aplikácii záleží na mentálnom veku žiaka a jeho schopnostiach. Úroveň mu môže nastaviť učiteľ alebo rodič.

- **nervové poruchy**

  Zväčša sa prejavujú trasľavým písmom ([Sup98]) alebo občasnými výkyvmi v písme. Pri takýchto poruchách budú posielané na cvičenia na rozcvičenie ruky, čím sa časom zlepší plynulosť písma.

Kapitola 4

Vlastné výsledky

V predchádzajúcej kapitole sme si ukázali, čo všetko je treba zohľadniť pri programovaní ideálnej aplikácie na výučbu písania. V tejto kapitole si povieme niečo o tom, ako zisťovať jednotlivé chyby.

Naprogramovala som aplikáciu, v ktorej je možné skúšať rôzne algoritmy na detekciu spomínaných chýb. Tento softvér je už ľahko rozšíriteľný na celú aplikáciu, ktorá má všetky vlastnosti spomínané v predchádzajúcej kapitole.

Odteraz budem používať nasledovné pojmy:

Definícia *Znak* je konečná postupnosť *kriviek*. *Krivka* je konečná postupnosť *bodov*, predstavujúca jeden ťah pera. *Bod* má svoje súradnice, ak je možné, tak aj tlak a sklon pera. Poradie bodov je určené časom.

Ako najlepší spôsob kontroly sa ukazuje postupná detekcia chýb akými sú posun, zväčšenie, zmenšenie, sklon. Následne môžme znak transformovať do vhodnej polohy a potom overiť správnosť znaku.

Experiment ukazuje, že najlepšie poradie tranformácií a detekcie chýb je nasledovné:

1. škálovanie
2. posun po x-ovej a y-ovej osi
3. zošikmenie
4. verifikácia

Použitím jednoduchých matematických transformácií sa dajú dosiahnúť vynikajúce výsledky. Teraz si popíšeme jednotlivé algoritmy, ktorými môžme detekovať už spomínané chyby.

4.1 Škálovanie

Kedže každý znak sa skladá z konečnej postupnosti bodov, vieme ho uzavrieť do obdĺžnika. Jeho rozmery budú $(x_{max} - x_{min}) \times (y_{max} - y_{min})$, kde x_{max} je maximálna, x_{min} je minimálna x-ová súradnica; y_{max} je maximálna a y_{min} je minimálna y-ová súradnica.

Algoritmus bude vyzerať nasledovne:

GLYPH SCALING

1 ▷ Each line has to be scaled
2 **for** $line \leftarrow 1$ **to** $min(length(model), length(scribble))$
3 **do**
4 ▷ Finding rectangle boundaries for both: model and scribble
5 $x_{min}^{(model)} \leftarrow min(x_{line,1\ldots\, length(model[line])}^{(model)})$
6 $y_{min}^{(model)} \leftarrow min(y_{line,1\ldots\, length(model[line])}^{(model)})$
7 $x_{max}^{(model)} \leftarrow max(x_{line,1\ldots\, length(model[line])}^{(model)})$
8 $y_{max}^{(model)} \leftarrow max(y_{line,1\ldots\, length(model[line])}^{(model)})$
9 $x_{min}^{(scribble)} \leftarrow min(x_{line,1\ldots\, length(scribble[line])}^{(scribble)})$
10 $y_{min}^{(scribble)} \leftarrow min(y_{line,1\ldots\, length(scribble[line])}^{(scribble)})$
11 $x_{max}^{(scribble)} \leftarrow max(x_{line,1\ldots\, length(scribble[line])}^{(scribble)})$
12 $y_{max}^{(scribble)} \leftarrow max(y_{line,1\ldots\, length(scribble[line])}^{(scribble)})$
13 ▷ Ratio of the model's and scribble's rectangle
14 $scale_x \leftarrow \frac{x_{max}^{(model)} - x_{min}^{(model)}}{x_{max}^{(scrible)} - x_{min}^{(scrible)}}$
15 $scale_y \leftarrow \frac{y_{max}^{(model)} - y_{min}^{(model)}}{y_{max}^{(scrible)} - y_{min}^{(scrible)}}$
16 ▷ Transformation of each point
17 **for** $point \leftarrow 1$ **to** $length(scribble[line])$
18 **do**
19 $x_{line,point}^{(scribble)} \leftarrow (x_{line,point}^{(scribble)} - x_{line,1}^{(scribble)}) \cdot scale_x + x_{line,1}^{(scribble)}$
20 $y_{line,point}^{(scribble)} \leftarrow (y_{line,point}^{(scribble)} - y_{line,1}^{(scribble)}) \cdot scale_y + y_{line,1}^{(scribble)}$

Uvedomme si, že každá krivka musí byť škálovaná samostatne, pretože dieťa ju píše samostatne. Zväčša má problémy práve s proporciami jednotlivých kriviek.

Problém tohto algoritmu je ten, že zmena výšky nezmení šírku znaku. Podľa [Sup98] platí zásada, že čím je písmo menšie, tým je širšie. Toto však nie je určené žiadnou normou. Z tohto dôvodu to do práce nezahrnieme.

Na nasledujúcom obrázku si ukážeme, ako algoritmus znak naškáloval.

Aplikácia v tomto prípade detekovala nesprávnu výšku a šírku znaku. To je pravdepodobne zapríčinené dlhým rozmýšľaním pri písaní znaku (kapitola 3).

Obr. 4.1: Škálovanie

Pre dieťa by bolo vhodné písmeno mu animovať a nechať ho kresliť podľa animácie.

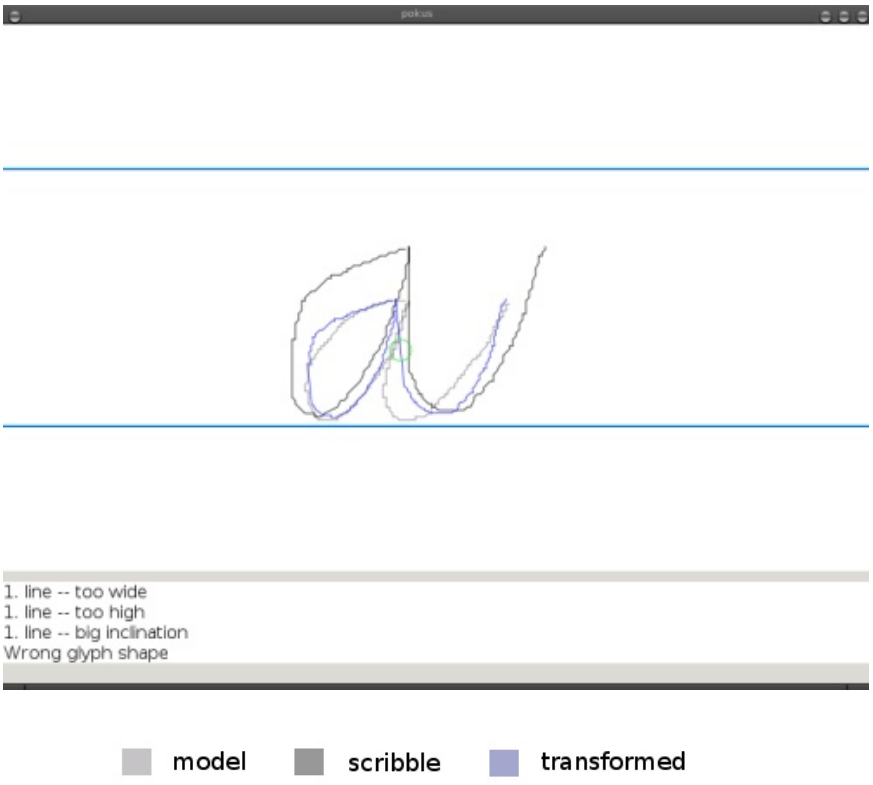
4.2 Posun

Otázka posunu písmena nemá ťažkú odpoveď z algoritmického hľadiska. Problém je didaktický. Je vhodné posúvať znak na linajku? Alebo je vhodné ho posunúť podľa prvého bodu krivky? Alebo je najlepšie posunúť celý obdĺžnik, do ktorého je znak vpísaný? Pozrime sa na všetky možnosti, ich výhody a nevýhody.

Ak písmeno nezačína na linajke, dieťa si vyberie pozíciu niekde v riadku, ktorá sa mu zdá vhodná proporcionálne. Potom pokračuje v písaní a linajku väčšinou trafí. To sa odrazí na veľkosti písma. Kedže krivku škálujeme pred

posunom, výsledok bude v prípade posunu na linajku rovnaký, ako keby sme posúvali podľa prvého bodu krivky. Tým sme však odhalili chybu, ktorú dieťa robilo. Netrafilo proporcie vrámci riadku, kde má začať písať. V tomto prípade je teda lepšie použiť posun podľa prvého bodu krivky. Ak by sme posúvali celý obdĺžnik, tak sme v rovnakej situácii. Viedlo by to k rovnakému výsledku, avšak by sme nepoukázali na správnu chybu.

Obr. 4.2: Posun podľa prvej súradnice



Shifting according to the first point

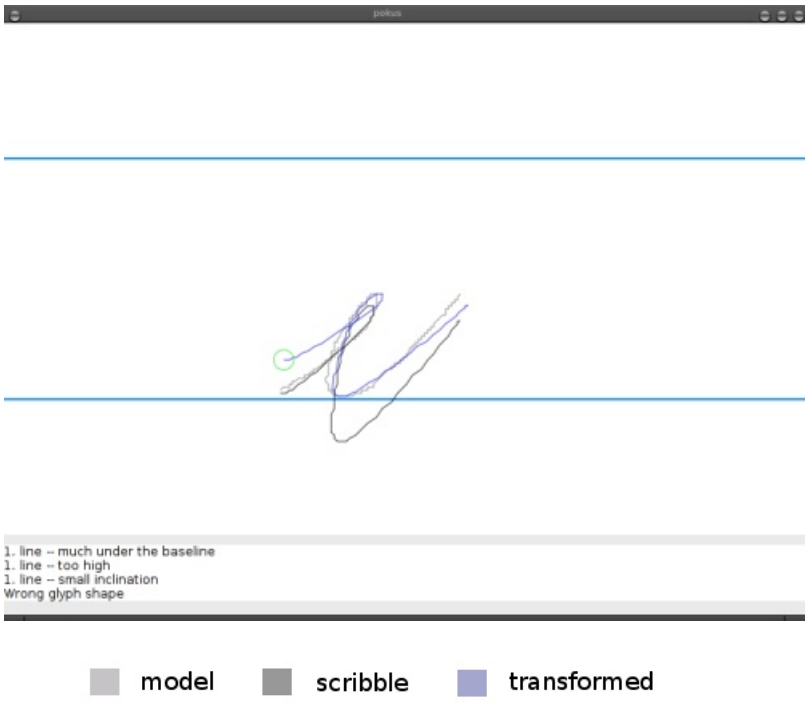
```
1  ▷ Each line has to be shifted
2  for line ← 1 to min(length(model), length(scribble))
3       do
4            ▷ Shift is difference between first points of model and scribble
5            shift_x ← x_{line,1}^{(model)} − x_{line,1}^{(scribble)}
6            shift_y ← y_{line,1}^{(model)} − y_{line,1}^{(scribble)}
7            ▷ Each point is going to be shifted
8            for point ← 1 to length(scribble[line])
9                 do
10                     x_{line,point}^{(scribble)} ← x_{line,point}^{(scribble)} + shift_x
11                     y_{line,point}^{(scribble)} ← y_{line,point}^{(scribble)} + shift_y
```

Ak písmeno začína na linajke, dieťa na linajke aj začína, pretože prvý bod pre ňho nie je problém trafiť. Pozrime si napríklad na malé písané e. Dieťa začne na linajke. Pokračuje a dolný zátrh sa dostane buď nad linajku alebo pod linajku. Ak chceme nájsť chybu, je lepšie sa pozrieť na dolný zátrh. Možný spôsob je pozrieť sa na minimálnu y-ovú súradnicu v druhej a tretej tretine krivky. Ak chceme krivku transformovať, nie je optimálny spôsob presunúť krivku podľa tohto bodu. V tomto prípade je lepšie presunúť krivku podľa hraníc opísaného obdĺžnika.

Shifting according to the rectangle

```
1  ▷ Each line has to be shifted
2  for line ← 1 to min(length(model), length(scribble))
3       do
4            ▷ Finding topleft rectangle's corner for both: model and scribble
5            x_min^{(model)} ← min(x_{line,1... length(model[line])}^{(model)})
6            y_min^{(model)} ← min(y_{line,1... length(model[line])}^{(model)})
7            x_min^{(scribble)} ← min(x_{line,1... length(scribble[line])}^{(scribble)})
8            y_min^{(scribble)} ← min(y_{line,1... length(scribble[line])}^{(scribble)})
9            ▷ Shift is difference between topleft corners
10           shift_x ← x_min^{(model)} − x_min^{(scribble)}
11           shift_y ← y_min^{(model)} − y_min^{(scribble)}
12           ▷ Each point is going to be shifted
13           for point ← 1 to length(scribble[line])
14                do
15                    x_{line,point}^{(scribble)} ← x_{line,point}^{(scribble)} + shift_x
16                    y_{line,point}^{(scribble)} ← y_{line,point}^{(scribble)} + shift_y
```

Obr. 4.3: Posun podľa opísaného obdĺžnika



4.3 Zošikmenie

Sklon je jedna z najdôležitejších vlastností písma. Prezrádza veľa o pohyblivosti ruky, správnom držaní pera a sedení dieťaťa. Napravenie práve týchto problémov pomáha dieťaťu vyvýjať sa, aby nemalo v dospelosti zdravotné problémy. Napravenie sklonu je teda niečo ako hygiena písania. Preto zošikmeniu venujeme v práci veľa priestoru.

Transformácia, ktorá presne modeluje sklon písma, je zošikmenie y-ovej osi. Algoritmus na zošikmenie podľa uhlu α, ktorý zachováva výšku znaku vyzerá nasledovne:

GLYPH SKEWING BY α

```
1  ▷ Each line has to be skewed
2  for line ← 1 to min(length(model), length(scribble))
3       do
4          ▷ Each point is going to be skewed
5          for point ← 1 to length(scribble[line])
6               do
7                  x^(scribble)_(line,point) ← x^(scribble)_(line,point) + y^(scribble)_(line,point) · cos α / sin α
8                  ▷ Y-coordinate doesn't change
```

Problémom je, ako nájsť vhodný uhol α, o ktorý treba natočiť znak. Jedna z možností je skúšať uhly vzdialené o seba o malú konštantu. Pre každý z týchto uhlov vypočítame pomocou verifikačného algoritmu pravdepodobnosť, že znak opisuje model a vyberieme uhol s najvyššiou pravdepodobnosťou. V prípade použitia DTW algoritmu na verifikáciu bude časová zložitosť tohto algoritmu až $O(N \cdot M \cdot A)$, kde N je počet bodov v modely, M je počet bodov v znaku a A je počet uhlov, ktoré skúšame.

FINDING α WITH CONSTANT STEP

```
1  ▷ Each line has it's own α
2  for line ← 1 to min(length(model), length(scribble))
3       do for α ← −φ/4 to φ/4
4            do if verify(skew(α)) > best
5                 then best ← verify(skew(α))
6                      best_α ← α
```

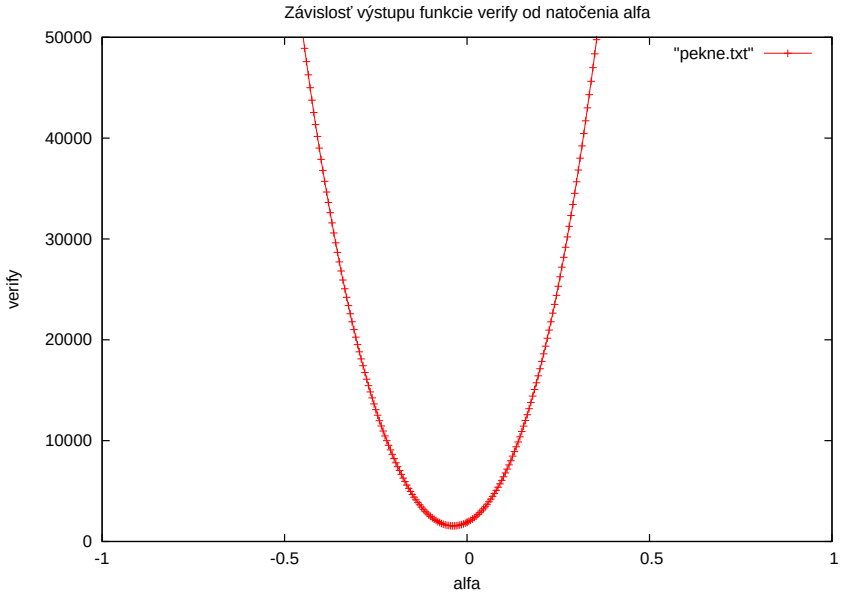
Ak by bola funkcia *verify* unimodálna[1], mohli by sme na nájdenie vhodnej α použiť binárne vyhľadávanie, čím by sme algoritmus zefektívnili. Problém je, že táto funkcia unimodálna nie je. V okolí optimálnej α dosahuje najvyššie čísla, avšak sú aj iné lokálne maximá.

Pozrime sa na funkciu *verify* pri použití DTW algoritmu.

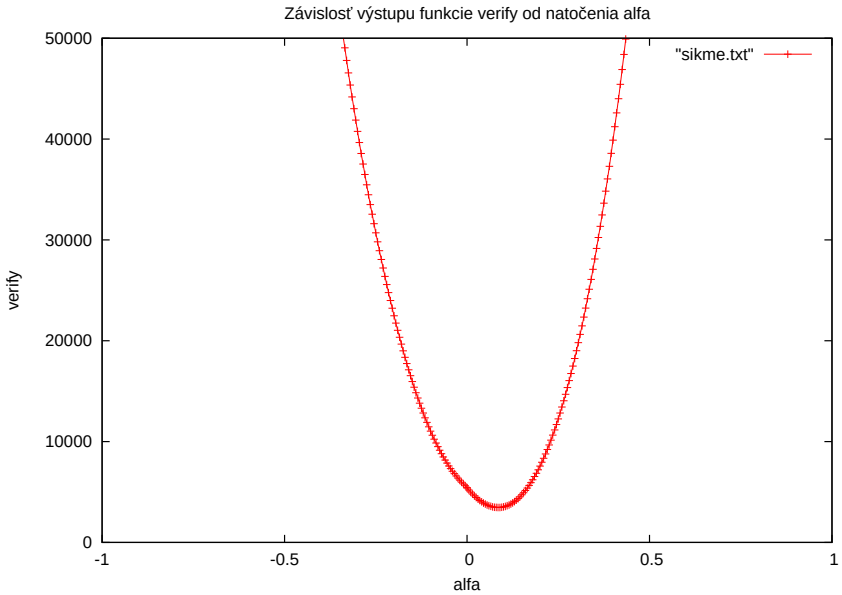
Na takto vyzerajúce funkcie sa bežne používa algoritmus, v ktorom sa najprv dostaneme do okolia globálneho maxima a následne ternárnym vyhľadávaním nájdeme maximum. Binárne vyhľadávanie nie je vhodné, pretože pracujeme s reálnymi číslami, takže občas môže zaokrúhľovanie spôsobiť problém s rovnosťou. Potom nevieme isto povedať, ktorým smerom je treba sa vybrať.

[1] mala iba jedno lokálne maximum

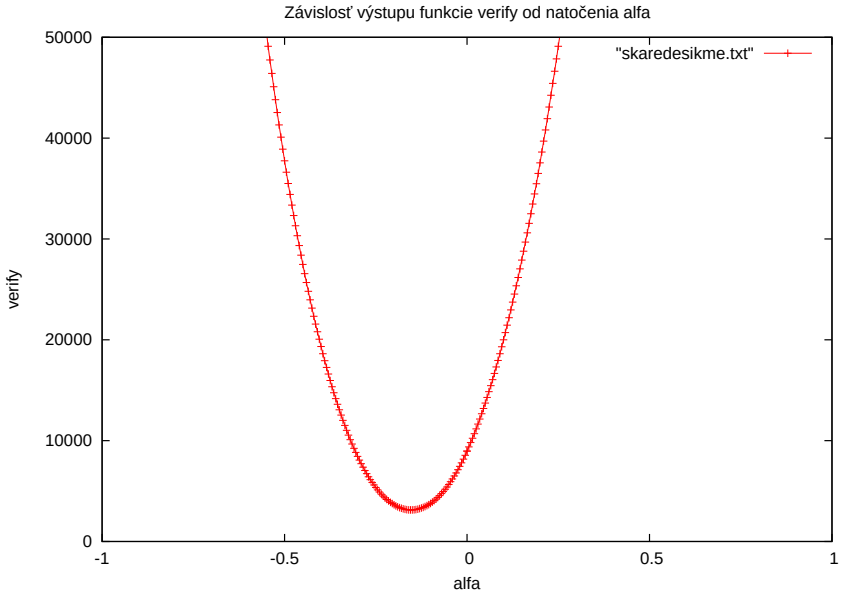
Obr. 4.4: Verify v pekne napísanom písmene



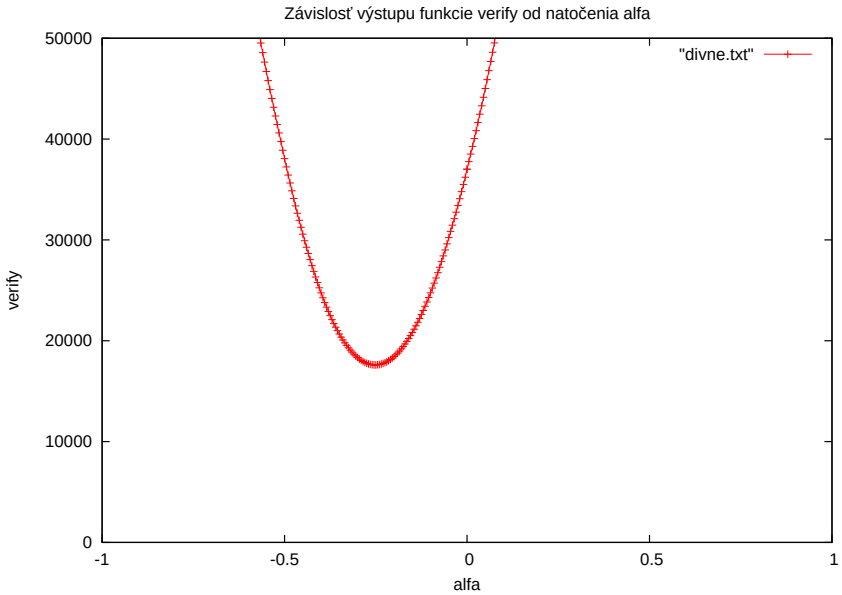
Obr. 4.5: Verify v šikmo napísanom písmene



Obr. 4.6: Verify v šikmo a škaredo napísanom písmene



Obr. 4.7: Verify v nepodobajúcom sa písmene



4.3.1 Ternárne vyhľadávanie v okolí maxima

Okolie maxima nájdeme pomocou procedúry FINDING α WITH CONSTANT STEP. Krok v tomto prípade bude veľký, napríklad stotina celého intervalu. Následne budeme robiť ternárne vyhľadávanie v okolí nájdeného maxima s polomerom veľkosti kroku. V okolí maxima sa už funkcia bude správať unimodálne.

Majme teraz interval $< a, b >$. Niekde v tomto intervale sa nachádza globálne maximum. Pozrime sa na prvok g v jednej tretine intervalu a na prvok h v dvoch tretinách intervalu. Vypočítajme funkciu v bode g a bode h. Ak je funkcia v bode g menšia (hľadáme maximum), môžme prvú tretinu intervalu vynechať a hľadať maximum v intervale $< g, b >$. Obdobne pre $f(g) > f(h)$ sa maximum bude nachádzať v intervale $< a, h >$.

Prečo si môžme dovoliť takéto niečo spraviť? Uvedomme si, že od globálneho maxima sú všetky prvky menšie, teda aj $f(g)$ a $f(h)$. Teraz sú dve možnosti:

- g **aj** h **sú na jednej strane od maxima**

  Bez ujmy na všeobecnosti nech je to na pravej strane. Menšie bude $f(h)$ kvôli unimodálnosti funkcie. A maximum sa naozaj nachádza v intervale $< a, h >$.

- g **aj** h **sú na rozdielnej strane od maxima**

  Nech je menšia ľubovoľná z hodnôt $f(g)$ a $f(h)$, interval $< g, h >$ ponecháme.

Dôležité je uvedomiť si, že krok nemôže byť príliš veľký. Povedzme, že polovica znaku je písana nejakým sklonom a druhá polovica iným sklonom. Takto budeme mať minimálne dve lokálne maximá. Budú však od seba dosť vzdialené. Časová zložitosť je logaritmická od požadovanej presnosti.

FINDING α WITH TERNARY SEARCH

```
1  ▷ Each line has it's own α
2  for line ← 1 to min(length(model), length(scribble))
3       do
4          ▷ Finding maximum's neighbourhood
5          for α ← −φ/4 to φ/4
6               do
7                  if verify(skew(α)) > best
8                     then best ← verify(skew(α))
9                          best_α ← α
10         ▷ Ternary search
11         min ← best_α − step
12         max ← best_α + step
13         while max − min < ε
14              do
15                 g ← min + (max−min)/3
16                 h ← min + 2 · (max−min)/3
17                 if f(g) < f(h)
18                    then max ← h
19                    else  min ← g
20         return (max+min)/2
```

4.3.2 Transformácia pomocou nájdeného sklonu

Pozrime sa na výsledky zošikmenia na obr. 4.8.

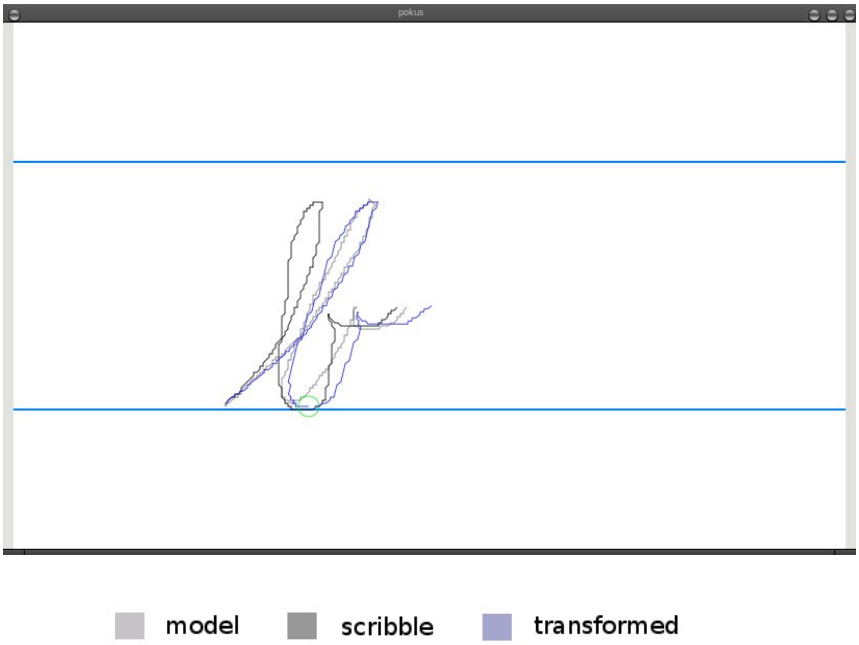
Toto písmo bolo hodnotené ako reverzné písmo. Ako sme si prečítali v predchádzajúcej kapitole, problém takéhoto písma je v nesprávnom držaní pera alebo v nesprávnom sedení. Po takomto výsledku je vhodné zaradiť cvičenie, v ktorom sa tento problém vyrieši.

4.4 Verifikácia

V aplikácii je na verifikáciu použitý algoritmus DTW. Vysvetlime si dôvod, prečo bolo lepšie použiť DTW algoritmus ako Skryté Markovovské modely.

Najprv si uvedomme, že sme použili všetky základné transformácie. Ostáva nám zistiť, či je písmeno chybné. Dieťa písalo podľa predlohy, takže pred sebou videlo obraz písmena, ktoré má napísať. Chyby, ktoré po transformácii budú nastávať, budú rôzne lokálne výkyvy, prípadne viacnásobné oblúky. V Skrytých Markovovských modeloch vieme povedať iba s akou pravdepodovnosťou je znak

Obr. 4.8: Zošikmenie písmena

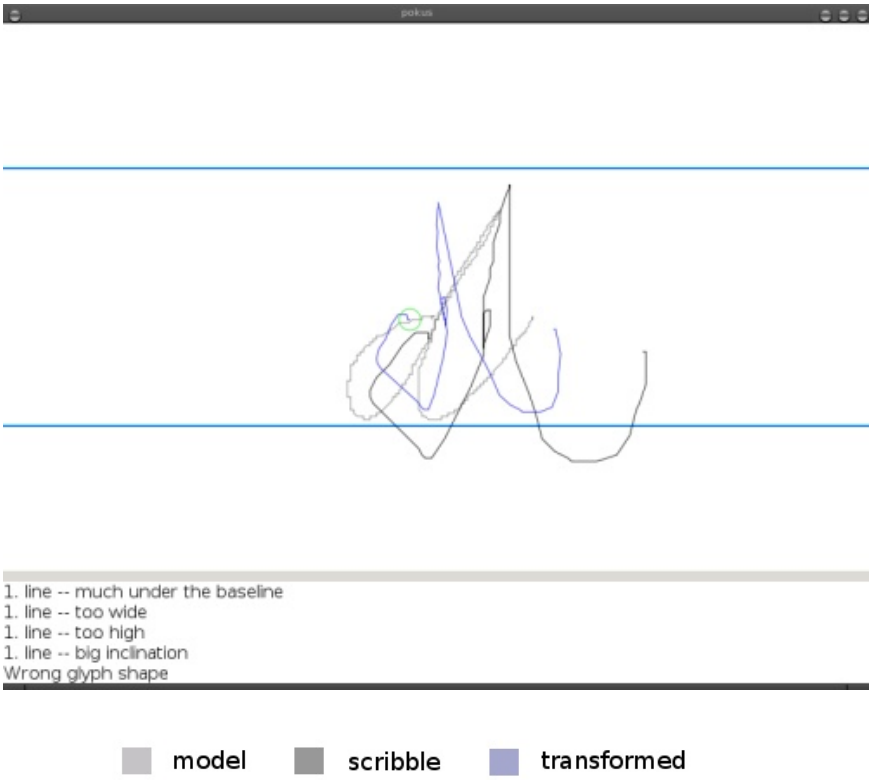


daným písmenom. V DTW algoritme je miera podobnosti presnejšia. Toto sa nám odrazí v časovej zložitosti riešenia. Neskôr si ukážeme, čo nám použitie DTW algoritmu prinesie.

Teraz sa pozrieme, aké výsledky získame použitím algoritmu.

Ako na obr. 4.9 vidno, dieťa sa snažilo napísať zadané písmeno, avšak sa mu to nepodarilo. Výstup algoritmu bol okolo 30000, akceptovaný znak by musel mať výstup do cca. 5000. Pri takomto výsledku by malo byť dieťa poslané na nacvičenie si písania znaku. Buď by to mala byť animácia so zvukovým popisom, alebo sledovanie znaku podľa animácie.

Obr. 4.9: Verifikácia pomocou DTW algoritmu



Dynamic time warping

```
1  for line ← 1 to min(length(model), length(scribble))
2       do
3          ▷ Initialization
4          for i ← 1 to length(scribble[line])
5               do DTW[0, i] ← ∞
6          for i ← 1 to length(model[line])
7               do DTW[i, 0] ← ∞
8          DTW[0, 0] ← 0
9          ▷ Algorithm's body
10         for i ← 1 to length(model[line])
11              do
12                 for j ← 1 to length(scribble[line])
13                      do cost ← dist[s[i], t[j]]
14                         DTW[i, j] ← cost + min(DTW[i − 1, j],
15                                                DTW[i, j − 1], DTW[i − 1, j − 1])
16         return DTW[length(model[line]), length(scribble[line])]
```

4.5 Detekcia miesta chyby pomocou DTW algoritmu

Otázkou býva, nielen aké chyby dieťa spravilo. Zaujímavé je tiež zistiť, kedy začalo robiť chybu vrámci znaku. Ako niečo takéto môžme zistiť?

Stačí drobná úprava DTW algoritmu. V prvom rade potrebujeme nájsť priradenie bodov medzi modelom a znakom. To dosiahneme tým, že si nebudeme pamätať len aký je najlepší výsledok pre prefixnú podpostupnosť modelu a znaku. Zapamätáme si tiež, či sme pridali nové priradenie bodov ($[i-1, j-1]$), alebo sme pridaný bod spojili s bodom, ktorý už pár má ($[i, j-1]$), alebo sme pridanému bodu iba priradili ďalší pár ($[i-1, j]$).

DTW binding in Algorithm's body

```
1  DTW[0,0] ← 0
2  Mem[0,0] ← (−1,−1)
3  ▷ Algorithm's body
4  for i ← 1 to length(model[line])
5        do
6           for j ← 1 to length(scribble[line])
7                 do cost ← dist[s[i], t[j]]
8                    DTW[i,j] ← cost + min(DTW[i−1,j],
9                                          DTW[i,j−1], DTW[i−1,j−1])
10                   Mem[i,j] ← min_arg(DTW[i−1,j],
11                                         DTW[i,j−1], DTW[i−1,j−1])
12 return DTW[length(model[line]), length(scribble[line])]
```

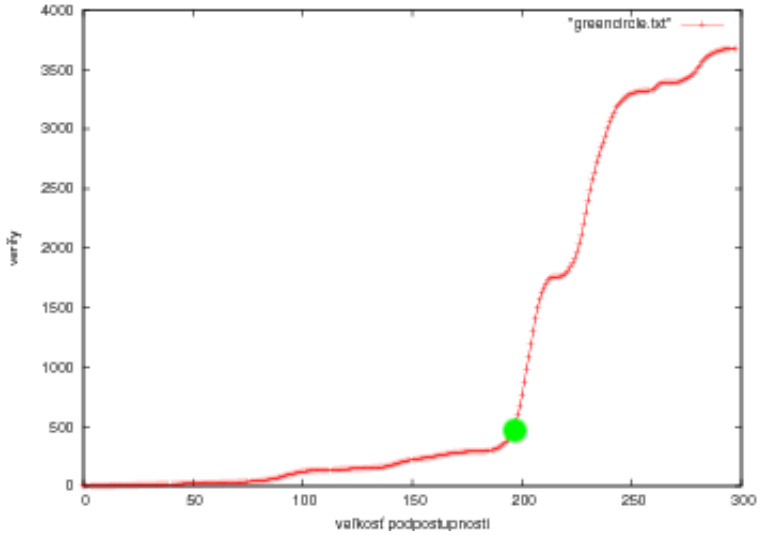
Aby sme zistili priradenie, stačí nám ho už iba spätne zrekonštruovať. Začneme v poslednom bode a postupne prechádzame po súradniciach poľa DTW podľa hodnoty v poli Mem.

DTW binding reconstruction

```
1  Z_0 ← Mem[length(model[line]), length(scribble[line])]
2  index^(model) ← Z_0^(model)
3  index^(scribble) ← Z_0^(scribble)
4  while (index^(model) > 0) ∨ (index^(scribble) > 0)
5        do
6           Z_next ← Mem[index^(model)][index^(scribble)]
7           index^(model) ← Z_last^(model)
8           index^(scribble) ← Z_last^(scribble)
9  return reverse(Z)
```

Pozrime sa teraz, ako vyzerá výstup DTW algoritmu na prefixnej podmnožine priradení (obr. 4.10)

Obr. 4.10: Výstup DTW algoritmu na prefixnej podmnožine priradení



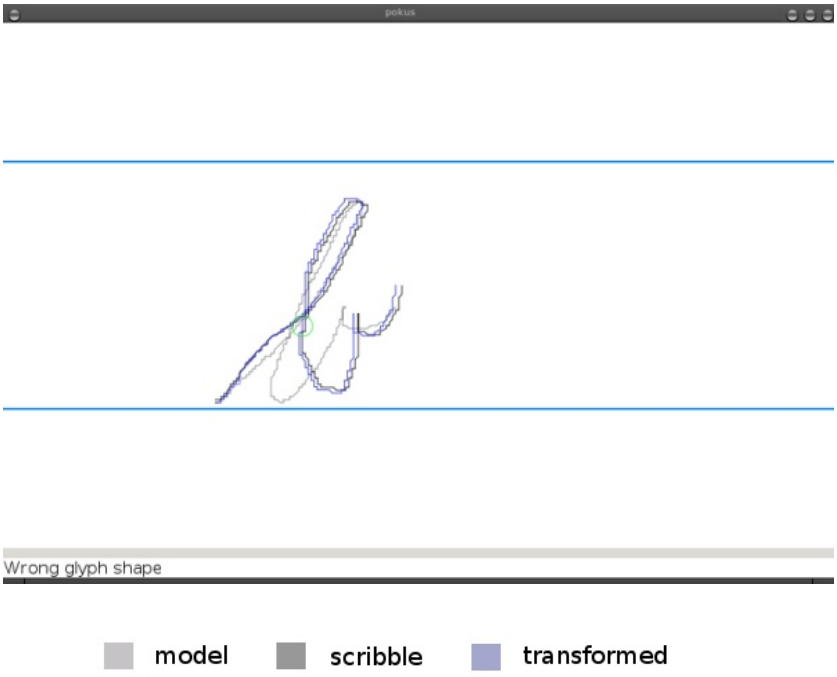
V zelenom bode nastala veľká zmena. Toto miesto stojí v detskom znaku za povšimnutie. Tam dieťa prešlo do nesprávneho smeru.

Na obr. 4.11 môžme toto miesto vidieť označené zeleným krúžkom. Ak by sme mali v aplikácii označené aj jednotlivé časti znaku (čo treba kvôli cvičeniu s animáciou a zvukovým popisom častí znaku – Kapitola 3), mohli by sme povedať, kde v písmene nastala chyba, či je to v hornej slučke alebo v dolnom zátrhu. Ak by bola chyba v hornej slučke, aplikácia môže vyberať v ďalších cvičeniach písmená v kategórii "lastovičkové", čím sa prehĺbi znalosť písania tejto časti znaku.

Čo ak je takýchto miest viac? Zaujímavé by bolo použiť algoritmus *zošikmenia* od miesta chyby. Znova nájdeme najlepší sklon a možme v zošikmenej časti znaku hľadať ďalšie miesto chyby. Zaújmavosťou je, že **týmto spôsobom nájdeme aj nerovnaký sklon znaku**, ktorý je veľmi dôležitý pri písaní slov, pretože podľa [Sup98] je prejavom malej pohyblivosti ruky zľava doprava, keď dieťa má položenú ruku na písanke a hýbe iba zápästím.

Zatiaľ sme si nepovedali, ako toto miesto algoritmicky detekovať. V aplikácii som si vybrala spôsob, v ktorom sa pozeráme na prírastok vo výstupe vzniknutý pridaním konštantného počtu priradení. Ak chyba nenastala, tento prírastok by nemal byť väčší ako $15 \cdot p$, kde p je počet priradení, na ktorých zmenu sledujeme.

Obr. 4.11: Zobrazenie miesta chyby v detskom znaku



Číslo 15 nebolo celkom náhodné. Ak dieťa píše podľa čiary (po transformácii), má drobné odchylky, ktoré sa zmestia do 15 pixelov. V aplikácii som položila $p = 10$. Tento algoritmus sa javil ako veľmi dobrý a výborne detekoval chyby, čo si môžme všimnúť aj na obr. 4.11.

FINDING SHAPE ERROR

1 **for** $i \leftarrow 10$ **to** $length(Z)$
2 **do**
3 $diff \leftarrow D[Z_i^{(model)}][Z_i^{(scribble)}] - D[Z_{i-10}^{(model)}][Z_{i-10}^{(scribble)}]$
4 **if** $diff > 15 \cdot 10$
5 **then**
6 ▷ Error found at Z_{i-10}

Kapitola 5

Záver

V tejto práci sme sa venovali popisu aplikácie a algoritmov, použiteľných na výučbu písania. Aplikácia by mala brať do veľkej miery ohľad na didaktiku písania a mohla by byť použitá na hodiny písania, aby tak pomáhala učiteľovi analyzovať chyby v žiakovom písme a robiť prácu, ktorá je pre učiteľa časovo náročná alebo nemožná, čím by sa mu uvoľnil priestor na prácu so žiakmi ako jednotlivcami.

Motiváciou k tejto práci bolo práve množstvo prác, používajúcich bežné algoritmy na rozpoznávanie písma, ktoré sa neopierajú o didaktiku písania, pričom to, čo je práve dôležité vo výučbe zanedbávajú. My sme sa to touto prácou snažili napraviť a ukázať na iný smer uvažovania.

Rozobrali sme návrh aplikácie aj po obsahovej a aj po implementačnej stránke, pričom sme sa venovali hlavne algoritmom na detekciu chýb, akými sú nesprávny sklon, veľkosť, posun a rôzne odchýlky od znaku. Práve tieto chyby hrajú kľúčovú rolu v didaktike písania. Tiež sme poukázali na dôvody, prečo vôbec aplikáciu na hodinách písania využívať a prečo sú chyby také signifikantné.

5.1 Do budúcnosti

V budúcnosti by bolo dobré sa venovať aj detekcii trasľavosti písma, ktorá sa dá robiť porovnaním s vyhladenou krivkou. Taktiež sa treba pozrieť, či by tieto chyby vhodným algoritmom nevedeli detekovať Skryté Markovovské modely a nielen chyby zanedbávať, tak ako to bežné algoritmy robia. Tým by sme vedeli algoritmy zefektívniť. Otázkou tiež ostáva, či nevieme zefektívniť DTW algoritmus, kedže problém najdlhšej spoločnej podpostupnosti efektívnejšie riešenie má.

Bolo by dobré, keby aplikácia bola schopná detekovať nesprávne poradie čiar, prípadne písanie z iného smeru. Síce znak animujeme, dieťa môže túto chybu aj tak spraviť. Po detekcii takejto chyby by bolo vhodné ho poslať na cvičenie prehlbujúce kinetickú znalosť procesu písania písmena. Dá sa to robiť buď algoritmami offline rozpoznávania písmen alebo skúšaním rôznych permutácií kriviek. Pre efektívnosť nie je vhodné naozaj permutovať krivky, ale použiť dynamické programovanie.

Onedlho sa pokúsim naprogramovať celú aplikáciu podľa popísaného návrhu v programe Google Summer of Code 2008. Aplikácia bude open-source. Dúfam, že sa táto aplikácia bude môcť využívať aj na hodinách písania, čím sa nielen, že prehĺbi využívanie informačných a komunikačných technológií na prvom stupni základnej školy, môže to tiež pomôcť k zviditeľneniu dobrých pedagogických open-source aplikácií a hlavne môže to pomôcť niektorým žiakom, ktorí by mali bežne na hodinách problémy.

Algoritmy by bolo vhodné vyskúšať aj na väčšej vzorke detí, aby sme mali aj kvantitatívny výskum. Ja som vhodnosť algoritmov overovala iba na malej vzorke študentov FMFI UK.

Dodatok A

Obsah CD

Na CD sa nachádza skomprimovaný archív scribbler.tar.gz. Po jeho rozbalení nájdete v adresári *scribbler* zdrojové kódy aplikácie. V adresári *scribbler/data* sú data jednotlivých znakov, ktoré už v aplikácii existujú.

Na skompilovanie aplikácie treba mať okrem kompilátora *gcc* nainštalované *gtkmm* najlepšie verzie gtkmm-2.12 (aj keď verzie od 2.4 by mali fungovať). V prípade inej verzie treba príslušné prepínače kompilátora prepísať v *Makefile*. Kompiláciu pustíte príkazom *make*.

Na spustenie aplikácie použite príkaz *./scribbler*.

Literatúra

[Cech] Lukáš Čechovič: Neurónové siete pri rozpoznávaní písma http://frtk.fri.utc.sk/~cechovic/dajzdrojak.pdf

[Sra07] Rasťo Šrámek: The on-line Viterbi algorithm, KAI FMFI UK, Bratislava, máj 2007

[Tom07] Marek Tomacha: Počítač - učiteľ písania (rozpoznávanie písaných písmen pomocou HMM), KAI FMFI UK, Bratislava, máj 2007

[Cer07] Jozef Červeň: Počítač učiteľ písania (počítač pomáha učiť písať v 1.ročníku ZŠ), KZVI FMFI UK, Bratislava, máj 2007

[WFST] Mehryar Mohri , Fernando Pereira , Michael Riley: Weighted Finite-state Transducers in Speech Recognition, Computer Speech & Language, 2002

[Sup98] Božena Šupšáková: Písmo a písanie, Univerzita Komenského, Bratislava, 1998, ISBN 80-223-1293-2

[Rab89] Lawrence R. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proceeding of the IEEE, volume 77, no. 2, February 1989

[SalCha] Stan Salvador, Philip ChanFast: DTW: Toward Accurate Dynamic Time Warping in Linear Time and Space http://cs.fit.edu/ pkc/papers/tdm04.pdf

[MR81] C. S. Myers, L. R. Rabiner: A comparative study of several dynamic time-warping algorithms for connected word recognition, The Bell System Technical Journal, 60(7):1389-1409, september 1981

[Jacks] Philip Jackson: Dynamic Time Warping, slides for lecture at University of Surrey http://info.ee.surrey.ac.uk/Teaching/Courses/eem.ssr/slides02.pdf

[Virg06] Lýdia Virgovičová: Moje prvé čiary, blok prípravných cvikov pre 1. ročník ZŠ, Orbis Pictus Istropolitana, Bratislava, 2008, ISBN 80-7158-650-1

[Sla06] Michaela Slavíková: Reedukace specifických poruch učení v doplňovacích hodinách na 1. stupni ZŠ, Masarykova univerzita, Brno, 2006 `http://is.muni.cz/th/65823/pedf_m/`

[Tes07] Ivana Tesařová: Možnosti využití počítačů ve vzdělávání žáků se specifickými poruchami učení, Masarykova univerzita, Brno, 2007 `http://is.muni.cz/th/43727/pedf_m/`