Babeş – Bolyai Tudományegyetem
Matematika – Informatika kar
Informatika szak

UJJLENYOMATOK FELISMERÉSE

Ujjlenyomatképek feldolgozása, osztályozás neuronális hálókkal,
azonosítási célú összehasonlítás

Vezetőtanár:
Dr. **Soós Anna** adjunktus

Végzős hallgató:
Csomai András

Kolozsvár, 2003

BEVEZETŐ

A biometrika és az azonosítástechnika az utóbbi évtizedben egyre nagyobb teret hódít. Egyre több biometrikai jegy megbízhatósága és egyedisége bizonyított. A legelterjedtebb és a legnagyobb hagyományokkal rendelkező azonosítási forma azonban továbbra is az ujjlenyomat-vizsgálat. A kriminológiai felhasználási lehetőségeket nem kell ecsetelnünk, azonban legalább ennyire lényeges a polgári, kereskedelmi célú azonosítástechnikai jelentősége (pl. Online Banking, beléptető rendszerek).

A tökéletes ujjlenyomat-felismerő rendszerre azonban még várni kell. Jelen pillanatban egyetlen 100 százalékosan megbízható rendszer sem működik. Természetesen a kutatások igen intenzíven folynak, jelen munkánkban mi is megkíséreltünk hozzájárulni a terület fejlődéséhez.

Célunk a létező ujjlenyomat-felismerési eljárások áttekintése, rendszerezése, valamint egy újabb módszer kidolgozása volt. A dolgozatot is ennek megfelelően strukturáltuk, azaz az első fejezetben magát a biometrikát definiáljuk, majd részletesen ismertetjük az ujjlenyomat tulajdonságait, a felismerésben alapvető szerepet betöltő lokális (minutia) és globális (delta, mag) jegyeket és a klasszikus osztályozási formákat, melyek a katalogizálás megkönnyítését szolgálják. Itt kap helyet a szakirodalomban megjelenő eljárások és módszerek ismertetése. A két alaptechnikát, a binarizáláson alapuló, illetőleg a direkt szürkeárnyalatú felismerési eljárásokat részletekbe menően tárgyaljuk.

A modern ujjlenyomat-felismerést automatizált rendszerek végzik, melyek alapvető funkciói az ujjlenyomatkép minőségi javítása, osztályozás, az azonosításban használt jegyek meghatározása (feature extraction), az adatok tárolása valamint az összehasonlítások megvalósítása. Az általunk javasolt rendszer struktúrája ezzel azonos. Ennek megfelelően az első lépést számunkra is a második fejezetben bemutatott képminőség-javítási eljárás jelenti. Egy korszerű technikát alkalmazunk, apróbb változtatásokat eszközölve, az A.K. Jain és S. Pankanti szerzőpáros által kifejlesztett Gábor szűrős eljárást. A módszer egy olyan adaptív anizotropikus szűrőt használ, mely magvát a lokális redőfrekvencia- és orientáció-paramétereknek megfelelően folyamatosan állítjuk. Az általunk eszközölt változtatás a legjelentősebb paraméter, az orientációkép meghatározására terjed ki.

A második lépés az ujjlenyomatok osztályozása, mely tulajdonképpen a keresések könnyítését és az adatbázis felosztását szolgálja. Ezen folyamatot a harmadik fejezetben

ismertetjük. Az osztályozás alapját az ujjlenyomaton található redők mintázata képezi. A mintázatok ábrázolását a redők irányainak függvényében végezzük. Az osztályozást egy általunk épített, a hiba hátraterjedésének módszerével tanított többrétegű perceptron valósítja meg.

Az ujjlenyomatok összehasonlítása, mint ez a negyedik fejezetből kiderül, a lokális jegyek azaz minutiák alapján történik. Az összehasonlítás legnagyobb problémáját a rotáció- és transzláció- invariancia képezi. A jelenleg alkalmazott rendszerek túlnyomó többsége a globális jegyek alapján igyekszik esztimálni az említett paramétereket, és ennek függvényében korrigálja a képet. Ez azonban a rendszerek megbízhatóságának csökkenéséhez vezet. Ezzel szemben mi javasoltunk egy alternatív módszert, mely nem tesz kísérletet a rotáció és transzláció mértékének meghatározására, hanem egy olyan reprezentálási formát alkalmaz, mely ezen paraméterekre nézve invariáns. Az említett ábrázolási mód, nem más, mint egy szimbólumsorozat, mely alapját a különböző minutiákat összekötő szakaszok hossza és az ezek által bezárt szög képezi. Az így kapott szimbólumsorozatok összehasonlítása azonban komputacionális szempontból igen költséges, ezért kifejlesztettünk egy gyorsított összehasonlító algoritmust, melyet szintén a negyedik fejezetben ismertetünk.

A dolgozat előbb említett fejezeteiben csupán az eljárások elméleti hátterét tárgyaljuk, a gyakorlati megvalósításokat a programozói dokumentációban taglaljuk. Ennélfogva az ötödik fejezet szerves része a dolgozatnak. A szűrési technikák megvalósítása, valamint a többrétegű perceptron tanítási algoritmusának ismertetése mellett a már említett gyorsított összehasonlító algoritmust is részletezzük. Ugyancsak ebben a fejezetben kapott helyet a javasolt továbbfejlesztési módok és lehetőségek ismertetése.

Összegezve az elmondottakat, sikerült létrehoznunk egy rotáció és transzláció invariáns, rendkívül rossz minőségű ujjlenyomatokkal is dolgozni képes, automatizált ujjlenyomat-felismerő rendszert.

1. BIOMETRIKA ÉS AZ UJJLENYOMAT-FELISMERÉS

1.1 BIOMETRIKA

A biometrika az egyén bizonyos fiziológiai illetve pszichológiai jellemzőinek mérését jelenti. Ez főként azonosítási célokat szolgál, ezért a biometrika lassan összeolvad az azonosítástechnika fogalmával. A mérésekben felhasznált jegyek skálája igen széles: antropológiai jellemzők (csontozat, testfelépítés, koponyaforma, stb.), írisz, testillat, DNS, arc, hang, fülcimpa, talp, gépelési dinamika, aláírás, mozgás-analízis, stb. [4].

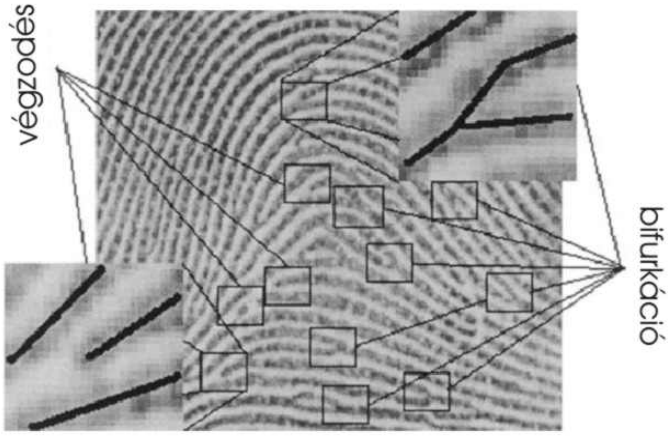
Egy biometrikai azonosító rendszer több funkcióval rendelkezik, legfontosabbak a jellemzők mérése valamilyen szenzor segítségével, a speciális jellemzők kivonása (ezek alapján dönthető el két minta azonossága), és összehasonlítás (két különböző mintáról eldönti, azonosak e). A biometrikai azonosítási rendszerek egyre nagyobb teret hódítanak, polgári célú alkalmazásuk egyre nagyobb (beléptető-rendszerek, banki azonosítási rendszerek, stb.).

Az ujjlenyomat a leginkább elterjedt, és a legszélesebb körben elfogadott biometrikai jegy. Világszerte minden bűnüldözési és ítélethozó szerv elfogadja az ujjlenyomatot bizonyítékként, a polgári célú felhasználtsága is az ujjlenyomatnak a legnagyobb.
Több érv szól e széleskörű felhasználás mellett, többek között az egyediség, az érzékelés egyszerűsége, időbeni állandóság.

1.2 TÖRTÉNELMI ÁTTEKINTÉS

Az ujjlenyomatok azonosítás céljából történő felhasználása meglehetősen nagy múltra tekint vissza. Az ókori Babilóniában agyagtáblákon találtak hitelesítő lenyomatokat, Kínában pecséteken, Perzsiában hivatalos papírokon.

A modern ujjlenyomat-vizsgálat (*dactyloscopia*) alapjait Sir Francis Galton helyezte le „Fingerprints” című munkájában 1892-ben. Ő volt az első, aki felismerte az ujjlenyomatok egyediségét (még az egypetéjű ikrek ujjlenyomatai is különbözőek) valamint időbeni állandóságát (az ujjlenyomat mintázata még a születés előtt kialakul és a baleseteket leszámítva az egyén haláláig nem változik). Az ujjlenyomatok azonosítására ő vezette be a *minutiák* fogalmát [1,2], ezek az úgynevezett *lokális jellemzők*, melyek tulajdonképpen a mintázaton található jegyek, úgymint a végződés és a bifurkáció.



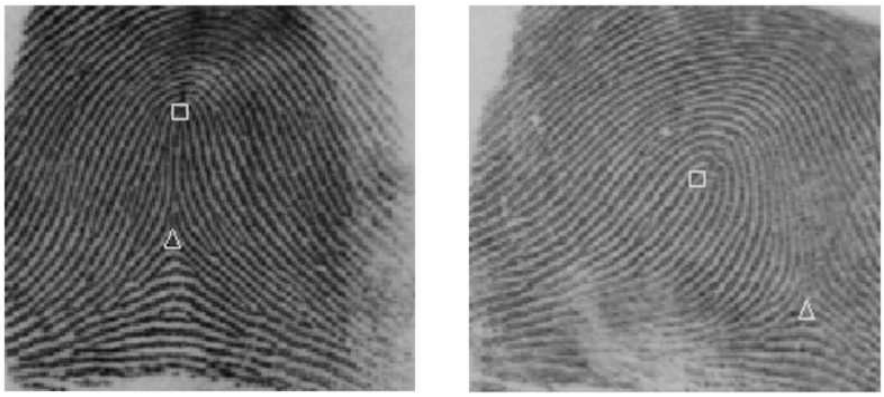
1. ábra: Minutiák: végződés és bifurkáció

Ő volt az első ujjlenyomat-osztályozási rendszer megalkotója is.

1901-ben Angliában és Walesben bevezették az ujjlenyomatok azonosításának bűnüldözési célú alkalmazását. Az alkalmazott osztályozási rendszer készítője Sir Edward Richard Henry, aki Galton módszerére építette saját rendszerét. Az angol nyelvű államokban nagyrészt még mindig ezt a rendszert használják.

1.3 STRUKTURÁLIS VAGY GLOBÁLIS JEGYEK

Ezek a jegyek első ránézésre megállapíthatóak, azonosításban nem, csak az osztályozásban van szerepük [1,2]. Ide tartozik az ujjlenyomat redőinek mintázata (később kifejtve), valamint a mag (illetőleg közép) és delta pont. A magpont, nagyjából az ujjlenyomat közepén található, a redővonalak ebben a pontban megközelítőleg 180 fokos kanyart írnak le. A delta pont jellegzetessége, hogy a redővonalak Y betűhöz hasonló hármas kereszteződésben találkoznak.



2. ábra: Delta és magpont az ujjlenyomaton

Ide tartozik még a redőszám, ami nem más mint a delta és magpontokat összekötő képzeletbeli szakaszt metsző redők száma. Sok esetben ez nem egyértelműen meghatározható. A meghatározást nehezítő tényezők lehetnek a mag illetve deltapont hiánya esetleg ezek többszörös jelenléte. Még az ideális esetben (egy mag és egy deltapont) sem mindig egyértelmű a redőszám, ezért általában $^{+}$/-2 vonalas eltérést engedélyeznek.

1.4 OSZTÁLYOZÁS

Az osztályozás célja a rendszerezés és a visszakeresések megkönnyítése és gyorsítása. A Henry Osztályozási Módszer (Henry Classification System) [3] alapját négy csoport képezi, ezek az ív, csigavonal, jobb és balhurok:

3. ábra: A legfontosabb ujjlenyomat típusok: ív, csigavonal, hurok

A Henry Osztályozási Rendszer természetesen ennél sokkal bonyolultabb, ennek ismertetésére most nem térünk ki.

Egy kor- és egyszerűbb módszer az NCIC osztályozás. Ez a módszer is a strukturális jegyeket alkalmazza, minden egyénhez hozzárendelnek egy húszjegyű karaktersorozatot, minden ujjat két karakter szimbolizál, ami a típus kódja illetőleg hurok vagy csigavonal esetén a redőszám. Más típusok esetében ez azért nem releváns, mert egyrészt hiányozhat a delta illetőleg magpont, vagy akár kettő is lehet belőlük.

1.5 UJJLENYOMATOK ÉRZÉKELÉSE

Alapvetően két esetről beszélhetünk, az egyik az amikor a vizsgált alanytól veszünk mintát, a második az úgynevezett latens ujjlenyomatok érzékelése. A *latens ujjlenyomatok* a különböző tárgyak érintése folytán a tárgy felületén maradó, szabad szemmel nem feltétlenül érzékelhető nyomatok, melyeket különböző módszerekkel tesznek láthatóvá. Ha egyenesen az alanytól veszünk mintát, két alapvető lehetőségünk van, az egyik a *klasszikus tintalenyomat* (az ujjat pecsétpárnára majd papírra nyomjuk) másik pedig a *„live-scan”* (egy speciális

„optical frustrated total internal reflection” szkenner segítségével). A tintalenyomat nagy hiányossága, hogy az ujjlenyomat egyes részei elmosódnak, a redők közé befolyt tinta is korrumpálhatja a végeredményt, álredőket képezve.

1.6 AUTOMATIZÁLT FELISMERŐ RENDSZEREK.

A különböző osztályozási eljárások egyszerűbbé teszik ugyan az ujjlenyomat visszakeresését, de egy nagyobb adatbázis esetében (pl a F.B.I <<Federal Bureau of Investigation>> adatbázisa 7 millió lenyomatot tartalmaz), ez még nem elég. Szükség volt a keresések automatizálására.

Egy automatizált ujjlenyomat-felismerő rendszer (AFIS- Automated Fingerprint Identification System) legfontosabb feladatai közé tartozik az ujjlenyomat típusának meghatározása, tárolása és beazonosítása [4,5]. Lehet polgári vagy bűnüldözési jellegű. Futószalagként is felfogható. Első lépés az ujjlenyomat bevitele vagy érzékelése (lásd később), majd a kép *minőségi javítása* (a beviteli eszközök hiányosságai által okozott torzulások kiküszöbölése), a globális jellemzők megállapítása és *osztályozás*, a lokális jellemzők (minutiák) lokalizálása, majd a feladatnak megfelelően az adatbázishoz csatolás illetőleg *összehasonlítás* a már létező bemenetekkel.

Az ujjlenyomatok tárolására két alapvető módszer létezik, egyik a teljes értékű, melynek során a teljes felvételt tárolják, másik pedig az információvesztéses, amikor csupán a lokális jellemzőket illetőleg ezek kódolt formáját tárolják. Az információvesztéses tárolási módszer esetében a lenyomat rekonstrukciója teljességgel lehetetlen [6].

Képminőség javítás és minutia lokalizálás.

A *képminőség javítás* az egyik legmeghatározóbb lépés, hiszen a bemeneti adatok megbízhatósága az egyik legbefolyásolóbb tényező. Alapvetően két módszert különböztethetünk meg: a ***binarizáció-véknyítást*** valamint a ***direkt szürkeárnyalatú*** (direct grayscale) javítást.

A ***binarizáció-véknytás*** [8, 9] módszer is több lépésből áll. Rendszerint az első lépés valamilyen *szűrő* alkalmazása, a szennyeződések (pl. só-bors) eltávolítása végett. Ez lehet egy egyszerű Gauss, vagy Wiener. Második lépésben kerül sor az adaptív *hisztogramm kiegyenlítésre*, mely célja a kontrasztnövelés illetőleg a megvilágítás változékonyságának megszűntetése. A kiegyenlítést általában 11x11 ablakokban lokálisan végzik. A kontrasztnövelésnek köszönhetően egyes elmosódott képrészletek hangsúlyosabbá, könnyebben érzékelhetőkké válnak.

Megjegyezzük, hogy az első és második lépés sorrendje az igényeknek megfelelően felcserélhető.

A Gauss szűrés nem más, mint egy diszkrét Gauss mag konvolúciója az eredeti képpel. A diszkrét Gauss mag képzését a 2.5 fejezetben tárgyaljuk. Mit is jelent tulajdonkeppen a konvolúció. Legyen egy szürkeárnyalatú kép az *I(m,n)* mátrix, melynek egyes elemei a pixelek intenzitását jelentik. A diszkrét Gauss mag egy négyzetes mátrix, jelölje ezt $G(n_g,n_g)$, ahol n_g a Gauss mag mérete. Legyen t_g a mátrix elemeinek összege, azaz:

$$t_g = \sum_{i=1}^{n_g} \sum_{j=1}^{n_g} G(i,j) \tag{1.1}$$

A konvolúció során az *I* kép minden egyes pixelének értékét megváltoztatjuk, a környezetét figyelembe véve:

$$I(i,j) = \frac{1}{t_g} \sum_{k=-n_g/2}^{n_g/2} \sum_{l=-n_g/2}^{n_g/2} I(i+k, j+l) G(k+(n_g/2), l+(n_g/2)) \tag{1.2}$$

A Wiener szűrő, a Gauss – hoz hasonlóan a környezetet figyelembe véve származtatja a vizsgált pixel új intenzitásértékét. Egy adott *I(i,j)* pixel környezetét alkotó *n* x *n* ablakban (az ablak középpontja *i,j*) kiszámítjuk az intenzitásértékek szórását és várható értékét, legyen ez σ^2 és μ. A szemét feltételezett szórása legyen υ^2. Ekkor a szűrés eredményeképp kapott *W* kép a következőképpen számítható:

$$W(i,j) = \mu + \frac{\sigma^2 - \upsilon^2}{\sigma^2} \left(I(i,j) - \mu \right) \tag{1.3}$$

Az adaptív hisztogramm kiegyenlítés, ugyancsak a környezetet figyelembe véve határozza meg az új intenzitásértékeket, de az előbbi szűrőktől eltérő módon, nem rendelünk minden egyes pixelhez egy ablakot, hanem a képet indításkor felosztjuk *n* x *n* méretű ablakokra, és a változtatásokat ezen ablakok szintjén, lokálisan végezzük. A művelet célja, hogy az egyes pixelintenzitások eloszlása minden ablakban egyenletes legyen. Az intenzitások tulajdonképpen *0* és *255* között vehetnek fel értékeket. Legyen *k* egy intenzitásszint, $0 \le k \le 255$, a kiválasztott ablakban található *k* intenzitású pixelek számát jelölje n_k , az ablak pixeleinek száma n^2 , ekkor egy *k* intenzitáshoz hozzárendelt s_k új intenzitásérték :

$$s_k = \sum_{j=1}^{k} \frac{n_j}{n} \tag{1.4}$$

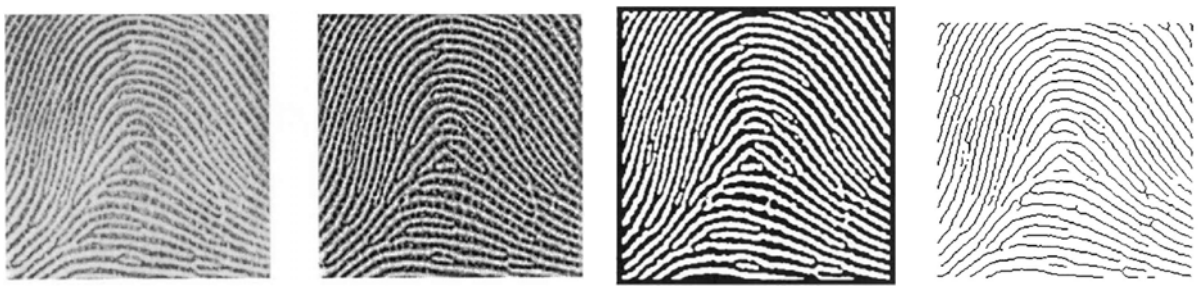
Tehát minden *k* intenzitású pixel intenzitását s_k – ra cseréljük.

A harmadik lépés lenne a *binarizáció*, melyet szintén adott méretű ablakokban lokálisan végzünk, a képlet egyszerű, a kiválasztott régióban megkeressük az intenzitások várható értékét, minden ennél nagyobb intenzitású képpont fehér, minden ennél alacsonyabb intenzitású pedig fekete lesz. Tehát legyen *N(w,w)* egy *w* x *w* ablak az *I(m,n)* szürkeárnyalatú képben. A *0* intenzitás a fekete, a *255* a fehér megfelelője. Legyen az *N* ablakban az intenzitások várhatóértéke μ. Ekkor a *B* bináris kép, az *I*-hez hasonlóan ablakokból épül fel, egy adott *N* –nek megfelelő $B_n(w,w)$ ablak pixeleinek kiszámítása:

$$B(i,j) = \begin{cases} 255, & \text{ha } N(i,j) > \mu, \\ 0, & \text{ha } N(i,j) \le \mu, \end{cases} \tag{1.4}$$

Elméletileg a binarizáció után kapott képen minden redőhöz tartozó képpont fekete és minden völgyhöz tartozó képpont fehér.

Következik a *véknyítás*. Ennek célja, hogy a több pixel vastagságú redővonalakat redukálja egy 1 pixel vastagságú vonallá. Több elfogadható eredményt produkáló véknyító algoritmus létezik, de többségük igen lassú és torzít.



4. ábra: A feldolgozás fő lépései: eredeti, hisztogramm-kiegyenlített, binarizált és véknyított kép

A binarizáció és véknyítás során igen sok torzulás jelentkezik az ujjlenyomatképen, legjellegzetesebb a *híd* és a *szakadás*. Ezek javítására egy ún. *morfologikus szűrőt* alkalmaznak. Híd minden olyan rövid redővonal mely összekapcsol két redővonalat és közel merőleges a domináns redőirányra. A hidak maximális hosszúságát empirikusan 10 pixelben állapították meg. A szakadás nem más, mint két végpont, melyek egy adott (empirikusan 15 pixelben megállapított) átmérőjű körön belül vannak, és összekötésük egy, a domináns

redőiránnyal párhuzamos szakaszt eredményez. A tulajdonképpeni szűrés az így azonosított hidak törlése és szakadások kitöltése.

Természetesen ezen hibák detektálásában egy megfelelően tanított neuronháló is rendkívül jó eredményeket mutathat. Erre a célra több neuronális háló modellt is használnak, a legegyszerűbb MLP azaz többrétegű perceptron modell (lásd 3.1 fejezet). A tanításhoz szükséges adatbázist valódi és hamis minutiákat ábrázoló képrészletekből építik fel. A minutia környezetét is tartalmazó képrészletek lehetnek egyszerű négyzetablakok, de használnak kör és ellipszis alakú kivágásokat is (az ellipszis a leghatékonyabb). A minutiák valódiságát humán szakértő segítségével döntik el, az adatbázis minden elemét ennek megfelelően címkézik. A tanításhoz körülbelül 1000 – 1500 ily módon alkotott képrészletre van szükség. Megjegyezzük, hogy hasonló módszerrel megkísérelték a minutiák fölfedezését a még javítatlan szürkeárnyalatú kép esetében, de ezek a megközelítések még nem hoztak igazán bíztató eredményeket (Maio-Maltoni).

5. ábra: A leggyakoribb hibák a véknyított képen: szakadás és híd

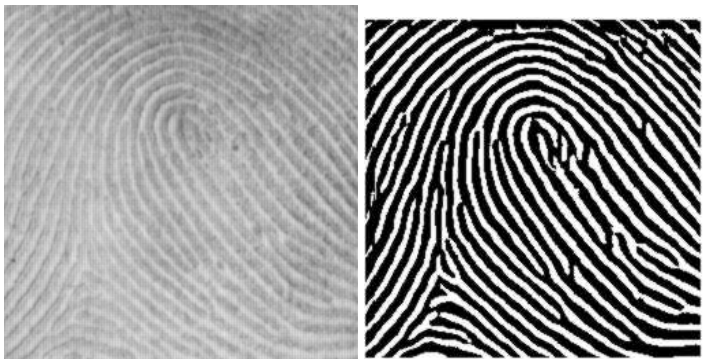
A *minutiák meghatározása* a javított, véknyított kép alapján történik, minutia minden olyan redőpont, melynek van legalább három, egymással nem szomszédos redőpont szomszédja (bifurkáció) illetőleg melynek csupán egy redőpont szomszédja van (végződés).

A fent említett módszernek több hiányossága is van. Elsőként említeném a lassúságot. Mint látható, a folyamat sok lépésből áll, ezek közül a véknyítás fokozottan időigényes. Ugyancsak a véknyítás fogyatékosságainak egyenes következményei a minutiapontok helyeinek torzulása, a nem sikeresen detektált szakadások és hidak okozta hamis minutiák nagyszámú jelenléte. Végkövetkeztetésként leszögezhetjük, hogy a binarizáció-véknyítás technika csupán jó minőségű ujjlenyomatképek biztonságos feldolgozására alkalmas.

A ***direkt szürkeárnyalatú*** feldolgozási módszer [8, 9, 12, 13, 14] szintén egy többlépéses folyamat. Első lépés a kép normalizálása. Második az orientációkép meghatározása. Az orientációkép pontjai tulajdonképpen a lokális domináns redőirányt jelentik. A harmadik a frekvenciakép, melynek pontjai a lokális domináns redőközi távolságot jelenti. Ezek kiszámítására is több technika létezik, ezekre a következő fejezetben kitérünk. A

tulajdonképpeni szűrés csak ezután következik, erre a célra Gábor (lásd 3. fejezet) vagy más anizotropikus szűrőket alkalmaznak, melyek magvát a már kiszámított frekvencia és irányparamétereknek megfelelően alakítják.

Az eljárás előnye, a rendkívül nagy tolerancia a rossz képminőségre nézve. Hátrányként megemlíthetjük a lassúságot. A tulajdonképpeni szűrésnek a legnagyobb a számítási igénye, ez teljesítményvesztéssel némileg kiküszöbölhető ha előre legenerálnak egy a Gábor szűrő értékeit megközelítő szűrőkből álló halmazt. Ez a próbálkozás csak akkor gyorsabb, ha relatív kisszámú szögre generálják le a szűrőket, ellenben pont ez okozza minőségbeli hátrányát.



6. ábra:A Gábor szűrő eredménye: eredeti és feldolgozott kép

Létezik egy gyors és meglehetősen megbízható módszer, a Maio és Maltoni által kidolgozott *redőkövetéses minutia-meghatározás*. Az eljárás lényege, hogy a szürkeárnyalatú képen megkeresik a redőket, majd ezeket követve megkeresik a minutiákat. Első lépésben meghatározzák az ujjlenyomat orientációképét, majd azon egyszerű kijelentésből kiindulva, melyszerint a redő azon pontok sorozata, melyek a redőirány mentén lokális maximumok, feltérképezik a redőket. Az így kapott pontokat összekötve megkapjuk a redők poligoniális közelítését. Ezek után a minutiák meghatározása triviális. Megjegyezzük, hogy a módszer azért nem ennyire egyszerű, és a kép minősége nagyon nagy mértékben befolyásolja az algoritmus eredményességét.

Osztályozás

Mint már korábban említettük, az osztályozás célja az adatbázisban való keresés egyszerűsítése. A többféle létező osztályozási rendszer közül, a leggyakrabban használt a Henry-féle csoportosítás (csigavonal, ív, sátor-ív, bal és jobbhurok). A különböző

alkalmazások igen nagy hibaszázalékot mutattak a két ívtípus megkülönböztetésénél, ennélfogva a legtöbb esetben ezeket egy csoportnak tekintik.

Több szempontot illetve jellemzőt is figyelembe vehetünk az osztályozásnál. Ezek a redőiránykép, a delta és magpont, szimmetriatengely.

A legegyszerűbb osztályozási rendszerek alapvetően a delta és magpont valamint a szimmetriatengely információkat veszik figyelembe. Az osztályozási séma relatív egyszerű: két magpont esetén egyértelmű a csigavonal, delta és magpont hiányában egyértelmű az ív, egy delta és egy magpont hurkot jelent, az irány eldöntésében segít a szimmetriatengely. A módszer nehézsége és fogyatékossága a delta és magpont helyes meghatározásában áll. Azokban az esetekben, ha az ujjlenyomatkép nem teljes, és valamely jellemző pont hiányzik a képről, egyértelmű a hibás osztályozás. Egy megbízhatóbb alternatíva az orientációkép elemzése. Erre a célra a legmegfelelőbb eszköz egy neuronális háló. Az optimális megoldás a két módszer ötvözete.

Maio és Maltoni kidolgozott egy alternatív megoldást [10], mely tulajdonképpen egyet sem használ a fent említett jellemzőkből. Módszerük lényege, hogy bizonyos redővonal-jellemzőket figyelembe véve, régiókra osztják az ujjlenyomatképet, a régióknak megfelelően felépítenek egy súlyozott gráfot. Empirikus módszerekkel megalkották a különböző osztályok ősgráfmintáit, majd egy inegzakt gráfillesztési algoritmussal vizsgálják az osztályozandó ujjlenyomat valamint az ősmintagráfok hasonlóságát, és ez alapján történik a csoportosítás.

Összehasonlítás

Az *összehasonlítás* célja két ujjlenyomat (bemeneti és adatbázisban tárolt) esetén annak eldöntése, hogy ugyanattól az egyéntől származnak e vagy sem. Tulajdonképpen meg kell határoznunk egy olyan teret, melyben az ujjlenyomatok különbözősége mérhető.
Alapvető követelmény, hogy a rendszer rotáció és transzláció invariáns legyen (a méretinvariancia elhanyagolható). Más komoly problémák is felmerülnek az összehasonlítás folyamán, ezek a hamis minutiák, a valódi minutiák hiánya, valamint a bőr elaszticitása okozta torzulás. Hamis minutiáknak nevezzük az olyan jegyeket, melyek a bőrön található szennyeződések vagy vágások miatt jelentkeznek. A szennyeződések a redők közé kerülve, két hamis elágazást, a vágások pedig minden metszett redő esetében két végződést eredményeznek.

Ha ujjlenyomatot veszünk, tulajdonképpen az ujj háromdimenziós felületét két dimenzióba vetítjük. A bőr elaszticitásának köszönhetően, a lenyomat különböző mértékű nyomás esetén különböző mértékben torzul. Eredményképp egyes valós minutiák nem a nekik

megfelelő pozícióban jelentkeznek. Az előbb említett hibák kiküszöbölésére is több módszer létezik. A hamis-valódi minutia kérdés eldöntésére az előző alfejezetben már taglalt neuronhálós megközelítés a legelfogadottabb módszer. A bőrelaszticitás kiküszöbölésére egy lehetséges módszer a Kovács-Vajna Miklós Zsolt [7] által javasolt háromszögeléses eljárás, de bármely módszer átalakítható, ha az összehasonlításnál egy bizonyos toleranciaszintet engedélyezve, a minutia-koordinátákat intervallumként kezeljük.

Léteznek olyan rendszerek is, melyek nem használják a lokális jellemzőket, ezek eredményei még nem biztatóak, megbízhatóságuk nagyban függ a bemeneti képek minőségétől, ezért csupán a „klasszikus” megközelítést részletezzük.

A legtöbb *ujjlenyomat-felismerő rendszer* a következő háromlépéses sémára épül:

1. Egy referenciapont valamint a rotáció és transzláció mértékének megállapítása
2. A minutiák reprezentálása egy a referenciapontnak és a meghatározott paramétereknek megfelelő poláris koordináta-rendszerben .
3. Az így kapott minutiahalmazok között egy illeszkedési pontszám meghatározása

Megjegyezzük, hogy a poláris koordináta rendszerben való ábrázolás által megoldottnak tekinthető a rotáció és transzláció-invariancia.

A már említett rotáció és transzláció mértékének meghatározására több módszer létezik, pl. a szinguláris pontok (mag és delta) pozícióinak, a minutiák eloszlásának felhasználásával. Más lehetséges megoldás például a végződés-minutiákhoz tartozó redők irányainak segítségével történő esztimáció.

A leggyakoribb eljárás egy szimbólumsorozat generálása, melyben minden elem egy-egy minutia poláris koordinátáit (esetleg más mellékinformációkat, pl. redőirány, minutia típusa, stb.) tartalmazza, így az összehasonlítás redukálódik a két szimbólumsorozat hasonlóságának vizsgálatára.

1.7 A DOLGOZATBAN JAVASOLT MÓDSZER ISMERTETÉSE.

A dolgozat és az alkalmazás megírásakor egy olyan ujjlenyomat-felismerő rendszer megalkotása volt a cél, mely megfelel az általunk legfontosabbnak tartott követelményeknek. Ezek a rotáció- és transzláció-invariancia, biztonságos felismerés, az ujjlenyomat hiányosságával szembeni nagy tolerancia, rossz minőségű ujjlenyomatok felismerése, gyorsaság. Sajnos ez utóbbit nem sikerült maradéktalanul megvalósítani, a program számítási igénye elfogadhatónak nevezhető ugyan, de nagyobb adatbázisokon valószínűleg igen lassúnak bizonyulna.

A rossz minőségű ujjlenyomatok felismerhetőségének növelése érdekében a jelenleg legjobbnak kikiáltott képminőség-javítási technikát alkalmaztuk, névszerint a Gábor szűrős, direkt szürkeárnyalatú eljárást (néhány változtatást eszközölve). A minutiák lokalizálására egy morfológikus értelmező algoritmust használunk, melyet a későbbiekben ismertetünk. Ezen újszerű megoldás apropója a véknyítási folyamat kiküszöbölésének szükségessége (a nagy számítási igény és a torzítások miatt) volt.

Az osztályozást egy a hiba hátraterjedésének módszerével tanított többrétegű neuronális hálóval végeztük. Bemeneti adatokként az orientáció-képet használtuk. A kísérleti eredmények alapján kijelenthetjük, hogy az eredmények jók, de növelhetőek egy jobb tanítási adatbázissal.

Az összehasonlítás szintén újszerű megoldással történik. Mivel nem találtuk megbízhatónak rotációt és transzlációt esztimáló eljárásokat, igyekeztünk egy olyan ábrázolási formát találni, mely ezekre nézve független leírást ad. A módszer lényege, hogy nem egy, hanem több referenciaminutiát választunk, minden választott referenciapontra legenerálunk egy szimbólumsorozatot, mely jellemzi az ujjlenyomatot, és ezeket használjuk fel az összehasonlításhoz. A módszer tökéletesen rotáció és transzláció- invariáns. Hiányossága a relatív lassúság.

A továbbiakban az említett módszert bővebben kifejtjük.

2. KÉPMINŐSÉG JAVÍTÁS

Korábban már indokoltuk a Gábor szűrős technika használatának szükségességét, most részletesen ismertetjük ezt a módszert. Az egyik leghasználhatóbb közelítés, a Hong, Jain, Wan [12] kidolgozta eljárás, mely relatív egyszerűen implementálható, és számítási igénye is elfogadható. Mis is ezt a technikát alkalmaztuk, néhány változtatást eszközölve. A továbbiakban ezt a módszert ismertetjük, a változtatásokra is kitérve.

2.1 JELÖLÉSEK

A szürkeárnyalatú *ujjlenyomatképet* jelölje *I*, ez tulajdonképpen egy *N* x *N* mátrix, ahol *I(i,j)* jelöli az *i*. sor *j*. oszlopában található pixel intenzitását. Az ujjlenyomatkép *várhatóértéke M* és *szórása S,* ahol:

$$M(I) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} I(i,j) \tag{2.1}$$

$$S(I) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (I(i,j) - M(I))^2 \tag{2.2}$$

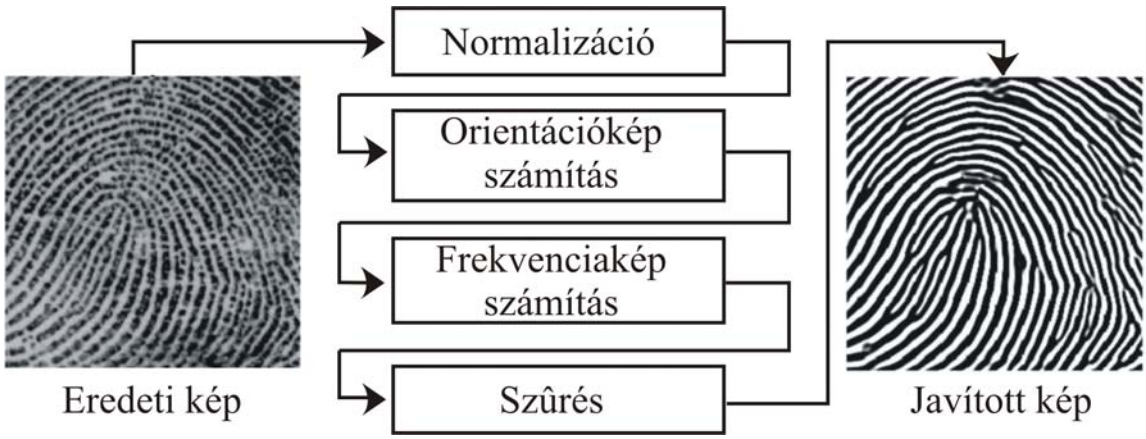
G(i,j) jelöli a normalizált ujjlenyomatképet.

Legyen az *orientációkép, O*, mely szintén egy *n*x*n* mátrix, ahol *O(i,j)* a lokális redőirány az *(i,j)* pixelben.

A frekvenciakép, F, az orientációképhez hasonlóan egy *n*x*n* tömb, melyben *F(i,j)* a lokális redőfrekvenciát jelzi. A frekvenciát *w* x *w* méretű ablakokban, lokálisan számoljuk. Azon blokkok esetében, ahol minutiák vagy szennyeződések jelentkeznek, a frekvencia nem egyértelműen meghatározható. Ezen blokkok esetében a frekvenciát a szomszédos blokkok értékeiből származtatjuk.

2.2 ALGORITMUS

Az alábbi sémán láthatóak az algoritmus fő lépései:



7. ábra: az algoritmus sémája

- Normalizáció: a bementi ujjlenyomatképet úgy módosítjuk, hogy előre meghatározott a szórás és a várhatóérték.
- Lokális orientáció meghatározás: a fent értelmezet orientációképet számítjuk, a már normalizált ujjlenyomatképből
- Lokális frekvencia meghatározás: a normalizált ujjlenyomatképből és az orientációképből kiindulva számítjuk a már értelmezett frekvenciaképet.
- Szűrés: a lokális orientációnak és frekvenciának megfelelően hangolt Gábor szűrőket alkalmazunk.

2.3 NORMALIZÁCIÓ

Az eredeti elképzelésben [12] Hong és társai a teljes képen alkalmazták a normalizációt. Azokban az esetekben, ha a kép megvilágítása nem egyenletes (azaz a kép fele sötét fele pedig világos), a fontos részletek elvesztéséhez vezet. Ennélfogva, javasoltunk egy ablakolásos módszer, mely azt jelenti, hogy a képet w x w (esetünkben 14x14) méretű részekre osztjuk, és az alább leírt módszerrel blokkonként normalizálunk.

Legyen $I_{u,v}$ az u,v blokk az I képből, u.h., $I_{u,v}(i,j)$ az adott blokk i. sorának j. oszlopában található pixel intenzitása. $M_{u,v}$ és $S_{u,v}$ az aktuális szórás és várhatóérték az u,v blokkban. M_0 és S_0 a kívánt várhatóérték és szórás. A $G(i,j)$ normalizált képet a következőképpen számíthatjuk:

$$G_{,m}(i,j)=\begin{cases} M_0+\sqrt{\dfrac{S_0(I_{m,n}(i,j)-M_{m,n})^2}{S_{m,n}}} & \text{, ha } \mathrm{I_{m,n}(i,j) > M_{m,n}} \\ M_0-\sqrt{\dfrac{S_0(I_{m,n}(i,j)-M_{m,n})^2}{S_{m,n}}} & \text{, különben} \end{cases} \tag{2.3}$$

$$G(i,j)=G_{\lfloor i/w\rfloor,\lfloor j/w\rfloor}(\mathrm{i \bmod w, j \bmod w}), \text{ahol i}=\overline{1,\mathrm{m}}\text{, j}=\overline{1,\mathrm{n}} \tag{2.4}$$

Ahol az m és n az I kép méreteit jelentik.

A normalizálás egy úgynevezett pixelszintű művelet, ami azt jelenti, hogy a változtatások nem módosítják a redőstruktúrát. A következő ábra mutatja a normalizáció eredményét:

8. ábra: eredeti és normalizált kép

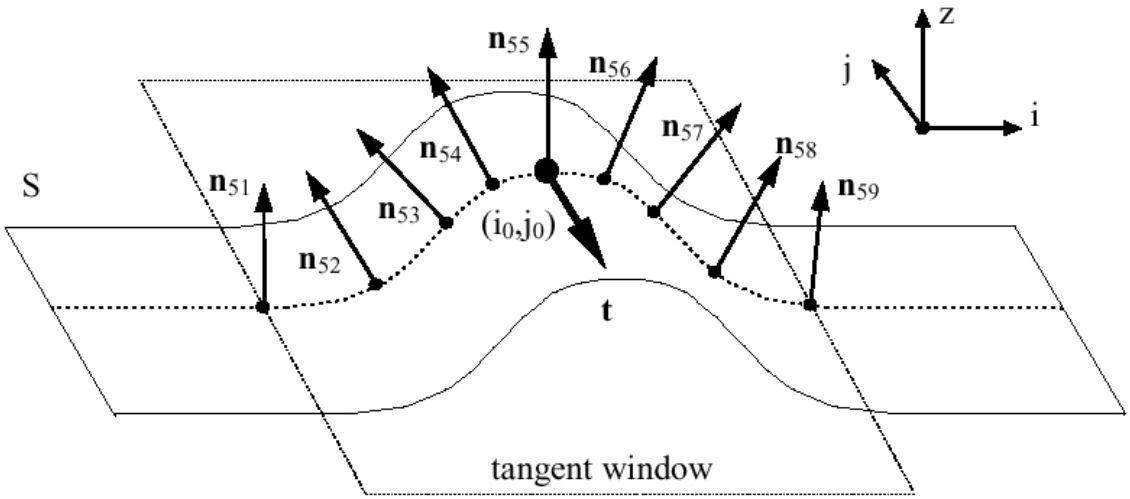
2.4 ORIENTÁCIÓKÉP SZÁMÍTÁS

Az orientációkép számítására több módszer létezik a szakirodalomban. Szinte kivétel nélkül mindegyik a lokális gradienseket használja a számításokhoz. A Hong és társai által javasolt módszert lassúnak és kevésbé megbízhatónak találtuk, ezért a Maio Maltoni szerzőpáros által javasolt eljárást implementáltuk [8]. A továbbiakban is ezt ismertetjük.

A számítások során a már normalizált *G* képet használjuk. Elsősorban leszögezzük, hogy jelen esetben az ujjlenyomatképet felületként kezeljük. A harmadik koordináta a pixelintenzitás. Legyen (i_0,j_0) az a pixel, ahol a orientációt kell számolnunk. Legyen T a tangens ablak, mely egy olyan (i_0,j_0) középpontú négyzetes mátrix, melynek oldalmérete α, értékei pedig a *G* képből átvett intenzitások. Az így kapott ablak minden (i_h,j_k) pixeléhez hozzárendelünk egy n_{hk} egységvektort, mely merőleges a $z=G(i,j)$ felületre. A tangensvektor, ha meghatározott, az *ij* síkon fekszik, és merőleges a hozzátartozó n_{hk} egységvektorra. Az

átlag tangensvektor, mely jelzi a kívánt φ_0 irányt (orientációt), az ij síkon fekvő, a már kiszámított n_{hk} vektorokra „legmerőlegesebb” egységvektor.

A következő ábrán látható a tangensvektor ábrázolása:



9. ábra: Tangens-ablak, tangensvektor

Az átlag tangensvektor kiszámítása a következőképpen zajlik:

Legyen $a_1=T(i_{h+1},j_{k+1})$, $a_2=T(i_{h-1},j_{k+1})$, $a_3=T(i_{h-1},j_{k-1})$ és $a_4=T(i_{h+1},j_{k-1})$ a tangens ablak h,k pixelének szomszédságába tartozó négy pixel intenzitásértéke.

Minden n_{hk} egységvektor meghatározható a következő módon:

$n_{hk} = [a_{hk}, b_{hk}, 1]$, ahol

$$a_{hk} = \frac{-a_1 + a_2 + a_3 - a_4}{4}, \qquad b_{hk} = \frac{-a_1 - a_2 + a_3 + a_4}{4} \tag{2.5}$$

Legyen $v_{hk}=(a_{hk},b_{hk})$, $h=1,..\alpha$, $k=1,..\alpha$ azon vektorok, melyeket a megfelelő n_{hk} vektorokból a z komponens elhagyásával nyerünk. Legyen $t=(t_1,t_2)$. Formálisan ez egy négyzetes minimalizálási módszer:

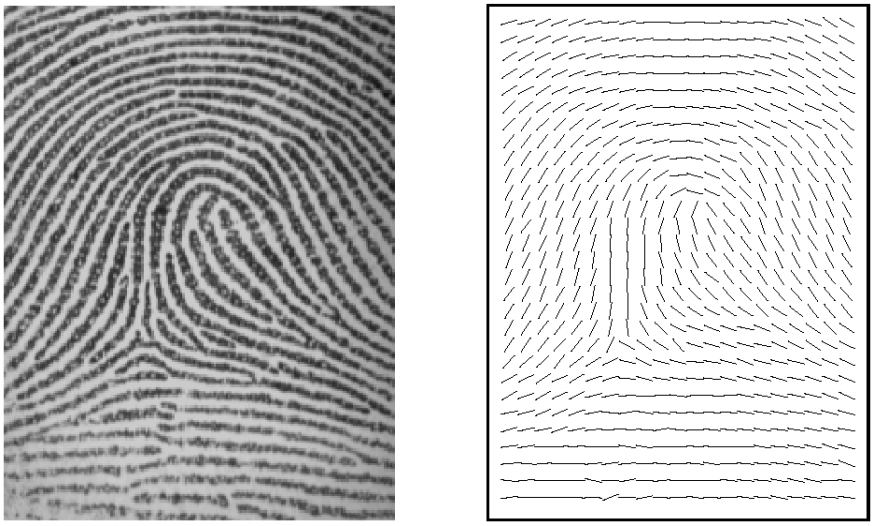
$$\min \sum_{\substack{h=1..\alpha \\ k=1..\alpha}} \left|\langle v_{hk}, t\rangle\right|^2 \quad \text{, ú.h. } \| t \| = 1 \tag{2.6}$$

$$A = \sum_{\substack{h=1..\alpha \\ k=1..\alpha}} (a_{hk})^2 \ , \ B = \sum_{\substack{h=1..\alpha \\ k=1..\alpha}} (b_{hk})^2 \ , \ C = \sum_{\substack{h=1..\alpha \\ k=1..\alpha}} a_{hk} b_{hk} \tag{2.7}$$

$$t = \begin{cases} \left[1,\ \frac{B-A}{2C} - \operatorname{sgn}(C)\sqrt{\left(\frac{B-A}{2C}\right)^2 + 1}\right] & \text{ha } C \neq 0 \\ [1,0] & \text{ha } C = 0,\ A \leq B \\ [0,1] & \text{ha } C = 0,\ A > B \end{cases} \tag{2.8}$$

Ezek után a φ_0 irány könnyedén meghatározható:

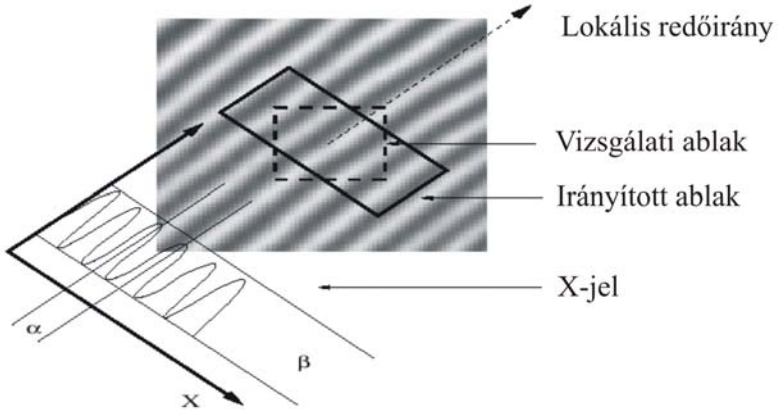
$$\varphi_0 = \begin{cases} \arctan\left(\frac{t_2}{t_1}\right) & \text{ha } t_1 \neq 0 \\ \frac{\pi}{2} & \text{különben} \end{cases} \tag{2.9}$$



10. ábra: Ujjlenyomat és a hozzátartozó orientációkép ábrázolása

2.5 FREKVENCIAKÉP MEGHATÁROZÁSA

Olyan, a képből kiragadott részletekben, ahol nincsenek minutiák vagy szennyeződések, a redők és völgyek szerkezete egy szinuszoid görbével jellemezhető, tehát beszélhetünk a redők frekvenciájáról. Az alábbi képen látható egy minutiamentes képrészlet, és a redőszerkezetet jellemző görbe.



11. ábra: Az x-jel kiszámításához használt irányított ablak

A korábbi jelöléseknek megfelelően legyen *G* a normalizált kép, *O* az orientációkép. A frekvencia-meghatározási eljárás lépései a következők:

Felosztjuk *G*-t *w* x *w* (16x16) méretű ablakokra

Minden *(i,j)* középpontú ablakhoz hozzárendelünk egy *l* x *w* (32x16) méretű irányított ablakot (lásd a fenti ábrát).

Minden *(i,j)* középpontú ablakban kiszámítjuk az x-jelet (x signature), mely értékei *X[0],X[1], . . X[l-1],* ahol

$$X[k]=\frac{1}{w}\sum_{d=0}^{w-1}G(u,v),\quad \mathrm{k}=0,1,..1\text{-}1$$

$$u=i+\left(d-\frac{w}{2}\right)\cos O(i,j)+\left(k-\frac{l}{2}\right)\sin O(i,j), \tag{2.10}$$

$$v=j+\left(d-\frac{w}{2}\right)\sin O(i,j)+\left(\frac{l}{2}-k\right)\sin O(i,j)$$

Ha nincs minutia vagy szennyeződés az irányított ablakban, az x-jel egy diszkrét szinuszoid jellegű formát mutat, melynek frekvenciája megegyezik a redőstruktúra frekvenciájával. Ennélfogva a frekvenciát meghatározhatjuk az x-jel alapján. Legyen *T(i,j)* az *(i,j)* középpontú ablakhoz tartozó x-jel két egymást követő csúcsa közötti pixelek számának átlaga, ekkor a frekvencia az adott blokkra a következőképpen számítható: *Ω (i,j) =1/T(i,j)*. Ha nem biztonságosan meghatározhatóak a csúcsok, a frekvenciakép értékét -1 –re állítjuk.

4. Az F.B.I. által is javasolt, és a leggyakrabban (általunk is) használt 500 dpi felbontással szkennelt képek esetében a frekvencia az [1/5,1/20] tartományban van.

Ennélfogva, ha a kiszámított frekvenciaérték nincs ebben az intervallumban, az azt jelenti, hogy az adott blokkban a helyes érték nem meghatározható, tehát a kapott eredményt figyelmen kívül hagyjuk, illetve értékét -1 –re állítjuk, hogy megkülönböztessük a helyes értékektől.

5. Azon blokkok esetében, ahol a frekvencia-megállapítás nem bizonyult sikeresnek, a lokális frekvencia értékét a szomszédos blokkok értékeiből kell interpolálnunk. Az interpoláció menete a következő:

(i) minden *(i,j)* középpontú ablakban

$$\Omega'(i,j)=\begin{cases}\Omega(i,j) & \text{ha } \Omega(i,j)\neq -1\\ \dfrac{\sum_{u=-w_\Omega/2}^{w_\Omega/2}\sum_{v=-w_\Omega/2}^{w_\Omega/2}W_g(u,v)\mu(\Omega(i-uw,j-vw))}{\sum_{u=-w_\Omega/2}^{w_\Omega/2}\sum_{v=-w_\Omega/2}^{w_\Omega/2}W_g(u,v)\delta(\Omega(i-uw,j-vw)+1)} & \text{különben}\end{cases} \tag{2.11}$$

ahol

$$\mu(x)=\begin{cases}0, & \text{ha } x\le 0\\ x, & \text{különben}\end{cases}$$

$$\delta(x)=\begin{cases}0, & \text{ha } x\le 0\\ 1, & \text{különben}\end{cases}$$

W_g egy diszkrét Gauss mag, ahol a várhatóérték 0, a szórás 9, és $w_\Omega = 7$ a mag mérete.

(ii) Ha még létezik legalább egy olyan ablak, melyben a frekvencia értéke meghatározatlan (illetve -1), cseréljük fel Ω és Ω' és ismételjük az első lépést.

6. A redők közötti távolság méreteiben nincsenek nagy ugrások, ezért a kapott frekvenciakép finomítására ajánlott egy szűrő használata. Így megszüntethetőek a nagy ugrások a frekvenciaértékekben. A frekvenciakép végső értékei tehát a következő módon számíthatóak:

$$F(i,j)=\sum_{u=-w_l/2}^{w_l/2}\sum_{v=-w_l/2}^{w_l/2}W_l(u,v)\Omega'(i-uw,j-vw) \tag{2.12}$$

ahol W_l egy kétdimenziós „low-pass” szűrő, w_l=7 a szűrő mérete.

A diszkrét Gauss mag kiszámítása a Gauss-eloszlásfüggvény alapján történik, tulajdonképpen egy *n* x *n* mátrix:

$$W_g(i,j)=\frac{1}{2\pi\sigma^2}e^{-\frac{i^2+j^2-m}{2\sigma^2}}\quad,\ \text{ahol } i=\overline{-n/2,n/2}\ ,\ j=\overline{-n/2,n/2} \tag{2.13}$$

2.6 SZŰRÉS

A redők és völgyek párhuzamos szerkezete, jól meghatározott irány és frekvencia paraméterek esetén nagy segítséget nyújthat a nemkívánatos szennyeződések eltávolításában. A már említett szinuszoid görbék jellege lokálisan csak kismértékben változik, ennélfogva egy megfelelően hangolt ú.n. „bandpass" szűrő hatékony lehet a szemét eltávolításában, anélkül, hogy a valós redőszerkezetet lényegesen módosítaná. A Gábor szűrők, frekvencia- és orientáció-függésüknél fogva erre a célra tökéletesen megfelelnek.

A kétdimenziós „even-symmetric" (a komplex rész elhagyva) Gábor – szűrő általános alakja:

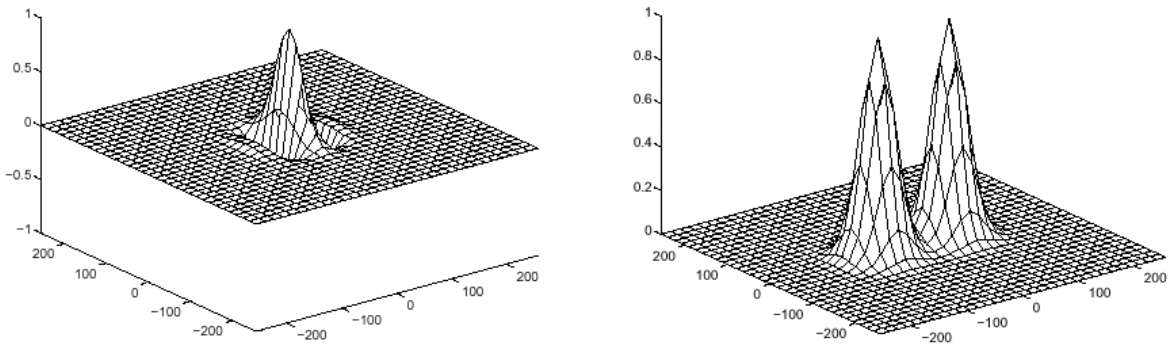
$$h(x,y:\phi,f)=\exp\left\{-\frac{1}{2}\left[\frac{(x\cos\phi)^2}{\delta_x^2}+\frac{(y\sin\phi)^2}{\delta_y^2}\right]\right\}\cos(2\pi fx\cos\phi) \qquad (2.13)$$

ahol ϕ a Gábor szűrő orientációja, f a frekvencia, δ_x, δ_y konstansok.

A „modulation transfer function" (MTF) a következőképpen ábrázolható:

$$\begin{aligned} H(u,v:\phi,f)=2\pi\delta_x\delta_y\exp\left\{-\frac{1}{2}\left[\frac{(u-2\pi/f)(\sin\phi)^2}{\delta_u^2}+\frac{v(\cos\phi)^2}{\delta_v^2}\right]\right\}+ \\ 2\pi\delta_x\delta_y\exp\left\{-\frac{1}{2}\left[\frac{(u-2\pi/f)(\sin\phi)^2}{\delta_u^2}+\frac{v(\cos\phi)^2}{\delta_v^2}\right]\right\} \end{aligned} \qquad (2.14)$$

ahol $\delta_u = 1/2\pi\ \delta_x$ és $\delta_v = 1/2\pi\ \delta_y$. Az alábbi ábrán látható az „even-symmetric" Gábor szűrő, és a hozzátartozó MTF.



12. ábra:Gábor szűrő 0^o orientáció és 1/10 frekvencia, valamint a hozzátartozó MTF

Ahhoz, hogy egy Gábor szűrőt alkalmazzunk, meg kell határoznunk három paraméter értékét: a szinuszoid görbe frekvenciáját, a szűrő orientációját, valamint a δ_x , δ_y konstanspárt. Értelemszerűen, frekvenciaparaméterként a már kiszámított lokális redőfrekvenciát, orientációparaméterként pedig a lokális redőorientációt használjuk. A δ_x , δ_y paraméterek kiválasztása empirikusan történik. Ha ezen értékek nagyobbak, a szűrő hatékonyabban szünteti meg a szemetet, de nagyobb a hamis redők létrehozásának esélye. Ellenkező esetben, értelemszerűen a szűrő kevésbé hatékony, de kevesebb hibát is vét. A mérési adatoknak megfelelően Hong és munkatársai mindkét paraméter értékét 4.0 –ben határozták meg.

Legyen tehát *G* a normalizált bemeneti kép, *O* az orientációkép, *F* a frekvenciakép, az *E* javított kép a következő módon számítható:

$$E(i,j) = \sum_{u=-w_g/2}^{w_g/2} \sum_{v=-w_g/2}^{w_g/2} h(u,v : O(i,j), F(i,j)) G(i-u, j-v) \tag{2.15}$$

ahol $w_g = 11$ a Gábor szűrő méretét jelzi.

3. OSZTÁLYOZÁS

Az osztályozás célja az adatbázis felosztása, ez lehetővé teszi azt, hogy egy azonosítási folyamat során ne keressünk a teljes adatbázisban, hanem csak azon részében, melybe a vizsgált ujjlenyomat is tartozik. Mint említettük, esetünkben legcélszerűbb négy osztályba sorolni a mintákat. Ha bonyolultabb sémát választunk, a hibás osztályozások száma igen nagy. A korábbiakban már vázoltuk a négy osztályt (ív, csigavonal, jobb és balhurok).

Az osztályozást egy a hiba hátraterjedésének módszerével tanított többrétegű perceptron modell segítségével végezzük.

3.1 A TÖBBRÉTEGŰ PERCEPTRON MODELL (MULTILAYER PERCEPTRON)

A többrétegű perceptron modell (továbbiakban MLP) [15, 16] három fontosabb részből áll: bemeneti réteg (egy neuronhalmaz), ezen keresztül kapja meg az MLP a bemeneti információkat, egy vagy több rejtett réteg valamint egy kimeneti réteg. A bemeneti jel rétegről-rétegre terjedve halad át a hálón *(lásd ábra)*.

Egy réteg nem más, mint egy neuronhalmaz, melyek elemei egymással nincsenek összeköttetésben. A rejtett illetve kimeneti rétegek neuronjainak bemenetei, az őket megelőző réteg neuronjainak kimenetére vannak csatolva. A bemeneti jel "áthaladásán", azt a folyamatot értjük, melynek során a bemeneti réteg neuronjainak kimenetein keletkezett jel feldolgozásra kerül a következő réteg neuronjaiban, majd az itt keletkezett jel hasonló módon kerül a következő rétegbe, ugyanígy rekurzívan, amíg el nem jut a kimeneti rétegbe.

Az MLP leggyakoribb tanítási formája a hiba hátraterjedésének módszere (error back-propagation, továbbiakban BP). Ez egy felügyelt tanítási forma, egyszerűségének és hatékonyságának köszönhetően igen népszerűvé vált, tulajdonképpen ennek köszönhető az MLP-k elterjedése.

Alapvetően a tanítás két lépésből áll, első lépésben a bemeneti rétegnek bemutatjuk a bemeneti információt, majd ez a bemeneti jel előrehalad az MLP-ben, a kimeneti rétegig, ezt nevezzük előrelépésnek. A kimeneti rétegben kiszámítjuk a hibajelet (a kívánt és a kapott eredmény különbsége), majd ezt a jelet hátrafelé léptetjük a hálóban, és a kívánt mértékben módosítjuk a szinaptikus súlyokat.

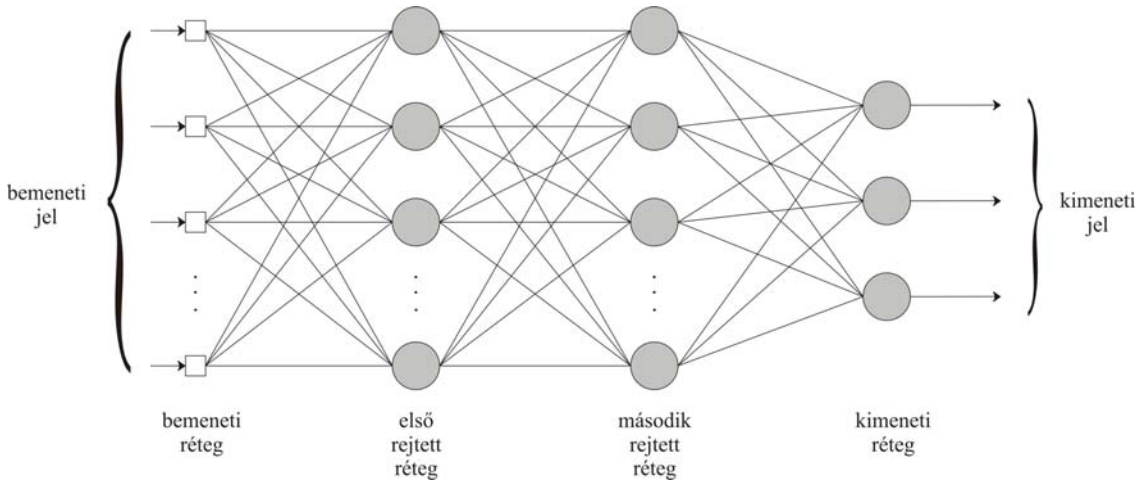
A többrétegű perceptron modell három jellemző tulajdonsággal rendelkezik:

- A modell minden neuronja rendelkezik egy *nemlineáris aktivációs függvénnyel*. A leggyakrabban használt aktivációs függvény a szigmoidális, azaz

$$y_j = \frac{1}{1+\exp(-v_j)}$$

ahol v_j a j neuron belső aktiváltsága, azaz a súlyozott bemenetek összege; y_j a neuron kimenete.

- Egy vagy több rejtett réteg, melyek neuronjai nem tagjai sem a kimeneti sem a bemeneti rétegnek. Ezek a rejtett rétegek teszik képessé a hálót bonyolult feladatok megoldására, ugyanis a bementi mintákból fontos jellemzőket vonnak ki.
- magas szintű konnektivitás, azaz a tulajdonképpeni szinapszisokat megvalósító kapcsolatok. Bármely változás ezen kapcsolatrendszerben a szinapszisok súlyainak változtatásának szükségességét vonja maga után.



15. ábra: Két rejtett rétegű MLP arhitektúrája, szinaptikus kapcsolatok

3.2 JELÖLÉSEK ÉS ELNEVEZÉSEK.

Mint már említettük két típusú jelet különböztetünk meg az MLP-ben. Az egyik a stimulus, ami nem más mint a bemeneti jel, a másik a hibajel, mely a kimeneti rétegben keletkezik és visszafelé halad a hálóban.

A továbbiakban *i,j* és *k* a háló különböző neuronjait jelentik. Különböző és egymást követő rétegekben vannak, sorrendjük megfelel az előrehaladási iránynak. Ebből következik, hogy *j* egy rejtett réteghez tartozik.

Az *n.* iteráció jelenti az n-edik tanítási minta bemutatását.

$E(n)$ jelenti a négyzetes hibák összegét az n iterációban. Az átlagos hiba azaz E_{av} nem más mint az $E(n)$értékek számtani közepe.

$e_j(n)$ – a j neuron hibajele az n iterációban.

$d_j(n)$ – a j neurontól elvárt válasz az n iterációban

$y_j(n)$ – a j neuron válasza (kimeneti jel)

$w_{ji}(n)$ – az i neuron kimenetét a j neuronnal összekötő él súlya az n iterációban. Az említett élen végrehajtott korrekció: $\Delta w_{ji}(n)$

$v_j(n)$ – a j neuron aktiváltsága az n iterációban

$\varphi_j(\cdot)$ – a j neuron válaszfüggvénye

$b_j(n)$- A bias a j neuronban, ez nem más, mint egy +1 értékű állandó bemenetre csatolt él

$x_i(n)$ – a n bemeneti minta i eleme

$o_k(n)$ – a kimeneti vektor k eleme

η – a tanítási paraméter (0 és 1 közöti konstans)

m_l – az l réteg mérete (neuronok száma), $l = 0, 1, \ldots, L$ ahol L a rétegek száma azaz az MLP mélysége

3.3 A HIBA HÁTRATERJEDÉSÉNEK ALGORITMUSA

A j neuron hibajele az n iterációban a következőképpen határozható meg:

$$e_j(n) = d_j(n) - y_j(n) \text{ , ahol } j \text{ egy kimeneti neuron} \tag{3.1}$$

Értelmezés szerint legyen a négyzetes hiba értéke a j neuron esetében $\frac{1}{2}e_j^2(n)$. A négyzetes hiba az MLP-re nézve tehát a kimeneti neuronok hibáinak összege, azaz:

$$E(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n) \tag{3.2}$$

Ahol C a kimeneti rétegben található neuronok halmaza. Legyen N a tanítási minták száma, ekkor az átlagos hiba értéke:

$$E_{av}(n) = \frac{1}{N}\sum_{n=1}^{N} E(n) \tag{3.3}$$

Megállapíthatjuk, hogy a négyzetes hiba $E(n)$ és az átlagos hiba E_{av} a neuronháló paramétereinek azaz a súlyok és bias-ok függvénye. Egy adott tanítási minta esetén E_{av} költségfüggvényként tekinthető, és általa mérhető a tanulási teljesítmény. A tanítási folyamat tehát nem más, mint a már említett paraméterek változtatása annak érdekében, hogy E_{av} értékét minimalizáljuk. E cél elérése érdekében egy egyszerű módszert alkalmazunk, mely szerint a súlyokat minden egyes újabb minta bemutatása után javítjuk (a gradiens módszerrel). A teljes tanítási mintahalmaz bemutatását korszaknak nevezzük. A korszak végén kiértékelhető az MLP aktuális pontossága. Az előbb említett, minden minta esetében végbemenő módosítások számtani közepe tulajdonképpen úgy tekinthető, mint annak a módosításnak a becslése, melyet a korszak végén E_{av} –t kiértékelve kapnánk.

Tekintsünk egy j neuront, egy rejtett rétegből. Ekkor a neuron bemeneteit az őt megelőző réteg neuronjainak kimeneti jelei adják. Ekkor a j neuron aktiváltsága:

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \tag{3.4}$$

ahol m a bemenetek száma (a biast leszámítva). A w_{j0} súly (a rögzített y_0=+1 bemenetnek megfelelő) nem más mint a bias. Így elkerülhetjük azt, hogy a számításokban különbséget tegyünk a bias és a súlyok között.
A j neuronban keletkező kimeneti jel az n iterációban tehát:

$$y_j(n) = \varphi(v_j(n)) \tag{3.5}$$

A hiba hátraterjedésének algoritmusa egy a $\partial E(n)/\partial w_{ji}(n)$ parciális deriválttal arányos $\Delta w_{ji}(n)$ értékkel javítja az élek súlyait. A parciális derivált kifejezhető:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_j(n)} \tag{3.6}$$

A (3.2) egyenlet mindkét oldalát differenciálva $e_j(n)$ szerint kapjuk:

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \tag{3.7}$$

A (3.1) egyenlet mindkét oldalát differenciálva $y_j(n)$ szerint kapjuk:

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{3.8}$$

Következőként a (3.5) egyenletet differenciáljuk $v_j(n)$ szerint:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi'(v_j(n)) \tag{3.9}$$

Végül a (3.4) egyenletet $w_{ji}(n)$ szerint differenciálva:

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{3.10}$$

A (3.7)–(3.10) egyenleteket behelyettesítve a (3.6) egyenletbe:

$$\frac{\partial E(n)}{\partial e_j(n)} = -e_j(n)\varphi'(v_j(n))y_i(n) \tag{3.11}$$

Az élekre alkalmazott módosítás értéke:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{3.12}$$

ahol η a tanítási paraméter, $-\partial E(n)/\partial w_{ji}(n)$ a leszállási irány.

A (3.11) egyenletet a (3.12)–be behelyettesítve kapjuk:

$$\Delta w_{ji}(n) = -\eta \delta_j(n) y_i(n) \tag{3.13}$$

ahol a lokális gradiens, $\delta_j(n)$ a következő:

$$\begin{aligned} \delta_j(n) &= -\frac{\partial E(n)}{\partial w_{ji}(n)} \\ &= -\frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \\ &= e_j(n)\varphi'(v_j(n)) \end{aligned} \tag{5.14}$$

A (3.14) egyenletből látható, hogy a kívánt változtatás a neuronnak megfelelő hibajel és a választott válaszfüggvény deriváltjának szorzata.

Mivel láthatjuk, hogy a hibajelnek kulcsszerepe van a $w_{ji}(n)$ meghatározásában, és mint tudjuk, ezt egyelőre csak a kimeneti rétegben ismerjük, ezért két esetet különböztetünk

meg, attól függően, hogy a j neuron rejtett vagy kimeneti. Természetesen a rejtett neuronok súlyvektorait is módosítanunk kell, ennek érdekében meg kell állapítanunk, hogy egy rejtett neuron milyen mértékben felelős a végső hiba méretéért, azaz ki kell számolnunk minden rejtett neuron hibajelét. Ezt a hibajel hátrafelé történő terjesztésével érjük el (innen az elnevezés).

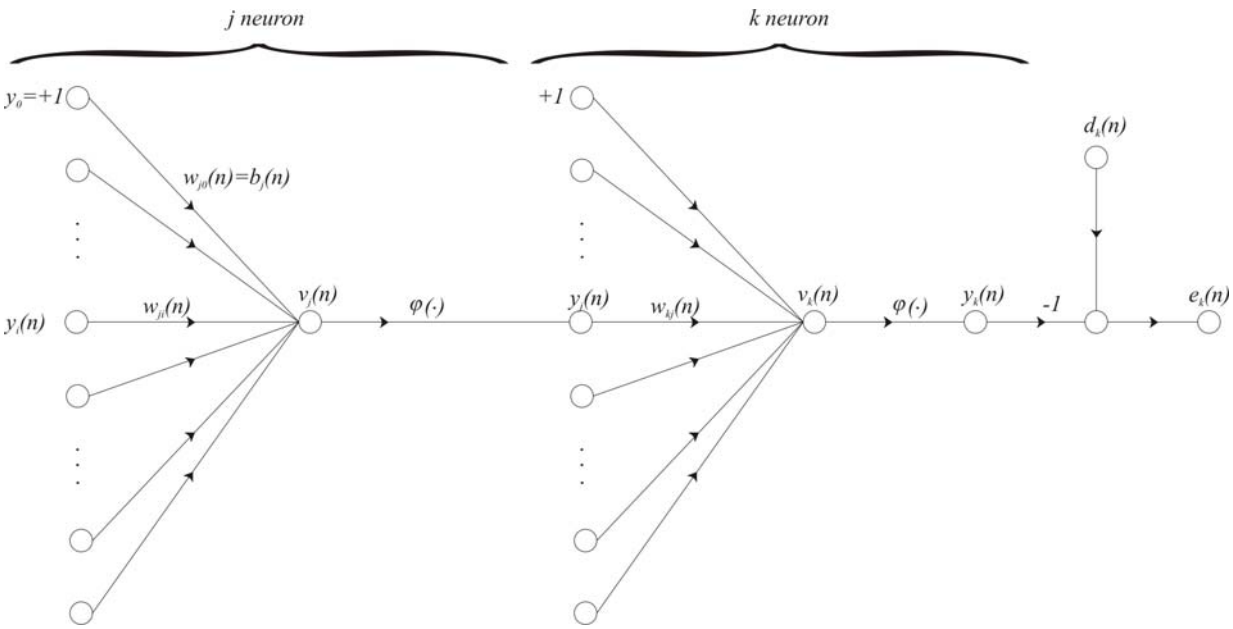
A már említett két eset tehát:

1. a *j* egy kimeneti neuron

Ha egy neuron a kimeneti rétegben van, ismerjük a várt kimeneti jelet. A (3.1) egyenlet segítségével könnyedén kiszámíthatjuk a hibajelét. Az $e_j(n)$ értékét ismerve a lokális gradiens kiszámítása a (3.14) segítségével triviális.

2. a *j* egy rejtett neuron

Ha egy neuron egy rejtet rétegben van, nem ismerjük a várt kimeneti jelet, tehát meg kell állapítanunk azt, mégpedig úgy, hogy rekurzívan visszavezetjük azon neuronok hibajeléből, melyekkel összeköttetésben áll.



16. ábra: Előrehaladó lépés folyama az MLP-ben

A (3.14) egyenlet alapján újradefiniálhatjuk a lokális gradienst:

$$\begin{aligned}\delta_j(n) &= -\frac{\partial E(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} \\ &= -\frac{\partial E(n)}{\partial y_j(n)}\varphi'(v_j(n)) \quad , j \text{ neuron rejtett}\end{aligned} \tag{3.15}$$

A $\partial E(n)/\partial y_j(n)$ parciális derivált kiszámítása következik. Értelmezés szerint, az (3.2) alapján

$$E(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n), \quad \text{ahol } k \text{ egy kimeneti csúcs} \tag{3.16}$$

A (3.16) egyenlet differenciálja az $y_j(n)$ kimeneti jel szerint:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \tag{3.17}$$

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{3.18}$$

Ugyancsak értelmezés szerint, és az amint az ábrán is látható:

$$\begin{aligned} e_k(n) &= d_k(n) - y_k(n) \\ &= d_k(n) - \varphi_k(v_k(n)) \ , k \text{ kimeneti neuron} \end{aligned} \tag{3.19}$$

következésképp

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)) \tag{3.20}$$

A k neuron aktiváltsága:

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n) \tag{3.21}$$

ahol m a k neuron bemeneteinek száma. Ezt az $y_j(n)$ szerint differenciálva kapjuk:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{3.22}$$

A (3.20) és (3.22) egyenleteket felhasználva, a (3.18) egyenlet alapján megkapjuk a kívánt parciális deriváltat:

$$\begin{aligned} \frac{\partial E(n)}{\partial y_j(n)} &= -\sum_k e_k(n)\varphi_k'(v_k(n))w_{kj}(n) \\ &= -\sum_k \delta_k(n) w_{kj}(n) \end{aligned} \tag{3.23}$$

a második sorban felhasználtuk a lokális gradiens (3.15) egyenletben adott értelmezését.
És végül a (3.23) és (3.15) egyenleteket felhasználva felírhatjuk a lokális gradiens hátraterjedésének egyenletét:

$$\delta_j(n) = \varphi_j'(v_j(n))\sum_k \delta_k(n) w_{kj}(n), \quad \text{ha a } j \text{ neuron rejtett} \tag{3.24}$$

Összefoglalva az algoritmus a következő képen írható:

$$\begin{pmatrix} \text{súly-} \\ \text{korrekció} \\ \Delta w_{ji}(n) \end{pmatrix} = \begin{pmatrix} \text{tanítási} \\ \text{paraméter} \\ \eta \end{pmatrix} \begin{pmatrix} \text{lokális} \\ \text{gradiens} \\ \delta_j(n) \end{pmatrix} \begin{pmatrix} j \text{ neuron} \\ \text{bemeneti jelei} \\ y_i(n) \end{pmatrix} \tag{3.25}$$

A tanítási paraméter megválasztása egy igen kényes feladat. Csak 0 és 1 között vehet fel értékeket, $0 < \eta < 1$. Minél nagyobb a η annál gyorsabban közelít az algoritmus, és fordítva. Azonban van egy probléma, ha a választott érték túl nagy, előfordulhat, hogy az algoritmus oszcillálni kezd. Ha a választott érték kicsi, hosszabb ugyan a tanítás menete, de jobb eredmények érhetőek el.

3.4 A VÁLASZFÜGGVÉNY

Minden neuron esetében a lokális gradiensek számításához szükséges az adott neuron válaszfüggvényének a deriváltjának ismerete. A deriválhatóság megköveteli, hogy a válaszfüggvény folytonos legyen (elméletileg ez az egyetlen feltétel amit teljesítenie kell).

Az egyik legnépszerűbb, és általunk is használt válaszfüggvény az úgynevezett szigmoidális, melynek alakja:

$$\varphi(x) = \frac{1}{1+\exp(-ax)} \tag{3.26}$$

Az MLP esetében alakja:

$$\varphi_j(v_j(n)) = \frac{1}{1+\exp(-av_j(n))} \quad , \text{ ahol } a > 0 \text{ és } -\infty < v_j(n) < \infty \tag{3.27}$$

ahol $v_j(n)$ a j neuron aktiváltsága. Mint láthatjuk, a kimeneti jel 0 és 1 között van: $0 \le y_j \le 1$
A (3.27) egyenletet differenciálva $v_j(n)$ szerint:

$$\varphi_j'\left(v_j(n)\right) = \frac{a\exp(-av_j(n))}{\left[1+\exp(-av_j(n))\right]^2} \tag{3.28}$$

Felhasználva azt, hogy $y_j(n) = \varphi_j(v_j(n))$, kiküszöbölhetjük az exponenciális kifejezést:

$$\varphi_j'\left(v_j(n)\right) = ay_j(n)(1-y_j(n)) \tag{3.29}$$

Egy a kimeneti rétegben levő *j* neuron esetében $y_j(n) = o_j(n)$, ennélfogva a *j* neuron esetében a lokális gradiens a következőképpen számítható:

$$\begin{aligned}\delta_j(n) &= e_j(n)\varphi_j'(v_j(n)) \\ &= a\left[d_j(n)-o_j(n)\right]o_j(n)\left[1-o_j(n)\right], \quad \text{ha a } j \text{ kimeneti neuron}\end{aligned} \tag{3.30}$$

Egy rejtett rétegben levő *j* neuron lokális gradiense pedig:

$$\begin{aligned}\delta_j(n) &= \varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n) \\ &= ay_j(n)\left[1-y_j(n)\right]\sum_k \delta_k(n)w_{kj}(n), \quad \text{ha a } j \text{ neuron rejtett}\end{aligned} \tag{3.31}$$

3.5 AZ OSZTÁLYOZÁST VÉGZŐ MLP

Az ujjlenyomatok osztályozását tehát egy MLP segítségével végezzük. A kiválasztott háló egy bemeneti, egy kimeneti és két rejtett rétegből áll. A bemeneti réteg a tanítási adatok dimenziójának megfelelően 400, a kimeneti, mivel ez egy osztályozó háló, az osztályok számával egyenlően 4 neuront tartalmaz. Mindkét rejtett réteg 50 neuronból áll.

A betanított háló esetében, egy adott minta bemutatása után, a kimeneti rétegben az annak az osztálynak megfelelő neuron tüzel, melybe a minta tartozik.

Az alkalmazott válaszfüggvény szigmoidális. A tanításhoz a fentebb ismertetett hiba hátraterjedésének algoritmusát használtuk.

3.6 BEMENETI ADATOK GENERÁLÁSA, TANÍTÁS

A tanítási folyamatban használt mintákat az ujjlenyomatképekből képezzük. Alapkövetelmények, hogy a dimenziók száma konstans legyen minden lenyomat esetében, valamint, hogy a minta reprezentatív legyen. Legalkalmasabb a redőirány adta információ, hiszen ez könnyen számítható valamint tökéletesen jellemzi az osztályokat. A bemeneti

adatok reprezentatív jellege és mérete közötti egészséges egyensúly megtartásának érdekében, a minta dimenzióját 400-ban határoztuk meg (empirikus úton). Tulajdonképpen a 3.4 fejezetben már ismertetett orientációképből indulunk ki. A képre “ráhúzunk” egy 20 x 20 méretű hálót, és minden csomópontban megvizsgáljuk a lokális orientációt. A 400 csomópontból álló mátrixból képezzük a bemeneti vektort, tulajdonképpen ez azt jelenti, hogy a mátrix sorait egymás után fűzzük.

A tanítási folyamatban használt mintahalmaz 40 elemből áll, azaz minden osztályhoz tartozik 10 minta. Ez a mennyiség csak kísérleti célokra elegendő, az osztályozás megbízhatósága érdekében ezt növelni kell.

A módszer jelentős hiányossága, hogy nem rotáció invariáns. A neuronháló magas hiba-toleranciaszintje megenged bizonyos mértékű forgatást, mindazonáltal lényeges, hogy a bemutatott ujjlenyomat hozzávetőlegesen függőleges helyzetben legyen.

4. UJJLENYOMATOK ÖSSZEHASONLÍTÁSA

Az összehasonlítás képezi a tulajdonképpeni feladatot. Azok után, hogy az ujjlenyomatot feldolgoztuk, besoroltuk valamely osztályba, meg kell vizsgálni, hogy milyen mértékben azonos az adatbázisban tárolt lenyomatokkal. Mint már említettük, az összehasonlítást a minutiák alapján végezzük. A kriminológiában általánosan elfogadott nézet, hogy ha két ujjlenyomat legalább nyolc minutiája egyértelműen egyezik, a két ujjlenyomatot azonos személytől származónak tekinthető. Automatizált azonosítási rendszerekben ez az eljárás nem célszerű, kézenfekvőbb egy pontozási rendszert bevezetni, meghatározni a két ujjlenyomat illeszkedésének mértékét, és a kapott eredmény alapján eldönteni, hogy azonosak-e vagy sem.

4.1 MINUTIA KERESÉS

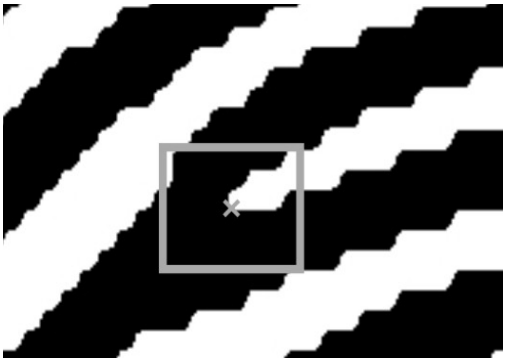
A szűrés eredményeképp kapott képet binarizálva, megkapjuk a bináris redőképet, melyben minden pixel, mely egy redőhöz tartozik fekete, minden völgyhöz tartozó képpont pedig fehér.

13. ábra: Bifurkáció és a komplementerkép

Megfigyelhető, hogy a bifurkáció, tulajdonképp egy végződés komplementere. Ennélfogva elegendő a bifurkáció detektálásra koncentrálnunk, a végződések keresése hasonló. A feladat redukálható egy olyan jellemző meghatározására, mely egyértelműen meghatároz minden bifurkációt. Elsősorban el kell döntenünk, hogy az adott bifurkáció mely pixelét vegyük figyelembe a koordináták meghatározásakor. Úgy döntöttünk, hogy a völgy végződéspontja, azaz egy fehér pixel legyen a meghatározó pont. A bináris redőkép minden pixelét meg kell vizsgálnunk. A vizsgálathoz természetesen elemeznünk kell a környezetet is. Ennélfogva minden képponthoz hozzárendelünk egy *w* x *w* méretű ablakot, melynek középpontjában az aktuálisan vizsgált képpont van.

Legyen n_r a redőpontok, n_v pedig a völgypontok száma a vizsgált ablakban. Legyen $k = \frac{n_r}{n_v}$, a redő és völgypontok aránya.

Az alábbi ábrán látható a bifurkáció és a vizsgálati ablak.



14. ábra: Bifurkáció és a vizsgált környezet

Annak a feltétele, hogy egy pont bifurkációhoz tartozik, a következőképpen írható:

- A vizsgált pont völgypont
- A *k* értéke meghalad egy bizonyos *t* küszöbértéket
- A redőpontok a vizsgált ablakban egy összefüggő régiót alkotnak

Az első feltétel biztosítja, hogy nem egy redő belső pontjáról van szó. A *t* küszöbérték kiválasztása egy kényes feladat, alacsony érték esetén egy egyszerű hajlat is minutiaként értékelhető, túl magas érték esetén pedig fennáll a minutiák figyelmen kívül hagyásának a veszélye. A harmadik feltétel biztosítja, hogy két, egymáshoz közel fekvő redő ne értékelődjék minutiaként.

A fentebb vázolt feltétel egy bifurkáció esetén több képpontra is igaz. A megfelelő képpont kiválasztása mindössze azt jelenti, hogy azt a képpontot választjuk, mely esetében a *k* értéke maximális. Ezen kiválasztás során, legrosszabb esetben, valódi helyzethez viszonyított maximális deviancia értéke $w\frac{\sqrt{2}}{2}$. Ez az eltérés a véknyítási folyamat okozta eltolódásokhoz viszonyítva elhanyagolható.

Az általunk végzett kísérletek alapján kijelenthetjük, hogy az imént ismertetett eljárás kevesebb hamis minutiát eredményez és gyorsabb mint a véknyításos technika.

4.2 AZ ILLESZKEDÉS FOGALMA

Tekintsünk a sík egy ponthalmazát. Ezt reprezentáljuk egy súlyozott gráf segítségével, oly módon, hogy a gráf minden csúcsa össze van kapcsolva minden csúcssal, az élek súlya pedig a pontok közötti távolság. Ez egy redundáns, de tökéletes leírás.

Szükségünk van néhány fogalom bevezetésére:

A *teljes súlyozott gráf*, egy olyan súlyozott gráf, melyben bármely két csúcs között létezik él.

Egy teljes súlyozott gráf *teljes részgráfja* a csúcsok és élek egy részhalmaza úgy, hogy a kiválasztott csúcsok és élek egy teljes súlyozott gráfot alkotnak.

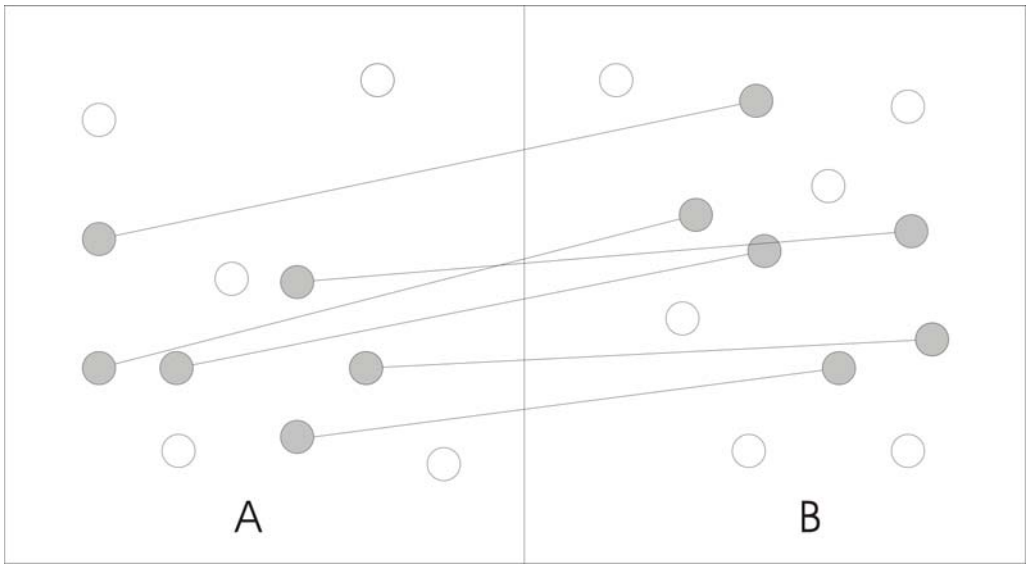
Egy teljes súlyozott gráf egy I csúcsából kiinduló élek halmazát jelölje I_e .

Egy A és B teljes súlyozott gráf *ekvivalens*, ha a csúcsok száma azonos, és A bármely I csúcsát választva létezik a B egy olyan J csúcsa, melyekre igazak a következő kijelentések:

- I_e és J_e elemeinek száma azonos, azaz $card(I_e)=card(J_e)$
- I_e bármely elemét választva létezik a J_e egy olyan eleme, mely azonos az I_e választott elemével, azaz $\forall x \in I_e,\ \exists y \in J_e,\quad x = y$

Tekintsük két ponthalmaz az imént leírt módszerrel képzett gráfreprezentációját. A két ponthalmaz *illeszkedő ponthalmazait* az őket ábrázoló teljes súlyozott gráfok ekvivalens teljes részgráfjai képezik.

Az összehasonlítás problémája tehát nem más, mint két ponthalmaz esetében a legnagyobb illeszkedő részhalmaz megkeresése. Az alábbi ábrán igyekeztünk szemléltetni a feladatot.



17. ábra: Két ponthalmaz legnagyobb illeszkedő részhalmaza.

A szürkével jelölt pontok képezik az illeszkedő ponthalmazokat. Jól látható, hogy az illeszkedést nem befolyásolja más, csupán a részhalmazok elemeinek egymáshoz viszonyított helyzete.

4.3 A MINUTIÁK REPREZENTÁCIÓJA

Mint az 1.6 és 1.7 fejezetekben említettük, egy automatizált felismerő rendszernek teljesítenie kell néhány követelményt:

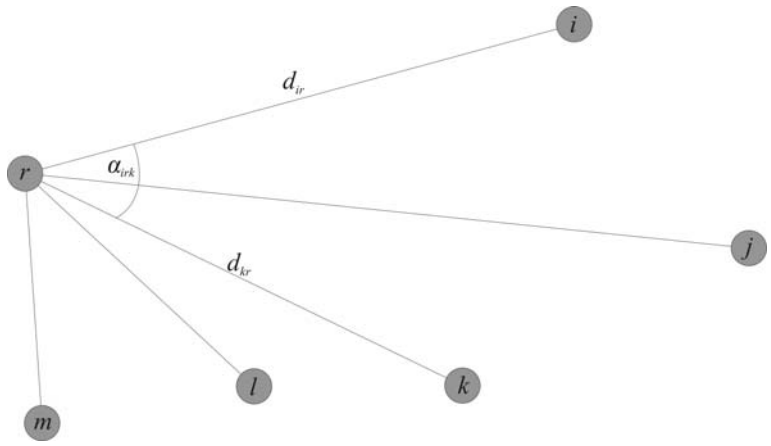
- rotáció invariancia
- transzláció invariancia
- a bőrelaszticitás okozta torzulások kezelése

A rotáció és transzláció invariancia tulajdonképpen azt jelenti, hogy az ujjlenyomat felismerését nem befolyásolhatja az ujjlenyomat elfordulása vagy elcsúszása. Ez érthető, hiszen két különböző mintavétel esetén szinte lehetetlen, hogy ugyanabban a helyzetben legyen az ujj. Az 1.6 fejezetben tárgyaltuk a bőrelaszticitás okozta torzulás problémáját. Ismét összefoglalva, ujjlenyomatvételkor, a különböző nyomáserősségeknek köszönhetően a bőr különböző mértékben torzul. Ez a minutiapontok egymáshoz viszonyított helyének módosulásához vezet. Az összehasonlítás során figyelembe kell vegyük ezt a torzulást.

A megoldás az első két követelmény esetében egy olyan ábrázolási mód megtalálása, mely nem használja a pozícióra vonatkozó információkat. A 4.2 fejezetben ismertetett teljes súlyozott gráf ezen követelményeket teljesíti. Azonban felmerül egy probléma: a részgráfok megtalálása komputacionális szempontból nagyon költséges.

Úgy döntöttünk, hogy egy kicsit módosítunk a leíráson. A továbbiakban a ponthalmazt nem a gráffal reprezentáljuk, hanem egy szimbólumsorozattal. Elsősorban a minutiahalmazból kiválasztunk egy úgynevezett referenciaminutiát. A ponthalmazt a referenciaminutiától mért távolságok, és a különböző pontokat a referenciaponttal összekötő szakaszok bezárta szögekkel írjuk le.

Az alábbiakban látható egy minutiahalmaz sematikus ábrája.



18. ábra: Minutiahalmaz és a képzésben használt jellemzők

Legyen *A* egy minutiahalmaz. Legyen *r* a halmaz egy kiválasztott eleme, a referenciaminutia.

A halmaz *i* pontját a referenciaminutiával összekötő szakasz hosszát jelölje d_{ir}.

A halmaz két, *i, j* pontját a referenciaminutiával összekötő szakaszok bezárta szöget jelölje α_{irj}.

A minutiahalmazt leíró *S* szimbólumsorozat minden eleme egy $s=(\alpha_{xry}, d_{xr}, d_{yr})$ alakú háromtagú szimbólum.

Legyen *A* egy minutiahalmaz, *r* egy referenciaminutia, ekkor az *S'* szimbólumsorozat képzése a következő:

$$S' = \left\{ s \middle| s = (\alpha_{irj}, d_{ir}, d_{jr}), i, j \in A, i \neq j, i \neq r, r \neq j \right\} \tag{4.1}$$

Ez még mindig nem egy tökéletes leírás, hiszen redundáns, az *i,j* és *j,i* leírása tulajdonképpen ugyanaz, ezért elegendő lenne, ha egyszer szerepelne a sorozatban, megegyezés szerint a hosszabb távolság szerepelne elsőként a szimbólumban. Ennek megfelelően tehát az *S* szimbólumsorozat képzését egy sematikus algoritmussal írjuk le:

1. $A = A / r$
2. *kiválaszt* $i \in A$
3. $A = A / i$
4. $S = S \cup \left\{ s \middle| s = (\alpha_{irj}, \max(d_{ir}, d_{jr}), \min(d_{ir}, d_{jr})), j \in A, i \neq j, r \neq j \right\}$
5. *amíg* $A \neq \phi$ *vissza a 2. lépéshez*

Ahol *r* a kiválasztott referenciaminutia.

A szimbólumsorozat elemeinek száma:

$$card(S) = \frac{V_{n-1}^2}{2} = \frac{(n-2)(n-3)}{2} \tag{4.2}$$

A majdani összehasonlítás megkönnyítése érdekében, a kapott sorozatot rendezzük mindhárom paraméter szerint növekvő sorrendben. A számítási igény csökkentése érdekében ajánlott egy gyorsrendező algoritmus használata.

Fölmerül egy probléma. Amikor két ponthalmazt akarunk összehasonlítani, lényeges, hogy a szimbólumsorozatok képzéséhez olyan referenciapontokat válasszunk, mely mindkét halmazban biztosan szerepel. Ennek eldöntése lehetetlen, ennélfogva kénytelenek vagyunk minden pontot referenciaminutiaként kezelni. Ez tulajdonképpen azt jelenti, hogy egy *n* elemű minutiahalmaz teljes leírását *n* darab szimbólumsorozattal adjuk meg, mindegyikben más-más a referenciaminutia.

4.4 INEGZAKT ILLESZTÉSI ALGORITMUS

Mint említettük, a feladat a két minutiahalmaz maximális illeszkedő részhalmazának a keresése. Tulajdonképpen az illeszkedés mértékét kellene számszerűen kifejeznünk. Hogy ezt mérhetővé tegyük, egy *illeszkedési pontszámot* határozunk meg, mely a minutiahalmazok és az illeszkedő részhalmazok számosságának a függvénye.

Az ábrázolás folytán, a 6. fejezetben említett problémák közül a rotáció és transzláció invarianciát kiküszöböltük, a bőrelaszticitás okozta torzulások kezelése az összehasonlító algoritmusra hárul. A legkézenfekvőbb megoldásnak az tűnik, ha két szimbólum összehasonlításakor engedélyezünk egy bizonyos küszöbérték alatti devianciát. Ennek figyelembevételével értelmezzük két szimbólum egyenlőségét. Legyen *a,b* két szimbólum, t_α a szögekre, t_d a távolságokra vonatkozó küszöbérték, ekkor:

$$a(\alpha_a, d_{a1}, d_{a2}) = b(\alpha_b, d_{b1}, d_{b2}) \Leftrightarrow (\alpha_b - t_\alpha) \le \alpha_a \le (\alpha_b - t_\alpha),$$
$$(d_{b1} - t_d) \le d_{a1} \le (d_{b1} - t_d), (d_{b2} - t_d) \le d_{a2} \le (d_{b2} - t_d) \tag{4.3}$$

Az összehasonlítási folyamat tulajdonképpen a két szimbólumsorozat párhuzamos bejárását jelenti, végeredményként az egyenlő szimbólumok számát kapjuk.

A következőkben ismertetjük a párhuzamos összehasonlítás menetét. Legyen *a* és *b* két szimbólumsorozat, szimbólumaik száma n_a valamint n_b. Jelölje $a(i)=(\alpha_{ai}, d_{a1i}, d_{a2i})$ az *a* szimbólumsorozat *i*. elemét, hasonlóan $b(i)=(\alpha_{bi}, d_{b1i}, d_{b2i})$. A feladat, minden $a(i)$, $i = \overline{1, n_a}$

elem megfelelőjének keresése a *b* sorozatban. Nem véletlenül rendeztük a sorozatot. Amikor *a(i)* elem megfelelőit keressük, elég a *b* azon részét vizsgálni, melyben a *b(j)* szimbólum α paramétere, a tolerancia figyelembevételével, egyenlőnek tekinthető *a(i)* elemével. Tulajdonképpen a *b* szimbólumsorozat egy növekvő részsorozatát keressük. Legyen a részsorozat legkisebb elemének indexe j_{inf}, a legnagyobbé j_{sup},

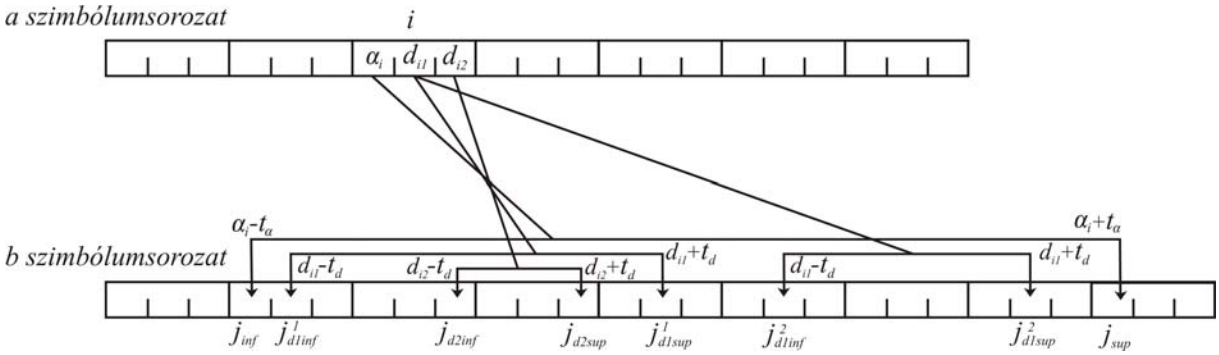
$$\alpha_{bj(\inf-1)} < \alpha_{ai} - t_\alpha \le \alpha_{bj\inf} \qquad (4.4)$$
$$\alpha_{bj(\sup-1)} < \alpha_{ai} + t_\alpha \le \alpha_{bj\sup}$$

Ha ennek megfelelően meghatároztuk a j_{inf} , j_{sup} értékét, akkor a kapott *b*' részsorozatba a *b* azon elemei kerülnek, melyek első paramétere megfelel az *a(i)* első paraméterének.

A kapott részsorozatot tovább szűkíthetjük. Mint tudjuk, a szimbólumsorozat másodlagosan rendezve van a második, valamint harmadlagosan a harmadik paraméter szerint. Tehát a *b*' sorozatban található elemek, melyeknek első paramétere azonos, a második szerint növekvően vannak rendezve. Azaz a *b*' sorozatot tovább bonthatjuk *k* darab, a d_1 szerint növekvő részsorozatra, melyeket a továbbiakban b^k-val jelölünk. Minden b^k esetében meghatározhatjuk a j_{d1inf}, j_{d1sup} értékét, a j_{inf} –hoz hasonlóan, azaz:

$$d_{b^k 1(j\inf-1)} < d_{a1i} - t_d \le d_{b^k 1 j\inf} \quad , \quad d_{b^k 1(j\sup-1)} < d_{a1i} - t_d \le d_{b^k 1 j\sup} \qquad (4.5)$$

A mellékelt ábrán igyekeztünk szemléltetni a felosztást.



19. ábra: A tolerancia-intervallumok és részsorozatok meghatározása a szimbólumsorozatokban

A következőkben tovább bontjuk a részsorozatokat, ha egyáltalán léteznek, a harmadik paraméter, a d_2 szerint, az előzőekkel megegyező módon.

Ha létezik legalább egy ilyen harmadszintű részsorozat, az azt jelenti, hogy megtaláltuk az *a(i)* – nek megfelelő elmet, és leállítjuk a keresést. Ekkor továbbléptetjük az *i*-t, és hozzáigazítjuk a j_{inf} és j_{sup} értékét. Tehát tulajdonképpen csak előrelépés történik, és egyszerre léptetjük az *i* változót valamint a toleranciaintervallumot (innen a párhuzamos elnevezés).

Megjegyezzük, hogy valós esetekben nem túl gyakran fordul elő, hogy több másodszintű sorozatot kapjunk, tehát az algoritmus nem annyira sokágú, mint a fentebb ismertetett "legrosszabb-eset forgatókönyv" esetében.

Az összehasonlítást, a két minutiahalmazból származtatott szimbólumsorozatok minden lehetséges párosítására el kell végeznünk, azaz ha a két halmaz számossága *n1* ill. *n2*, a szükséges összehasonlítások száma: *n1*n2*. Ez még egy kis magyarázatra szorul. A 4.3 fejezetben bemutattuk a minutiák reprezentációját. Tehát a minutiahalmazt egy szimbólumsorozat – halmaz segítségével írtuk le. Minden minutiát egyszer és csakis egyszer referenciaminutiának választottunk, és hozzárendeltünk egy szimbólumsorozatot. Ha két minutiahalmazt akarunk összehasonlítani, mindkettő leíró halmazából egy olyan szimbólumsorozatot kell kiválasztani, melyek generálásakor ugyanazt a minutiát használtuk referenciaminutiaként. Mivel a minutiák nem sorszámozhatók, nem tudhatjuk, mikor bukkantunk egy ilyen párosításra, ezért minden lehetséges párosítás esetén el kell végeznünk az összehasonlítást, kiszámoljuk az illeszkedési pontszámot (lásd 4.5). Az így kapott illeszkedési pontszámok maximumát tekintjük az összehasonlító algoritmus végeredményének.

4.5 AZ ILLESZKEDÉSI PONTSZÁM

Az ujjlenyomatok hasonlóságát *illeszkedési pontszám* segítségével határozzuk meg. Az illeszkedési pontszám a két összehasonlított szimbólumsorozat hosszának és az egyenlő szimbólumok számának a függvénye. Legyen n_a , n_b a két szimbólumsorozat hossza, n_j az inegzakt illesztési algoritmus által meghatározott illeszkedő szimbólumok száma. Ekkor a *p* illeszkedési pontszám:

$$p = \frac{100 n_j}{(n_a + n_b)/2} \tag{4.6}$$

Tökéletes illeszkedés esetén értéke 100, abszolút különbözőség esetén pedig 0. A gyakorlatban, két nem azonos ujjlenyomat összehasonlítása általában 0 és 25 közötti értéket ad. Két ujjlenyomatot akkor nevezünk azonosnak, ha az illeszkedési pontszám legalább 50 fölött van.

4.6 AZ ALGORITMUS GYORSÍTÁSA

A fentiekben leírt összehasonlítási algoritmus számítási igénye igen nagy, de bizonyos módszerekkel növelhetjük a hatékonyságot, és csökkenthetjük a számítási igényt.

Visszatérnék a 4.4 fejezetben elmondottakra. Már megemlítettük, hogy nem minden esetben szükséges a másodszintű részsorozat megállapítása. Láthattuk, hogy az elsőszintű részsorozat olyan elemekből áll, melyek első szimbóluma valamilyen tolerancia-intervallumban van, ezek több értéket vehetnek fel. Az elsőszintű részsorozatot olyan szekvenciákra oszthatjuk, melyek szimbólumainak első eleme, az α azonos. Ezek a szekvenciák a második paraméter szerint növekvő sorrendbe vannak rendezve és ezek képezik a másodszintű részsorozatok alapját, azaz minden másodszintű részsorozat egy-egy ilyen módon meghatározott szekvencia részhalmaza. Nevezzünk egy ilyen szekvenciát t -nek. Nagy a valószínűsége annak, hogy nem létezik a másodszintű részsorozat. Ha megvizsgáljuk a t sorozat első és utolsó elemét, amennyiben az első elem értéke nagyobb, mint a $d_{1ai}+t_d$, illetőleg az utolsó kisebb mint $d_{1ai} - t_d$, a másodszintű részsorozat keresését elvethetjük. Ha mégis létezne, ugyanezzel a módszerrel megvizsgáljuk, létezik e a harmadszintű.

Másodsorban láthattuk, hogy csupán azokat az eseteket fogadjuk el, melyekben az illeszkedési pontszám nagyobb mint 50. Ez nagyjából azt jelenti, hogy a minutiák 50%-a azonos kell legyen. Tehát amennyiben az összehasonlítás során kiderül, hogy ez a pontszám már semmiképp sem érhető el, megszakítjuk az algoritmust.

A 4.4 fejezetben elmondtuk, hogy a két minutiahalmazból generált összes szimbólumsorozat valamennyi párosítására el kell végeznünk az összehasonlítást, hiszen nem tudjuk, melyik a referenciaminutia. Ezt a kijelentést most egy kicsit módosítjuk. Mint említettük, feltételként szabtuk meg, hogy legalább a minutiapontok fele szerepeljen mindkét halmazban. Ez azt jelenti, hogy ha a két halmaz teljesíti ezt a feltételt, akkor a szimbólum-sorozatok halmazának első felében már meg kellett találnunk egy olyan párosítást, melyben azonos a két referenciaminutia. Tehát elegendő az egyik szimbólumsorozat-halmaz elemeinek a felét összehasonlítani a másik halmaz minden elemével.

Mint láttuk, a két szimbólumsorozat összehasonlításakor az első szimbólumsorozat bejárása lényegesen kevesebb lépést igényel, mint a másodiké, tehát érdemes a rövidebb sorozatot elsőként szerepeltetni, és ugyancsak érdemes a kisebb számosságú halmazból képzett sorozatokat felezni.

FELHASZNÁLT KÖNYVÉSZET

1. http://onin.com/fp/fphistory
2. http://www.reachoutmichigan.org/funexperiments/agesubject/lessons/prints_ext
3. http://brazoria-county.com/sheriff/id/fingerprints
4. James L. Wayman. *NATIONAL BIOMETRIC TEST CENTER COLLECTED WORKS 1997-2000*. San Hosé State University 2000
5. Anil Jain, Sharath Pankanti: *Fingerprint Classification and Matching*. MSU-CPS 1999
6. Paul Collier, Fernando Podio. *Common Biometric Exchange File Format*. NIST/ITL 1999.
7. Kovács-Vajna Miklós Zsolt. *A Fingerprint Verification System Based on Triangular Matching and Dynamic Time Warping*. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 2000.
8. Dario Maio, Davide Maltoni. *Direct gray-scale minutiae detection in fingerprints.* IEEE Transactions on Pattern Analysis and Machine Intelligence 19, 1997.
9. Shlomo Greenberg, Mayer Aladjem, Daniel Kogan. *Fingerprint Image Enhancement using Filtering Techniques*. Real-Time Imaging 8, 2002.
10. Dario Maio, Davide Maltoni. *Inexact Graph Matching for Fingerprint Classification.* Machine GRAPHICS & VISION Special Issue on Graph Trasformations in Pattern Generation and CAD 8, 1999.
11. Bing Jian. *Fingerprint Minutiae Matching Based on Relationship Examination*
12. Lin Hong ,Anil Jain. F*ingerprint image enhancement: algorithm and performence evaluation*. IEEE Transaction on Pattern Analysis and Machine Intelligence 20, 1998.
13. Lin Hong, Anil Jain, Sharathcha Pankanti, Ruud Bolle. *Fingerprint Enhancement.* IEEE WACV, Sarasota, Florida, 1996
14. Jianwei Yang, Lifeng Liu, Tianzi Jiang, Yong Fan. *A modified Gaborfilter design method for fingerprint image enhancement*. Elsevier Science 2003.
15. D. Dumitrescu, Hariton Costin. *Retele Neuronale*. Teora, 1996
16. Simon Haykin. *Neural Networks*. Prentice Hall, 1999.
17. Sergiu Nedevschi. *Prelucrarea imaginilor si recunoaşterea formelor.* Grupul Microinformatica, 1998.
18. Salil Prabhakar, Anil K. Jain, Jianguo Wang, Sharath Pankanti, Ruud Bolle. *Minutia Verification and Classification for Fingerprint Matching.*

TARTALOMJEGYZÉK