A számítógépes mozgásminta-felismerés egy alkalmazása

Japán és más „egzotikus” betűk tanítása számítógéppel

Tartalomjegyzék

1. Bevezetés

Az emberiség egyik legnagyobb vívmánya a megszerzett, felismert tudás tárolása és visszakeresése, az írás és olvasás képessége. Ez tette lehetővé, hogy a tapasztalatok generációk sokaságán át fennmaradjanak, bővüljenek, folyamatosan hozzájáruljanak a közösség fejlődéséhez, gyarapodásához: a mások által korábban verejtékes munkával megszerzett ismeretek ne vesszenek el, ne kelljen azokat újra és újra felfedezni. Az ókori írások ma is érdekes, sokszor tanulságos, megdöbbentő emlékek.

A különböző kontinenseken különbözőféle írások alakultak ki. A magyarok ősei rovásírást használtak. A képírás és hangírások számos különböző változata terjedt el. A legutóbbi évszázadokig az írás és olvasás „tudománya" csak a kiváltságosoknak adatott meg. Mára az írás és olvasás ismerete széles körben elterjedt, s egyre inkább felmerül az igény, hogy gondolatainkat idegen nyelveken is fejezzük ki, foglaljuk írásba.

Bár az angol világnyelvnek tekinthető, a világon legtöbben mégsem ezt használják anyanyelvükként. Az „egzotikus", nem latin betűs nyelvek (japán, kínai, orosz, héber, arab…) nemcsak lelkes érdekelődők figyelmét ragadják meg, hanem a globalizálódó világunkban ismeretük hétköznapi előnyt is jelenthet.

Ezért érdemes megvizsgálnunk, hogyan segíthetjük ezek írásjeleinek tanulását-tanítását. A dolgozatban bemutatok egy eljárást, mely a japán, kínai, orosz, héber, arab… nyelvek betűinek tanítása mellett természetesen latin betűs nyelvek írásjeleinek oktatásában is használható. Így tehát az idegen nyelvek betűinek tanítása mellett érdemes végiggondolni, hogy a módszer mennyiben alkalmazható írási nehézségekkel küzdő gyerekek, felnőttek oktatásában.

Munkám során célul tűztem ki a mozgásminta-felismerés prototípus jellegű megvalósítását és kipróbálását egy egzotikus nyelv, például a japán nyelv betűinek tanításakor. Mivel a japán nyelv több ezer írásjelet tartalmaz, csak arra vállalkoztam, hogy a módszert néhány írásjel esetében kipróbáljam. A megoldás a mozgásminták számítógépi felismerésén alapul.

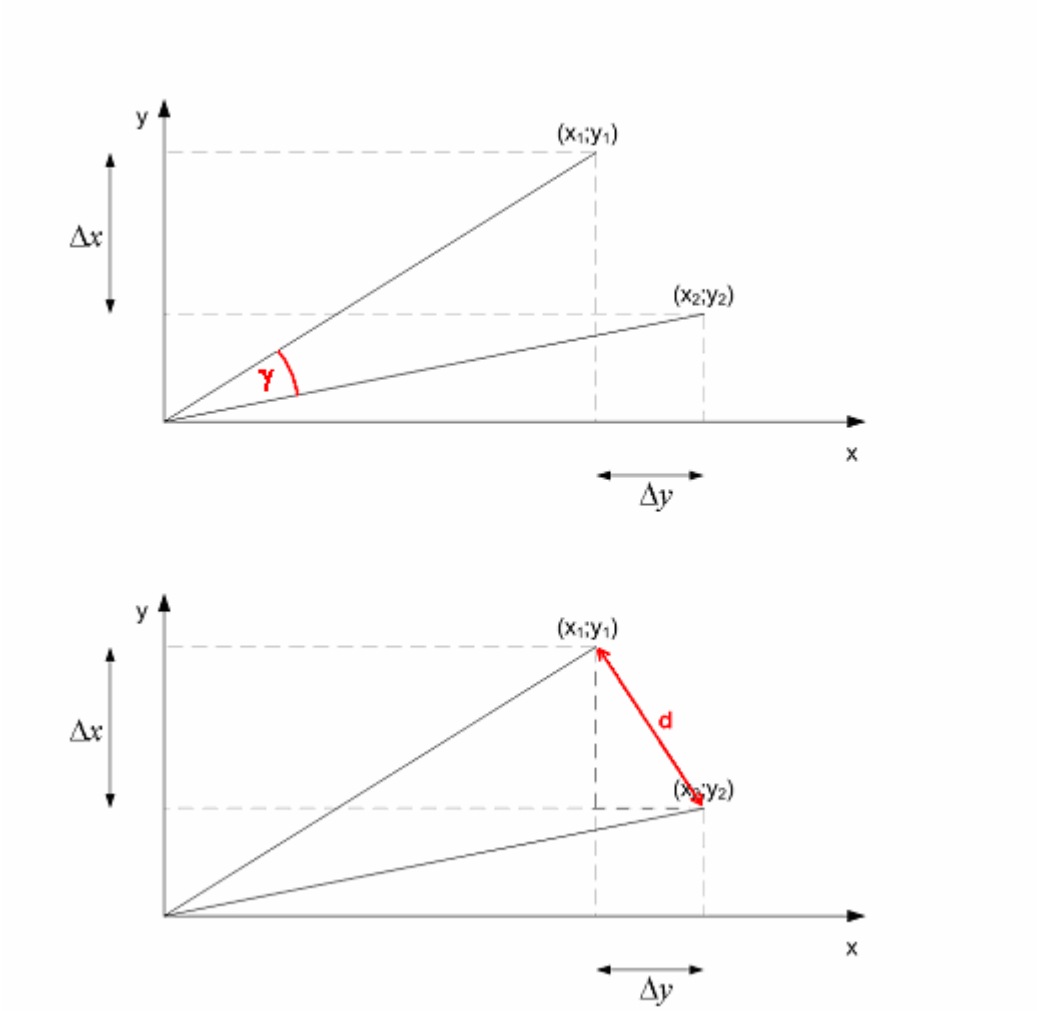
2. Mozgásminták felismerése

A mozgásminták felismerésének, mint feladatnak két fő eleme van: a mozgásminták beolvasása, és ezek összehasonlítása.

A **mozgásminta-beolvasás** lényege az, hogy a mozgás teljes időtartamát rövid (néhány századmásodperces vagy ezredmásodperces) időintervallumokra osztjuk és megnézzük, hogy a felhasználó ujja mennyit mozdult el (feltéve, hogy az ujjával rajzolta be a mintát). Ez többek között egy laptop touch padja segítségével lehetséges, amely azért előnyös, mert a programozás szempontjából a touch pad tejesen azonos az egérrel, ezért könnyen programozható, de a felhasználó tud rá rajzolni az ujjával. A beolvasás módszere tehát az lehet, hogy a rövid időintervallum elején, az egér pozícióját egy meghatározott helyre állítom, ezt követően az egér elmozdítható, majd az időintervallum végén megnézem, hogy hova került az egér, melynek segítségével a felhasználó ujjának elmozdulását mérem. Így vektorok sokaságát kapom, melyek leírják a mozgást.

A feladat másik fontos eleme a bevitt mozgásminták (a mozgásokat leíró vektorsorozatok) összehasonlítása. Mivel két mozgásminta nem azonos hosszúságú, ezért olyan módszert kellene keresni, mely két nem azonos hosszúságú jelsorozatot (vektorsorozatot) is össze tud hasonlítani.

Nézzünk erre egy konkrét példát! Vajon hogyan hasonlíthatók össze a *telefon* és a *mikrofon* szavak (jelsorozatok)? Sokan ismerjük azt a játékot, ahol egyik szóból kell eljutni a másikig úgy, hogy minden lépésben csak egy betű cserélhető ki. Ez a „játék” bővíthető azzal, hogy a betűk kicserélése mellett kivehetünk, illetve beilleszthetünk egy-egy írásjelet. További bővítésként mondhatjuk, hogy mind a három műveletnek valamekkora költsége van. Ekkor a célunk az, hogy a lehető legkevesebb költséggel jussunk el egyik szóból (jelsorozatból) a másikig. A Dynamic Time Wrapping (továbbiakban DTW) algoritmus hasonlóan működik, mint a játék. A DTW egy lehetséges eljárás két, akár különböző hosszúságú jelsorozat hasonlóságának meghatározására: meghatározza, hogy milyen minimális költséggel állítható elő az egyikből a másik. Figyelembe vehetjük azt is, hogy két betű távolsága nem mindig azonos. A két betű kicserélésének költsége kisebb lehet, ha pl. e-t kell é-re cserélünk, mintha e-t x-re cserélnénk. A mozgásminták beolvasásakor kapott jelsorozatok nem betűkből állnak, hanem vektorokból. Ezek összehasonlításáról a 2. 1. fejezetben írok. A DTW-algoritmus részleteit a 2. 2. fejezetben mutatom be.



1. ábra: Vektorok összehasonlítása bezárt szögeik (fent), ill. végpontjaik távolsága (lent) alapján

A mozgásminták beolvasásán és összehasonlításán alapulva egy „egzotikus” nyelv (japán, kínai, orosz, arab, héber, stb.) betűinek tanítását segítő program úgy működhet, hogy a program megkéri a felhasználót egy meghatározott betű beírására, és összehasonlítja azt előre megadott, az adott betű helyes leírását tartalmazó mintákkal. Így lehet gyakorolni a betűk helyes leírását.

2.1. Vektorok összehasonlítása

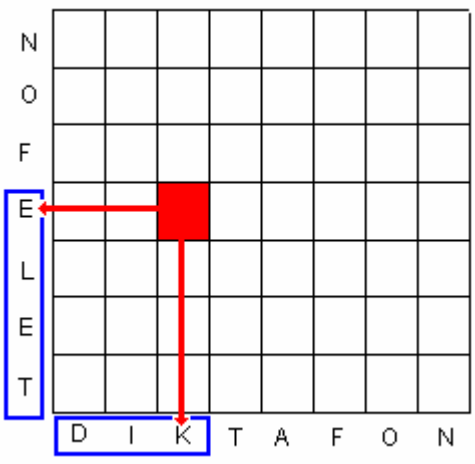
A mozgásminták beolvasása során kapott jelsorozat nem betűkből, hanem vektorokból áll. Ezért szükséges, hogy össze tudjunk hasonlítani két vektort, azaz meg tudjuk mondani, hogy az egyik vektor másikra cserélésének mennyi a költsége. Két vektor hasonlóságának megállapítására több módszer is adódik. Egyik lehetőség a vektorok egymással bezárt szögének mérése. Másik megoldás, ha a két vektort közös kezdőpontba helyezem, és végpontjainak távolságát Pitagorasz-tétellel számítom ki (1. ábra).

A bezárt szög két (x_1,y_1), (x_2,y_2) vektor esetén szögfüggvényekkel határozható meg: ha α az egyik, β a másik vektor x tengellyel bezárt szöget jelöli akkor: $tg\alpha = \frac{y_1}{x_1}$, $tg\beta = \frac{y_2}{x_2}$. Így ha a két szög tangensét ismerem, könnyen meghatározhatjuk a két szöget is. Ekkor a két vektor által bezárt szög: $\gamma = |\alpha - \beta|$ (1. ábra).

2.2. DTW algoritmus

A DTW algoritmus két jelsorozatot hasonlít össze, azt adja meg, hogy a két jelsorozat milyen mértékben különbözik. A DTW azt adja meg, hogy meghatározott költséggel rendelkező szerkesztési lépések segítségével egyik jelsorozatból mekkora költséggel állítható elő a másik. A DTW háromféle szerkesztési lépést kezel: egy jel beszúrása, egy jel kihagyása, egy jel cseréje.

A DTW algoritmus futása közben egy táblázatot tölt ki, ahol a táblázat oszlopainak száma az egyik, a sorainak száma a másik jelsorozat hosszával egyenlő. „Képzeletben” odaírjuk a táblázat mellé oldalt az egyik, alá pedig a másik jelsorozatot (2. ábra). Ekkor a táblázat minden egyes cellája kijelöl egy-egy pozíciót az egyik, illetve a másik jelsorozatban. A táblázat adott cellájába kerülő szám azt adja meg, hogy a jelsorozatok adott pozícióig tartó részsorozatainak mekkora a különbsége.



2. ábra: A DTW-táblázat adott cellájához tartozó részsorozatok

4	8	12	16	20	24	28
3	7	11	15	19	23	27
2	6	10	14	18	22	26
1	5	9	13	17	21	25

3. ábra: A cellák kitöltési sorrendje

Az eljárás a cellákat a 3. ábrán látható sorrendben tölti ki. A kitöltés tehát a dél-nyugati cellával kezdődik, észak felé halad, majd ha nem lehet továbblépni északra, akkor az eggyel keletre lévő legdélebbi cellán folytatódik.

Az egyes cellák kitöltésének menete a következő (tegyük fel, hogy már néhány cellát kitöltöttünk, s például a 3. ábra szerinti 11-es cellánál járunk):

1. Meghatározzuk a cellához tartozó két jel különbségét, jelöljük ezt *k*-val.
2. Tekintjük a cellától dél-nyugatra, délre valamint nyugatra elhelyezkedő cellákat (6, 7, 10). Ezek valamelyikéből fogunk az aktuális cellába „belépni”. Ha dél-nyugatról lépünk, az annak felel meg, hogy mindkét jelsorozatban előre léptünk egyet, tehát nem történt „nyújtás” (azaz nem történt kihagyás vagy beszúrás). A keleti illetve déli cellából való belépés során az egyik jelsorozatban kihagyás, a másikban beszúrás történik, ezért ekkor számolnunk kell a nyújtás (beszúrás, illetve kihagyás) költségével. A nyújtás költségét jelöljük *b*-vel.
3. Arra törekszünk, hogy a lehető legkisebb költséggel állítsuk elő egyik jelsorozatból a másikat, ezért az alábbi három érték közül a legkisebb kerül az aktuális cellába (*dny*, *d*, *ny* a délnyugati, déli, illetve nyugati szomszédos cellákban található értékeket jelölik):

   $dny + k$
   $d + b + k$
   $ny + b + k$

Ha nem létezik mind a három szomszéd (például legelső oszlop kitöltésekor), akkor csak a létező szomszédokkal számolunk. A legelső (dél-nyugati) cellát a két jelsorozat első jeleinek különbségével töltjük ki.

Most, hogy ismerjük az algoritmus alapszabályait, nézzünk meg egy konkrét példát! Vizsgáljuk meg, hogy mekkora a különbsége a *telefon* és a *diktafon* szavaknak! A vizsgálat előtt meg kell határoznunk, hogy hogyan állapítjuk meg két betű különbségét. Az egyszerűség kedvéért első példánkban a nem azonos betűk különbsége legyen 1, az azonosaké 0. A nyújtás költsége pedig legyen 1,5.

A 4. ábrán azt láthatjuk, hogy hogyan tölti ki a DTW algoritmus a táblázatot a fenti költségek figyelembe vételével. A két eredeti jelsorozat különbsége mindig a táblázat észak-nyugati (azaz utoljára kitöltött) cellájában lévő érték.

A TELEFON F-jének és a DIKTAFON O-jának összehasonlításakor jobban járunk, ha F-t és F-et összehasonlító cellából (nyugati szomszéd) „nyújtjuk”, mintha az E-t és F-et összehasonlító cellából (dél-nyugati szomszéd) lépnénk, még akkor is, ha a nyújtás költsége nagyobb mint két eltérő betű költsége.

N	16	14,5	13	11,5	10	8,5	7	**6,5**
O	13,5	12	10,5	9	7,5	6	6,5	9
F	11	9,5	8	6,5	5	6,5	9	11,5
E	8,5	7	5,5	4	6,5	9	10,5	13
L	6	4,5	3	5,5	8	9,5	12	14,5
E	3,5	2	4,5	7	8,5	11	13,5	16
T	1	3,5	6	7,5	10	12,5	15	17,5
	D	I	K	T	A	F	O	N

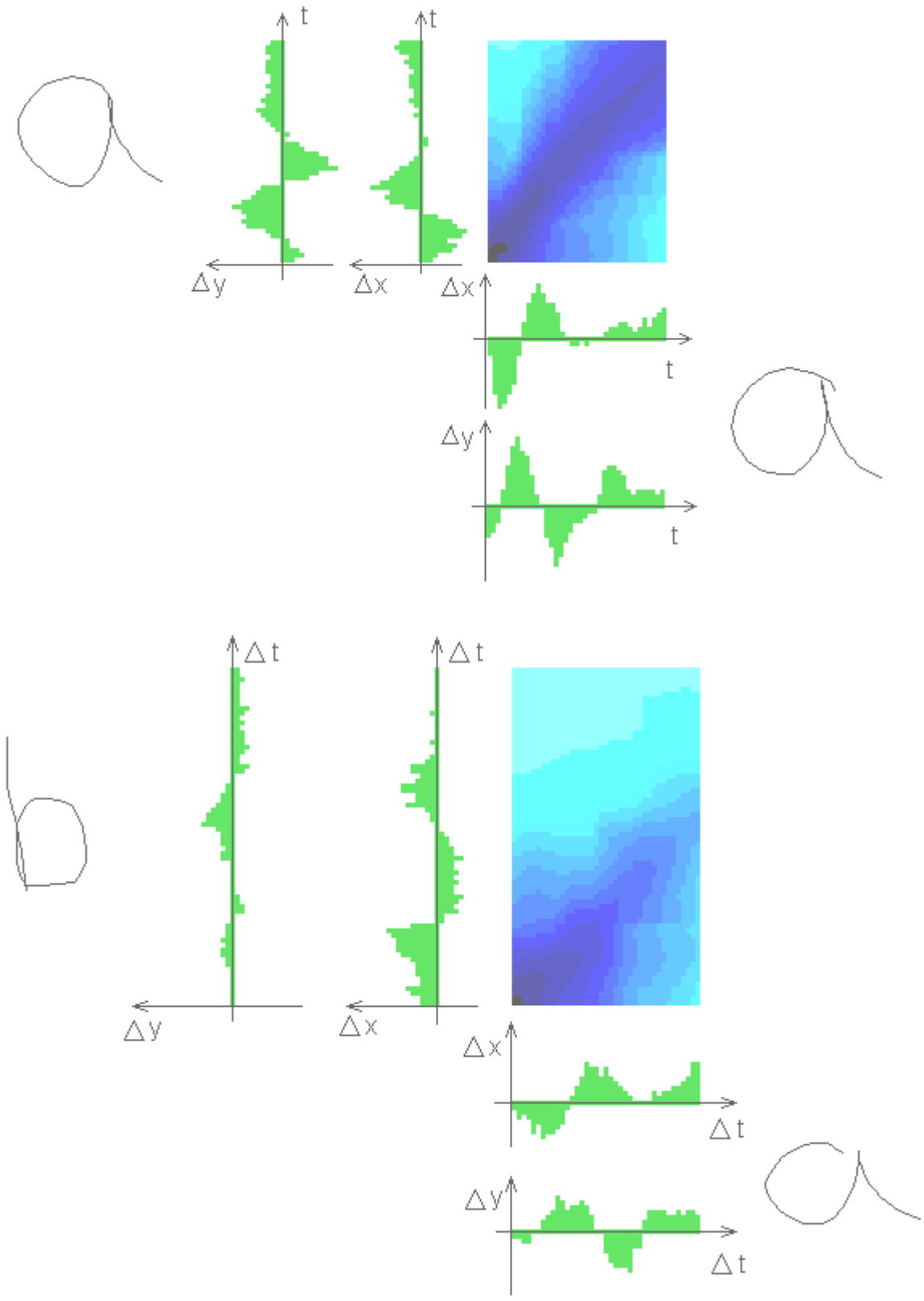
4. ábra: A TELEFON és DIKTAFON szavak különbségének meghatározása DTW-vel

Az 5-dik ábrán azt mutatjuk be, miként működik a DTW algoritmus mozgásminták összehasonlításakor. Mivel a mozgásminták hosszú jelsorozatok (40-70 vektorból állnak), a DTW-táblázatok nagyok és nehezen áttekinthetőek. Ezért a táblázatok celláiban lévő számokat színekkel helyettesítettük: a világosabb árnyalatok nagyobb számokat jelentenek, sötétebbek kisebbeket. Jól látható, hogy két *a* betű különbsége sokkal kisebb, mint egy *a* és egy *b* betűé.

2.3. DTW paramétereinek megválasztása

Már a kezdeti kísérletek során tapasztaltuk, hogy az emberek egyazon írásjelet (mozgásmintát) nem állandó sebességgel, nem mindig ugyanakkora méretben írnak le. Célunk, hogy a vektorsorozatok összehasonlítása csak kevéssé legyen érzékeny a sebességkülönbségekre. Ennek érdekében több lehetőséget vizsgálunk. Eközben végig arra törekszünk, hogy különböző jelek mozgásmintái minél inkább elkülönüljenek egymástól, ugyanazon jeleké viszont minél kevésbé.

Ha két vektor különbségét a 2. 1. fejezetben bemutatott módon a Pitagorasz-tétellel számoljuk, az érzékeny lesz a sebességkülönbségekre. Elsőként tehát ezt a különbségszámító eljárást próbáljuk „megjavítani”. Az eljárás lényegét egy példán szemléltetjük. Tegyük fel, hogy az egyik mozgásminta a következő vektorokból áll: (0,1),(0,1),(0,1),(2,0),(2,0),(2,0) a másik pedig ezekből: (1,15),(0,16),(-1,14),(28,1),(32,-1),(29,0). Az eredeti távolságszámítás szerint a két minta jelentősen különbözik, hiszen a vektorok különbsége nagy. Ugyanakkor látható, hogy mind a két vektorsorozat ugyanazon alakzat lerajzolásakor keletkezik, csak az egyik esetben sokkal, kb. 15-ször kisebb méretben rajzoljuk le azt. Ha a második mintát elosztanánk 15-tel akkor a két minta már nagyon hasonló lenne. Ezt az ötletet fogjuk általánosítani, s normalizálásnak hívni. A kérdés csupán az, hogy mivel osszunk le.



5. ábra: Két *a* betű, illetve egy *a* és egy *b* betű leírásakor keletkező mozgásminta és a két minta összehasonlítása DTW-algoritmussal. Az elmozdulásvektorok *x* és *y* irányú komponenseit az idő függvényében ábrázoltuk. A DTW-táblázatban a világosabb tónus nagyobb számot, a sötétebb kisebb számot jelöl.

Megvizsgáljuk a következő eseteket:

1. nem osztunk semmivel, az eredeti mintákat hasonlítjuk össze
2. a vektorsorozatok elemeit komponensenként a legnagyobb abszolútértékű számmal osztjuk le, és az így kapott mintákat hasonlítjuk össze (azaz az előbbi példában az első sorozat első komponensét 2-vel, a másodikat 1-gyel osztanánk, a második sorozat első komponensét 32-vel, másodikat 16-tal osztanánk)
3. a vektorsorozatok elemeit komponensenként az elmozdulások abszolútértékeinek átlagával osztjuk le, és az így kapott mintákat hasonlítjuk össze (azaz az előbbi példában az első sorozat első komponensét 1-vel, a másodikat 0.5-tel osztanánk, a második sorozat első komponensét 15.16-dal, másodikat 7.84-dal osztanánk)

Ha egy mintát lassabban írunk le, nemcsak a vektorok komponensei lesznek kisebbek, hanem – mivel több ideig tart a művelet – a vektorsorozat is hosszabb lesz. Ezért, annak érdekében, hogy az összehasonlítás ne legyen érzékeny a sebességre, a nyújtás költségét kicsire érdemes állítani.

A következő táblázatban *a* és *b* betűk távolságait látjuk mind a három esetben (1. táblázat). A 2. táblázatban az azonos és különböző betűk átlagos távolságait látjuk a három esetben (egy-egy mozgásminta önmagával való különbségét nem számítottuk be a táblázatbeli számokba).

A különböző táblázatokban különböző mintákat használtunk. Például az 1. táblázatbeli a1-es minta (egy *a* betű) nem azonos a 3. táblázat a1-es mintájával (ez egy „másik” *a* betű). Ugyanazon táblázat különböző szakaszaiban viszont ugyanazon mintákat használtuk. (Tehát például az 1. táblázat felső szakaszában használt a1-es minta azonos az 1. táblázat középső ill. alsó szakaszában használt a1-es mintával.) Ezért tehát csak egyazon táblázaton belüli eredmények hasonlítandók össze egymással.

A 3. táblázat közel azonos sebességgel és mérettel írt mintákon mutatja be a távolságokat különböző normalizálások esetén. A táblázat 10-10 mintát hasonlít össze háromféle normalizálás (semmilyen, legnagyobbal való leosztás, átlaggal való leosztás) mellett.
A táblázatokról látható, hogy ha az elmozdulások közel azonos nagyságúak, akkor keveset javít a normalizálás. A táblázatokról leolvasható, hogy ha az elmozdulások nem azonos nagyságúak (írás sebessége, mérete más, 1-2. táblázat), akkor az átlaggal való normalizálás a leghatékonyabb a vizsgáltak közül. További normalizáló eljárás lehetne még a legnagyobb 10 átlagával való leosztás.

Amikor a vektorok különbségét végpontjaik Pitagorasz-tétellel számolt távolságaként számoltuk, azt a „hibát” igyekeztünk orvosolni, hogy a felismerés érzékeny a minta méretére és rajzolási sebességére. Vektorok különbségét azonban máshogy is számíthatjuk: amint a 2.1. fejezetben bemutattuk, bezárt szögeik alapján is számíthatjuk a hasonlóságukat.

Az 5-8. táblázatok hasonlóan készültek, mint az első négy. Annyi különbség azonban van, hogy ezekben az esetekben a DTW algoritmusban a vektorok különbségét már nem a végpontjaik távolsága jelenti, hanem a bezárt szögük. A mérések alapján látható, hogy ha a bezárt szöget vesszük alapul, akkor felesleges normalizálni, a normalizálás hatása elhanyagolható (a 8. táblázatban látható, hogy a normalizálás az arányon csak 1-2 századot változtat). Ez nem meglepő, mert a normalizálás alapvetően nem változtat a bezárt szögön. (Egy „kicsit” azonban változtat, mivel a normalizálást komponensenként külön végezzük, nem ugyanazzal a számmal osztjuk az *x* és az *y* irányú elmozdulásokat.)

A minták végpontjainak Pitagorasz-tétellel számított különbségekor a legjobb eljárás az volt, amikor a mintákat az átlaggal normalizáltuk. A 9. és 10. táblázatban ezt hasonlítottuk össze azzal, amikor a vektorok különbségét bezárt szögükkel számítottuk és nem használtunk normalizálást.

	a1	a2	a3	a4	a5	b1	b2	b3	b4	b5
a1	0	556	375	250	494	662	598	834	546	577
a2	556	0	669	354	76	445	100	640	175	318
a3	375	669	0	392	601	826	696	1050	633	742
a4	250	354	392	0	345	510	394	710	372	410
a5	494	76	601	345	0	500	140	687	225	378
b1	662	445	826	510	500	0	339	273	290	161
b2	598	100	696	394	140	339	0	571	122	218
b3	834	640	1050	710	687	273	571	0	510	374
b4	546	175	633	372	225	290	122	510	0	171
b5	577	318	742	410	378	161	218	374	171	0
a1	0	120	129	71	65	152	154	138	129	172
a2	120	0	230	57	67	74	74	61	54	88
a3	129	230	0	188	174	266	269	253	245	284
a4	71	57	188	0	36	87	87	74	63	102
a5	65	67	174	36	0	105	101	88	81	117
b1	152	74	266	87	105	0	13	24	36	25
b2	154	74	269	87	101	13	0	28	37	24
b3	138	61	253	74	88	24	28	0	32	38
b4	129	54	245	63	81	36	37	32	0	49
b5	172	88	284	102	117	25	24	38	49	0
a1	0	137	146	93	92	206	215	197	189	221
A2	137	0	259	80	88	107	115	98	97	123
A3	146	259	0	210	206	324	331	316	317	349
A4	93	80	210	0	72	132	142	125	119	148
A5	92	88	206	72	0	151	154	137	136	168
B1	206	107	324	132	151	0	27	35	53	31
B2	215	115	331	142	154	27	0	47	52	35
B3	197	98	316	125	137	35	47	0	63	48
B4	189	97	317	119	136	53	52	63	0	57
B5	221	123	349	148	168	31	35	48	57	0

1. táblázat: *a* és *b* betűk mozgásmintáinak különbségei különböző normalizálások mellett
(fentről lefele: nincs normalizálás, maximummal való normalizálás, átlaggal való normalizálás,
a vektorok különbségét a végpontjaik távolságaként számoljuk Pitagorasz-tétellel, a nyújtás költsége: 5)

	1	2	3
azonos betűk átlagos távolsága (A)	357,1	72,2	91,6
különböző betűk távolsága egymástól (K)	526,72	132,72	184,68
arány (K/A)	1,474	1,838	2,016

2. táblázat: Az 1. táblázatbeli azonos és különböző betűk mintáinak átlagos távolsága különböző normalizálások mellett (balról jobbra: nincs normalizálás, maximummal való normalizálás, átlaggal való normalizálás)

	a1	a2	a3	a4	a5	b1	b2	b3	b4	b5
a1	0	125	182	163	221	352	373	408	569	374
a2	125	0	111	118	175	287	321	344	507	304
a3	182	111	0	72	108	222	246	276	424	238
a4	163	118	72	0	110	212	236	262	417	229
a5	221	175	108	110	0	198	217	238	380	201
b1	352	287	222	212	198	0	73	96	244	106
b2	373	321	246	236	217	73	0	90	198	109
b3	408	344	276	262	238	96	90	0	188	100
b4	569	507	424	417	380	244	198	188	0	252
b5	374	304	238	229	201	106	109	100	252	0
a1	0	61	120	114	139	208	217	216	272	178
a2	61	0	68	68	93	160	171	169	226	130
a3	120	68	0	18	43	108	116	119	170	80
a4	114	68	18	0	42	105	112	114	166	76
a5	139	93	43	42	0	83	93	92	146	53
b1	208	160	108	105	83	0	24	22	80	42
b2	217	171	116	112	93	24	0	16	66	53
b3	216	169	119	114	92	22	16	0	69	51
b4	272	226	170	166	146	80	66	69	0	110
b5	178	130	80	76	53	42	53	51	110	0
a1	0	98	150	144	177	285	289	289	337	255
a2	98	0	87	94	124	225	235	231	278	196
a3	150	87	0	44	64	165	173	171	215	140
a4	144	94	44	0	68	166	171	172	215	140
a5	177	124	64	68	0	139	149	145	191	111
b1	285	225	165	166	139	0	42	39	96	59
b2	289	235	173	171	149	42	0	34	79	70
b3	289	231	171	172	145	39	34	0	81	64
b4	337	278	215	215	191	96	79	81	0	124
b5	255	196	140	140	111	59	70	64	124	0

3. táblázat: közel azonos sebességgel és méretben írt *a* és *b* betűk mozgásmintáinak különbségei különböző normalizálások mellett (fentről lefele: nincs normalizálás, maximummal való normalizálás, átlaggal való normalizálás, a vektorok különbségét a végpontjaik távolságaként számoljuk Pitagorasz-tétellel, a nyújtás költsége: 5)

	1	2	3
azonos betűk átlagos távolsága (A)	142,175	65,075	87,025
különböző betűk távolsága egymástól (K)	313,4	143,2	203,32
arány (K/A)	2,204326	2,2	2,33634

4. táblázat: A 3. táblázatbeli azonos és különböző betűk mintáinak átlagos távolsága különböző normalizálások mellett (balról jobbra: nincs normalizálás, maximummal való normalizálás, átlaggal való normalizálás)

	a1	a2	a3	a4	A5	b1	b2	b3	b4	b5
a1	0	146	49	59	61	444	52	206	370	48
a2	186	0	213	229	160	350	230	98	258	219
a3	38	161	0	27	72	492	25	236	401	17
a4	51	189	22	0	80	494	18	242	399	24
a5	78	128	91	101	0	416	103	185	334	93
b1	651	475	681	697	600	0	691	399	239	684
b2	40	179	20	18	80	465	0	225	380	28
b3	257	115	282	298	234	306	291	0	201	285
b4	459	324	466	482	423	215	476	236	0	469
b5	36	185	17	28	75	468	32	226	382	0
a1	0	146	47	57	59	445	52	205	368	45
a2	183	0	211	225	152	352	227	98	253	213
a3	36	162	0	25	69	513	25	226	417	15
a4	49	173	20	0	79	505	16	240	416	22
a5	75	123	88	98	0	417	100	179	333	88
b1	649	475	681	695	600	0	689	396	229	681
b2	41	172	20	16	78	465	0	224	379	24
b3	259	114	285	299	230	305	293	0	198	285
b4	459	319	468	483	422	205	476	234	0	468
b5	35	173	15	26	70	473	28	226	385	0
a1	0	145	46	57	59	445	52	205	369	46
a2	184	0	211	226	156	351	228	98	256	216
a3	35	161	0	25	70	523	24	240	426	15
a4	49	179	20	0	79	502	16	232	391	23
a5	76	126	89	99	0	416	102	181	334	90
b1	647	472	679	692	596	0	686	393	229	679
b2	41	177	20	16	79	465	0	224	380	25
b3	259	114	285	299	232	304	293	0	198	285
b4	460	322	469	483	423	207	477	234	0	469
b5	35	178	15	27	71	470	29	225	383	0

5. táblázat: *a* és *b* betűk mozgásmintáinak különbségei különböző normalizálások mellett (fentről lefele: nincs normalizálás, maximummal való normalizálás, átlaggal való normalizálás, a vektorok különbségét bezárt szögeik nagyságaként számoljuk, a nyújtás költsége: 5)

	1	2	3
azonos betűk átlagos távolsága (A)	220,975	218,5	218,675
különböző betűk távolsága egymástól (K)	257,52	257,8	257,44
arány (K/A)	1,1653	1,1765	1,177272

6. táblázat: Az 5. táblázatbeli azonos és különböző betűk mintáinak átlagos távolsága különböző normalizálások mellett (balról jobbra: nincs normalizálás, maximummal való normalizálás, átlaggal való normalizálás)

	a1	a2	a3	a4	a5	b1	b2	b3	b4	b5
a1	0	124	114	115	134	114	111	118	120	125
a2	101	0	34	23	30	35	44	41	29	29
a3	93	43	0	24	56	58	42	54	63	66
a4	94	31	22	0	58	47	30	52	51	52
a5	109	26	40	46	0	45	50	42	37	33
b1	94	37	51	42	49	0	24	16	19	29
b2	91	50	40	30	57	28	0	26	44	47
b3	100	43	47	48	45	16	22	0	34	41
b4	99	29	53	44	40	17	38	32	0	24
b5	104	27	54	42	36	26	40	38	22	0
a1	0	123	110	115	132	110	108	116	116	122
a2	101	0	34	23	30	31	41	39	25	29
a3	89	42	0	25	55	50	38	51	56	62
a4	94	31	22	0	57	44	28	50	49	53
a5	107	25	40	45	0	39	49	40	31	30
b1	89	32	44	39	43	0	22	14	17	26
b2	88	46	36	28	56	26	0	25	40	43
b3	94	40	44	46	44	14	22	0	30	36
b4	94	25	47	41	34	15	34	28	0	22
b5	101	27	50	42	31	23	37	33	20	0
a1	0	125	114	114	133	113	110	116	118	123
a2	102	0	34	23	30	34	42	38	27	29
a3	93	43	0	24	56	56	40	52	60	64
a4	93	31	22	0	57	44	27	48	48	51
a5	108	25	40	45	0	42	49	40	33	30
b1	92	35	49	39	46	0	23	15	18	26
b2	88	47	38	27	56	27	0	25	42	44
b3	96	40	45	44	44	15	21	0	31	36
b4	97	27	51	41	36	17	36	29	0	22
b5	102	27	51	41	32	24	37	33	20	0

7. táblázat: Közel azonos sebességgel és méretben írt *a* és *b* betűk mozgásmintáinak különbségei különböző normalizálások mellett (fentről lefele: nincs normalizálás, maximummal való normalizálás, átlaggal való normalizálás, a vektorok különbségét bezárt szögeik nagyságaként számoljuk, a nyújtás költsége: 5)

	1	2	3
azonos betűk átlagos távolsága (A)	47,5	45,675	46,325
különböző betűk távolsága egymástól (K)	56,8	53,36	54,5
arány (K/A)	1,195789	1,168254	1,176471

8. táblázat: Azonos és különböző betűk mintáinak átlagos távolsága különböző normalizálások mellett (balról jobbra: nincs normalizálás, maximummal való normalizálás, átlaggal való normalizálás)

	a1	a2	a3	a4	a5	b1	b2	b3	b4	b5
a1	0	53	59	53	42	112	114	117	124	134
a2	53	0	27	76	53	77	76	84	86	95
a3	59	27	0	78	58	64	64	72	74	82
a4	53	76	78	0	44	133	132	140	145	154
a5	42	53	58	44	0	103	102	114	118	129
b1	112	77	64	133	103	0	24	25	24	35
b2	114	76	64	132	102	24	0	25	29	39
b3	117	84	72	140	114	25	25	0	21	30
b4	124	86	74	145	118	24	29	21	0	31
b5	134	95	82	154	129	35	39	30	31	0
a1	0	62	81	43	24	89	90	98	100	108
a2	48	0	25	74	47	48	53	59	64	72
a3	64	23	0	85	54	50	48	57	64	69
a4	53	97	108	0	59	120	121	126	130	130
a5	24	60	70	49	0	95	99	107	111	111
b1	72	44	48	97	77	0	33	31	23	47
b2	73	46	44	99	77	31	0	34	29	42
b3	79	50	50	104	83	28	32	0	12	27
b4	81	54	59	110	86	19	27	12	0	31
b5	87	57	59	109	86	41	37	23	28	0

9. táblázat: *a* és *b* betűk mozgásmintáinak különbségei különböző vektor különbség számítási módszerek mellett (a végpontok távolsága fent, szögek mérése lent, a nyújtás költsége: 5)

	1	2
azonos betűk átlagos távolsága (A)	41,3	43,425
különböző betűk távolsága egymástól (K)	105,8	81
arány (K/A)	2,561743	1,865285

10. táblázat: Azonos és különböző betűk mintáinak átlagos távolsága különböző vektor távolság számítási módszerek mellett (a végpontok távolsága balra, a szögek mérése jobbra)

A 10. táblázat egyértelműen mutatja, hogy sokkal jobb módszer a két vektor végpontjainak távolságát alapul venni, mint a szöget, hiszen ez a módszer mintegy 35%-kal rosszabb eredményt mutat.

Ekkor azonban még nem vettük figyelembe, hogy a végeredményt befolyásolja a nyújtás költségének mértéke is. A következő részben azt vizsgálom, hogy vajon mekkora mértékben befolyásolja a végeredményt a nyújtás költségének mértéke.

	a1	a2	a3	a4	A5	b1	b2	b3	B4	b5
a1	0	89	69	37	38	126	116	114	122	131
a2	89	0	73	101	107	196	180	181	187	196
a3	69	73	0	74	76	155	142	144	150	163
a4	37	101	74	0	32	108	96	94	103	111
a5	38	107	76	32	0	101	89	89	94	102
b1	126	196	155	108	101	0	38	35	29	28
b2	116	180	142	96	89	38	0	22	22	33
b3	114	181	144	94	89	35	22	0	21	32
b4	122	187	150	103	94	29	22	21	0	18
b5	131	196	163	111	102	28	33	32	18	0
a1	0	144	99	52	58	191	161	159	172	186
a2	144	0	98	171	182	316	280	281	292	306
a3	99	98	0	119	126	250	217	219	230	248
a4	52	171	119	0	37	158	126	124	138	151
a5	58	182	126	37	0	146	114	114	124	137
b1	191	316	250	158	146	0	58	55	44	38
b2	161	280	217	126	114	58	0	22	27	43
b3	159	281	219	124	114	55	22	0	26	42
b4	172	292	230	138	124	44	27	26	0	23
b5	186	306	248	151	137	38	43	42	23	0
a1	0	199	129	67	78	256	206	204	222	241
a2	199	0	123	241	257	436	380	381	397	416
a3	129	123	0	164	176	345	292	294	310	333
a4	67	241	164	0	42	208	156	154	173	191
a5	78	257	176	42	0	191	139	139	154	172
b1	256	436	345	208	191	0	78	75	59	48
b2	206	380	292	156	139	78	0	22	32	53
b3	204	381	294	154	139	75	22	0	31	52
b4	222	397	310	173	154	59	32	31	0	28
b5	241	416	333	191	172	48	53	52	28	0
a1	0	254	159	82	98	321	251	249	272	296
a2	254	0	148	311	332	556	480	481	502	526
a3	159	148	0	209	226	440	367	369	390	418
a4	82	311	209	0	47	258	186	184	208	231
a5	98	332	226	47	0	236	164	164	184	207
b1	321	556	440	258	236	0	98	95	74	58
b2	251	480	367	186	164	98	0	22	37	63
b3	249	481	369	184	164	95	22	0	36	62
b4	272	502	390	208	184	74	37	36	0	33
b5	296	526	418	231	207	58	63	62	33	0
a1	0	309	189	97	118	386	296	294	322	351
a2	309	0	173	381	407	676	580	581	607	636
a3	189	173	0	254	276	535	442	444	470	503
a4	97	381	254	0	52	308	216	214	243	271
a5	118	407	276	52	0	281	189	189	214	242
b1	386	676	535	308	281	0	118	115	89	68
b2	296	580	442	216	189	118	0	22	42	73
b3	294	581	444	214	189	115	22	0	41	72
b4	322	607	470	243	214	89	42	41	0	38
b5	351	636	503	271	242	68	73	72	38	0

11. táblázat: *a* és *b* betűk mozgásmintáinak különbségei különböző nyújtási költségek mellett (a nyújtás költsége fentről lefelé: 5, 10, 15, 20, 25)

	5	10	15	20	25
azonos betűk átlagos távolsága (A)	48,7	73,2	97,7	122,2	146,7
különböző betűk távolsága egymástól (K)	131,6	193,6	255,6	317,6	379,6
arány (K/A)	2,702259	2,644809	2,616172	2,599018	2,587594

12. táblázat: Azonos és különböző betűk mintáinak átlagos távolsága különböző nyújtási költségek mellett (a nyújtási költségek balról jobbra: 5, 10, 15, 20, 25)

A 12. táblázatról leolvasható, hogy a nyújtás költsége habár befolyásolja az eredményt, nem kimondottan nagymértékben. Ahogyan azt fentebb is említettem, ha a vektortok különbségét a végpontjaik távolsága alapján számítjuk, akkor érdemes a nyújtás költségét kisebbre venni, ezzel is lehet csökkenteni a beírás sebességéből és méretéből adódó különbségeket.

Ezek a vizsgálatok nem széleskörűek, mert jeltípusonként csak 5 mintát és csak két jeltípust vizsgáltunk. Az optimális paraméterek kikísérletezése további kutatások tárgyát képezheti.

3. A program megvalósítása

A megvalósított program célja, hogy az egzotikus betűk írásának tanítását támogassa. A programba előre be kell táplálni az egzotikus betűket, mindet ötször. Gyakorláskor a program kiírja a képernyőre, hogy mely betűt kell berajzolni. Ezután felhasználó által rajzolt mintát összehasonlítja mind az öt ahhoz a betűhöz tartozó előre betáplált mintával. A legkisebb különbséget veszi figyelembe. Ha ez egy meghatározott különbség (az öt előre betáplált minta átlagos különbségének 1,5-szerese) alatt van, akkor kiírja, hogy a felhasználó már jól írja be a mintát, a különbség fölött azt írja ki, hogy mennyivel lépte túl a felhasználó a korlátot.

A program egy-egy mintát úgy ír le, hogy két darab 600 elemből álló tömbben eltárolja az egyes vektorokat. Van egy kiegészítő változó is, mely a minta hosszát tartalmazza. A következő részben kifejtésre kerülnek a program egyes részei.

3.1. Eljárások

A programot Pascal nyelven valósítottam meg. A program főbb részeit külön-külön eljárásokba (procedure-kba) szerveztem. A következőkben ezeket írom le.

A 2.2. fejezetben leírt *DTW algoritmust* egy önálló eljárásban valósítottam meg. Mivel a DTW-táblázatból egyszerre mindig csak két oszlop értékei fontosak, ezért memória-gazdálkodási okokból csak ezt a kettőt tároljuk. Az eljárás paramétere a nyújtás költsége.

A *mintabeolvasó eljárás* először a mintára vonatkozó összes változó értékét 0-ra állítja. Ezt követően grafikus mód inicializálására kerül sor 640×480 képponttal és 16 színnel. Ez után inicializáljuk az egeret. Az egeret egyből láthatatlanná tesszük.

Ezt követően egy ciklus következik, amely egy változó értékét növeli egyesével 1 és 600 között. A ciklus elején az egeret a képernyő előre meghatározott helyére (középpontjára) teszi. Ezt követően 1 századmásodpercig vár, és megnézi, hogy mennyit mozdult el az egér. Ezt követően vonal húzása történik. Ez azért jó, mert a felhasználó látja, hogy eddig mit rajzolt. Ha volt elmozdulás, akkor ezt a mozgást, a mintát leíró tömbbe tárolom el. (Az x és y irányú komponenseket külön-külön.)

Ha billentyű-leütés történik akkor kilép a ciklusból és a mintabeolvasó eljárásnak vége. Ha nem, akkor csak akkor van vége az algoritmusnak, ha a ciklus változója eléri a 600-at.

Az eljárás paraméterezhető úgy is, hogy ha adott ideig nem volt elmozdulás, akkor is lépjen ki a ciklusból.

A *mintamentő eljárásnak* az a lényege, hogy a mozgásmintát fájlba menti. Ennek az eljárásnak 3 paramétere van: a mintát tartalmazó változó, a fájl neve, és a megjegyzés. (Minden mozgásmintához egy megjegyzés „kapcsolható".) A megjegyzés bekerül a fájl első sorába, ahonnan a minta-olvasó eljárás kiolvassa. A megjegyzést követő 600 sor tartalmazza a vektorok *x* irányú koordinátáit, a következő 600 sor pedig a vektorok *y* irányú koordinátáit. Ezután a fájl lezárásra kerül, az algoritmusnak vége.

A *mintaolvasó eljárás* egy fájlba mentett mozgásmintát olvas be. Ennek az eljárásnak is három darab paramétere van: a mintát tartalmazó változó (ebbe lesz beolvasva a minta), a fájl neve és a megjegyzést tartalmazó változó (ennek értékét az eljárás fogja beállítani). Az eljárás megnyitja a fájlt, az első sorból kiolvassa a megjegyzést, és beleteszi az erre létrehozott változóba, melyet paraméterként kapott. Ezt követően beolvassa a mintát tartalmazó változóba a vektorok koordinátáit, végül a fájlt lezárja.

A *minta-megjelenítő eljárás* az idő függvényében felrajzolja az elmozdulás-vektor *x* és *y* irányú komponenseit.

A 2. 3. fejezetben leírt normalizálást is külön eljárás végzi.

3.2. Felhasználói leírás

A program indítása a *Start.exe*-vel történik. Ezt követően öt lehetőség közül választhatunk: új minta bevitele, minta törlése, gyakorlás, minták ábrázolása, kilépés. A megfelelő gombra kattintva elérhetjük ezeket a lehetőségeket. Ha új mintát viszünk be, akkor először meg kell adnunk annak nevét, majd ötször egymás után be kell rajzolnunk a mintát. (Két berajzolás között le kell nyomni a billentyűzet egy tetszőleges gombját.) Ha mintát akarunk törölni, akkor csak meg kell adnunk annak nevét. (Ha a név helyére semmit nem írunk, akkor a program kiírja a most meglévő minták nevét.) Ha gyakorolni szeretnénk, akkor arra a gombra kattintva ezt megtehetjük. Ekkor a program a meglévő minták közül véletlenszerűen kiválaszt egyet, kiírja a nevét. Az üzenet olvasása, és egy billentyűleütés után berajzolhatjuk a mintát. A berajzolásnak billentyűleütéssel vethetünk véget. Ekkor a program kiírja eredményünket, ha már jól rajzoljuk a mintát akkor azt, ha nem akkor azt, hogy a két minta különbsége az elfogadható szint hány százaléka (ez természetesen egy 100 fölötti érték lesz, az elfogadható szintet tekintem 100 százaléknak). Ha a kilépés gombra kattintunk, akkor a program leáll.

4. Mozgásminta felismerés további lehetséges alkalmazásairól

A kidolgozott technológia széleskörben alkalmazható, így például az azonosításban. A hagyományos esetben az azonosítás egy „titkon" (jelszó, PIN-kód, stb.) alapul. Ha valaki megszerez egy ilyen „titkot", akkor könnyen hozzá tud férni az adott rendszerhez. Tegyük fel, hogy Béla egy mozgásmintát használ jelszóként. Ha András ismeri Béla mozgás-mintáját (tudja róla, hogy az egy napocska), kisebb eséllyel tud hozzáférni a védett rendszerhez, mert nem tudja épp úgy rajzolni, mint ahogy azt Béla szokta.

Ezek alapján a jelszó alapú azonosítást kiváltó program a következőképpen működhet: először beolvassa minden felhasználó kódjelét. (Egy felhasználónak lehet akár több kódjele is.) Ez után a program indításakor kéri majd a kódjelet, és ha ez bármelyikhez hasonló, akkor a személy beléphet a programba az adott felhasználó kódjával.

Hasonló feladat még az írásfelismerés is. Ekkor a beolvasott mozgásmintát a program összeveti a már tárolt mozgásmintákkal, melyeknél ismert, hogy az adott mozgásminta melyik betű írásakor keletkezett. A program az új mozgásmintákat betűkké alakítja, és a futása végén fájlba írja.

Ekkor azonban természetesen gondolni kell arra, hogy az emberek nem ugyanúgy írnak, tehát a programban kell, hogy legyen lehetőség saját betűkészlet bevitelére, így ténylegesen egyénivé lehet tenni a programot, ugyanazt a mozgásmintát két külön számítógépen akár másfajta betűnek is felismerheti, hiszen nincs nagy különbség az írott *a* betű és az írott *d* betű között.

Különféle szociológiai vizsgálatokat is érdemes kipróbálni a módszerrel. Érdemes például megvizsgálni, hogy mekkora különbség van az idősebbek és a fiatalabbak mozgásmintái között. Az is érdekes lehet, hogy az írás alapján meghatározható-e, hogy valaki hol lakik (nagyváros, kisváros, falu). Vajon mekkora a különbség egy városi és egy falusi ember írása között?

Továbbá pszichológiai vizsgálatokra is alkalmas lehet a módszer: ha valaki ideges, akkor szinte biztos, hogy máshogy ír, mintha nyugodt lenne.

A technológia alkalmazható még írásjelek megfejtésére, melyek elmosódtak, vagy csak részben maradtak fenn. Ez különösen jól alkalmazható, egy régi és elmosódott szöveg betűinek megfejtésére (megállapítására, hogy milyen betű volt eredetileg). Ekkor feltesszük, hogy minden betűből van legalább egy teljesen épen maradt.

5. Összefoglalás

Az írás és az olvasás az emberek egyik legfontosabb ismerete. A betűk írását megtanulni – főleg ha az anyanyelvének betűitől eltérő betűkről van szó – nem mindenkinek könnyű. Az írás tanulása sok nehézséget okozhat, főleg az „egzotikus” nyelvek esetén.

Ezért célul tűztem ki a mozgásminta-felismerés prototípus jellegű megvalósítását és kipróbálását egy egzotikus nyelv, például a japán nyelv betűinek tanításakor. A feladatot egy számítógépes programmal kívánom megoldani, melynek fő részei: minta beolvasása és eltárolása, minta törlése, minták koordinátáinak megjelenítése, és gyakorlási lehetőség.

A írásjelek helyes írását tanító és ezt ellenőrző programot különféle írásjelekkel kipróbáltam.

A dolgozatban bemutatott módszer továbbfejleszthető és sokféle területen alkalmazható lehet, a tanítástól, az azonosításon és a szociológiai vagy pszichológiai vizsgálatokon át egészen az írásfelismerésig.

Megállapítható tehát, hogy az új technológia széleskörűen alkalmazható, olcsó és könnyen kivitelezhető.

6. Irodalomjegyzék

[1] Pirkó József: Turbo Pascal 5.5 Budapest, 1990, kiadó: LSI oktatóközpont
[2] Marton László: bevezetés a Pascal nyelvű programozásba, Novodat Kiadó, 1999
[3] Wolfgang Hadamitzky – Kazár Lajos: KANJI és KANA
A japán írásrendszer kézikönyve és szótára, Aula Kiadó KFT.
[4] bmfnik.hu/iar/2003_2004/haver/project.htm
[5] www.ee.oulu.fi/~jylipekk/projects/dtrend/dtrend-1.0/docs/html/srs/node12.html