Online aláírásfelismerés projekt

Bordé Sándor, Csernai Kornél

2011. december 11.

1. Feladat

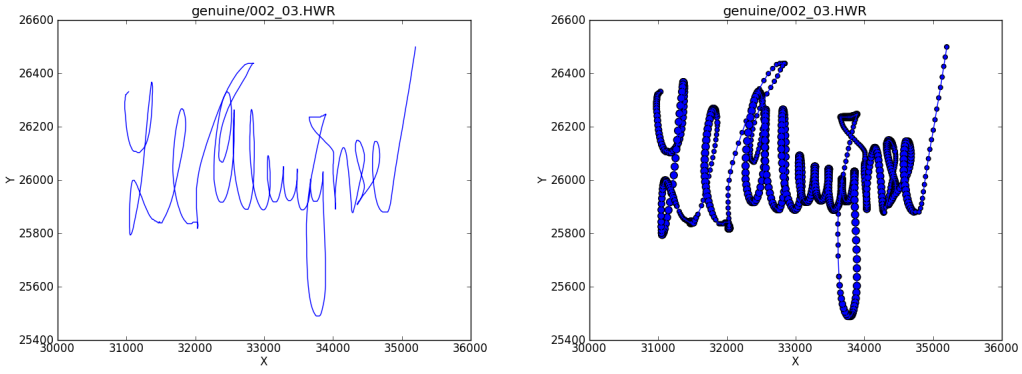
Feladatunk egy online aláírásfelismerési verseny tanítóadataihoz versenyképes tanuló módszereket adni[2]. A verseny leírása a http://forensic.to/webhome/afha/SigComp.html oldalon található. Megoldásunkhoz csupán a "Holland online" adatbázist használhattuk.

2. Az adatbázis

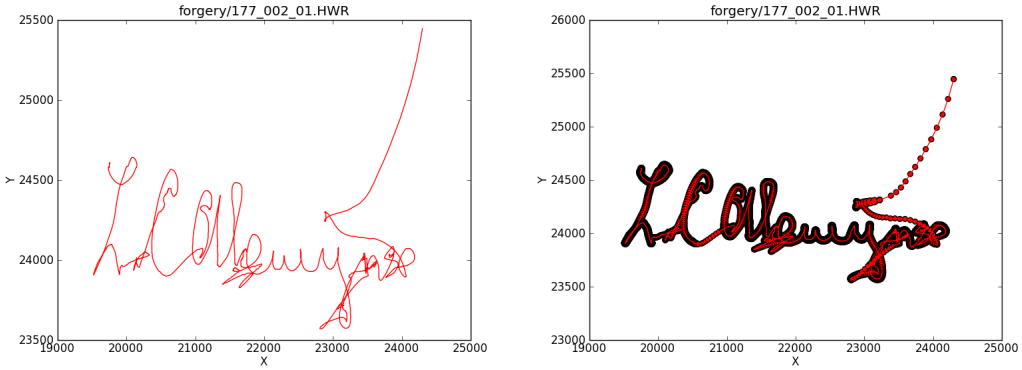
Az adatbázis WACOM Intuos3 A3 Wide USB Pen Tablet típusú eszközzel készült, amely x és y koordinátákat illetve nyomásértékeket rögzít, 200Hz frekvenciával.

Az adatokat letöltöttük, kicsomagoltuk, és áttekintettük. Az adatbázis tartalmaz egy valós aláírás adatbázist (*genuine*) szerzők egy csoportjától. Ezen szerzők aláírásait próbálták gyakorlott hamisítók reprodukálni (*forgery*).

Az aláíroskat vizualizáltuk, ezeket mutatják a következő ábrák.



1. ábra. Egy aláírás nyomásértékekkel és anélkül.



2. ábra. Az 1. ábrához tartozó egy hamisítás nyomásértékekkel és anélkül.

3. Megoldásunk

Megoldásunk többféle módon közelíti meg a problémát. Egyrészt, az adott lokális jellemzőket közvetlenül felhasználva egy *DTW* algoritmuson alapuló tanítót építettünk. Másrészt, a lokális attribútumokat egy-egy aláíráshoz aggregáltuk és így globális attribútumokat kaptunk. A globális attribútumokra *egydimenziós normál eloszlást illesztettünk* illetve *Euklideszi távolságot számoltunk*, és egyfajta anomália detekciót végeztünk. A végső megoldásunkban szavaztatni szerettünk volna, azonban időhiány miatt csupán a lokális tanítót használjuk.

Szóba jött még *Rejtett Markov Modell* és *Neurális Háló* alapú implementáció is, de ezeket időhiány miatt nem próbáltuk ki. Továbbá, szerettünk volna néhány megoldásra koncentrálni.

A megvalósítás *Java, Python, BASH*, és *gnuplot* eszközök segítségével történt. A lokális attribútumok feldolgozása és a globális attribútumok létrehozása Python scriptek segítségével történt. A DTW implementációk teljes egészében Java nyelven íródtak. Az ábrák elkészítéséhez és egyéb segédfeladatok elvégzéséhez a Python, BASH és gnuplot eszközöket használtuk. Munkánkat a git verziókövető rendszerrel kezeltük.

3.1. Lokális jellemzők

Adottak voltak az x, y, z értékek, viszont úgy láttuk, hogy hasznos lenne további, redundáns attribútumokat is kinyerni, ezzel segítve a tanulóalgoritmusokat. A három érték első, második, és harmadik deriváltjait kiszámoltuk a következőképpen:

$$\Delta^1 [f] (x) = x_{i+1} - x_{i-1}$$

$$\Delta^2 [f] (x) = x_{i+1} - 2x_i + x_{i-1}$$

$$\Delta^3 [f] (x) = \frac{x_{i+2} - 2x_{i+1} + 2_{i-1} - x_{i-2}}{2}$$

Tehát $3+3*3 = 12$ lokális attribútummal dolgoztunk. Implicite jelen volt az idő értéke is, azaz, hogy hanyadik adatpontról van szó az aláírásmintán belül.

3.1.1. DTW

Lokális attribútumok szerinti osztályozásra a DTW algoritmust próbáltuk ki. Az interneten elérhető egy FastDTW[1] nevű implementáció, mely Java nyelven, hely- és időtakarékosan számítja ki két idősorozat távolságát.

Ez a megvalósítás a hagyományos DTW algoritmust valósítja meg, változtatható ablakmérettel. Távolság számításhoz a négyzetes *Euklideszi algoritmust* használja, a bemenő adatokat pedig egy rögzített formátumú fájlból olvassa be. Két aláírás távolságának kiszámításakor apróbb módosításoktól eltekintve ezt az implementációt használtam.

3.1.2. Módosítások az eredeti kódon

A tesztelések során azt tapasztaltam, hogy az eredeti, négyzetes Euklideszi távolság túl nagy értékeket ad, és emiatt nagy hibaértékek jöttek ki. Átírva a hagyományos Euklideszi távolságra jelentősen javult az eredmény.

3.1.3. Aláírás osztályok

Az aláírásokat a DTW algoritmus használatához elegendő idősorokként tárolni (ehhez a csomaghoz tartozó `TimeSeries` és `TimeSeriesPoint` osztályokat használtuk), azonban szerettünk volna az aláírások felett magasabb absztrakciót biztosítani, ehhez írtunk egy Signature osztályt.

Ez az osztály adattagként eltárolja az aláírás idősorát és biztosít ehhez egy lekérdező metódust, aminek köszönhetően kompatibilis lesz az algoritmussal. Azonban az eredeti `TimeSeries` osztály hátránya, hogy egy bemeneti fájl teljes egészéből adatsort készít, nem lehet kijelölni, mely értékekre vagyunk kíváncsiak. A `Signature` osztály ezért egy előfeldolgozást is végez: csak a szükséges adatokat olvassa be a fájlból, és ebből hoz létre idősorokat.

Az osztályozónkban végül nem használtuk fel, de a Signature osztály képes az aláírást szakaszokra bontani (szakaszhatár a z koordináta értékének váltása 0-ról vagy 0-ra, tehát tollfelemelésenként bontjuk fel), ebből adódóan a szakasz hosszakat kiszámítani, valamint az egyes aláírásokra toll-fent és toll-lent időket számolni.

További fejlesztésekre gondolva, lehetőség van egy aláírás adott szegmensének adott hosszúságú újramintavételezésére. Újramintavételezéskor a köztes pontok értékét lineáris interpolációval számítja ki az adott időpont összes értékére.

3.1.4. Tanítóhalmaz osztály

A tanító aláírások könnyebb kezelhetősége érdekében ezt is egy magasabb szinten valósítottuk meg.

[1] Elérhető a http://code.google.com/p/fastdtw/ címen.

Az osztály a tanításra szánt aláírásokat várja, valamint az ablakméretet, amellyel számolja a DTW távolságot. Több féle ablakméretet próbáltunk, de eredményben nem hozott javulást, ezért az alapértelmezett 0 értéket választottuk.

A tanítóhalmaz kiszámítja a benne lévő aláírások távolságát (mindegyik aláírás mindegyik másiktól vett távolságát), és ezeket eltárolja egy távolságmátrixban. Kihasználjuk a szimmetriát, ezért csak a főátló alatti elemeket számítottuk ki.

A döntéshez később szükségünk lesz rá, ezért a gyorsabb elérés érdekében eltároljuk a halmaz aláírásai közötti legnagyobb, legkisebb és átlagos távolságot.

Szintén a későbbi bővítésre gondolva, a tanítóhalmaz képes kiszámolni a halmazban lévő aláírások átlagos toll-fent és toll-lent idejét. Ezt is fel lehet használni a döntéskor.

3.1.5. A DTW osztályozó

Magát a döntési folyamatot is egy külön osztályban valósítottuk meg. Az osztályozó megkap egy tanítóhalmazt (amelyben ki vannak számítva a halmaz elemei közötti távolságok), valamint egy küszöbértéket. Ez a küszöbérték jelenti az elfogadási kritériumot. A döntés képlete:

$$T < D \cdot t$$

Ahol T a tesztelendő aláírás távolsága a halmaztól, D a halmaz elemeinek távolsága, t pedig a küszöbérték.

Az, hogy a T és D mely távolságot jelenti pontosan, függ a döntési eljárástól. Három módra van lehetőség a programunkban: minimális, maximális és átlagos távolság. Mindháromra futtattunk teszteket, ezek közül az átlagos nagyon gyenge eredményt adott, így hamar elvetettük. A maximális távolságok jobbnak bizonyultak, de a minimális adott legjobb eredményt a három közül.

3.1.6. A döntéshozó osztály

A program belépési pontja, és a döntési folyamatot vezérlő osztálya a `DTWBasedClassifier` osztály. Két fő működése van: kapott teszt- és tanítóadatokra hoz egy döntést és ezt kiírja az `output.txt` fájlba. Paraméter nélküli indításkor pedig feltételez egy tanítóadatbázist (amit felhasználhattunk a projekt elkészítésekor), és ezekre kiszámítja az EER értéket valamint a hozzá tartozó küszöbértéket.

Globális küszöböt számoltunk, de egyszerűen át lehet írni a programot egyéni küszöbök használatára.

Az EER keresésekor a teszthalmazban szerepelnek a hamisítványok (amiből számítjuk az FAR értéket), valamint a valódi aláírások egy bizonyos része. A valós aláírások kiválasztására keresztvalidációt használtunk. A teszthalmaz aránya 50%, ahogy a kiadott feladatleírásban ajánlották.

Az EER értéket bináris kereséshez hasonló módszerrel határozom meg. A keresés pszeudokódja:

```
begin = 100.0
end = 0.0
threshold = begin - (begin-end)/2
while (threshold < end ) {
    findEER()
    if ( abs(far-frr) < 0.01 )
      break;

    if ( frr < far )
      begin = threshold
    else ( frr > far )
      end = threshold

    threshold = begin - (begin-end)/2
}
```

Mivel nem mindig jött ki pontos egyezés az FAR és FRR értékére, ezért engedélyezzük az 1% eltérést.

3.1.7. Kipróbált esetek

Többféle megközelítéssel próbálgattuk a DTW osztályozót, ezek közül néhány már meg lett említve.

Első lényeges javítást a négyzetesről a hagyományos Euklideszi távolságra való áttérés jelentette. Ekkor 13% körüli EER értékről sikerült 8% körülire lemenni.

Ekkor a távolságokként a maximális távolságokat vettük. Minimális értéket véve tovább sikerült néhány százalékot javítani.

Végül további apróbb javulást értünk el, ha csak a koordináták első deriváltjait használtuk mind a 12 adat helyett.

3.1.8. További javítási ötletek

Menet közben felmerültek további javítási ötletek.

1. érdemes kipróbálni az egyéni küszöbértéket az egyes aláírókhoz
2. attribútumok különféle kombinációit megvizsgálni
3. Maximális távolságot a minimális távolsággal összevetni
4. Improved DTW algoritmust megvalósítani

3.2. Globális jellemzők

Minden egyes tanulópéldán és bejövő tesztpéldán elvégezzük a lokális jellemzők aggregálását. Ez főképp a [3] alapján történt. A felhasznált jellemzők:

Minták száma (1)	Magasság (2)
Szélesség (3)	1 / 3 (5)
Átlagos x sebesség (9)	Pozitív sebességű pontok száma (11)
Toll-lent minták száma (14)	Átlagos nyomás (20)
Maximum nyomás (21)	Maximum nyomás pontja (22)
Nyomás terjedelem (28)	Maximum x sebesség (29)
Átlagos x gyorsulás (30)	Maximum y sebesség (31)
Átlagos y sebesség (32)	Negatív sebességű pontok száma / 14 (34)
Pozitív sebességű pontok száma / 14 (37)	

1. táblázat. A felhasznált globális jellemzők. Zárójelben a [3] cikkben szereplő sorszám.

3.2.1. Gauss-görbék illesztésén alapuló módszer

Ezen módszer esetében a globális attribútumokat tartalmazó tanító vektorokra először normális eloszlást illesztünk:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)},$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)^2.$$

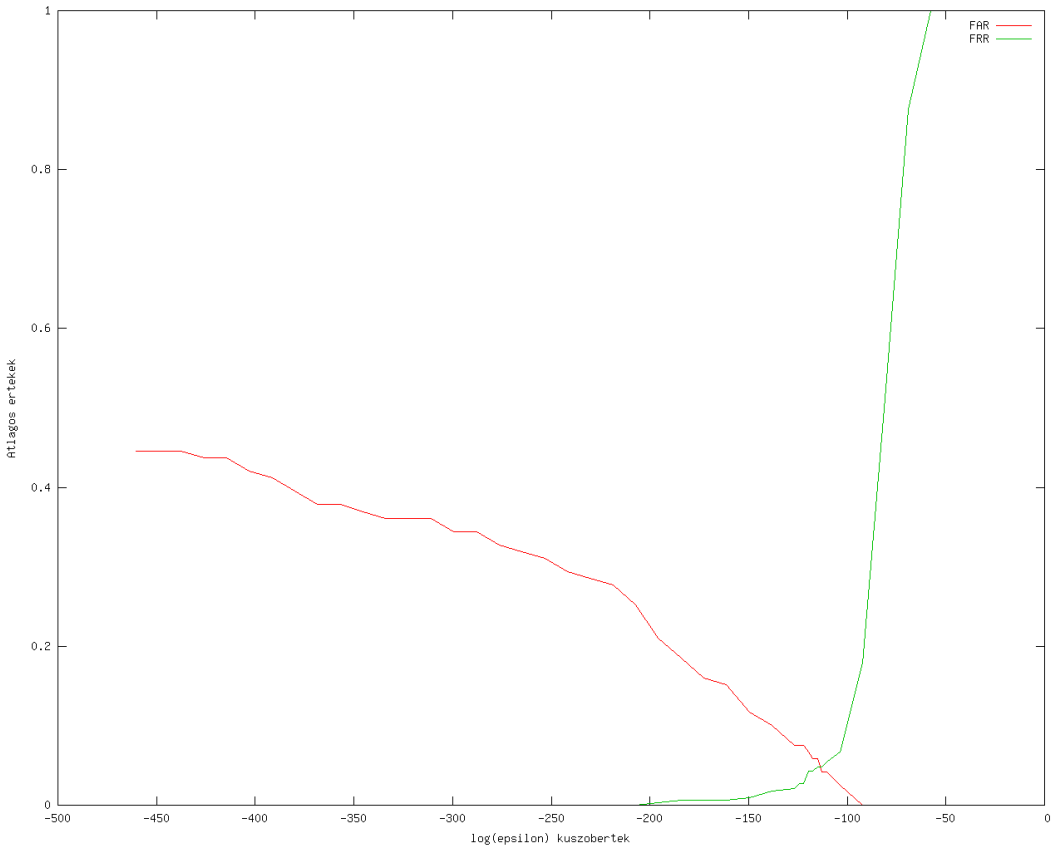
Ezután, ha egy ismeretlen teszt példányt kapunk, akkor elkészítjük annak globális jellemzővektorát (x), majd kiszámítjuk annak valószínűségét, hogy a komponensek külön-külön a megfelelő eloszlásba tartoznak:

$$p(x) = \prod_{j}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}).$$

Választunk egy ϵ küszöbértéket, az x aláírást hamisítványnak tekintjük akkor és csak akkor, ha

$$p(x) < \epsilon.$$

Az ϵ érték kompromisszumot jelent, amely a FAR és FRR értékek közötti átmenetet meghatározza. A következő ábrákat keresztvalidációs kiértékeléssel kaptuk.

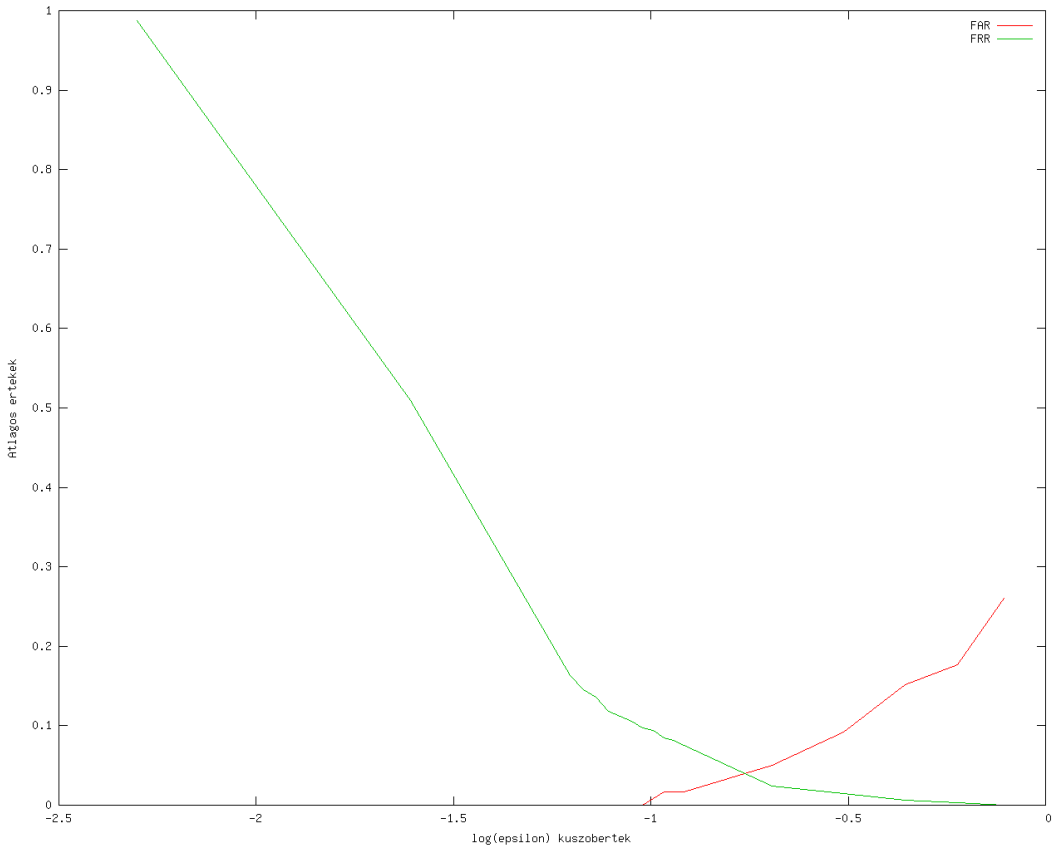


3. ábra. A Gaussian módszerhez tartozó FAR és FRR görbék. Az EER értéke 0.054, amelyet az $e^{-115,214}$ paraméterrel kapunk.

3.2.2. Euklideszi távolság számítása

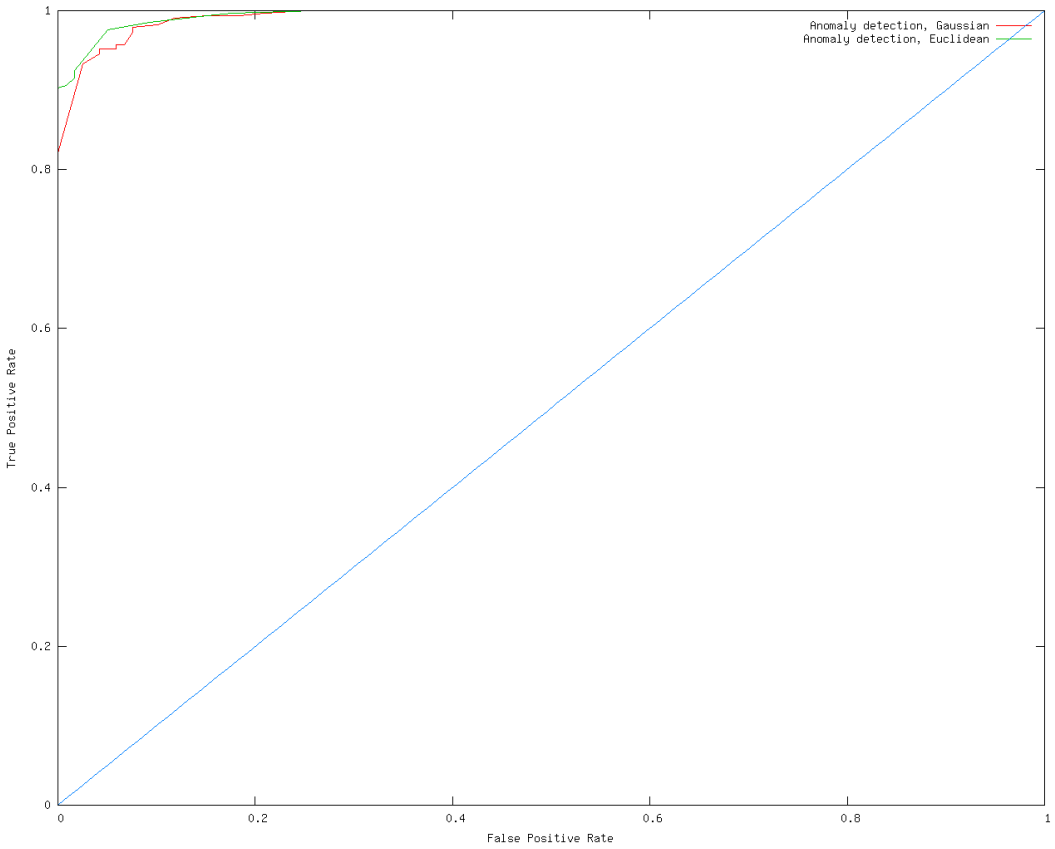
[1]alapján történik, a távolság:

$$\frac{1}{n}\sqrt{\sum_{j=1}^{n}\frac{(x_j-\mu_j)^2}{\sigma^2}}$$



4. ábra. Az Euklideszi távolsághoz tartozó FAR és FRR görbék. Az EER értéke 0.04, amelyet az 0.37 paraméterrel kapunk.

3.3. ROC görbék a három módszerhez



5. ábra. A tanulók ROC görbéi.

4. Tartalom

- `data/`: eredeti adatok
- `data-deriv/`: derivált adatok
- `doc/`: dokumentáció
- `Onsig/`: lokális tanulók
- `scripts/`: segédeszközök, globális tanulók
- `results/`: eredmények adatai

5. Feladatmegosztás

5.1. Bordé Sándor

- DTW
- Dokumentáció

5.2. Csernai Kornél

- Lokális jellemzők feldolgozása
- Globális jellemzők elkészítése
- Gauss-görbék illesztése, illetve Euklideszi távolság módszer
- FAR-, FRR-, és ROC görbék ábrázolása
- Dokumentáció

• Hivatkozások

[1] G. K. Gupta. Abstract the state of the art in on-line handwritten signature verification, 2006.

[2] Marcus Liwicki, Muhammad Imran Malik, C. Elisa van den Heuvel, Xiaohong Chen, Charles Berger, Reinoud Stoel, Michael Blumenstein, and Bryan Found. Signature verification competition for online and offline skilled forgeries (sigcomp2011). In *ICDAR*, pages 1480–1484. IEEE, 2011.

[3] Jonas Richiardi, Hamed Ketabdar, and Andrzej Drygajlo. Local and global feature selection for on-line signature verification. In *In Proc. IAPR 8th International Conference on Document Analysis and Recognition (ICDAR 2005*, pages 625–629, 2005.