

**XXXI. Országos Tudományos Diákköri Konferencia
Had- és Rendészettudományi Szekció
Budapest, 2013. április 16-18.
Nemzeti Köszolgálati Egyetem
Hadtudományi és Honvédtisztképző Kar**

Többszintű Tenyéralapú Biometrikus Azonosító Rendszer

Jelige: *Rókalyuk*

Dolgozat lezárásának időpontja: 2011, november

Tartalomjegyzék

1 Bevezetés	4
1.1 Biometrikus azonosítás	4
1.2 Célkitűzés.....	5
2 Irodalomkutatás	6
2.1 Fővonal alapú egyedi azonosító.....	8
2.2 Minutia alapú egyedi azonosító	9
2.2.1 A minutia pontok meghatározásának algoritmus.....	10
2.3 Véna alapú egyedi azonosító	12
2.4 Kontúr alapú egyedi azonosító.....	13
3 A végső rendszer tervezett működése, felépítése	14
3.1 Aktorok a rendszerben	14
3.2 A rendszer viselkedése.....	14
3.2.1 A felhasználók felvitele	15
3.2.2 A felhasználók hitelesítése.....	16
3.3 A rendszer állapotai	17
3.3.1 A tenyérképek kiértékelése	17
3.3.2 A felhasználók felvitele	18
4 A saját rendszerünk.....	19
4.1 Rendszerünk globális felépítése.....	19
4.2 Input alrendszer.....	20
4.2.1 Input eszköz	20
4.2.2 Színterek	21
4.2.3 Binarizálás	26
4.2.4 Egy komponens kialakítása	28
4.2.5 Kontúr bejárás.....	30
4.2.6 Kontúrfüggvény előállítása.....	31
4.3 Képfeldolgozó alrendszer	32
4.3.1 Referencia pontok meghatározása	32
4.3.2 Geometriai jellemzőn alapuló azonosítás	33
4.3.3 A ROI extraktum elkészítése	34
4.3.4 Főkomponens analízis (PCA)	35

4.4 Adatbázis alrendszer	39
4.4.1 Entity Framework	39
4.4.2 Az adatbázis felépítése.....	41
4.4.3 Saját tenyéradatbázis készítése	42
4.5 Összehasonlító alrendszer	43
4.5.1 Összehasonlító modul input adatai	43
4.5.2 Összehasonlítási módszerek	43
4.5.3 A döntési folyamat.....	44
4.6 Adminisztratív alrendszer, GUI, felhasználói útmutató	45
4.6.1 Funkció tesztelő	45
4.6.2 A fő ablak.....	46
4.6.3 Statisztikai ablak	49
4.7 A rendszer osztályai	52
4.7.1 Preprocess modul	52
4.7.2 Feature modul	53
4.7.3 Főmodul	54
5 Teszteredmények	56
5.1 Biometrikus rendszerek teljesítmény mutatói.....	56
5.2 Kézeometriai keresés első "Open-set" eredményei	57
5.3 PCA módszer első "Open-set" eredményei	59
5.4 Binarizáció és jellemzővektor módosításának eredménye	60
5.4.1 Kézeometriai statisztikák	60
5.4.2 PCA statisztikák.....	63
5.5 Kombinált teszteredmények	67
5.5.1 Optimalizálás	68
5.5.2 Feltételezett optimális beállítás teszteredménye.....	68
6 Összefoglalás	71
6.1 Fejlesztés, megvalósítás.....	71
6.1.1 Input eszköz	71
6.1.2 Szoftveres hibaszűrés.....	72
6.1.3 Sebesség.....	72
6.1.4 Egyéb rendszertulajdonságok	72

6.2 Abstract.....	73
7 Irodalomjegyzék	75
8 Melléklet	77
8.1 Az adatbázis montázsa.....	77

1 Bevezetés

1.1 Biometrikus azonosítás

Mai világunkban működő informatikai rendszerek nagy részében jelen van valamilyen felhasználói azonosítás. Ennek célja, hogy csak az tudja igénybe venni a szolgáltatást, vagy a rendszert, aki megfelelő jogokkal rendelkezik. Felhasználói azonosítást, jelszót, vagy valamilyen egyéb biztonsági kulcsot használnak például banki rendszereknél (ATM), magas biztonsági fokon őrzött épületbe való belépéskor, laptopok esetében és sok más eszközben is. Felmerülhet az a probléma, hogy egy esetleges csaló, aki a belépési azonosítónkat megszerzi (chip kártya, lemez), könnyedén vissza tud azzal élni. A biometrikus azonosítás előnye az, hogy egy olyan megoldást nyújt, amellyel ezeknek a biztonsági rendszereknek a hatásfoka sokszorosára növelhető, mivel az emberi test egyedi sajátosságaiból képezi le az azonosításhoz szükséges információt, így gyakorlatilag hamisíthatatlan. Továbbá leegyszerűsíti azt a kérdést, hogy a kulcsot, ami szükséges a belépéshez, vagy a szolgáltatás igénybevételéhez hol tároljuk, mivel a testi sajátosságainkat nem tudjuk elhagyni, vagy elfelejteni, mint egy PIN kódot.

Egy francia rendőr Alphonse Bertillon [1] volt az első, aki testi jellemzők összessége alapján próbált azonosítani, előtte csak a szemtanúk leírásaira hagyatkoztak. Kutatásai során az emberekről statisztikai adatokat gyűjtött, hogy meg tudja állapítani a test egyedi sajátosságait. A 20. század elején kezdett elterjedni a bűnözők azonosítása az ujjlenyomatuk alapján.

Az emberi test tulajdonságait akkor lehet azonosításra használni, ha a következő tulajdonságokkal rendelkezik:

- egyediség: mindenkinél egyedi,
- különbözőség: mindenkinél különböző,
- permanens: idő múlásával változatlan,
- mérhető: digitálisan mérhető,
- teljesítmény: az azonosításhoz szükséges idő megfelelően alacsony,
- elfogadhatóság: az emberek elfogadják-e a mintavételt,
- megbízhatóság: különféle módszerekkel kikerülhető-e a rendszer (hamisítás). [2]

1.2 Célkitűzés

A project fő célja, hogy létrehozzunk egy olyan rendszert, amely biometria jellemzőket alapul véve képes az azonosításra, vagyis a felismerés részre kerül a hangsúly, nem egy valós biztonsági rendszer kialakítására.

Választásunk a tenyérre esett, melynek első sorban azaz oka, hogy gyakorlati megvalósítása nem annyira elterjedt, másrészt mert számos lehetséges módszer rejlik benne.

Az azonosítás hatékonysága logikusan nőni fog, ha több, eltérő jellemző vizsgálatát végezzük, nem beszélve arról, hogy ezt egyetlen szenzor használatával tehetjük meg. Így az általunk megtervezni kívánt rendszer röviden a következő tulajdonságokkal szeretnénk, hogy bírjon:

Egy egyszerű asztali számítógéphez vagy laptophoz csatlakoztatható, lehetőleg olcsó eszközt használjon és az előbb említett elveknek megfelelően vegyen figyelembe több egyedi jellemzőt is, vagyis egy többszintű tenyéralapú azonosító rendszer létrehozása a feladatunk.

2 Irodalomkutatás

Ebben a fejezetben néhány alapvető információ található kéz- és tenyéralapú azonosítás témakörében. Ehhez kapcsolódó módszereket, eszközöket, algoritmusokat ismertetünk, melyekből számos elemet vettünk alapul saját rendszerünknel. Ezek a többnyire elméleti ismeretek segítették, hogy tudjuk milyen irányba érdemes elindulni a kutatásokkal és fejlesztésekkel.

Alapvetően ezeket a módszereket három eltérő csoportba lehet sorolni, melyek technikai (hardver) igényükben és a felhasznált képfeldolgozási elemekben is jelentősen eltérnek. Ez a három csoport a következő:

- Geometriai jellemző: szélesség, hosszúság, körvonal.
- Vonal jellemző: fővonalak, esetleg véna.
- Pont jellemző: minutia pontok.

Nem annyira elterjedtek a teljes kéz alapú rendszerek, mint a szem vagy ujjlenyomat alapúak, ennek ellenére létező és megbízható technológiáról van szó. Japánban kialakítottak olyan ATM hálózatot, melyben a tenyérerezet alapú biztonsági ellenőrzés része a rendszernek (1. ábra) [3].



1. ábra. Japán ATM tenyérerezet alapú azonosítással

A HandPunch széria egy ma is piacon lévő eszköz (egy biometrikus időzár), mely szabadon megvásárolható (2. ábra) [4].



2. ábra. Piacon kapható időzárás beléptető rendszer (HandPunch)

Folynak fejlesztések számítógéphez és laptophoz készülő eszközökkel kapcsolatban is (3. ábra) [5].



3. ábra. Laptophoz készült verifikációs eszköz

Ezeknek a rendszereknek a részletekbe menő működéséről nincs sok információ, de a szakirodalmakból megismerkedhetünk az induláshoz szükséges elméleti háttérrel. A fejezet további részeiben néhány, ezekben fellelhető algoritmusról lesz szó.

2.1 Fővonal alapú egyedi azonosító

A tenyéren lévő fővonalak hosszú ideig állandóak az embereknél, így azt hatékonyan fel lehet használni [6]. Végződés keresése egy viszonylag egyszerű és gyors művelet, így az ezeket kereső algoritmusok használata előnyös lehet. A 4. ábrán láthatunk egy példát.



4. ábra. Végpontok egyezése két mintánál (szürke sávok)

Nem szükséges a vizsgált pontoknak az egymáshoz való kötődését kiszámítani, elegendő csupán ezeknek a jellemző pontoknak a pozíciójukat és az orientációjukat vizsgálni a hozzájuk rendelt fővonalakhoz képest.

A pontok kinyerésének algoritmus a következő:

- A kép simítása az eredeti pixel és négy közvetlen szomszédos pixel átlaga alapján. A simítást a zajszűrés és a nagyon vékony vonalak figyelmen kívül hagyása miatt kell elvégezni.
- A simított kép binarizálása egy választott küszöbölési technikával a háttér és a vonalak megkülönböztetésére.
- Szűrések elvégzése, hogy eltávolítsuk a tévesen vonalaknak hitt egybefüggő régiókat. A maradék háttérpixelék között lesznek a számunkra releváns pontok.
- A pontok hozzárendelése a megfelelő fővonalhoz.
- A pontok orientációjának kiszámítása a hozzárendelt fővonalakhoz képest.

- Mivel a hamis tulajdonságú pontok még jelen vannak, ezért eltávolítjuk azokat az előző lépésben meghatározott pontokat, amelyeknek a legalacsonyabb az intenzitás értékük (30%-os küszöb alapján).

Az azonosítás során szükség van elforgatásra és eltolásokra, ami után a hasonlóságot az összerendelt pontok és fővonalak irányából és pozíciójából következtetjük.

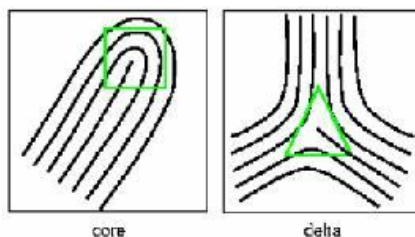
2.2 Minutia alapú egyedi azonosító

Az ujjlenyomat alapú azonosításhoz hasonló módszer a minutia alapú. A tenyérynymaton található fodorszálak nagyon hasonlítanak az ujjlenyomaton található fodorszálakhoz és a módszer is lényegében azonos az ott alkalmazottakkal. [7]

A fodorszálak végét, elágazását illetve összefutásának helyét minutia pontoknak nevezzük. Ezen pontok elhelyezkedése rendkívül jellemző az adott ujjfelületre (illetve tenyérre) nézve. Több azonosítási módszer kizárólag e pontok relatív elhelyezkedéséből dönti el két minta azonosságát vagy különbözőségét.

Minutiák típusai (példák az 5. ábrán):

- Végződés (Ending): a barázda az adott ponton véget ér.
- Elágazás (Bifurcation): ahol a barázda kettő barázdára ágazik szét.
- Sziget (Island): a rövid barázda.
- Pont (Dot): hasonló a szigethez, de pontszerűen rövid.
- Híd (Bridge): átszögellés két barázda között.
- Kiszögellés (Spur): a másik barázdát nem érintő kis nyúlvány ágazik el a barázdából.
- Körülzárás (Enclosure): kis távolságra kettéválik a barázda, majd megint egyesül.
- Delta (Delta): elágazás után folytatódik az eredeti barázda.
- Dupla elágazás (Double bifurcation): egymás után két elágazás.
- Hármás elágazás (Trifurcation): háromfelé ágazik el a barázda.



5. ábra. Fő minutia típusok

Az első két minutiából, az elágazásból és a végződésből, ki lehet alakítani az összes további minutia típust, valamint ezek önálló előfordulása a leggyakoribb, ezért általában az azonosító rendszerek ezt a két típust keresik. Ha fixen a tenyéren található delta pozíciók (6. ábra) mentén vizsgáljuk a minutiákat, tovább javíthatjuk a technika hatékonyságát és sebességét.



6. ábra. Delták elhelyezkedése a tenyéren

2.2.1 A minutia pontok meghatározásának algoritmusa

Orientáció meghatározása:

- Meg kell határozni a szálak orientációját, másképpen fogalmazva a szálak elhelyezkedését a képen, ezt egy választott szegmentációs technikával el lehet érni.

Szálak meghatározása:

- Két $W \times H$ nagyságú konvolúciós maszk használata, amelyekkel meghatározzuk a szálak irányát.
- Ha a pixel intenzitás nagyobb, mint egy bizonyos küszöb, akkor az szál.

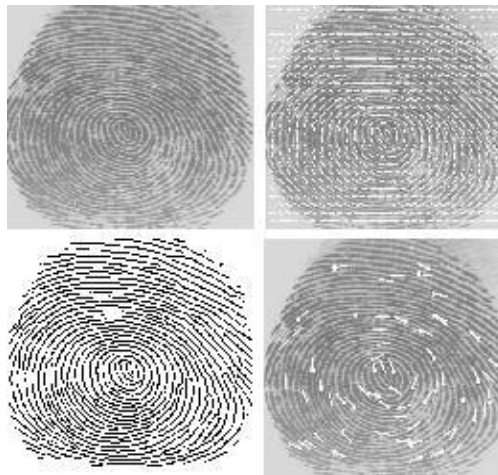
- Egy szálvékonyító algoritmus után megkapjuk a száltérképet.

Minutia pontok detektálása:

- Elágazások és végződések meghatározása.
- Hamis minutia pontok kiszűrése.
- Szakadások szűrése, szálak összekötése.
- Vékony „tüskék” (zajok) szűrése küszöböléssel.

Minutia pontok jellemzőinek eltárolása:

- X, Y koordináta.
- Lokális szál orientáció.
- Hozzárendelt szál.
- Két minta összehasonlítása során először forgatni kell a bemeneti képet a már eltároltra, utána átalakítani polárkoordinátákkal, majd összehasonlítani a kapott sztringeket. Az algoritmus szemléltetése a 7. ábrán látható. [8]



7. ábra. Minutia meghatározás algoritmusának szemléltetése

2.3 Véna alapú egyedi azonosító

A véna alapú azonosítás során egy infra alapú szkennerek segítségével készítenek felvételt a kézről, melyen az erek láthatóvá válnak (8. ábra), mivel az infra fény kimutatja a vérben lévő hemoglobint.

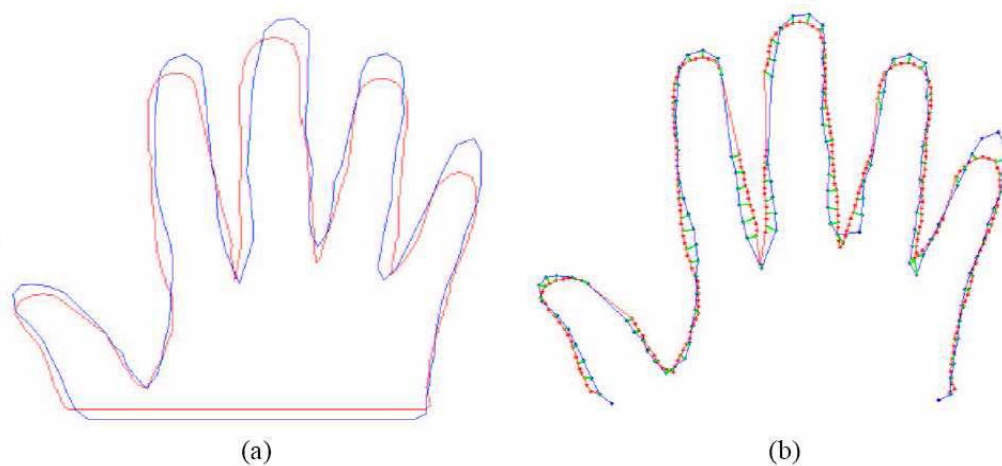


8. ábra. Tenyér vénái infraszkennerrel készült felvételen

A képen fehér színnel látható a tenyér és sötétebb színnel az erek hálózata. Hasonló technikával működik a retina alapú azonosítás is. A vénák által kirajzolt hálózat mindenkinél egyedi és időtálló, ahogy ezt a biometrikus azonosítás elvárja. A nehézség benne az, hogy a vénák mozognak, ahogy a szív pumpálja a vért az erekben, de ezt a megfelelő képfeldolgozási algoritmusokkal megoldják, hiszen léteznek ilyen rendszerek. Előnyös a technológia használata, mivel nem kell kontaktust létesíteni a szkennerekkel, mely eszköz egyébként olcsó és egyszerű. Hátránya természetesen az, ami a hasonló módszereknél, vagyis az érzékenység a kéz sérüléseire. [9]

2.4 Kontúr alapú egyedi azonosító

A kontúr alapú azonosítás a nevéből adódóan a kéz körvonalát veszi alapul. Mivel nem lehet két teljesen megegyező képet csinálni, ezért a keletkező deformációkat, torzulásokat kezelniük kell, és kéztartástól függetlenül jól felhasználhatóvá kell tenni. Ezt úgy tesszük meg, hogy a kéz kontúrját transzformációkkal (pl. hasonlósági, lineáris) módosítjuk, általában az ujjakra külön-külön elvégezve. Addig végzünk transzformációt, amíg a letárolt képhez képest a négyzetes eltérés minimális nem lesz (9. ábra). Előfordul, hogy ezt használják előfeldolgozásaként a geometriai sajátosságokat figyelembe vevő azonosításhoz is. [10]

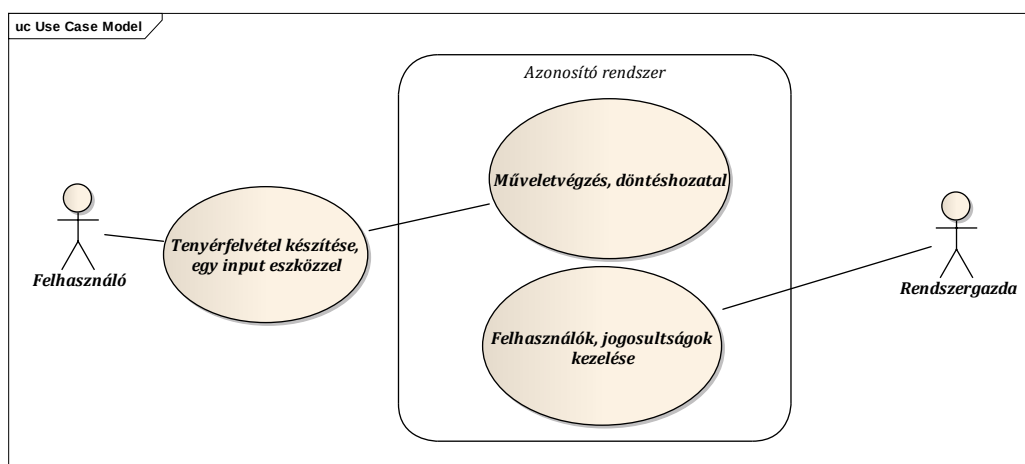


9. ábra. Kontúrillesztésre példa

3 A végső rendszer tervezett működése, felépítése

3.1 Aktorok a rendszerben

A rendszert két típusú aktor fogja irányítani, használni. Az egyik a felhasználó, akinek a tenyérfelvételét az alkalmazás kiértékeli, majd eldönti, hogy a vizsgált személy (a felhasználó) jogosult-e a rendszer által levédett, felügyelt eszköz használatára vagy belépésre az épületbe. A másik aktor a rendszergazda, aki kezelni fogja a rendszerben tárolt felhasználók jogosultságait, továbbá ő viszi fel az adatbázisba a felhasználókról eltárolt mintákat amivel a későbbiek folyamán azonosítjuk a felhasználókat. Az azonosító rendszer feladata az input eszközről kapott kép feldolgozása, kiértékelése és a döntés meghozatala a felhasználó kiléte felől. A rendszer és a felhasználók közötti kapcsolatok elvi felépítése a 10. ábrán látható.



10. ábra. Használati eset diagram

3.2 A rendszer viselkedése

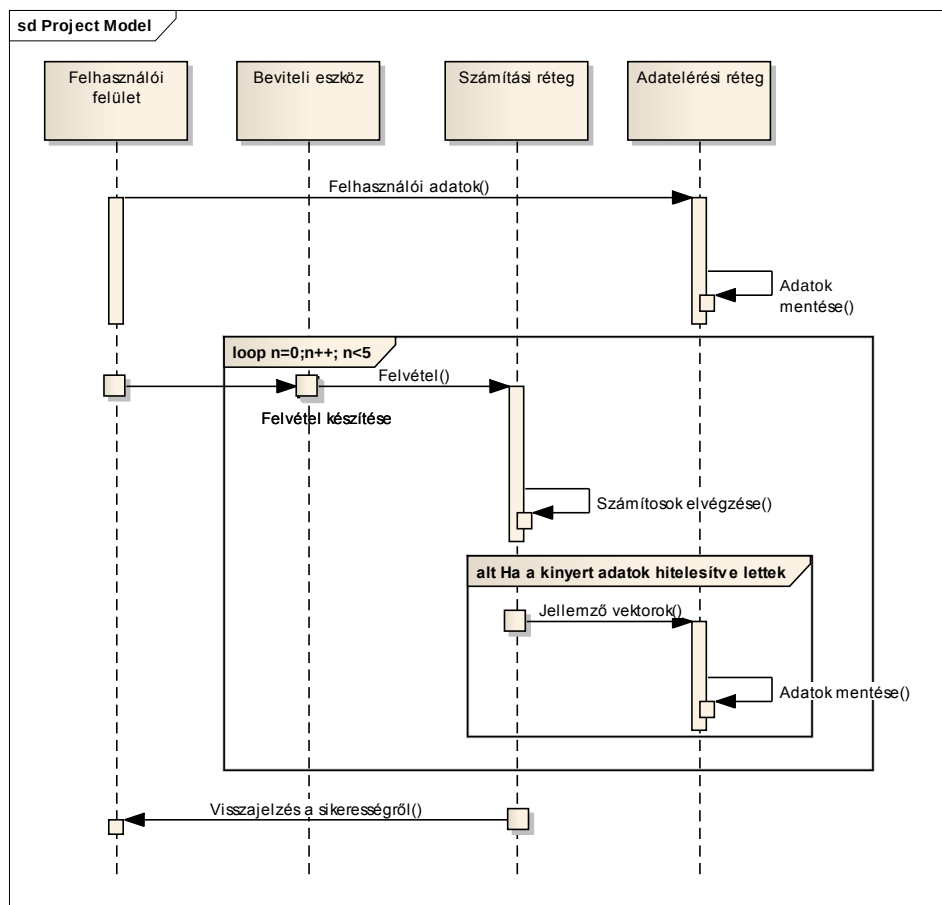
A rendszer viselkedését szekvencia diagramokkal írjuk le. A szekvencia diagramon az egyes rétegek közti kommunikációt, üzenetküldést, és belső folyamatokat modellezünk az idő függvényében. Az egyes rétegetek idővonalain a számítások, és a réteg műveleteinek, szekvenciájának az időtartama látható.

3.2.1 A felhasználók felvitele

A felhasználói felülettel való interakció során felvisszük az új felhasználó adatait, melyeket a validálás után az adatelérési rétegnek elküldjük egy üzenettel, mely az adatbázissal való kommunikációt követően elmenti azokat. Ezután egy újabb felhasználói interakcióval egy ciklusba lépünk.

A ciklus belsejében, az új felhasználó biometriai adatainak kiszámítása történik. Ehhez az kell, hogy a beviteli eszközről üzenetként kapott képet a számítási réteg feldolgozza. Ha a feldolgozás sikeres, és a kinyert adatok validálva lettek, a számítási réteg kommunikációba lép az adatelérési réteggel és menti az egyes felvételek adatait. A számítási réteg a folyamat befejeztével visszajelzést küld a felhasználói felületnek

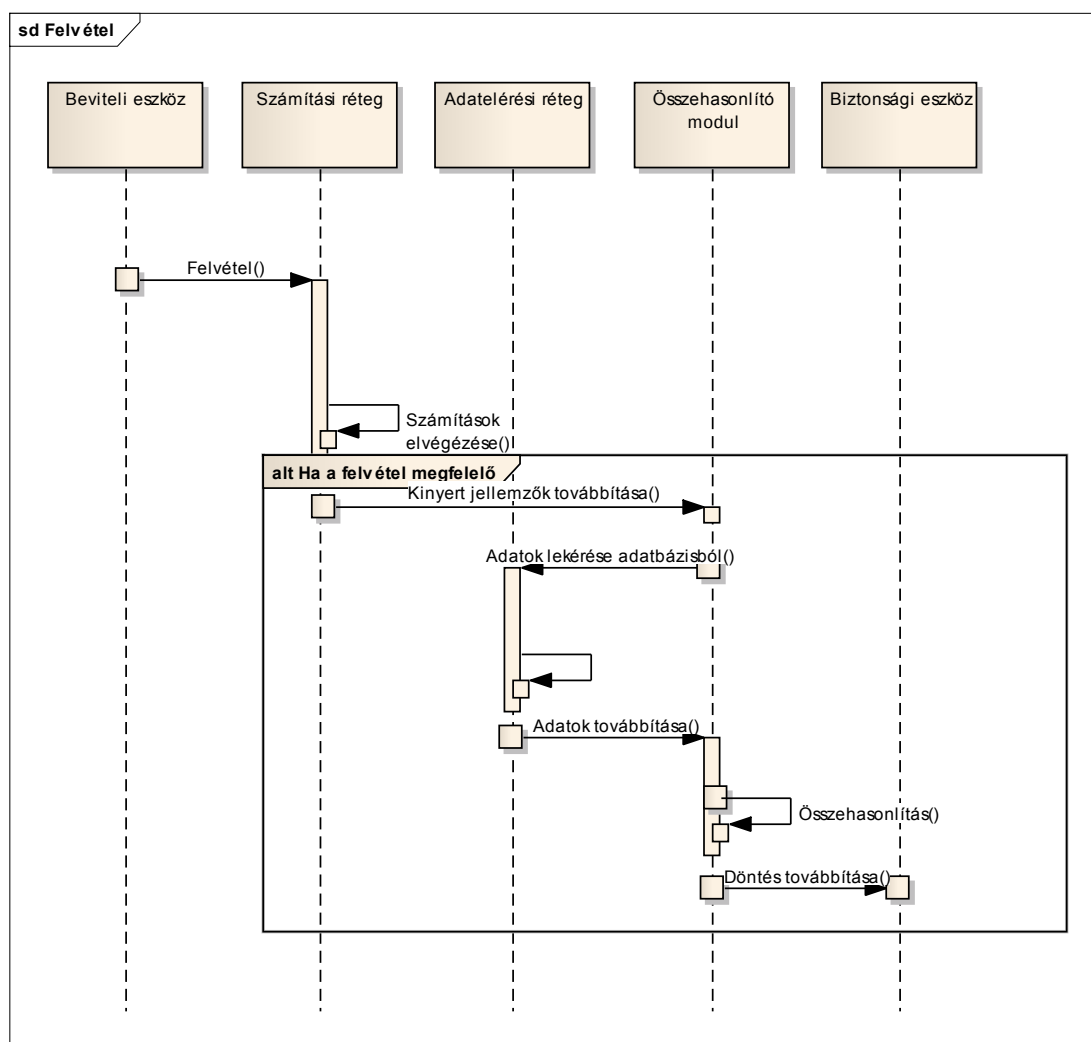
A felhasználók felvitelének szekvencia diagramja a 11. ábrán látható.



11. ábra. Felhasználó felvitelének szekvencia diagramja

3.2.2 A felhasználók hitelesítése

A felhasználók hitelesítése során, a felhasználó interakcióba lép a beviteli eszközzel, ami elküldi a tényerről készült felvételt a számítási rétegnek. A számítási rétegben a tényér jellemzőit számítjuk ki a kapott felvételből, és amennyiben az adatok megfelelőek, a kinyert jellemzőket az összehasonlító modulnak továbbítjuk. Az összehasonlító modul az adatelérési rétegtől elkéri a szükséges adathalmazt, amelynek segítségével az összehasonlítás történik. Ezután a kapott halmazt és az új felvételt összehasonlítja, majd a döntéshozó az eredmény kielemezésével továbbítja a döntést a biztonsági eszközöknek. Ennek szekvencia diagramját 12. ábra mutatja.



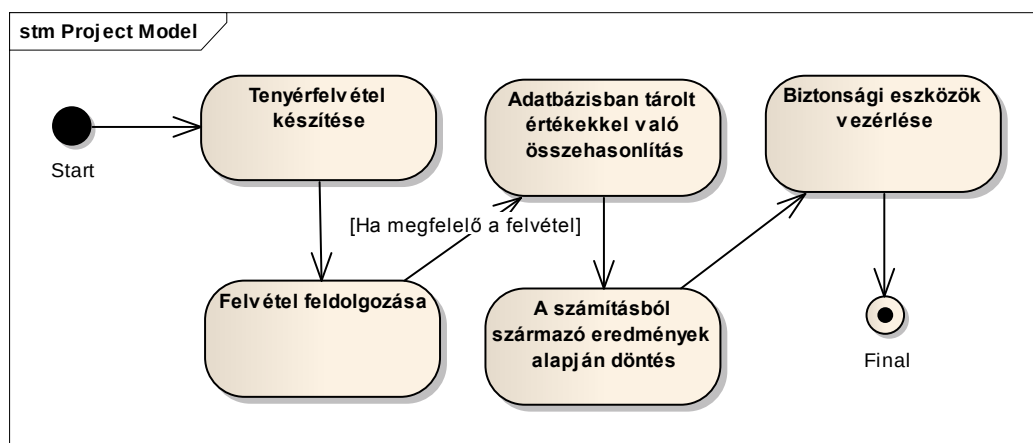
12. ábra. Felhasználó hitelesítésének szekvencia diagramja

3.3 A rendszer állapotai

A rendszer állapotait állapotdiagramokkal írjuk le. Az állapotváltozást, és a kapcsolatokat a nyilak jelzik.

3.3.1 A tényérképek kiértékelése

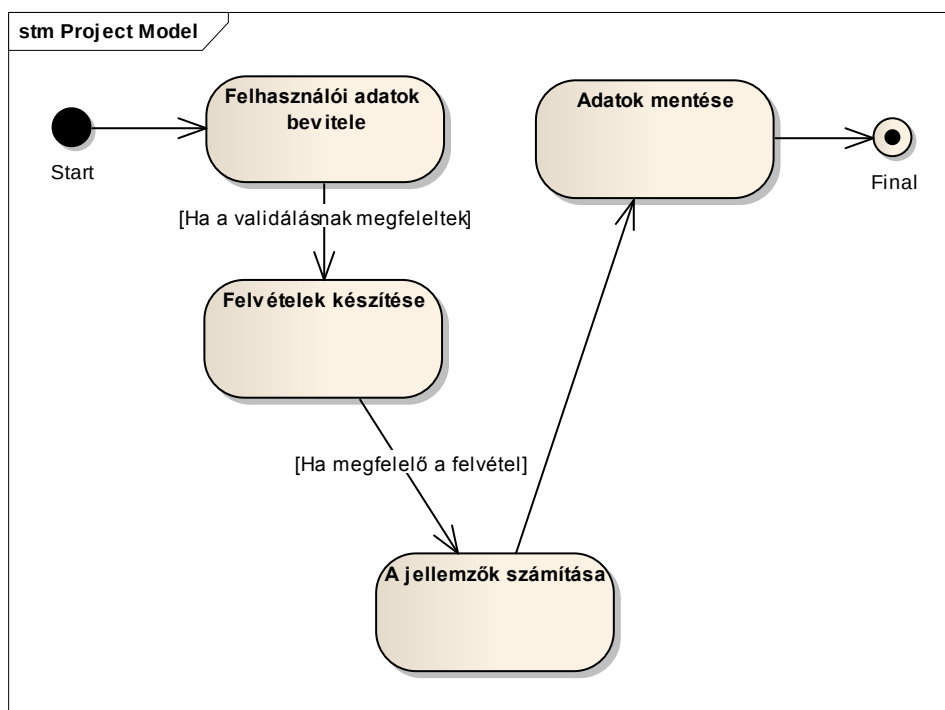
A rendszer a start állapotban vár a felhasználói interakcióra. Ezután az input eszköz elkészíti a felvételt, amit a következő lépésben feldolgoz, és ha az eredmény megfelelő, akkor az összehasonlítási folyamat során kiszámolja az adatbázisban eltárolt értékekhez viszonyított eltéréseket, majd ezt követően a rendszer döntést hoz. A döntéshozatal után, a biztonsági eszközök vezérlése történik, amivel elértünk a végállapotba. A folyamat állapotdiagramját mutatja a 13. ábra.



13. ábra. Egy tényérkép kiértékelésének állapotdiagramja

3.3.2 A felhasználók felvitele

A start állapotban a rendszer interakcióra vár. Miután felvittük a felhasználó adatait, melyek megfeleltek a validációnak, akkor felvételeket kell készíteni, és ha ezekben megtalálhatóak a számunkra releváns információk, akkor a jellemző adatok kiszámítása után, az adatok mentésével létrehoztunk egy új felhasználót, és a rendszer végállapotába lép. Az állapotdiagram a 14. ábrán látható.



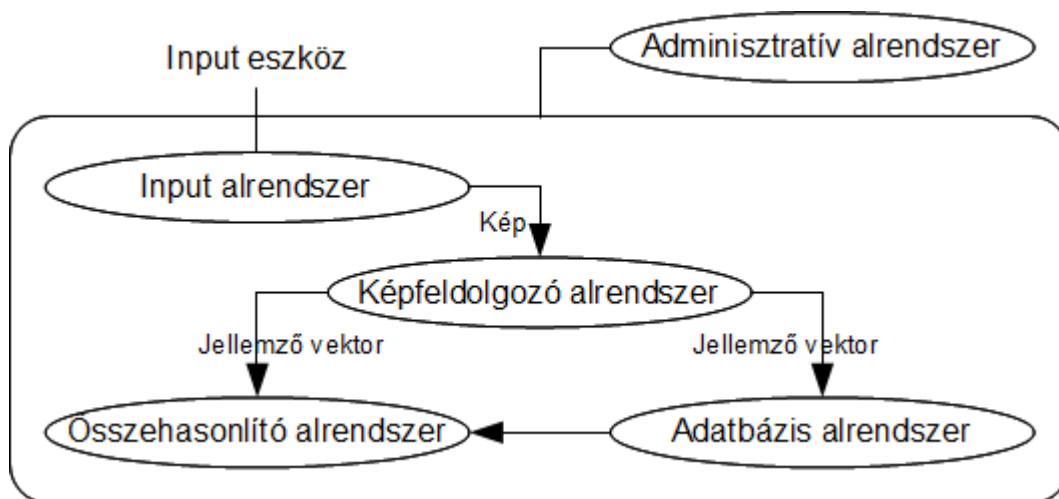
14. ábra. Egy felhasználó felvitelének állapotdiagramja

4 A saját rendszerünk

Ebben a fejezetben saját rendszerünket mutatjuk be részletesen beleértve a megvalósított rendszer globális felépítést, az alrendszereket és az azok közti kapcsolatokat. A megtervezésnél figyelembe vettük a modularizáció elvét, vagyis hogy az egyes alrendszerek közti kapcsolat minimális legyen, módosítások a lehető legkevésbé hassanak ki a "külvilágra". Részletekbe menően bemutatjuk a kész rendszer felépítését.

4.1 Rendszerünk globális felépítése

Input alrendszerünk végzi az általunk választott képkészítésre alkalmas eszköz vezérlését és a továbbítást a képfeldolgozónak, mely a megfelelő előfeldolgozást követően kinyeri az egyedi jellemzőket, összehasonlításra és tárolásra alkalmas információkká transzformálja azokat. Tovább előállítja az egyes módszerekhez tartozó, az adott képre jellemző vektorokat. Szükségünk lesz még egy adatbázisra, valamint egy összehasonlító alrendszerre, mely hatékonyan el tudja végezni az azonosítást. Természetesen az emberi irányításhoz és az eredmények megjelenítéséhez kell egy grafikus felület is. A felépítést 15. ábrán láthatjuk.



15. ábra. Alrendszerek és kapcsolataik

4.2 Input alrendszer

4.2.1 Input eszköz

Első lépés, hogy eldöntjük milyen eszközt célszerű használnunk az input képek elkészítésére, mik azok előnyei és hátrányai. Számunka elvárás volt, hogy az eszköz olcsó és könnyen használható és kezelhető legyen, de ez ne menjen túlságosan a rendszer hatékonyságának rovására. Szkenner, webkamera és digitális kamera használata között választhatunk. Mindegyiknek megvannak a maga előnyei és hátrányai.

Szkenner esetében alacsony költség mellett nagyon jó minőség érhető el. Egyszerű átalakítással homogén fényviszonyokat hozhatunk létre. Azonban felhasználói szempontból fontos tényező a gyors működés, és egy ennek az elvárásnak megfelelő (rövid szkennelési idejű) típus már kifejezetten drága lehet. Higiéniai szempontból sem előnyös a tényleges fizikai kontaktus, és ez a modern világunkban már jelentős tényezőnek számít.

A webkamera és a digitális kamera között több tényező dönthet. A digitális kameráknak általában jobb a képminősége és jóval több funkciójuk is van, de ezeket nem feltétlenül tudjuk használni számítógépes környezetben. A funkciók eléréséhez és vezérléséhez az eszköznek rendelkeznie kell adatkábel bemenettel (USB), melyen közvetlen kétirányú kommunikációt (P2P) tud végrehajtani. Ha ezek teljesülnek, akkor részletekbe menő vezérlést tudunk megvalósítani, ellenkező esetben, ha felismeri is a számítógép az eszközt, akkor is csak általános beállításokkal tudjuk használni, így egyáltalán nem lesz jobb, mint egy egyszerű webkamera.

A webkamerák kifejezetten számítógépes alkalmazásra készülnek, így azok használata minden esetben egyszerű programozói és felhasználói szempontból.

Összefoglalva tehát az eszközök legfontosabb tulajdonságai a következők:

Szkenner:

- Könnyebben használható kép.

- Lassú a bemeneti kép elkészítése.
- Higiénia gondok.

Kamera

- Komolyabb előfeldolgozást igényel.
- Gyors a bemeneti kép elkészítése.
- Nem szükséges közvetlen érintés.

Felhasználói indításra, vagy egy szenzor általi jelre a kép elkészül, melyen elkezdődnek előfeldolgozási folyamatok. Végso soron az input alrendszer feladata a mintavétel elvégzése mellett annak ellenőrzése is, hogy megfelelő kép készült-e el. Ebbe beleértjük, hogy megfelelően tartottuk-e a kezünket és hogy megfelelően elkülöníthető-e a tenyér a háttértől.

A mi választásunk hosszas mérlegelés után végül egy webkamera felhasználására esett. A tényleges mintavétel menetének mikéntjéről és körülményeiről részletesen a későbbiekben lesz szó.

4.2.2 Színterek

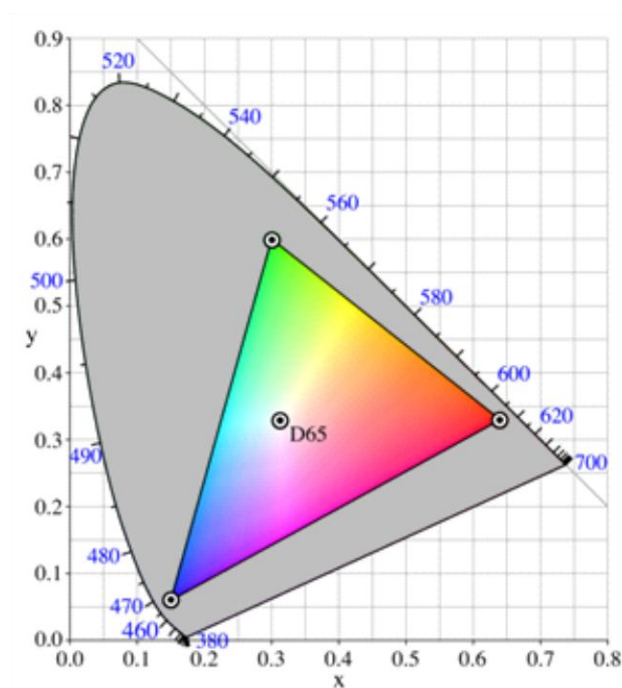
A hatékony feldolgozáshoz nem árt tisztában lennünk a színekkel kapcsolatos tudnivalókkal, ezért ebben a fejezetben az ehhez kapcsolódó, számunkra legfontosabb ismeretekről lesz szó. A színtér egy matematikai reprezentációja a színek világának, azok leírása számokkal, számsorozatokkal. A színekkel, színterekkel foglalkozó tudomány folyamatosan fejlődik, mivel nagyon összetett feadat a színek digitális reprezentációját minden szempontból optimálisan előállítani. Jelenleg tökéletes megoldás nem létezik. Ugyanakkor a színérzékelés teljesen relatív. Nem lehet biztosítani például, hogy mindenki ugyanazt a színt lássa, továbbá nincs minden szín, színárnyalat névvel ellátva, így nehéz több embernek egy adott színre asszociálni. Az általunk használt eszközök (monitor, telefon, fényképezőgép) más-más színtartományokban dolgoznak, működnek. Először a legelterjedtebb RGB színtérről lesz szó.

4.2.2.1 RGB

Az RGB színtér a három alapszín kombinációjából állítja elő a színeket.

Ezek a következők: vörös (R), zöld (G), kék (B), ami a színek angol nevéből ered, vagyis Red, Green és Blue. Egy színt ez a három paraméter határoz meg, melyeknek az értékei azonos intervallumban változhatnak. Ha vizuálisan akarjuk elképzelni, akkor az RGB színtér egy kockát fed.

Azért ezt a három színt használjuk, mivel az emberi szemben található csapok (a színérzékelésért felelős receptorok) alapvetően e három szín érzékelésére képesek. Több hátránya is van az RGB használatának. 16. ábrán látjuk az egyiket.



16. ábra. RGB által az ember számára látható tartományból lefedett rész [12]

Az ábrán a szürke rész jelenti az emberi szem által látható részt. A színes jelenti ebből az RGB színtér színeit. Látható, hogy korántsem fedi le az összes "létező", az emberi szám által érzékelhető színt.

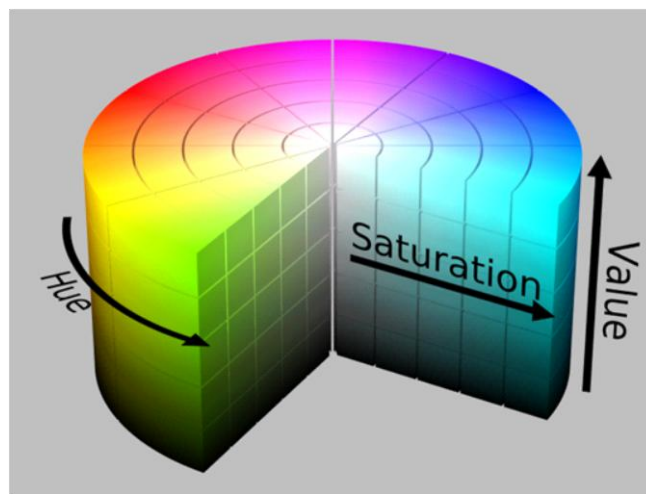
A másik és jelentősebb gond a színek közötti különbség meghatározásának a nehézsége. Ennek oka nyilvánvaló, hiszen három színkomponens határozza meg magát a színt és ezek eltéréséből kellene meghatározni mennyire hasonlóak vagy sem.

Ennek a problémának a kiküszöbölését teszi lehetővé a következő részben taglalt modell.

4.2.2.2 HSV

A HSV színtér az RGB színtérmodellben lévő színeket reprezentálja más geometria formában úgy, hogy áttanszformálja az RGB színhármast egy eltérő térbe.

Elsőként 1978-ban publikálták a modell alapjait. [11] Ebben az esetben is három paraméterrel lehet megadni minden egyes színt. Az értelmezést a 17. ábra segíti.



17. ábra. HSV henger [12]

A három paraméter a következőket jelenti:

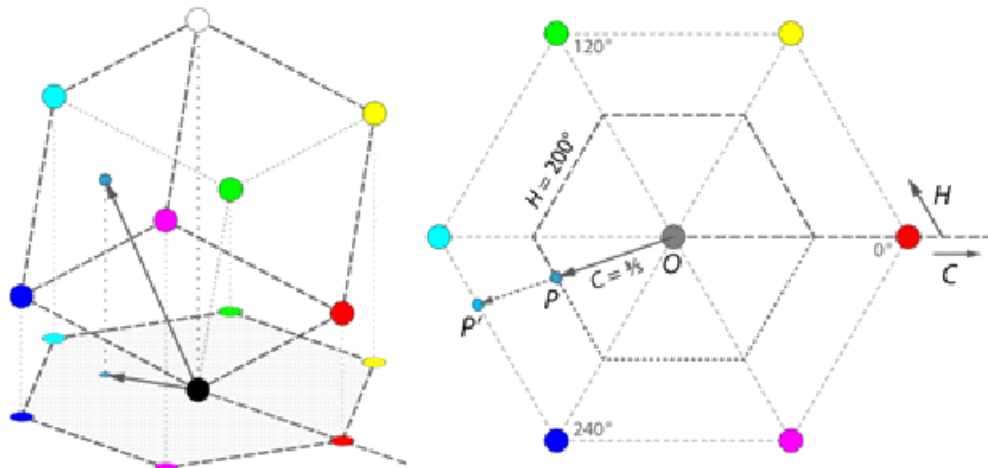
- Színárnyalat (Hue): a domináns szín, amit érzékelünk, mely egy szöget (irányt) fog jelenteni a hengeren felülnézetből.
- Színtelítettség (Saturation): a szín erőteljességét jelenti, mely a középponttól való távolságnak felel meg.
- Világosság (Value): ez határozza meg a fényerőt, vagyis a hengernél a magasság értékét.

Az előnye ennek a reprezentációnak, hogy az árnyékoktól és a megvilágítástól függetlenül tudjuk egy érték (intervallum) megadásával, hogy milyen színt keresünk. Vagyis esetünkben a bőr színének behatárolása könnyedén megoldható e módszer

felhasználásával. Ellenben ha RGB színtérben dolgoznánk három paraméterben kéne gondolkoznunk a színárnyalat meghatározásakor is.

4.2.2.3 RGB és HSV színtér közötti konverzió

Az alapötlet a HSV színtér létrehozásánál az volt, hogy az RGB kockát el kell forgatni a fekete csúcsára és le kell vetíteni a többi csúcsát a „színesség” síkjára, így létrejön egy hatszög, ami a tér alapját alkotja. Ezután meghatározzuk az egyes színek szögét, a vöröstől (0°) kezdve. Ez lesz megközelítőleg a színárnyalat ($H \in \mathbb{Q}\{0:360\}$). A szín origótól való távolsága megközelítőleg megadja a színesség értéket (ez a Chromaticity, jelölése C). Ezzel a leképezéssel megkapjuk megközelítőleg a H és C értéket. Ennek a menetét szemlélteti a 18. ábra.



18. ábra. RGB kocka elforgatása és leképezése [12]

Az ábra jobb oldali részén látható, hogy, a C érték meghatároz egy arányt, ami OP és OP' szakaszok hossza között áll fent, valamint meghatároz egy új hatszöget is. Ennek a két szakasznak az arányára úgy tekintünk, mint a legnagyobb és legkisebb RGB komponens különbségére.

Chroma (C) kiszámításának menete a következő:

$$\begin{aligned} M &= \max(R, G, B) \\ m &= \min(R, G, B) \\ C &= M - m \end{aligned}$$

A színesség érték azért képezhető le ennyire egyszerűen M és m különbségére, mert a

projekció során a természetes színek (fekete-fehér) leképzése során a színesség érték 0. A képletből adódik, hogy ha a három érték egyenlő (R, G és B), akkor azok a kocka fekete és fehér csúcsa közé fognak esni. Ezeken a helyeken a színesség értéke 0 lesz, vagyis kizárólag függőlegesen mozgunk a csúcsára forgatott kockában.

A H (Hue) érték a színesség érték alapján meghatározott vektor csúcsának levetítése a hatszög oldalára. Ennek a pontnak az előre definiált 0°-hoz képest a bezárt szöge adja az értéket.

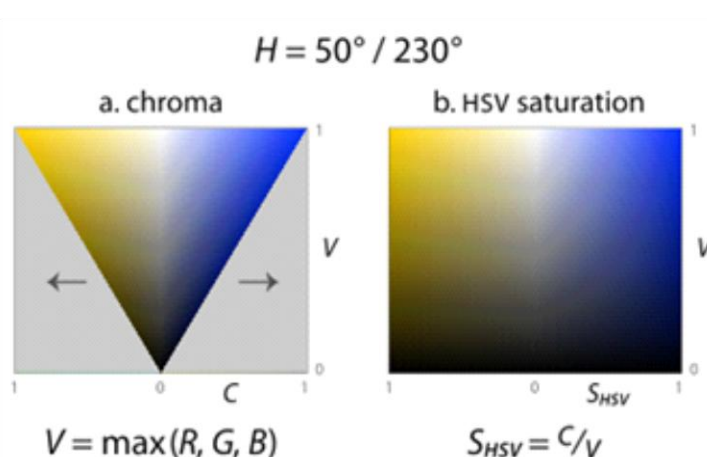
H érték kiszámításának képlete a következő:

$$H' = \begin{cases} \text{undefined,} & \text{if } C=0 \\ \frac{G-B}{C} \bmod 6, & \text{if } M=R \\ \frac{B-R}{C} + 2, & \text{if } M=G \\ \frac{R-G}{C} + 4, & \text{if } M=B \end{cases}$$

$$H = 60^\circ \times H'$$

Mivel polárkoordinátákat használunk ezért szükséges a hatszöget körré transzformálni.

A V (Value) érték meghatározására több módszer létezik feladattól függően. HSV szintérben a legnagyobb RGB komponens kiválasztásával a hatszög alapú gúla alapjába az összes elsődleges és másodlagos színt és a fehéret képezzük le (19. ábra bal oldala).



19. ábra. V (Value) és S (Saturation) értéke HSV színtartományban [12]

Ahogy az ábrán látható nem minden C értékhez tartozik értelmezhető V érték. Ennek az értéktartományban, amiben vagyunk, csak a felét használjuk ki, az értékek másik fele az RGB tartományán kívül esik. Ez számításoknál zavaró lehet ezért az értékeket átskálázva 0 és 1 közé megkapjuk az S értéket (19. ábra jobb oldala).

4.2.3 Binarizálás

Feladatunk, hogy jó pontossággal és gyorsan elkülönítsük a képet a háttértől, vagyis hogy egy olyan fekete-fehér maszkot hozzunk létre, ami pontosan elkülöníti a kívánatos régiókat a nem kívánatostól. Első nekifutásra ez könnyűnek tűnhet, de ha nem ideálisak a körülmények (fényviszony, élesség stb.), akkor hamar akadályokba ütközhetünk, melyek miatt a kívánt eredményt nem érjük el.

Alapvetően a binarizáláshoz használt input lehet álló-, vagy mozgókép. A lényegi különbség a felhasznált képek számában van. Használhatunk egy darab képet, majd választunk a szakirodalomban megtalálható rengeteg algoritmus közül és optimalizálunk a legjobb eredmény eléréséhez. Másik megközelítés, hogy valamilyen változást is figyelembe veszünk (időbeli változás, felvétel fókusz stb.). Mi kihasználtuk a kamera adta lehetőségeket és két képet használunk a binarizációs maszk elkészítéséhez. Az eszköz helyszíni felállítása után az azonosítás megkezdése előtt készítünk egy referencia képet, melyen csak a háttér látható, majd ha ezután egy minta készül, akkor elvárhatóan a legnagyobb eltérések a kezünk helyén lesznek.

4.2.3.1 Maszk első rétege Otsu algoritmussal

Legkézenfekvőbb megközelítés, hogy a két kép különbségét használjuk fel. Esetünkben ez abszolút szín eltérést jelent, ráadásul kizárólag vörös szín eltérését (így hatékonyabbnak bizonyult). A kapott szürkeárnyaltos képen egy mediánszűrő használata után a jól ismert Otsu algoritmust alkalmazzuk. [13] Ez az ideális képeken viszonylag elfogadhatónak tűnt. Első teszteredményeket kizárólag ennek alkalmazásával végeztük.

Hátránya, hogy mivel ez egy, az algoritmus által meghatározott globális küszöbértékkel dolgozik, ezért ez az árnyékosabb részekből (tenyér körvonalából például) gyakran levág. A finomítás és az eredmény javítása érdekében, ezért ezt célszerű lenne további

módszerrel (vagy módszerekkel) finomítani. E célból további maszk rétegeket fogunk felhasználni, melyek segítségével pontosabban megkapjuk a tenyér által határolt részt.

4.2.3.2 Maszk második rétege HSV színekkel

Második választásunk javulás eléréséhez a HSV színreprezentáció előnyeinek kihasználására esett, melyről már a korábbiakban részletesen szoltunk. Visszautalva, ennek színreprezentációnak a lényegi előnye számunkra, hogy a színárnyalatot egyetlen érték meghatározza a telítettségtől és a fényerőtől függetlenül. Ebben a rétegben kiszűrjük a bőr színétől leginkább eltérő színeket (mivel a háttér kék, ezért ez logikusan célravezető megoldásnak tűnik). Kiegészítő lépésként kiszűrjük a sötétebb részeket is, melyek a tenyér kontúrja körül jellemzőek (árnyékot vet a kéz). Itt küszöbértéknek a kép átlagos fényerejét vesszük alapul, melyet tapasztalati úton megfelelőnek találtunk.

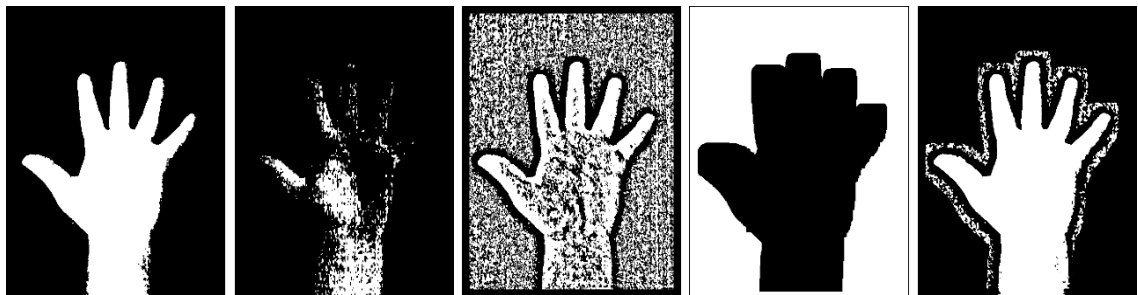
4.2.3.3 Maszk harmadik rétege lokális Niblack algoritmussal

A binarizálási módszer további fejlesztésére azért van szükség, mert nem tökéletesen homogén környezetben készültek a képek. Egy integrált és kontrollált rendszernél a megfelelő eljárás megvalósítása egyszerűbb lenne. A sötétebb képeknél még így is csökkenő pontosságot tapasztalunk. Ennek egyik oka lehet, hogy kisebb fényerőnél a színek eltérése csökken az olcsóbb kamera tulajdonságai következtében. Ez tulajdonképpen azt jelenti, hogy nem annyira jelentős a színárnyalatbeli eltérés, ami a HSV-ből származó előny elvesztésével jár. Ennek a megmaradó problémának a kiküszöbölése végett még egy réteget hoztunk létre.

Ebben a rétegben szeretnénk elérni azt, hogy közel minden helyzetben pontosan megkapjuk a körvonalakat. Ehhez globális küszöbölésről lokálisra kell váltanunk. Niblack algoritmusát használtuk fel, melynek során a kontúr környékén kapjuk a legpontosabb eredményeket, ahol a másik két módszer sötétebb képek esetén elégtelennek bizonyult. [14] Hátránya, hogy nagyobb képen nagyságrendekkel lassúbb, mint egy globális változat. Ennek a problémának a kezelésére, illetve enyhítésére integrál képeket generálunk, mely főleg a több processzoros gépeken jelentős gyorsítást eredményez.

4.2.3.4 Rétegek egyesítése

Az eltérő rétegeknek eltérő körülmények közt van meg az előnye. Egyesítésükkel egy olyan összetett eljárás jön létre, mely magas pontossággal működik az adatbázis összes képére. Amire szükség is van, mivel az összes további lépés ezt veszi alapul. Plusz lépésként az Otsu réteg másolatát dilatáljuk, invertáljuk és kivonjuk az egyesített maszkból, hogy a Niblack réteg által generált plusz komponensek nagy részétől megszabaduljunk, mely gyorsítja a későbbi feldolgozást. Ugyanebből a célból egy medián szűrőt is lefuttatunk. Példánkban (20. ábra) egy sötétebb képen mutatjuk be a folyamatot.



20. ábra. Binarizációs maszk részei (Otsu, HSV, Niblack, dilatált, egyesített)

4.2.4 Egy komponens kialakítása

Következő feladatunk annak a megvalósítása, hogy egyetlen komponens legyen a képen, ami nem más, mint a tenyér. Ezt egy úgynevezett kapcsolódó komponens analízissel fogjuk végrehajtani. [15] El kell tüntetnünk a binarizálás során elkerülhetetlenül megjelenő kisebb foltokat. Az algoritmus eredménye a 21. ábrán látható egy példában.



21. ábra. Komponens kapcsolás, tenyérkép normalizálása

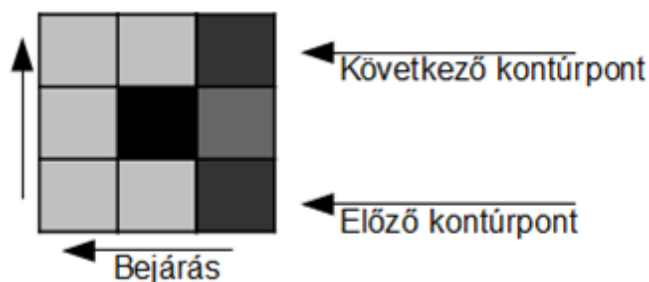
Feltételezzük, hogy tudjuk a háttér és az objektum színét a binarizált képen (melyik fekete és melyik fehér). Meg kell keressük az összefüggő részeket, aminek véghezvitelét az iteratív négyes szomszédságon alapuló algoritmussal oldottuk meg. Első lépésben pixelről pixelre végigmegyünk a bal felső sarokból jobbra és föntről lefelé. Elég vizsgálnunk az aktuális, egyel fölöttünk lévő és az egyel balra lévő pixelt. Ha ezeken a helyeken objektum pixel található és az aktuális is objektum pixel, akkor előbbieket azonosítójával látjuk azt. Amennyiben a két szomszédos pixel eltérő azonosítóval rendelkezik, azt eltároljuk egy szomszédsági listában. Ha nincs a szomszédban objektum, akkor új azonosítót kap. Értelmszerűen a legfelső sornál és a legszélső bal oszlopnál egyszerűsödik a teendő, mivel kevesebb szomszédot kell vizsgálni.

A második lépés, hogy az eltárolt kapcsolatok (pixel azonosítók) alapján egyesítenünk kell a komponenseket. Ahhoz, hogy a szomszédságok láncolatát feltérképezzük, rekurzív függvényt használunk fel. Sorra vesszük az összes azonosítót és rekurzívan az összes szomszédságban szereplőt az ő azonosítójára változtatjuk.

Először a háttérrel tekintjük objektumnak. Ilyenkor a háttérrel és minden tényéren belül lévő kisebb komponenset fogjuk megtalálni. A legnagyobb komponens maga a háttér lesz. Ezen kívül az összes komponenset a tényér színére változtatjuk. A kép invertálása után újra lefuttatjuk az algoritmust, ezáltal a háttérben lévő komponensek tűnnek el (16. ábra).

4.2.5 Kontúr bejárás

Binárizálás és komponens kapcsolás után elértük azt az állapotot, hogy a tényér képe egybefüggő. Következő feladatunk, hogy ki kell nyernünk a kontúrpontokat a megfelelő sorrendben a további feldolgozáshoz. Ehhez értelmeznünk kell egy csuklóvonalat, melynek bal pontjából indítjuk a bejárást a körvonalon, egészen addig, míg eljutunk a csuklóvonal jobb szélső pontjáiig. Ezt egy rekurzív algoritmus fogja megvalósítani. Ismernünk kell az egyel korábbi pont helyét és a nyolcas szomszédságban képesnek kell lennünk kinyerni a pixelek színét. A vizsgálatot az óramutató járásának megfelelő irányba végezzük. Vizuális szemléltetése a 22. ábrán látható.

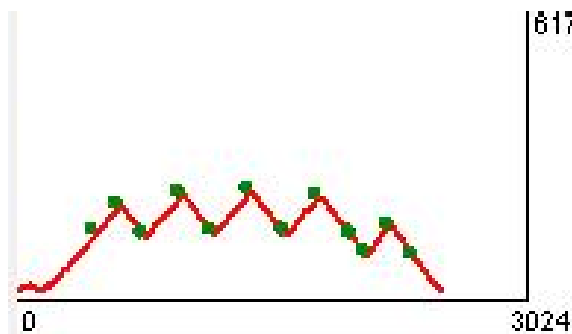


22. ábra. Kontúrbejárás szemléltetése

A vizsgálandó kezdő pixel a szomszédságban két pozícióval az előző kontúrpont után van és előző kontúrpontig vizsgálunk legfeljebb. A nyolcas szomszédságnak köszönhetően akkor is tovább tud haladni az algoritmus, ha csak átlósan csatlakozik egy pixel. Optimalizációs lehetőség, hogy ha az előző pixel nem átlósan helyezkedik el, akkor három pixellel odébbról kezdünk vizsgálni. Minden egyes lépésnél a koordinátákat eltárolom (esetünkben egy listába).

4.2.6 Kontúrfüggvény előállítása

A kontúrpontok megkeresése után definiálni kell a csukló felénél egy referencia pontot. Következő lépésben a kontúrpontok listáján végig menve kiszámoljuk minden elemhez az ahhoz tartozó Euklidészi távolságot ettől a csuklóponttól. Ezen művelet végrehajtása után eredményül kapjuk a kontúrfüggvényt, ami a későbbiekben az alapja lesz a további algoritmusainknak. 23. ábrán egy kontúrfüggvényre látunk példát.



23. ábra. Példa egy kontúrfüggvényre

4.3 Képfeldolgozó alrendszer

Ezen alrendszerünk végzi a jellemző vektorok létrehozását és tartalmazza az azok meghatározásához szükséges összes eljárást. Első lépésben fel kell tudnunk ismerni a kéz fontos pontjainak az elhelyezkedését (például ujj végei).

4.3.1 Referencia pontok meghatározása

Mind a geometriai jellemzők, mind a ROI meghatározásához szükségünk van a fontosabb referencia pontok meghatározására. Ezt a kontúrfüggvény felhasználásával tehetjük meg, melynek képe, ha megfelelő volt az input kép, öt csúcsból áll. Ez alkalmas eldönteni, hogy bal vagy jobb kézről van szó, mivel a hüvelykujj és a mutató ujj közötti nagyobb völgy található, mint a kisujjnál. A probléma megközelítése a következő: a kontúrfüggvény speciális alakja miatt (5 csúcs, 4 völgy) definiálni lehet egy algoritmust, amely megállapítja az ujjak csúcsait és a hozzájuk tartozó völgyeket. Az algoritmus lényege egy maximum és minimum keresés ötvözete. A tömb nagysága miatt igen nehéz a lokális maximumok illetve minimumok keresése mivel elég gyakran előfordulnak a függvény egyenetlensége miatt kisebb nagyobb csúcsok/völgyek. Ennek kiküszöbölésére be kell vezetni egy változót, ami figyel, hogy mióta nem történt változás az aktuálisan eltárolt lokális szélsőértékben és egy küszöbszám után az algoritmus átvált az ellentétes szélsőérték keresésére. Kísérletezés után ez a szám 80 pixel lett, de paraméterezhető, mert függ a kontúrfüggvény nagyságától (az input kép felbontásától) az egyenetlenség. Maximumkereséssel kezdődik majd, ha küszöbszám óta nem változott a lokális maximum, akkor átvált minimumkeresésre, majd megtalálja az első völgypontot és folytatódik tovább a tenyér végéig.

Miután megtalálta az összes pontot az algoritmus, definiálni kell a nagyujj „külső” részénél, a mutató ujj nagyujj felőli részénél, és a kisujj külső élénél referencia pontokat, hogy pontosan meg lehessen határozni az ujjakat, mivel ezeken a helyeken nincsenek völgyek. Ezeket a pontokat az adott ujj másik két rendelkezésre álló völgy és csúcspontjának távolságából lehet kiszámítani, lényegében csak a csúcsponttól kell rámérni a völgypont távolságát. Az algoritmus működése egyszerű, először megnézi az indexkülönbséget a meglévő két pont között, majd visszalép a helyes irányba az adott indexszel és megnézi, hogy merre kell tovább mennie (kisebb, vagy nagyobb-e az

ismert völgy és csúcs közti pixeltávolság), és ha megtalálta a megfelelő pontot leáll és eltárolja.

4.3.2 Geometriai jellemzőn alapuló azonosítás

Ezen módszerünkénél az ujjak hosszát és szélességét használjuk fel. A méretek függenek attól, hogy milyen távolról készül a kép, ezért két lehetőségünk van. Vagy fix távolságból készül a felvétel és abszolút méretekkel dolgozunk és pixelben számolhatunk, vagy arányokat veszünk figyelembe. Például szélességet az ujj hosszához, a hosszúságot pedig szomszédos ujjhoz viszonyítjuk.

3.3.2.1 Számításhoz felhasznált további pontok

A következő lépés az ujjak tőpontjainak megkeresése és ha ez megvan, akkor már közel járunk a geometriai jellemzők kinyeréséhez. Ezeket a pontokat egyszerű előállítani, csak meg kell felelni az adott ujj két oldalán lévő völgypontokat összekötő szakaszokat. Az ujjak tőpontjaival meghatározhatóak az ujj harmadol pontjai (a tőpont és a csúcspont között), ezt követően egy merőleges állításával ezekben a harmadoló pontokban meg tudjuk állapítani a geometriai jellemző vektort. A geometriai vektor tartalma minden ujj esetében: az ujj hosszúsága, illetve a szélessége a harmadoló pont. Felmerült a ujj szélességének figyelembe vétele az ujjak tövénél is, de ez sokszor pontatlan és változó értéket ad, így ez nem része a jellemzővektornak. A számításhoz és a jellemző vektorhoz használt pontokat és távolságokat 24. ábrán látjuk.



24. ábra. Geometriai jellemzővektor és az előállítása során felhasznált pontok

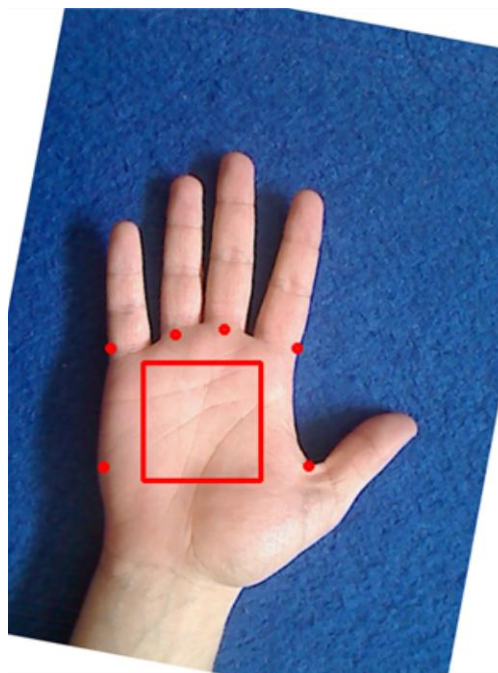
4.3.3 A ROI extraktum elkészítése

Egy másik azonosítási elv a tenyér mintázatának a figyelembe vétele. Ehhez szükség van a tenyér közepén található lényeges területet meghatározására, melyet ROI-nak (Region of Interest) nevezünk. Erre az összes tenyéralapú biometrikus rendszer hasonló algoritmusokat használ. Két lehetséges módszert is kipróbáltunk.

Az első csupán két referencia pontot vett figyelembe, a mutatóujj és kisujj tőpontjait. Az e két pont által alkotott egyenes alapján a kép forgatásra kerül, majd ennek az egyenesnek a középpontjától egy meghatározott értéket lemérve egy négyzetet húzunk ki, az előbb említett két ujj külső pontjáig. Ez a módszer elég bizonytalan eredményeket hozott, meglehetősen sok esetben elcsúszott. A másik módszer, mely implementálásra került, sokkal megbízhatóbb, mivel több referencia pontot használ, és sokkal inkább egybe esett az egyazon személytől nyert ROI-k elhelyezkedése.

Ehhez a módszerhez fel kell venni egy újabb referenciapontot a kisujj alatt, ami ugyanolyan távolságra van a kisujj bal oldali völgypontjától, mint a mutatóujj jobb oldali völgypontja és a hüvelykujj bal oldali völgypontja. Ezek után a kisujj bal oldali völgypontja és a mutatóujj jobb oldali völgypontjából létrejött egyenes segítségével

vízszintbe kell forgatni a képet. A nagyujj és a gyűrűsujj középpontja és a hozzájuk legközelebb álló pontok távolságának felénél lesz a ROI téglalapjának a két felső sarka. Mivel sok esetben a ROI kívül esett a tenyéren (az ujjak legalsó része is belekerült), ezért lejjebb kellett tolni a téglalapot egy konstans értékkel. A ROI magasságát a kisujj alatti, újonnan felvett referenciapont és a hüvelykujj völgypontja határozza meg. A 25. ábrán látható a módszer szemléltetése.



25. ábra. ROI kinyerésének szemléltetése

4.3.4 Főkomponens analízis (PCA)

A főkomponens analízis egy lineáris algebrai megközelítés, amely nagy és ködös adathalmazból képes meghatározni az adott halmazra legjellemzőbb struktúrát. [16] Használják például weboldalak indexelésénél, neurobiológiában, meteorológiában, ezen kívül még számos egyéb helyen is alkalmazható.

A főkomponens analízis célja azonosítani a leginkább jelentőségteljes változókat egy vektortérben, kiszűrve a zajt, és felfedve egy rejtett struktúrát, amivel kifejezhető az eredeti adathalmaz.

Tehát a feladatunk az, hogy megtaláljuk azt a transzformációt (legyen P), amivel létrehozható egy sokkal alacsonyabb dimenziószámmal rendelkező bázis (legyen Y), ami alacsony dimenziószáma ellenére megtartja az eredeti vektortérben rejlő jellemző eltéréseket, vagyis megmarad a variancia.

Általános egyenlet:

$$P X = Y$$

Ennek többféle értelmezése létezik:

- P egy mátrix, amely transzformálja X -et Y -á.
- Geometriai felfogással: P egy mátrix, amely nyújtja, elforgatja X mátrixot Y -á.
- P vektor sorai az új bázis vektorok, amelyekkel újra felépíthető X .

4.3.4.1 A zaj

A mérésből a zajt minimalizálni kell, különben az analízis technikától függetlenül nem lesz használható információink a jelről. Nincs skála a zaj nagyságára, de egy zaj és jel közötti aránypárral (Signal To Noise Ratio, azaz SNR) ki lehet fejezni a mértékét. Ez a jel variancia négyzetének és a zaj varianciánégyzetének aránya.

A magas SNR érték (közelít az 1-hez) magas pontosságú mérésre utal, míg az alacsony zajos mérésre. Ezen mérőszám segítségével kell kiválasztani a megfelelő bázistranszformációt. Mivel az adott problémára képesek vagyunk meghatározni a fix jellemzőket, következtetni lehet a zaj és a jel mibenlétére. Meg kell tehát alkotni egy szabályrendszert, ami alapján ki tudjuk szűrni a zajt, vagyis a számunkra irreleváns adatokat.

4.3.4.2 A kovariancia mátrix

Tételezzünk fel két mérést, melyek eredményeit egy vektorban tároljuk. Az egyes vektorelemek értékeikből az adott összetevőre vonatkozó átlagot levonjuk, ezt az összes összetevőre és vektorra elvégezzük, és az így kapott vektorokat használjuk fel a továbbiakban. A két mérés egyedülálló varianciája a következő módon van definiálva:

$$\sigma_A^2 = \frac{1}{n} \sum_i a_i^2 \quad \text{és} \quad \sigma_B^2 = \frac{1}{n} \sum_i b_i^2$$

A kovariancia egyszerű általánosítása:

$$\text{kovariancia}(A, B) \equiv \sigma_{AB}^2 = \frac{1}{n} \sum_i a_i b_i$$

A kovariancia a lineáris kapcsolatot méri két változó között. A magas pozitív érték korrelációt jelez, az abszolút értéke pedig a redundanciát méri. Két vektornak a kovariancia számítása nagyon hasonló egy skaláris szorzathoz. A létrejött $n \times m$ méretű X mátrixot a korábbiakban ismertetett módon normalizált vektorokból épül fel, minden egyes oszlopa egy-egy vektort reprezentál. Így elérkeztünk a kovariancia mátrix (C) definíciójához, melyet az X mátrix segítségével állíthatunk elő:

$$C_x \equiv \frac{1}{n} X X^T$$

Ez a mátrix jellemzi az összes mérés közötti kovarianciát. A magas diagonál értékek a feltételezések szerint jellemzik a rejtett struktúrát, amire szükségünk van és a többi magas érték pedig a redundanciát.

4.3.4.3 A kovariancia mátrix diagonál mátrixszá transzformálása

A célunk az, hogy minimalizáljuk a redundanciát, amit a kovariancia mér, és maximalizálni kell a jelet, aminek mérőszáma a variancia. Az optimalizációhoz a következő két tényezőnek kell teljesülnie:

- A főátlón kívüli elemek értékének zérusnak kell lennie.
- Minden dimenziót a variancia szerint rendezni kell.

Az eredmény a C_y diagonál mátrix lesz. Ezután a következő lépések elvégzésével lehet kinyerni főkomponenseket:

- A mátrix egyik dimenziójából kiválasztjuk azt a vektort, ahol a variancia a legnagyobb.
- Válasszunk egy másik dimenziót és ismételjük meg a lépést addig, amíg m db vektort ki nem választottunk.

Az eredményként kapott és rendezett P vektor lesz a főkomponens.

4.3.4.4 PCA megoldása sajátvektoros dekompozícióval

Szakirodalmi hivatkozásokban található bizonyítások alapján [16] kijelenthető, hogy a P transzformációs mátrix diagonál mátrixot hoz létre.

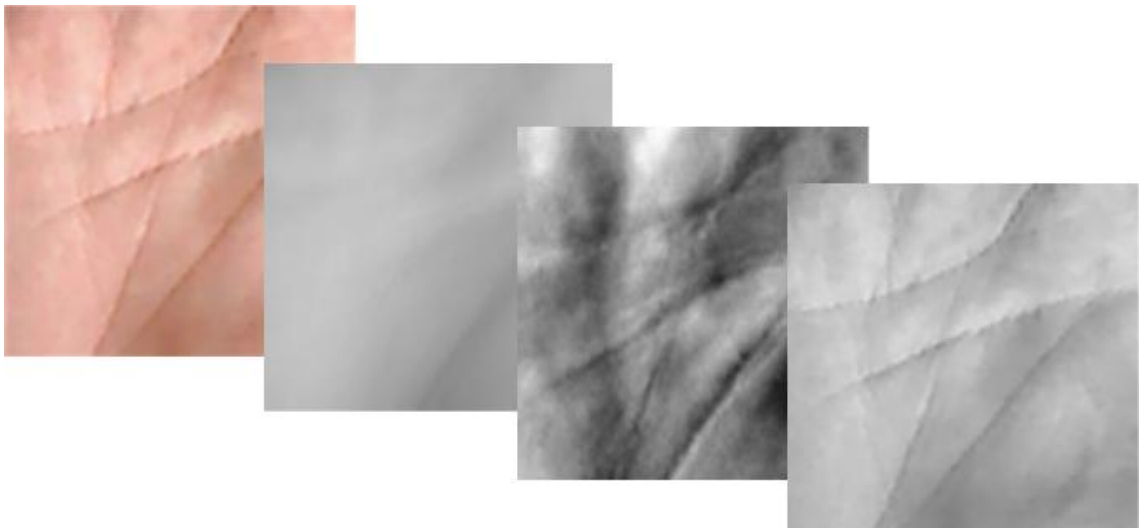
Összefoglalva:

- A főkomponens analízis eredménye C_y és P .
- X főkomponensei az sajátvektorai C_x mátrixnak.
- A C_y mátrix átlós értékei között megtalálhatóak X varianciái a P vektor i . elemében.

Főkomponens kiszámításának menete a gyakorlatban:

- Az X értékeiből ki kell vonni az átlagot, majd ki kell számolni a sajátvektorait C_x - nek.

Miután minden kép sajátértékeiről elkészült az extraktum, akkor matematikai távolságdefiníciók segítségével össze lehet hasonlítani őket, így létrejön egy hasonlósági lista, melyben a minél kisebb értékkel szereplő egyedek a leginkább hasonlóak a kiválasztott képhez. A 26. ábrán a PCA számítási folyamat néhány állomása látható.



26. ábra. Eredeti ROI, átlagkép, sajátkép, sajátképből visszaállított kép

4.4 Adatbázis alrendszer

A rendszer a működése során nagy mennyiségű adat tárolását és gyors összehasonlító algoritmus használatát igényli. A megvalósítás során két lehetséges módszer közt mérlegeltünk. Az egyik az XML alapú adattárolás, a másik az adatbázis és adatbázis szerver használata.

XML előnyei:

- Fájl alapú.
- Gyors elérést biztosít.
- Közvetlen elérés (I/O).
- Könnyű implementáció.

Adatbázis előnyei:

- A tárolást a szerver a végzi, nem terheli a klienst.
- Könnyen módosítható a későbbiekben.
- Biztonságos.

XML hátrányai:

- Nem biztonságos.
- Adatszerkezeteken a későbbi módosítás nehézkes és algoritmustól függ (jól átgondolt előzetes tervezést igényel).

Adatbázis hátrányai:

- Tárolt eljárások használata (hardcode) Entity Framework előtt.
- Nem megfelelő algoritmus esetén nagyon lassú lehet.

Az előnyök és a hátrányok tudatában az SQL adatbázis alapú megoldás mellett döntöttünk, mivel a .NET adta új lehetőségek (Entity Framework) megkönnyítik az implementációt, és a későbbi változtatásokat is lehetővé teszi. Mindemellett az adatbázis jóval nagyobb biztonsági faktorral rendelkezik, mint az XML alapú tárolás.

4.4.1 Entity Framework

Az Entity Framework megjelenése előtt az adatbázis elérés tárolt eljárások hívásával, és az üzleti objektumok manuális deklarációjával történt. Egy-egy eljárás-hívásban a név és paraméterek hardcode-olva voltak, a hívásból visszatérő recordset-ben a rekordoknak az oszlopait pedig string indexszel lehetett elérni. Az SQL oldalon lévő tárolt eljárásokban ügyelni kellett a táblák csatolásánál a join típusára és a pontos tábla definíciókra (idegen kulcsok). Ez megnehezítette a kódolást, az esetleges változtatásokat az objektumokban és az adatbázisban egyaránt. Különösen ügyelni kellett a helyes recordset visszaadására, és az azt meghatározó oszlopnevekre, melyekre a kódban hivatkoztunk.

4.4.1.1 Entitások

Az Entity Framework megjelenésével mindez leegyszerűsödött. [17] Az adatbázis elérést egy köztes absztrakt réteg biztosítja, az Entity Data Model, amit az adatbázisból lehet modellezni (mapping), illetve a modellből lehetséges az adatbázis leképezése. A Visual Studioban nincs más teendőnk, mint meghatározni az Entity Data Modellt, amiből képződik egy adatbázis, vagy magát a modellt is képezhetjük adatbázisból. Az Entity Data Modell meghatározása után az objektumaink kódja automatikusan generálódik. Az objektumok közötti relációk (melyek idegen kulcsokként jelennek meg az adatbázisban) egyszerűen tulajdonságokként vannak definiálva az objektumokban, ezzel jelentősen megkönnyítve az adatokkal való munkát.

4.4.1.2 LINQ

A tárolt eljárások helyettesítése LINQ (Language-Integrated Query) lekérdezésekkel történik. Ezek a C# kódban meghatározható lekérdezések sokban hasonlítanak az SQL utasításokhoz. A LINQ lekérdezések előny, hogy a visszatérési értékük nem egy adatbázisból érkező recordset, hanem már maga az objektum.

Példa egy LINQ lekérdezésre: azon számok kiválasztása, melyek kisebb vagy egyenlők 5-nél.

```
int[] numbers = new int[] {5, 7, 1, 4, 9, 3, 2, 6, 8};  
var smallnumbers = from n in numbers  
where n <= 5  
orderby n  
select n;
```

Egy LINQ lekérdezés Lamba kifejezés használatával: nevek kiválasztása az Employee táblából.

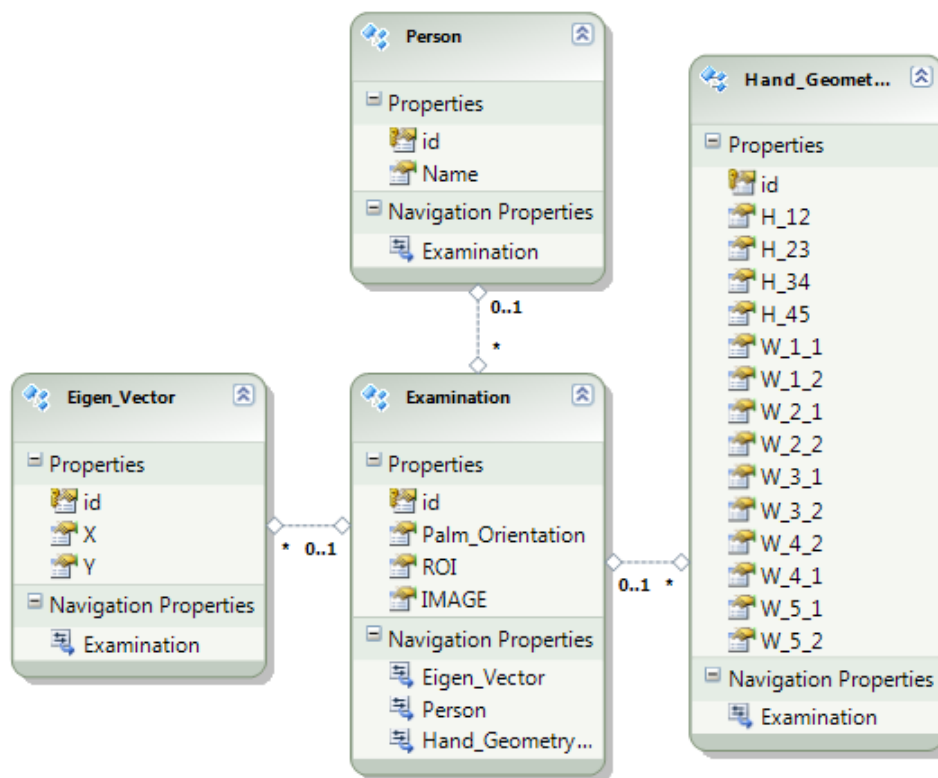
```
var iNames = employees.Select<Employee, string>(r => r.name);
```

A LINQ lekérdezések alapvetően nem túl gyorsak, viszont jelentős teljesítménybeli javulást érhetünk el a PreCompiledQueryk használatával. [18] Erre egy példa:

```
var _custByID = CompiledQuery.Compile<SalesEntities, int, Customer>
((ctx, id) => ctx.Customers.Where(c => c.ContactID == id).Single())
```

4.4.2 Az adatbázis felépítése

Az adatbázis felépítése során figyelembe vettük az egyes azonosítási módszerekhez szükséges input és output adatokat, ezt követően a táblák normalizálása után a 27. ábrán látható sémát kaptuk.



27. ábra. Adatbázis sémája

Egy adott személynek (Person) csak a nevét és az azonosító mezőjét tároljuk el. Több minta (Examination) tartozhat egy személyhez, és minden vizsgálatnál el kell tárolni, hogy az adott felvétel a jobb vagy a balkézzről készült-e (Palm_Orientation), továbbá a kéz geometriai jellemzővektorát (Hand_Geometry), a ROI sajátvektorát (Eigen_Vector) és a ROI-t magát. A ROI-t kép formájában tároljuk a merevlemezen, mivel annak adatbázisba való export, illetve importálása körülményes (mivel importálásnál át kell alakítani bináris típusúvá, az exportálásnál pedig fordítva) és lelassítja az adatbázis és a kliens kommunikációját.

4.4.3 Saját tenyéradatbázis készítése

Az adatbázis készítése során egy fix környezetet építettünk. A környezetben a kamera felülről fényképezi a tenyeret egy kék háttérrel, körülbelül 40 centiméteres fix magasságból.

Minden személy jobb kezéről (28. ábra jobb oldala) öt felvételt készítettünk és az egyes emberek képeinél figyelmet fektettünk arra, hogy más pozícióban legyen a kéz minden egyes felvételen.



28. ábra. Egy minta az adatbázisból és a hozzá tartozó referenciakép

Mind az öt felvétel mellé elmentésre került a referencia kép (28. ábra bal oldala) is, amely segítségével a binarizálást elvégezzük. Az adatbázisban közel hetven 18-27 év közötti személy szerepe. A mellékletben megtalálható az elkészült képeket tartalmazó montázs

4.5 Összehasonlító alrendszer

Az összehasonlító alrendszer feladata, hogy az input adatot, az azonosítani kívánt személy felvételéből kinyert egyedi vektorokat és jellemzőket összehasonlítsa az adatbázisban szereplő adatokkal, és döntést hozzon arról, hogy a személy szerepel-e az adatbázisban vagy sem.

4.5.1 Összehasonlító modul input adatai

Az egyes módszereknél eltérő input adatokkal kell számolni. A kézgeometriai módszer esetében ez az ujjak szélessége és hosszúsága. Az mindenképp elmondható, hogy valamilyen számértékek jelentik az információt. Ezekből az adatokból kell kiszámítani az egyes felvételek egyezőségének mutatóját. PCA módszer esetén a sajátértékek fogják a jellemzővektor értékeit adni. Ezekkel ugyanúgy számolhatunk, mint az előbbi esetben.

4.5.2 Összehasonlítási módszerek

Az összehasonlítást matematikai távolságokkal végezzük. [19] Egyezőnek, hasonlóknak tekintünk két mintát, ha a távolságokkal kiszámított érték egy előre meghatározott nagyságú küszöbön belül helyezkedik el. Az alábbi távolságok kerülnek implementálásra:

Minkowski-Manhattan távolság:

$$D = \sum_{i=1}^n |x_i - y_i|$$

Euklidészi távolság:

$$D = \left(\sum_{i=1}^n |x_i - y_i|^2 \right)^{1/2}$$

Csebisev távolság:

$$D = \max(|x_1 - y_1|, |x_2 - y_2|, \dots, |x_n - y_n|)$$

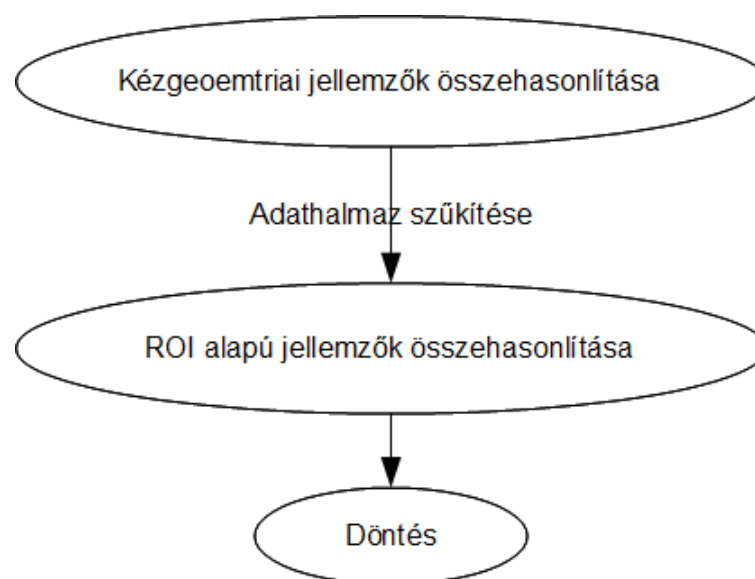
Mindegyik összefüggésre igaz, hogy x_i az egyik jellemzővektor i . elemét jelenti, míg y_i a másik jellemző vektor i . elemét.

Az egyes számítási módszerek eltérő hatékonysággal működnek. A tesztelési folyamat során derül ki, melyik a legoptimálisabb. Lehetséges javítani a módszereken a

mintavételezés folyamatának megváltoztatásával. Például, ha nagyobb egyedi jellemző vektorokat használunk, vagy ha több mintából számítjuk ki az egyes személyek egyedi jellemzőit, vektorait. A vektor növelésének hátulütője a számítások mennyiségének növekedése, így a rendszer lassulása. Tehát a feladat megtalálni az optimális nagyságú vektorokat, amelyek képesek magas biztonsági szinten azonosítani mindamellett, hogy a rendszer használhatósága nem romlik.

4.5.3 A döntési folyamat

A kézgeometriai jellemzők összehasonlítása elvben képes igen magas szintű verifikációra, de ebben az esetünkben, ahogy már korábban is említettük, ezzel nem elégszünk meg. A kézgeometriai jellemzővektor kiszámítása gyors és egyszerű. Ezt felhasználva szűkítjük a keresési adathalmazt, majd a kisebb adathalmazon várhatóan magasabb pontosságú ROI alapú technikát alkalmazzuk. A döntési folyamatot szemlélteti a 29. ábra.



29. ábra. Egy minta az adatbázisból és a hozzá tartozó referencia kép

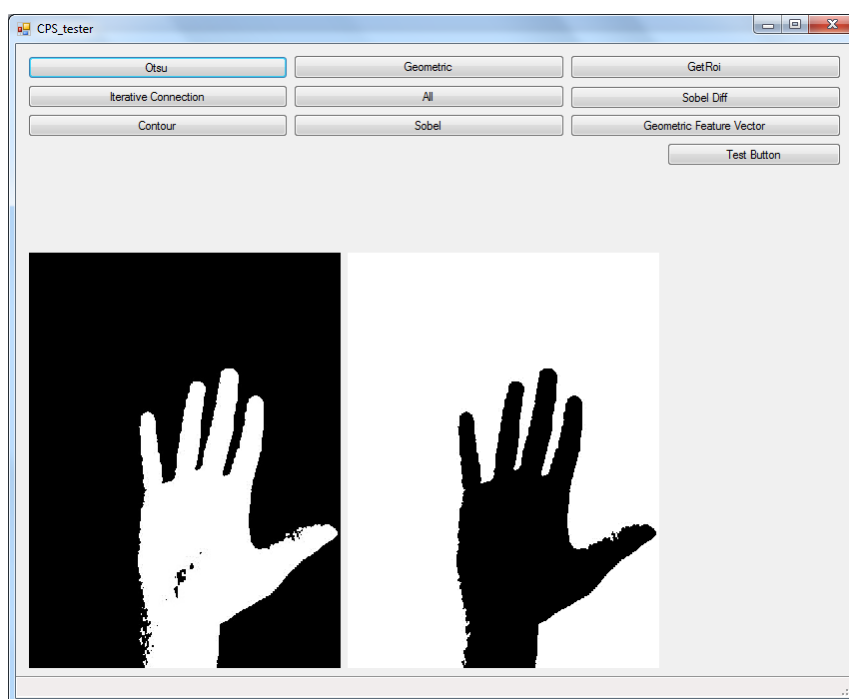
A szűrésnek köszönhetően komolyabb és lassabb ROI alapú módszerek használata is lehetségessé válik, és továbbra is megmarad a valós idejű működés lehetősége. Lényegében ezt jelenti azt, hogy a rendszerünk többszintű.

4.6 Adminisztratív alrendszer, GUI, felhasználói útmutató

Ebben a részben a grafikus kezelőfelületet és a program működését mutatjuk be. Alapvetően két program készült a project fejlesztése során. A korai szakaszban a metódusszintű tesztelésen volt a hangsúly, így szükséges volt egy erre a célra használt felület létrehozása. A rendszerszintű működéshez szükséges minimális szint elérése után lett létrehozva a jelenlegi felület, melynek kezelőfelülete már kisebb szemcsézettséggel tartalmaz funkcionális teszteléshez szükséges elemeket. Utóbbiban már az adatgyűjtés egyszerű végrehajtása és a működés statisztikai elemzéséhez szükséges funkciók megvalósítása a lényeges.

4.6.1 Funkció tesztelő

Ahogy azt korábban írtuk ez a project kezdeti fázisában volt használatban. A jelenlegi fázisban már nincs jelentősége. A felépítését a 30. ábrán látjuk.



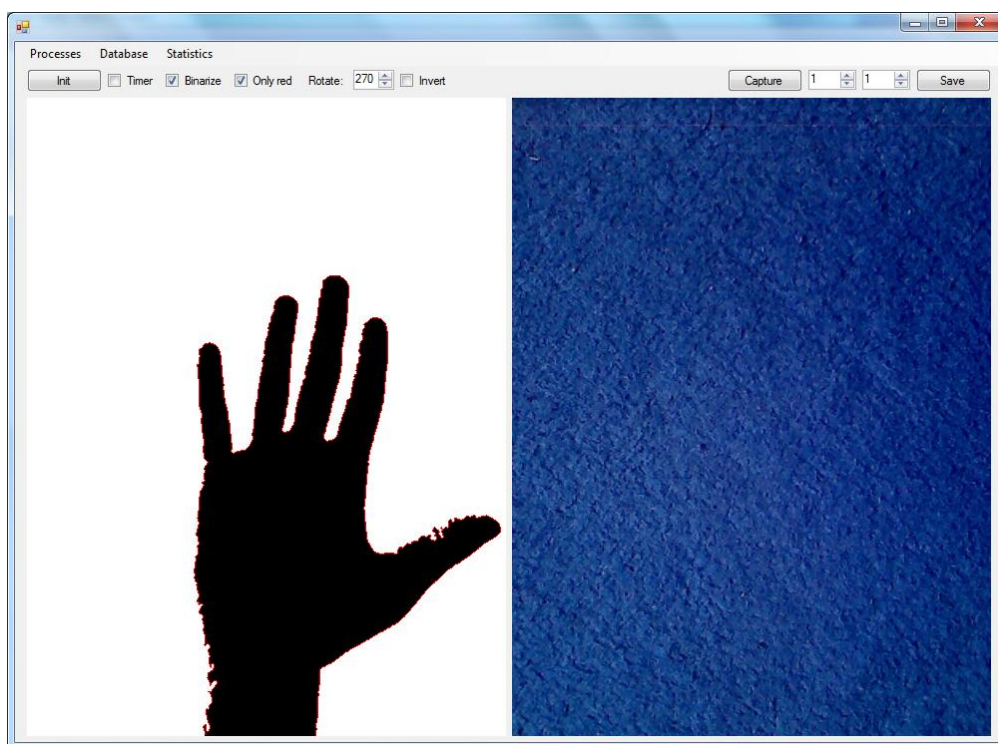
30. ábra. Funkció tesztelő felépítése

Három fő részből tevődik össze. A felső részen a legapróbb részletekig különválasztott funkciókhoz készült vezérlőgombokat látjuk. A gombokon keresztül elérhető funkciók

halmaza lépésről lépésre bővült a fejlesztés előre haladásával. Az ellenőrzés gyorsítására és megkönnyítésére a középső részen vizuálisan is megjelenik a művelet kimenete. További, elsődlegesen sebességre vonatkozó információt szolgáltat az alsó sorban található rész. A részletezés helyett térjünk rá inkább a fontosabb, jelenleg használt kezelőfelület bemutatására.

4.6.2 A fő ablak

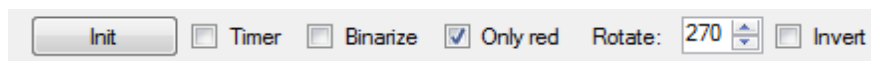
Tekintsük meg tehát, hogy mivel is dolgozunk jelenleg (31. ábra).



31. ábra. Fő ablak

Az előzőhöz hasonlóan jelen vannak a vezérlőelemek és az output rész.

Négy főbb részbe csoportosítottuk a rendszer funkcióit. Első lépés a minták gyűjtése az adatbázishoz. Ebben második sor elemei segítenek. Számos beállítást tudunk elvégezni (32. ábra).



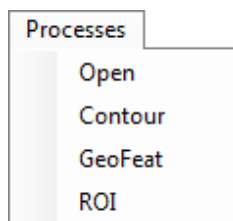
32. ábra. Binarizációs beállítások

Az Init gombra kattintva elkészül a binarizációhoz használatos referencia kép. A Timer-rel az időzítőt indíthatjuk el, mely segítségével folyamatosan frissülve látjuk a binarizáció eredményét. Amennyiben elégedettek vagyunk az inputtal (sikeres a binarizáció), akkor a Capture-re kattintva megállíthatjuk a folyamatot, majd a Save-vel az általunk megadott névvel elmentjük (33. ábra). Az első szám az személy azonosítója, a második a minta sorszáma.



33. ábra. Minta mentése

A Processes tartalmazza képfeldolgozási folyamatok teszteléséhez szükséges elemeket. Részletesebben a 34. ábrán láthatjuk.



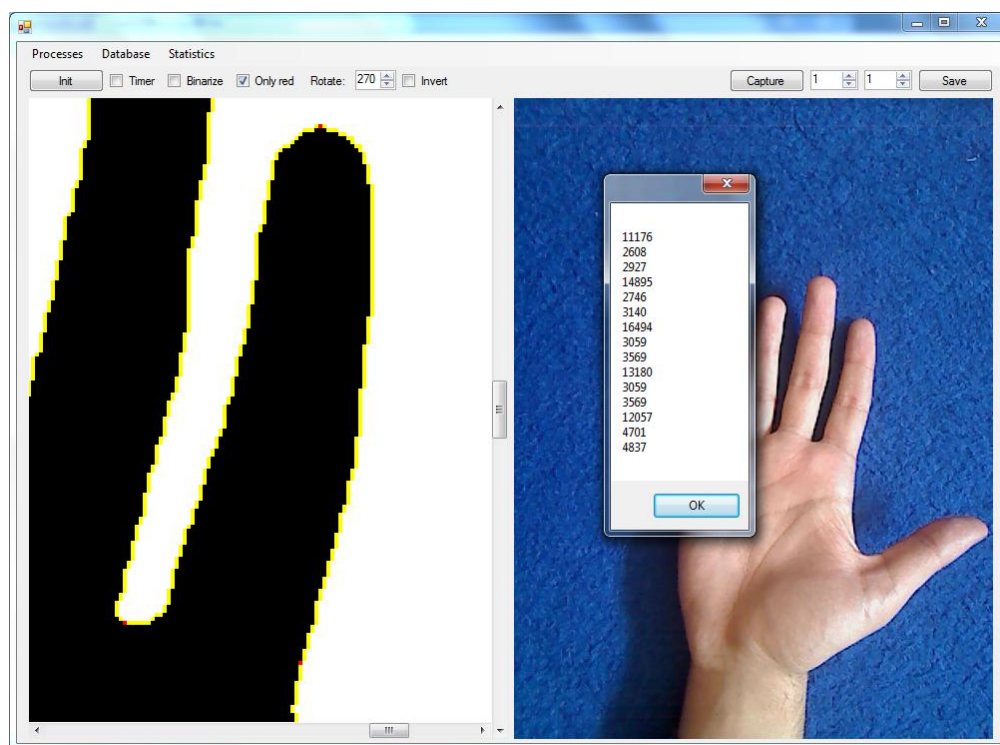
34. ábra. Processes fül tartalma

Az Open-re kattintva egy a gépen lévő általunk megadott mintát nyit meg. A bal oldali képen binarizálva, jobb oldalon pedig meg az eredeti színes kép fog megjelenni.

Ha a Contour-ra kattintunk, akkor a program meghívja a kontúrbejárási eljárást, és színes pixelekkal jeleníti meg az eredményt. Külön színnel jelzi a megtalált referencia pontokat.

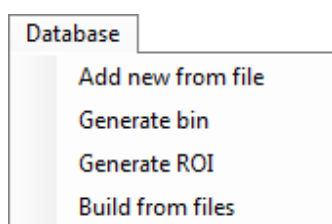
A GeoFeat-re kattintva külön ablakban megjelenik a geometriai jellemzővektor tartalma.

Tetszőlegesen nagyíthatunk a képen, és így könnyebben megvizsgálhatjuk a fontosabb pixelek helyzetét. Ezen funkciókat látjuk a 35. ábránkon. A ROI csak adatbázis létrehozása után működik, így erről később ejtünk szót.



35. ábra. Képfeldolgozási funkciók

Ha összegyűjtöttük a kellő mennyiségű mintát, akkor a következő feladatunk az SQL adatbázis létrehozása. Ennek funkcióit tartalmazza a Database fül (36. ábra).

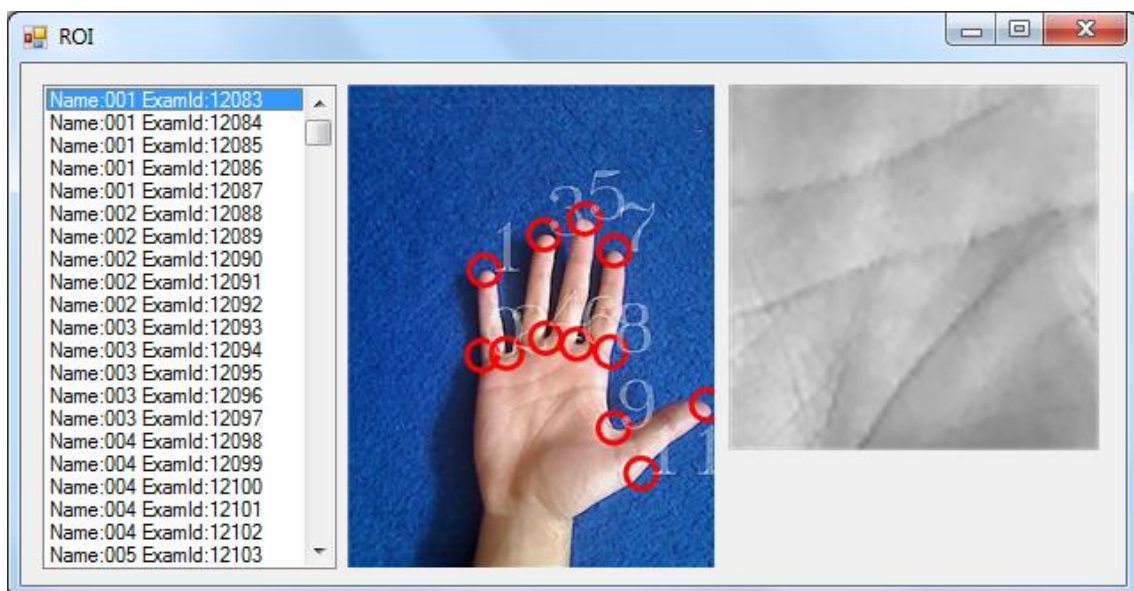


36. ábra. Adatbázis funkciók

Az első gombbal egy minta hozzáadását végezhetjük el. A teljes adatbázis létrehozásához a folyamatok szét lettek választva. Először a bináris képek létrehozása

történik meg. Ezt követi a ROI képek elmentése, majd végül az utolsó gomb segítségével létrehozuk a tényleges SQL adatbázist.

Ez után már tudjuk használni a korábban említett ROI funkciót (Processes fül alatt), mely egy új ablakot fog megnyitni (37. ábra).



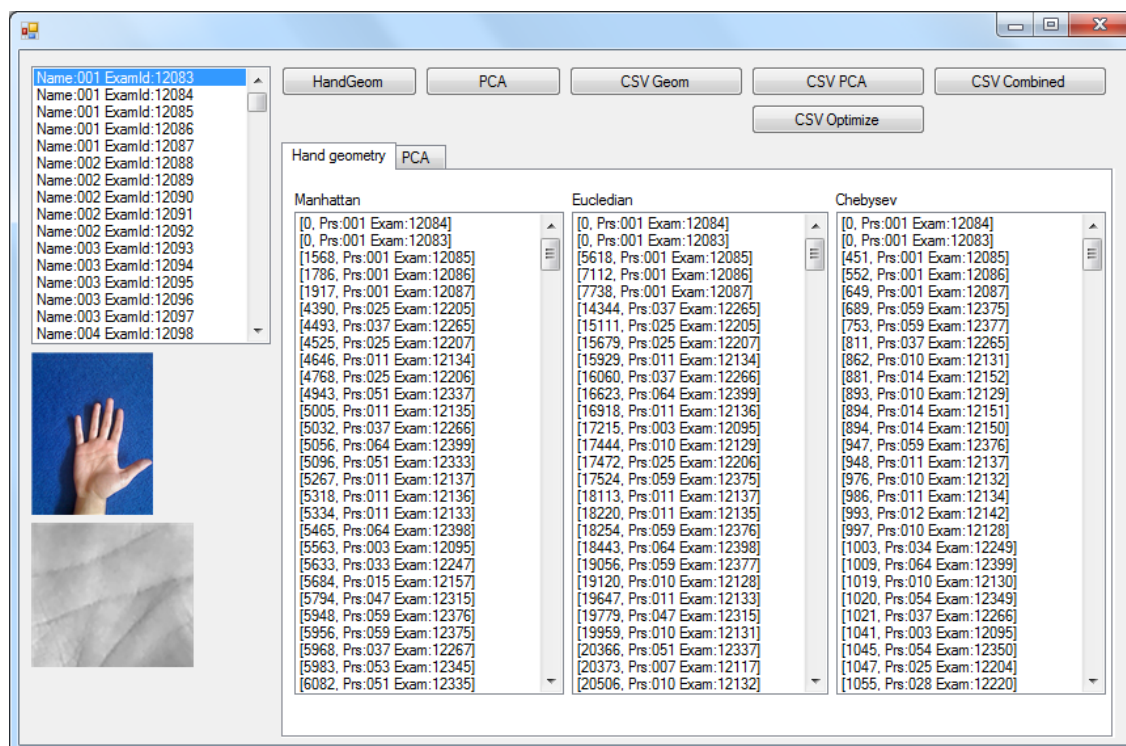
37. ábra. ROI ablak felépítése

A baloldali részen választhatjuk ki a mintát, középen látjuk a minta képét és az azon megtalált referencia pontokat, jobboldalt pedig az ez alapján kinyert ROI képét.

Egy elem maradt hátra, mégpedig a Statistics. Erre kattintva ismét egy új ablak jelenik meg.

4.6.3 Statisztikai ablak

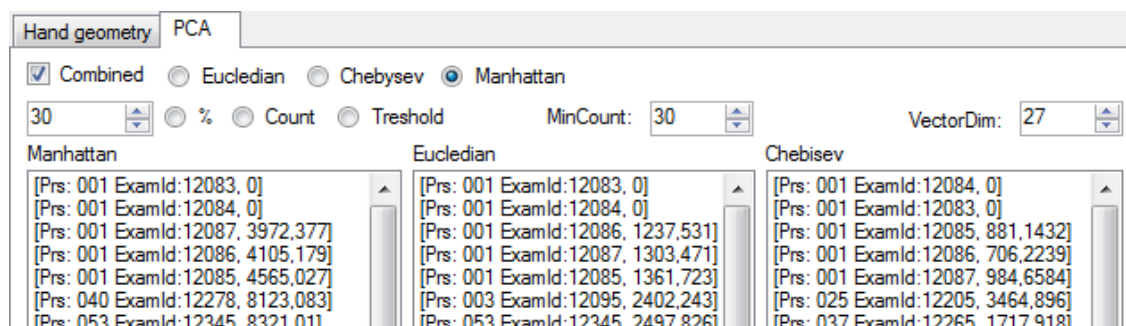
A statisztikai ablak elsődleges célja, hogy tényleges, számmal mérhető adatokat nyerjünk ki a teljes adatbázisra nézve. Ebbe beleértjük, hogy egy kiválasztott mintára a különböző távolságokkal milyen sorrendet kapunk a többihez viszonyítva, és átfogó az összes mintára kiterjedő statisztika számítását és CSV fájlba való lementését is. Felépítését a 38. ábra mutatja.



38. ábra. Statisztikai ablak felépítése

A baloldali részen választhatjuk ki, hogy melyik mintáról szándékozunk távolsági értékeket számolni. Kiválasztás után megjelenik a kiválasztott minta képe és a hozzá tartozó ROI képe is.

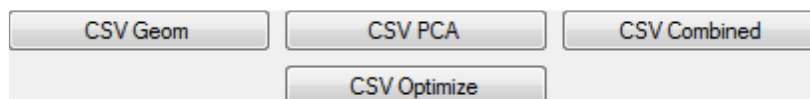
HandGeom gombra kattintva megjelennek a kiválasztott mintához tartozó kézgeometriai távolság értékek. A PCA-ra kattintva hasonló adatokhoz juthatunk, de itt számos részletesebb beállítási lehetőségünk is van (39. ábra).



39. ábra. PCA beállítási lehetőségek

Eldönthetjük, hogy a módszer önálló működésére, vagy a geometriai jellemzővektor által szűkített adathalmazzal való működés eredményére vagyunk-e kíváncsiak. Beállítások közt szerepel, hogy melyik távolsággal létrehozott listát vegyük alapul. Megadhatjuk a szűrés feltételét (bizonyos százalék, megadott darabszám, vagy küszöbérték). Ezen kívül meg kell adni a szűkített lista minimális méretét is. Utolsó feladatunk megadni, hogy hány dimenziós a sajátvektorunk. A gombra kattintás után megjelennek a mind a három távolsággal kiszámolt távolságértékek.

A fennmaradó négy gomb (a feliratok alapján nyilvánvalóan) átfogó statisztika számítására és fájlba való mentésére szolgál (40. ábra).



40. ábra. Hatékonysági statisztika számolása CSV-be

A Geom és a PCA a módszerek önálló működését veszik figyelembe. PCA a korábban említett beállított dimenziót veszi alapul. A kimeneti fájl tartalmazni fogja, hogy az eltérő küszöbértékek milyen hibát jelentenek a teljes adatbázisra nézve.

Az Optimize a dimenziók ideális számának meghatározására szolgál. Esetünkben a gomb 2 és 70 között számol, és minden dimenzió minden küszöbértékéhez megnézi a hibák számát, és a legkisebb kerül a kimenetbe. Ezt felhasználva megtudhatjuk, hogy melyiket célszerű használni.

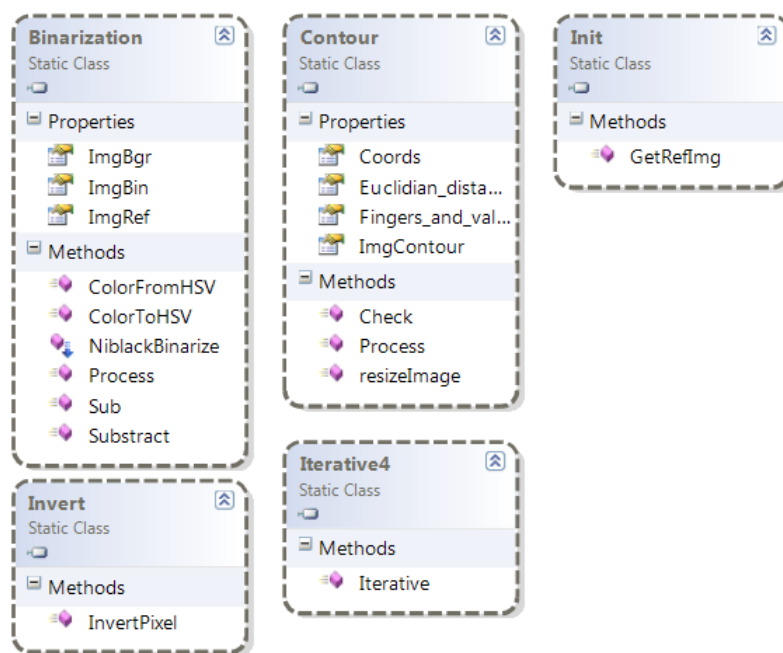
Utolsó gombunk a kombinált módszer értékeit adja eredményül, még hozzá a PCA fül alatt látott beállításokkal számolva.

4.7 A rendszer osztályai

- Preprocess modul: Feladata a kép előkészítése a jellemzővektorok kinyerésére.
- Feature modul: Feladata a preprocess modul által előkészített képből kinyerni a jellemző vektorokat.
- Főmodul: Feladata biztosítani a kezelőfelületet a rendszergazda számára, továbbá ez a modul végzi el az összehasonlítási feladatokat és a tesztelést is.
- A preprocess és a feature modulokban található metódusok nagy része private, ezek funkcionalitásokat tekintve elemi műveletvégzők.

4.7.1 Preprocess modul

A modulhoz tartozó osztályokat és metódusokat a 41. ábra mutatja.



41. ábra. Preprocess modul osztályai és publikus metódusai

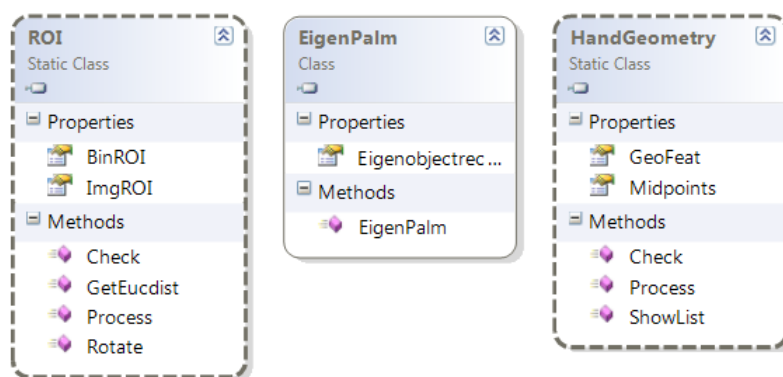
- Binarization: Ez a statikus osztály végzi a binarizálást. A ColorFromHSV, illetve a ColorToHsv végzi a színterek közti váltást. A NiblackBinarize valósítja meg a Niblack binarizálást. A Subtract illetve a Sub végzi a különbségkép generálást, és a

Process metódus vezérli az teljes folyamatot. Tartalmaz továbbá három tulajdonságot (ImgBgr,ImgBin,ImgRef), amik a folyamat eredményét adják.

- Contour: Ez a statikus osztály végzi a kéz kontúrjának a kinyerését, továbbá a kontúrfüggvény előállítását. A Check metódus ellenőrzi, hogy jó-e az adott kép, vagy esetleg hibás (rossz kéztartás, összeérő ujjak stb.). Amennyiben jó, akkor elvégzi a kontúrponatok meghatározását. A Process metódus vezérli a működést. A Coords tulajdonságban található a tenyérkoordináták listája. Az Euclidian_distance tömbben tároljuk az egyes kontúrponatok távolságát a referenciaponttól. A Fingers_and_valleys tömbben a meghatározott csúcsokat és völgyek koordinátáit tároljuk, míg az ImgContour-ban a kéz kontúrjának a képe található.
- Init: Az init osztály végzi a kamerakép inicializálását.
- Iterative4: Ez az osztály végzi el a soros komponens kapcsolást.
- Invert: Invertálja a képet.

4.7.2 Feature modul

A modul osztályairól kép a 42. ábrán.



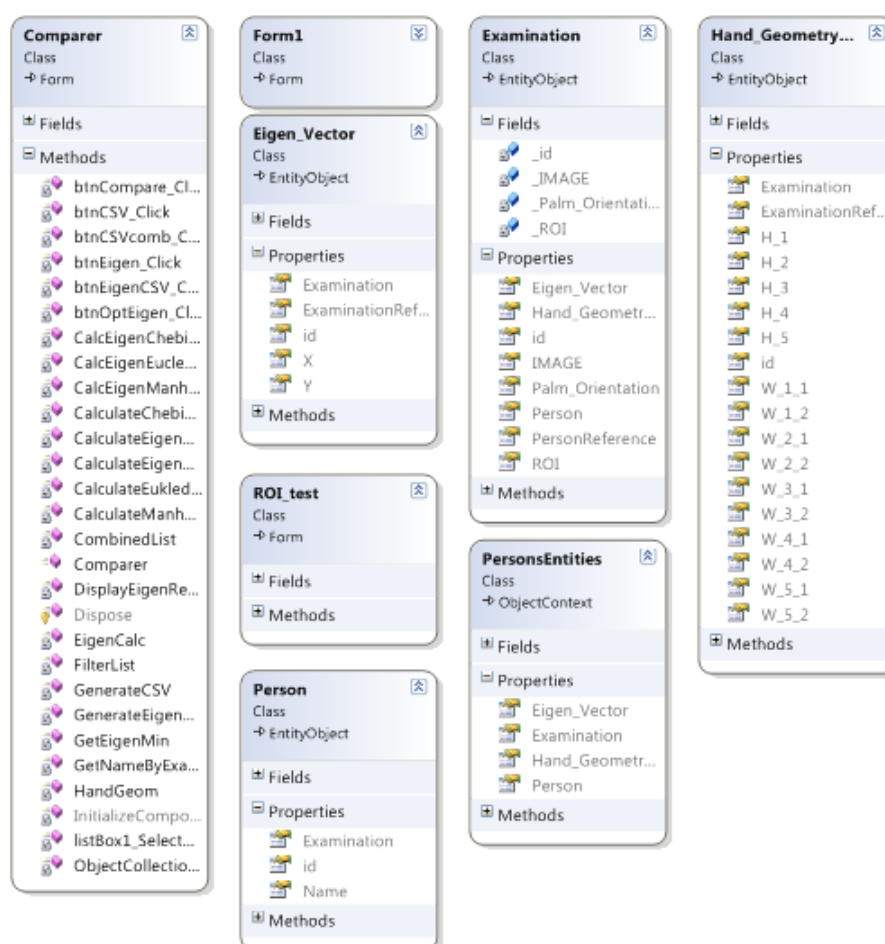
42. ábra. Feature modul osztályai és publikus metódusai

- Handgeometry: Ez a statikus osztály végzi a kézgeometria jellemzők kinyerését, a Process metódus végzi a vezérlést. Tartalmaz továbbá két tulajdonságot. A GeoFeat tárolja a jellemzővektort, míg a Midpoints az ujjtövek középpontjait.

- Roi: Ez a statikus osztály készíti el a geometriai jellemzők felhasználásával a ROI kinyerését. Rotate metódusa végzi a forgatást, amivel minden ROI egy orientációba kerül, a Process metódus végzi a műveletek végrehajtását. Tartalmaz továbbá két tulajdonságot. A BinRoi a binarizált ROI képét, míg az ImgROI az eredeti ROI képét teszi elérhetővé.
- Eigenpalm: Ez a statikus osztály valósítja meg a PCA algoritmust az Eigenobjectrecognizer tulajdonságban található az összes adat, ami az azonosításhoz szükséges.

4.7.3 Főmodul

Ez tartalmazza a legtöbb metódust. Számos különböző metódus szükséges a statisztikai adatok kinyeréséhez (43. ábra).



43. ábra. Főmodul osztályai és publikus metódusai

- Hand_Geometry_Vector, Examination, Person, Eigen_Vector: Az Entity Framework által generált osztályok. Ezeknek az elkészítését a fejlesztőkörnyezet végezte, az adatbázis modell alapján. Az osztályok a táblákat, míg a tulajdonságaik az egyes mezőket jelölik.
- PersonEntities: Az Entity Framework által generált osztály. Ez az osztály létesíti a kommunikációt az adatbázissal, továbbá kezeli az adatbázisban szereplő entitások közötti relációkat, és felállítja a dataset-et, amivel a műveleteket végezzük.
- ROI_test: Ez a form szolgál a ROI algoritmus tesztelésére.
- Comparer: Ez a form végzi a minták összehasonlítását, továbbá a teszteredmények generálását. A Calc metódusok számítják ki a matematikai távolságdefiníciók szerinti eredményeket. A Generate metódusok pedig legenerálják a teszteredményeket.

5 Teszteredmények

Néhány, a témakörben használt kifejezést tisztáznunk kell, mielőtt a grafikonok bemutatásába kezdenénk.

- Verifikációról (Verification) beszélünk, ha a személy azonosítani próbálja magát (akár egy jelszóval), majd a rendszer biometria jellemzők alapján igazolja azt.
- Személyazonosság azonosítása (Identification) az, amikor egy előre begyűjtött adatbázis alapján, kizárólag a személy jellemzői alapján próbáljuk meghatározni annak identitását. "Closed-set" esetén biztos, hogy szerepel a felhasználó (legalább is ezt valamivel bizonyítja, például mágneskártyával). "Open-set" esetén nem biztos, hogy része az adatbázisnak. Ilyen például az FBI ujjlenyomat kereső rendszere is (IAFIS [20]).
- Felismerésnek (Recognition) nevezzük, mikor egyezéseket keresünk adatbázisokban, de nem feltétlenül tartalmaz személyazonosságról információt.

A szükséges adatbázis tartalmát mi magunk hoztuk létre közel 70 embertől egyenként 5-5 tenyérkép felhasználásával. A konkrét eszköz egy Logitech C905-ös típusú középkategóriás webkamera. Ez az adatbázis már kellő méretű ahhoz, hogy algoritmusainkat finomíthassuk, és az egyezési határértékeket beállíthassuk a rendszerből kihozható optimális eredmény eléréséhez.

5.1 Biometrikus rendszerek teljesítmény mutatói

Valamilyen módszerrel mérnünk kell a rendszer hatékonyságát. Számos elterjedten használt mutató létezik, ezek egy részét be is mutatjuk a későbbi tisztább érthetőség érdekében.

Fontosabbak közé tartozik a FAR (False Acceptance Rate) vagy FMR (False Match Rate), azaz hibás egyezések, és FRR (False Reject Rate) vagy FNMR (False Non-Match Rate), azaz észre nem vett egyezések. Ez a két érték a határérték beállításától függően változik, mégpedig fordított arányban.

Ha a TAR-t (True Acceptance Rate) / TMR-t (False Match Rate) és a FAR-t / FMR-t közös diagramon ábrázoljuk (a két különböző tengelyen), akkor megkapjuk a ROC (Relative Operating Characteristic) görbét.

FAR és FMR egyazon diagramon ábrázolásával jól láthatóan megtalálhatjuk azt a határértéket, ahol a két különböző típusú hiba azonos arányban jelentkezik. Ez a két görbe metszéspontja, melyet EER-nek (Equal Error Rate) nevezünk.

Részletesebben is elmagyarázzuk e mutatók és diagramok használatát, a következő fejezetekben megjelenő valós eredményeken keresztül.

5.2 Kézgeometriai keresés első "Open-set" eredményei

Az első statisztikai adatokat kézgeometriai jellemzőkből nyertük ki. 60 személy mintáit használtunk fel. Ezt a tesztet egyrétegű binarizációs maszkkal hajtottuk végre, így nem a teljes adatbázist használtuk fel. A túl sötéteket kivettük, mivel azok nem voltak elegendően pontosak.

A teszt legfontosabb jellemzői a következők:

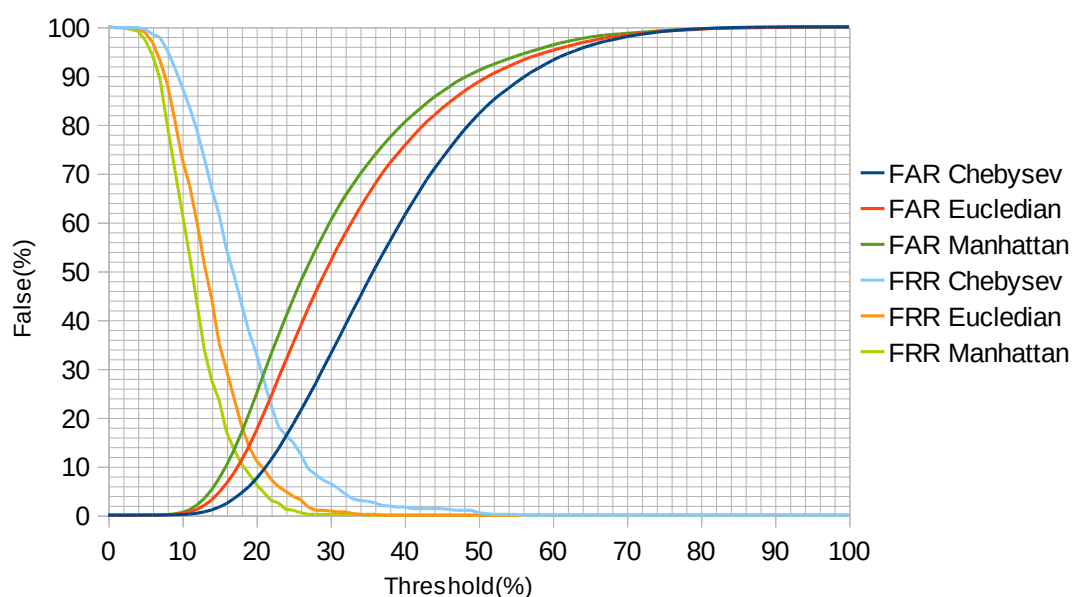
- Az adatbázist nyitottként kezeljük (Open-set).
- Binarizációs maszk első rétegét használja csak (vörös színt komponens különbsége képeznek Otsu algoritmussal történő binarizálása).
- Tartalmazza mind a három általunk használt távolságdefiníciót.
- Kézgeometriai jellemzővektor értékeit egymáshoz arányosítva adjuk meg.

A statisztikához arra volt szükség, hogy egy táblázatba kinyerjük, hogy a megadott határértékekre nézve egy adathalmazban mennyi a helyes és helytelen találatok száma (vagy aránya). Egy ilyen általunk használt táblából származik a 44. ábrán látható minta.

%	0	1	2	3	4	5	6	7	8	9	...
FAR Chebysev	0,00	0,00	0,00	0,00	0,00	0,00	0,01	0,01	0,02	0,05	...
FAR Euclidian	0,00	0,00	0,00	0,00	0,00	0,00	0,01	0,02	0,06	0,15	...
FAR Manhattan	0,00	0,00	0,00	0,00	0,00	0,01	0,01	0,03	0,10	0,25	...
FRR Chebisev	100,00	99,83	99,83	99,83	99,83	99,50	98,50	97,83	95,17	91,67	...
FRR Euclidian	100,00	99,83	99,83	99,83	99,67	98,83	96,83	93,17	88,33	81,00	...
FRR Manhattan	100,00	99,83	99,83	99,50	99,17	97,33	94,17	89,33	79,83	70,83	...

44. ábra. Hibák százalékos arányai a különböző határértékeknél

A módszer általános hatékonyságára voltunk kíváncsiak, így a teljes adatbázisra összesített FAR / FRR diagramot generáltuk a kinyert táblázatból, mind a három általunk használt távolságra (45. ábra). Egy ilyen diagramon a vízszintes tengely jelenti a küszöbértéket (megengedett távolság két minta között), míg a függőleges tengelyen jelenik meg a hibák százalékos aránya. Az FRR értékek 100-tól, az FAR értékek pedig 0-tól indulnak. Ez logikus is hiszen, ha a megengedett hiba 0, akkor nem tudunk téves egyezést generálni, viszont minél szűkebb a tűréshatár, az egyezések elmulasztása annál jobban növekszik (tehát fordított arányban áll a két érték).



45. ábra. Kézigéometria módszer FAR / FRR diagramja

A vízszintes tengely értékeit az előforduló legnagyobb távolsághoz kell viszonyítani, de típusonként külön-külön tényleges értéket reprezentálnak. Láthatjuk, hogy közel azonosak, de minimálisan a Manhattan kereszteződése (EER) található a legalacsonyabb értéken, vagyis ennek a távolságnak a használata eredményezi várhatóan a legkevesebb hibát. A célunk, hogy mind a két típusú hiba alacsony legyen.

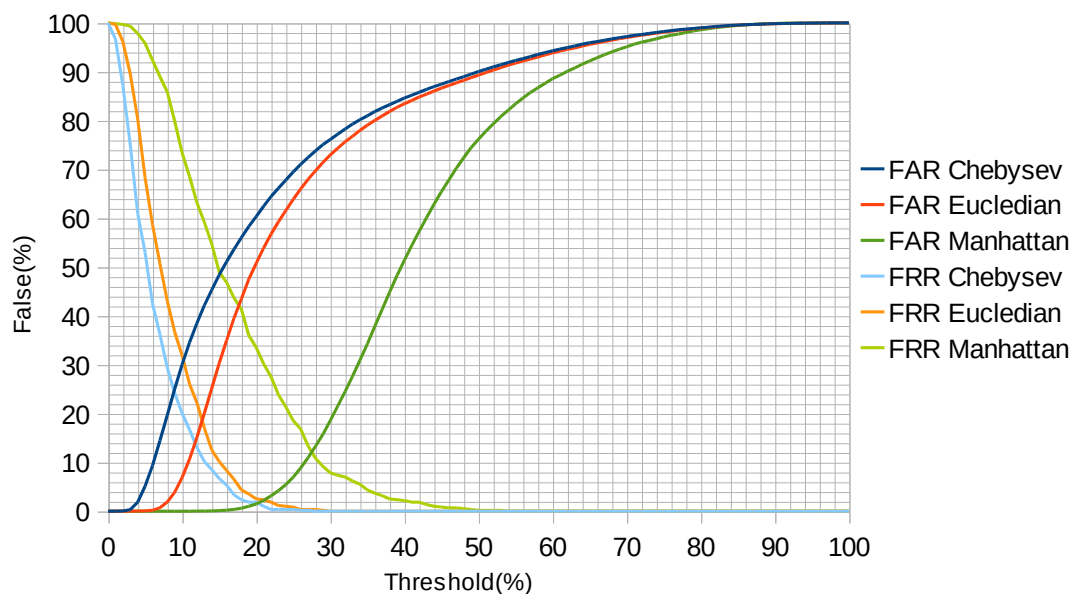
El lehetne gondolkozni, hogy melyik értéket célszerűbb alacsonyabban tartani, de mi erre a kérdésre nem fogunk kitérni. A rendszert legjobban az EER által megadott hibával lehet jellemezni, így ezentúl mi is ezt fogjuk kiemelni az összes teszt során.

Az értékek a következők hatékonysági sorrendben:

- Manhattan: 17%-os határértékkel, 13%-os várható hibával.
- Euklidészi: 19%-os határértékkel, 14%-os várható hibával.
- Csebisev: 24%-os határértékkel, 16%-os várható hibával.

5.3 PCA módszer első "Open-set" eredményei

Hasonlóan a kézgeometriaihoz, természetesen sajátvektoros módszernél is kíváncsiak voltunk, hogy milyen eredményt hoz (46. ábra). A sajátvektor dimenziójának 10-es értéket választottunk, és az előbbiekkal megegyező adatbázist és beállítást használtunk.



46. ábra. PCA módszer FAR / FRR diagramja

Láthatjuk, hogy itt már jelentősebb különbségek vannak a különböző távolsági módszerek használata között. Ennek ellenére meglepő módon, körülbelül azonos eredményt értünk el Manhattan távolsággal.

A konkrét adatok a következők:

- Manhattan: 27%-os határértékkel, 13%-os várható hibával.
- Euklidészi: 13%-os határértékkel, 19%-os várható hibával.
- Csebisev: 9%-os határértékkel, 25%-os várható hibával.

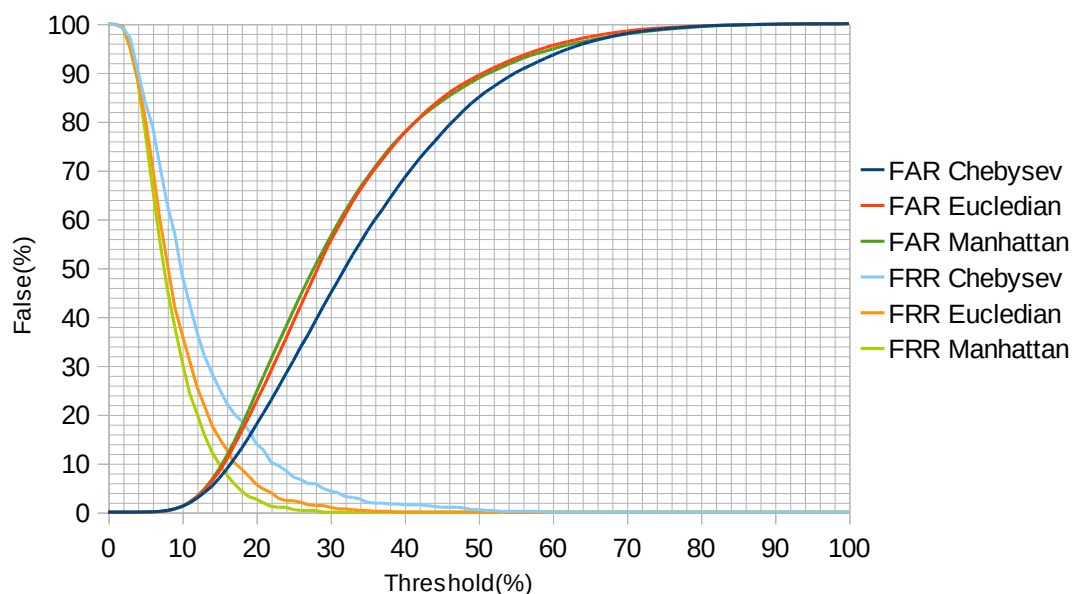
A PCA módszer hatékonysága függ a ROI kinyerésének módszerétől és a sajátvektor méretétől, így ennél jobb eredmény is lehetséges, e paraméterek módosításával (pl. dimenziók száma). Dimenzió optimalizációt egyelőre nem hajtunk végre.

5.4 Binarizáció és jellemzővektor módosításának eredménye

Az itt látható statisztikai eredményeket már a finomított maszkkal futtattuk (tartalmazza mind a három réteget). A jellemzővektorban arányokról konkrét pixelbeli hosszúságokra álltunk át, mivel az összes kép azonos távolságból készült, így nincs haszna viszonyszámokkal dolgozni.

5.4.1 Kézeometriai statisztikák

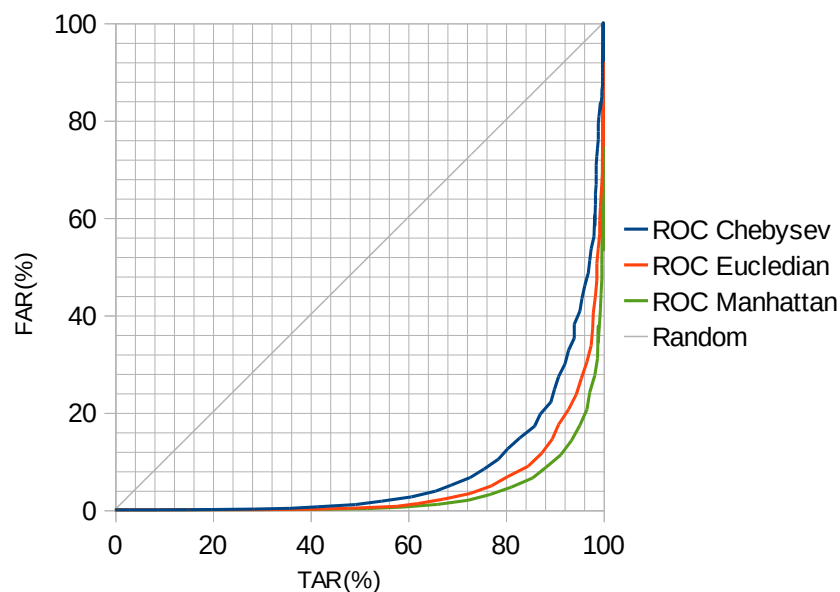
Tekintsük át először a kézeometriai módszer új eredményeit (47. ábra).



47. ábra. Kézeometriai módszer FAR / FRR diagramja fejlesztések után

Az eddigi 27%-os hiba helyett 19%-ot értünk el. Azt tudni kell, hogy az adatbázis készítésekor ellenőrizve lett, hogy a binarizáció megfelelően hajtódik-e végre. Mivel itt javult ennek pontossága (jobban közelíti a kéz kontúrját, mint a korábban), ezért tartalmaz több olyan képet melyeken az ujjak nem voltak kellően szétnyitva. Továbbá ebben az esetben már a korábban kihagyott sötét képeket is felhasználtuk. Ezen hátráltató tényezők ellenére is sikerült javulást elérni.

A következő, úgynevezett ROC diagram (48. ábra) a működési karakterisztikát szemlélteti. Sok esetben ezzel szokták a hasonló rendszerek teljesítményét jellemezni.



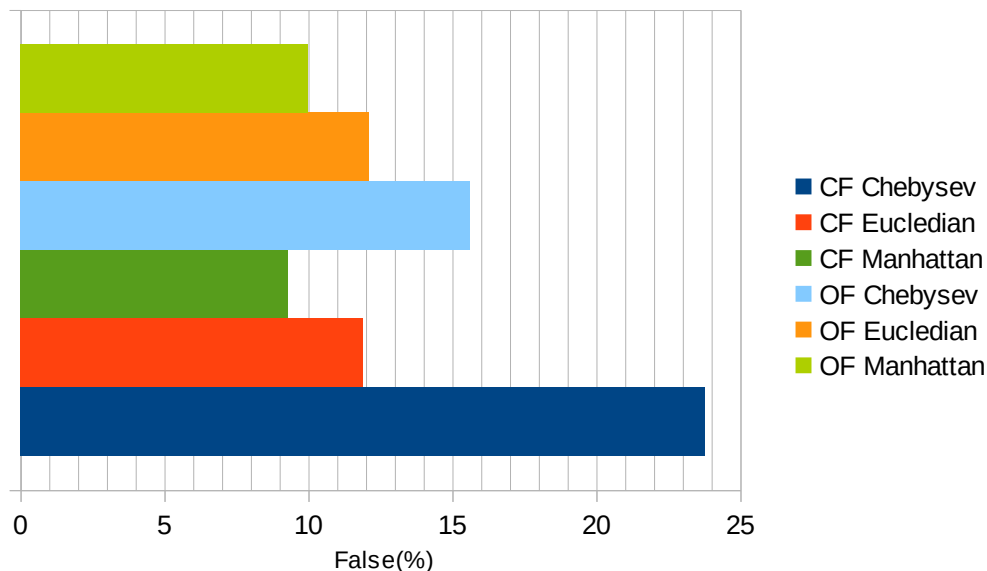
48. ábra. Kézeometriai módszer ROC diagramja

A hibás találatok és a sikeres találatok összefüggését mutatja be ez a diagram. Egy véletlenszerűen működő rendszernél az átlóban helyezkednének el az értékek (szürke vonal). A mérőszámot egy négyszög alakú terület nagysága adja meg, melynek három sarka a két felső és a bal alsó pontok, a negyedik pedig a görbén elhelyezkedő valamely pont. Jelen esetben nem ezzel a módszerrel fogjuk jellemezni a működést, de néhány esetben ezt a diagramot is közöljük.

5.4.1.1 Összesített hibadiagram (Open-set és Closed-set)

Ha zárt (Closed-set) rendszerről beszélünk, akkor nem szükséges megadnunk a határértéket. A személy előzetesen "bemutatkozik" (például egy mágneskártyával vagy egyedi azonosító/jelszóval) és csak azt kell eldönteni, hogy a leghasonlóbb elem az adatbázisban tőle származó minta-e vagy sem. Ha jobban átgondoljuk, ennek a módszernek is van jelentősége, hiszen egy magas pontosságú, másodlagos megerősítést jelenthet a biometria azonosítás egy már régóta elfogadott rendszer (mágnes kártya) mellé. Továbbá, abban az esetben, ha kártyánkat eltulajdonítják, esetleg jelszavunk kitudódik, akkor is nagy valószínűséggel a visszaélést elkövetni próbáló személy egyedi biometria jellemzői nem hasonlítanak a miénkre.

Az 49. ábrán egyazon diagramon hasonlíthatjuk össze, az Open-set rendszer által elérhető legjobb értékeket (OF), a Closed-set rendszer által elérhető értékekkel (CF).



49. ábra. Kézigometriai módszer egyesített hibadiagramja

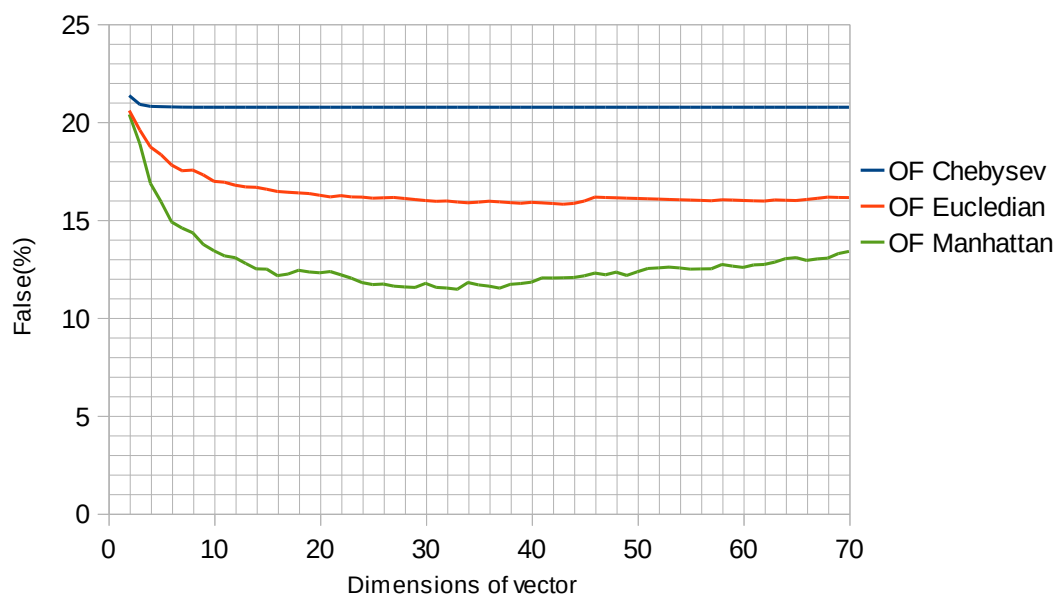
A konkrét hibaszázalék értékek a következők:

- Manhattan: Open-set 9,98 %, Closed-set 9,28 %.
- Euklidészi: Open-set 12,11 %, Closed-set 11,88 %.
- Csebisev: Open-set 15,61 %, Closed-set 23,77 %.

Zárt rendszer esetén nagyobb szórást tapasztalunk, de Manhattan távolság esetén mind a két típusú rendszerrel egyaránt 10%-on belülre kerülünk.

5.4.2 PCA statisztikák

Ezúttal szeretnénk megtalálni az optimális dimenziószámot is, így ennek számításával kezdjük. Ezt a következő módszer szerint keressük. Lefuttatjuk a mérést egy minimum és egy maximum dimenzióméretre megadva, és minden dimenzió belül az eltérő küszöbértékekhez tartozó lehető legjobbat keressük. Ennek grafikonját látjuk az 50. ábrán, mely egy nyitott rendszer eredményeit mutatja.



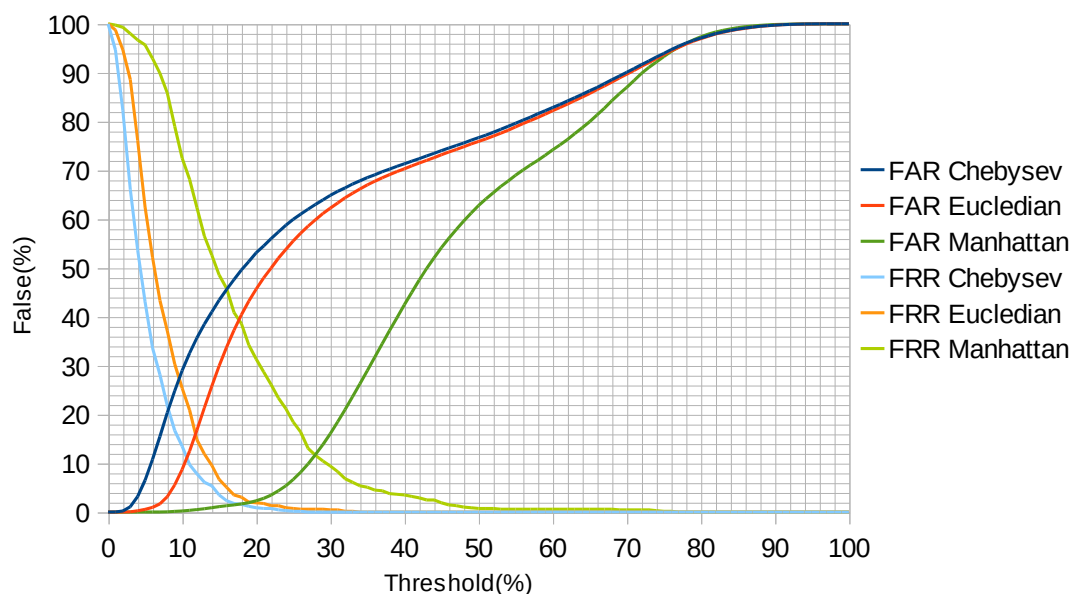
50. ábra. PCA optimalizálási diagramja (Open-set)

Alapvetően az várható, hogy egy bizonyos méret után a hatékonyság már nem fog nőni. Sőt, ahogy az ábrán is látjuk még romlani is kezdhet. Ezen okok miatt választottunk egy maximális 70 dimenziós vektort.

A legjobb értékek és hozzájuk tartozó dimenziók számai a következők:

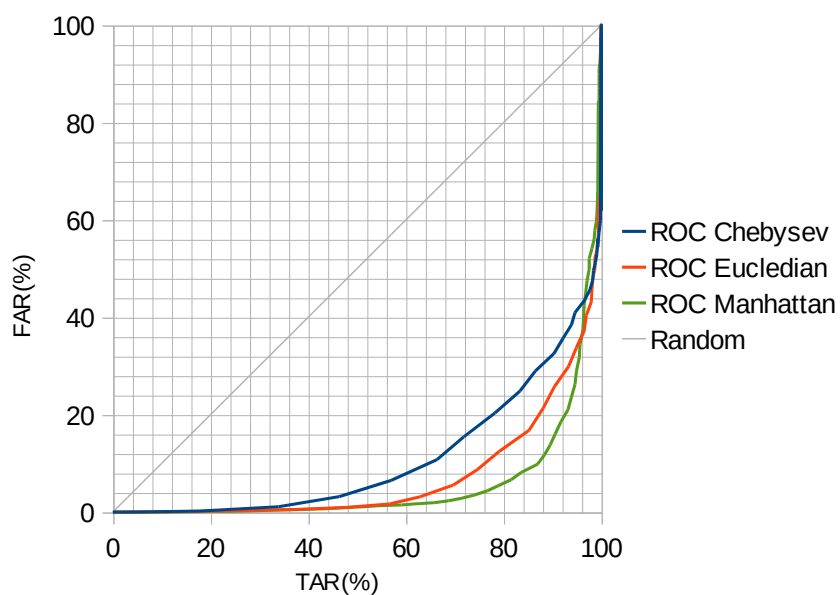
- Manhattan: 33 dimenziós, 11,44%-os várható hibával.
- Euklidészi: 43 dimenziós, 15,78%-os várható hibával.
- Csebisev: 8 dimenziós, 20,74%-os várható hibával.

A legjobb eredményekhez FAR / FRR diagramot is generálunk. Ez látható az 51. ábrán.



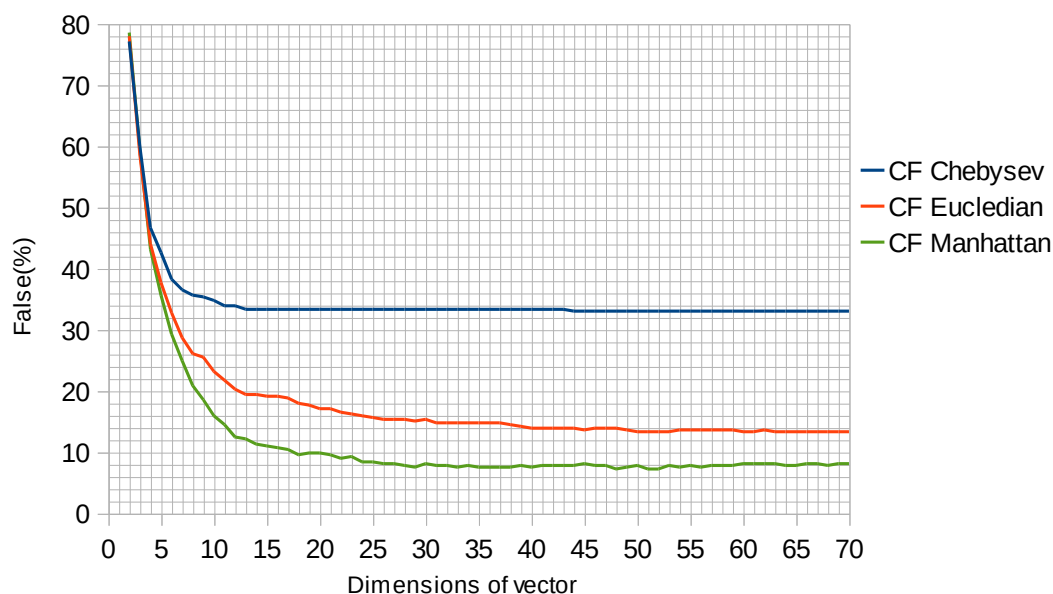
51. ábra. PCA optimalizálási utáni FAR / FRR diagramja (Open-set)

Az ehhez tartozó ROC diagramot is közöljük (52. ábra).



52. ábra. PCA módszer ROC diagramja

Viszont az optimalizáció eredménye nem feltétlenül lesz azonos Closed-set esetén, így azt külön kell keresnünk. A számításokat az 53. ábra mutatja.



53. ábra. PCA optimalizálási diagramja (Closed-set)

Várakozásunknak megfelelően eltérő értékeket kapunk. Szembetűnő, hogy jóval magasabb dimenziók jönnek ki legjobb értékre, bár elég alacsony különbségek jelentkeznek, így előfordulhat, hogy célszerűbb a sebességet inkább előnyben részesíteni, annál hogy minimálisan jobb eredményt kapjunk.

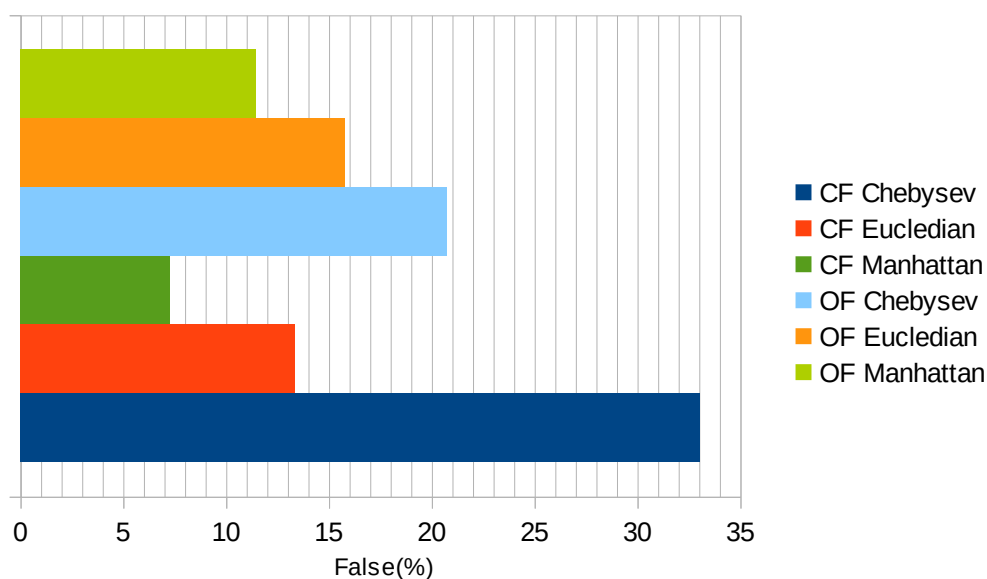
A pontos értékek a következők:

- Manhattan: 48 dimenziós, 7,25%-os várható hibával.
- Euklidészi: 50 dimenziós, 13,33%-os várható hibával.
- Csebisev: 44 dimenziós, 33,04%-os várható hibával.

A két típusú rendszer összesítése az 54. ábrán látható és a hozzá tartozó pontos értékeket itt adjuk meg az ábrának megfelelő sorrendben (fentről lefelé):

Sorrendben az értékek: 11,44%; 15,78%; 20,74%; 7,25%; 13,33%; 33,04%.

Dimenziók száma: 48; 50; 44; 33; 43; 8.



54. ábra. Összesített hibadiagram optimalizált PCA esetén

5.5 Kombinált teszteredmények

Ahogy azt korábban említettük, ennél az összetett módszernél azt az elvet követjük, hogy a PCA kisebb adathalmazra hatékonyabban működik (mivel nagy adathalmaz esetén csökken a variancia), így célszerű szűkített adathalmazzal dolgozni. A kézgeometriai jellemzővektor alapján meghatározott részhalmazzal használjuk fel a továbbiakban. A szűkítésre több különböző megoldást használhatunk, melyeket a programban implementáltunk is. Továbbadhatjuk az első megadott fix darabszámot, vagy a teljes adathalmaz egy bizonyos százalékát. Ezek egyszerűbb lehetőségek, de terveztünk egy összetettebbet megoldást is.

Ennek logikája, hogy a kézgeometriai statisztikából ismert lehető legalacsonyabb küszöbértéket választjuk, de úgy, hogy (megközelítőleg) nulla legyen az FRR. Ez az eltérő távolságoknál körülbelül 30-40 %-os értéket jelent.

Másik tapasztalati információ, hogy közelítőleg ismerjük azt a dimenziószámot az egyes távolságoknál, melyekkel elérjük a közel maximális pontosságot (ennél nagyobb már a sebesség drasztikus lassulását okozza, jelentős nyereség nélkül). Azt feltételezve, hogy adathalmaz méretétől függetlenül ezek eredményezik a legjobb értékeket, meghatározhatjuk, hogy a geometriai algoritmus által talált legjobb hány találatot

(legkisebb távolságú mintákat az adatbázisban) kell átadni (minimálisan), még akkor is, ha a küszöbérték szerint nem kellene. Azért van ez a minimális mennyiség, mert az adathalmaz méretének nagyobbnak kell lennie a dimenziók számánál.

5.5.1 Optimalizálás

Ezen a szinten a statisztikai adatok készítése már rendkívül lassú. Ennek oka részben, hogy minden mintához más szűkített adathalmaz tartozik. Vegyük észre, hogy ez valójában nem okoz gondot, hiszen egy minta esetén az eddigiekkel megegyezően csak egyszer kell sajátértékeket számolni a PCA módszerhez.

Számos dimenzióra kellene lefuttatni ezt a tesztet, ráadásul a különböző távolságokhoz eltérő a dimenziók optimális száma a korábbi tapasztalatok szerint. Rengeteg teszt után arra a következtetésre jutottunk, hogy Closed-set rendszer esetén továbbra is a Manhattan távolság hozza a legjobb eredményt, azonban Open-set rendszer felhasználása esetén megváltozott a sorrend és az Euklidészi bizonyult a legjobbnak.

Az előzőek tükrében tesszük közzé a végső eredményeket.

5.5.2 Feltételezett optimális beállítás teszteredménye

A legjobb eredményt a Manhattan távolsággal hozta a geometriai módszer, így itt ezt használjuk, 30 %-os küszöbértékkel.

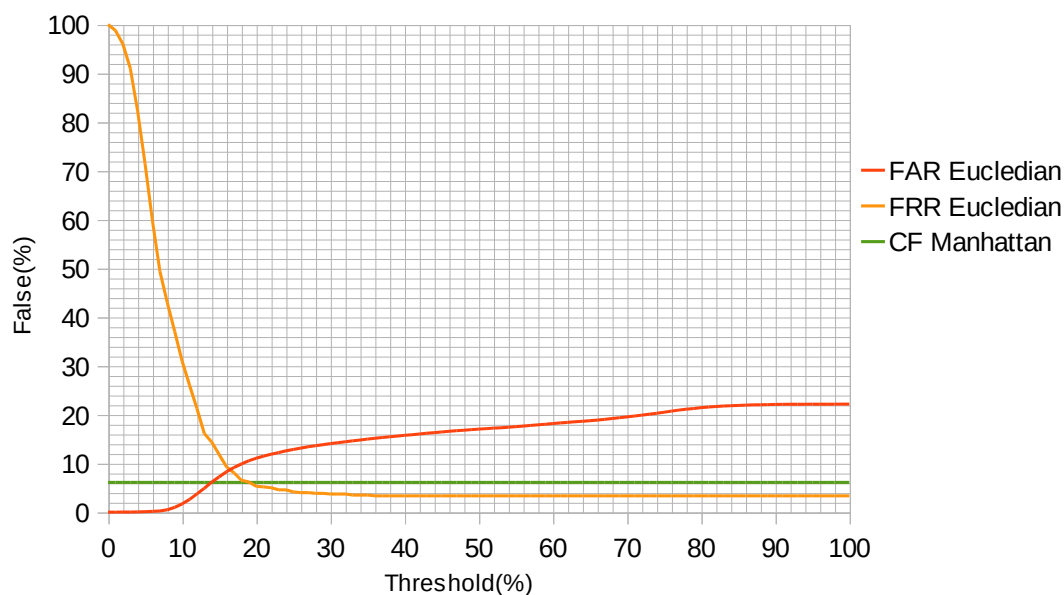
A korábban leírt elveket követjük. A Manhattan távolsághoz 33 dimenziós vektort, míg az Euklidészihez 43 dimenziós vektort használunk.

Tehát összefoglalva, a lefuttatott teszt fontosabb tulajdonságai a következők:

- Háromrétegű binarizációs maszk.
- 15 elemből álló geometriai jellemzővektor abszolút pixel távolságokkal.
- Manhattan távolság használata a geometriai összehasonlításhoz.
- 30%-os küszöbérték az adathalmaz szűkítéshez.
- Minimum 34 elemből álló szűkített adathalmaz, Closed-set rendszer esetén.
- Minimum 44 elemből álló szűkített adathalmaz, Open-set rendszer esetén.
- 33 dimenziós sajátvektor, Closed-set rendszer esetén.

- 43 dimenziós sajátvektor, Open-set rendszer esetén.
- Manhattan távolság használata PCA-hoz, Closed-set rendszer esetén.
- Euklidészi távolság használata PCA-hoz, Open-set rendszer esetén.

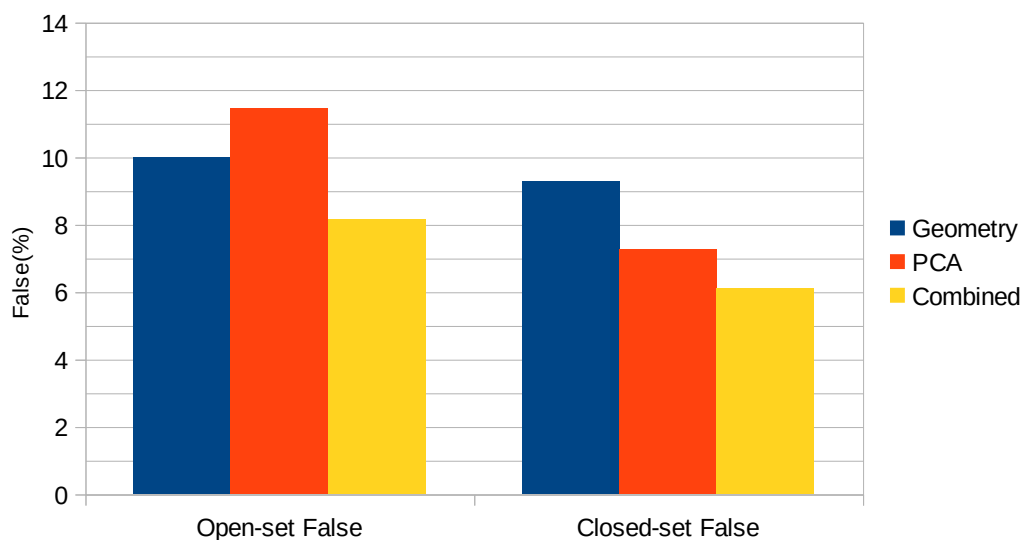
A kombinált teszt FAR / FRR diagramját az 55. ábrán láthatjuk, mely tartalmazza a Manhattan távolság felhasználásával elért Closed-set eredményt is.



55. ábra. Kombinált módszer FAR / FRR diagramja Closed-set értékkel együtt

Az első szembetűnő észrevétel, hogy a FAR meg sem közelíti a 100%-ot. Ez, ha belegondolunk egyértelműen várható volt, hiszen a jelentősen különböző méretekkel rendelkező kezek nagyobb részét kiszűrjük. Vegyük észre, hogy az FRR viszont nem éri el a 0-át, mivel esetenként hibásan kiszűrődik néhány azonos személytől származó minta is, így akármilyen küszöbértéke van a PCA-nak nem lesz találat az adatbázisban.

Zárásként egy összehasonlítást mutatunk be, mely a különböző módszerek által elért legjobb eredményeket kívánja prezentálni (56. ábra).



56. ábra. A különböző módszerek által elért legjobb eredmények

A pontos adatok a következők:

Open-set rendszer:

- Kézteometriai: Manhattan távolsággal, 9,98% várható hiba.
- PCA: Manhattan távolsággal, 11,44% várható hiba.
- Kombinált: Euklidészi távolsággal, 8,16% várható hiba.

Closed-set rendszer:

- Kézteometriai: Manhattan távolsággal, 8,28% várható hiba.
- PCA: Manhattan távolsággal, 7,25% várható hiba.
- Kombinált: Euklidészi távolsággal, 6,09% várható hiba.

Sikerült megmutatni, hogy igaz az eredeti feltevés, és a módszerek kombinálásával azonos adatbázisból jobb eredményt lehet kihozni.

Valószínűleg van még lehetőség a hatásfok további növelésére, de nem tartjuk kis eredménynek, hogy sikerült 90% fölötti eredményt elérnünk, mely bizonyítja, hogy kiváló rendszert lehet létrehozni kéz és tenyéralapú technikákkal is.

6 Összefoglalás

Az elért eredményekkel elégedettek vagyunk. Több elméleti módszert megismertünk a szakirodalmakból, melyek jó alapnak bizonyultak, annak ellenére, hogy gyakorlati megvalósítással nem nagyon találkoztunk.

Egy igen nagyméretű adatbázissal dolgozhattunk, melyet mi magunk állítottunk össze. Az adatbázis sok, eltérő körülmények között (világosban, szórt fényben, árnyékosabb időszakban) készült képet tartalmaz, köztük néhány nem megfelelőt is (homályos, hibás kéztartásos). Tapasztalataink azt mutatják, hogy ezen körülményekre is jól felkészítettük a rendszerünket.

Megvalósításra került két eltérő módszer (geometriai és PCA). Ezeknek már a vizsgálatok kezdeti szakaszában tapasztalt hatásfoka is egész jónak bizonyult. Az algoritmusok csiszolásával, módosításával és a módszerek eredetileg tervezett kombinálásával ezen jelentős mértékben sikerült javítani.

További jelentős potenciált látunk a projektben. A továbbiakban a továbbfejlesztési lehetőségeket és a későbbi hasznosíthatóságot tárgyaljuk a dolgozat befejezéseként.

6.1 Fejlesztés, megvalósítás

A megvalósított elemek felhasználásával, átalakításával teljesen reális alapját látjuk annak, hogy egy tényleges integrált biztonsági rendszert lehessen létrehozni. Persze ehhez számos további feltételnek, követelménynek kell teljesülnie. A megvalósításához szükséges legfontosabb feltételekről lesz szó ebben az alfejezetben.

6.1.1 Input eszköz

Az inputeszköz védelmének érdekében szükség van egy megerősített kamera dobozra, mely védi a hardverelemeket. Az általunk használt kameránál komolyabbra van szükség. Emellett képesnek kell lennie érzékelni, hogy élő ember kezéről készül-e a kép. Ezt több módszerrel is meg lehet állapítani. Egy infrakamerával például érzékelni lehet vérereket, és ha egy rövid felvételt készítünk vele, akkor annak elemzésével már eldönthető a kérdés. De helyette, vagy mellette egy hőkamerát is fel lehet használni ugyanerre a célra. Az, hogy a kezünket a dobozban egy átlátszó lapra helyezzük-e, és

vannak-e ujjelválasztók, vagy csak a levegőben tartjuk-e a kezünket, és úgy készül felvétel már részletkérdés, hiszen szoftveresen mindenképp ellenőrizni kell, hogy megfelelő-e az input kép vagy sem.

6.1.2 Szoftveres hibaszűrés

A legeggyértelműbb jele annak, hogy a személy kezéről nem megfelelő kép készült az az, ha a kontúrbejárás, komponens analízis hibát jelez. Ha az ujjak jelentősen összeérnek, azt az ujjközök és ujjvégek keresésekor részben ki tudjuk szűrni, hiszen valószínűleg ebben az esetben is hibát eredményez. Ha mégse, és a megfelelő mennyiségű csúcsot és völgyet találja meg az algoritmus a kontúrfüggvényen, akkor annak további elemzésével lehet hibát szűrni (részben összeérő ujjak). Emellett, ha a ROI képébe háttérpixel kerül, az is egyértelmű jele annak, hogy nem megfelelő az input kép. Ha mindezek ellenére jónak bizonyul az input, akkor az már elegendő valószínűséggel megfelelő.

6.1.3 Sebesség

Nem nagyon esett szó eddig az algoritmusok sebességéről, de az ismert, hogy egy teljes ellenőrzés a teljes adatbázisban is bőven 10 másodperc alatt van egy gyengébb teljesítményű számítógépen is. Ha ezres nagyságrendű adatbázisról beszélünk, akkor sem lehetne probléma több okból kifolyólag sem. Ha kontrollált fényviszonyok között, jó kamerával készül a felvétel, akkor a most megvalósított binarizációnál sokkal egyszerűbb és gyorsabb algoritmus is elegendő. Jelenleg a folyamat egyik leglassúbb eleme a binarizációs maszk Niblack rétege, melyre az előbb említett esetben egyáltalán nincs is szükség. A nagyobb adathalmaz valóban lassítja a PCA-t, de vegyük figyelembe, hogy a pontosabb binarizálás pontosabb geometriai jellemzőket is jelent, így a szűkítés is nagymértékben javulhat. Továbbá fontos megjegyezni, hogy a folyamatok nagy része olyan, hogy jól ki tudja használni a korszerű számítógépek adta lehetőségeket, így e téren is jelentős sebességnövekedést várhatunk el.

6.1.4 Egyéb rendszertulajdonságok

Az általunk használt adatbázishoz hasonlóan célszerű több mintát eltárolni a magasabb találati esély érdekében.

Mivel a tenyér az idő előre haladtával együtt fokozatosan változik, ezért érdemes lehet megfontolni, hogy minden sikeres autentikáció során lecseréljük a legrégebbi mintát. Ezt könnyedén meg lehet tenni a felhasználó számára láthatatlanul. Azonban figyelembe kell venni, hogy egy esetleges hibás egyezés esetén ennek milyen hatása lenne.

Láthattuk a statisztikai eredményekben, hogy zárt rendszer esetén kisebb hibaszázalékkal dolgozhatunk. Ennek megvalósítása is lehetséges, ha kiegészítjük az integrált eszközünket, például egy chip kártyás modullal, melybe akár PIN kód használata is beépíthető.

Jelenleg a rendszer csak jobb kézre működik, de egy olyan metódus beépítése, mely el tudja dönteni, hogy melyik kézről van szó, nem okoz problémát. A kontúr függvény vagy geometriai jellemző vektor alapján könnyedén meg tudjuk állapítani ezt.

6.2 Abstract

We decided to do this project because we were interested in computer vision, and biometrical recognition. Our primary objective was to construct a system which uses a number of recognition methods and a multi level decision system, with a high precision level.

The process of recognition that we are using in our system is the following: First step is the procurement of a signature, and contour of the hand by preprocessing methods. Second, with the contour coordinates the system can extract the geometric feature vector from the palm, which will be used for the geometric recognition module. The next step is to extract the Region of Interest from the palm. Our definition for the ROI is a rectangle which is located in the middle of the palm. With this ROI we built a recognition method, which is based on the variance of the patterns (Principal Component Analysis), which is also the second tier of our system. By these two methods we have two feature vectors: the first is the geometric feature vector (15) and the second is the eigenvector (from PCA) with a customizable size. We made the comparison of our samples with three different mathematical distance definitions: Manhattan-Minkowsky, Chebysev, and Euclidian. We tested the algorithms on a

database with 69 people, and from each of them we gathered 5 pictures of their right palm. We built up the database ourselves, by making a fix environment, with a homogenous blue background, and shooting from 35-45 cm height. We used two measurements for the successfulness of recognition: the False Acceptance Rate, and the False Rejection Rate, and we experimented with databases set to be open-set and closed-set. Open set is not using any other methods to authenticate (PIN, chipcard), but using thresholds, while a closed set is using preidentification and the problem reduces to the decision that the sample which is closest to his, is valid or not.

First, we calculated the results of the geometrical module; we decided to filter these result by the effectiveness of each mathematical distance. The PCA recognition method is based on the variance of an individual sample related to the whole database. If the possible set of matches is low, the system can recognize an individual sample much easier, because if this set is too large the variance between the samples lowers.

With these two levels of our system, we reached the following result:

Only geometrical			Only PCA		Combinated	
	Failure		Failure		Failure	
	Open-set	Closed-set	Open-set	Closed-set	Open-set	Closed-set
Manhattan	9,98%	8,28%	11%	7,25%		
Eucledian					8,16%	6,09%

7 Irodalomjegyzék

- [1] Robert D. Olsen, Sr., "A Fingerprint Fable: The Will and William West Case", Identification News, Vol. 37, No. 11, 1987
- [2] Anil K. Jain, Aroon Ross, Salil Prabhakar, "An Introduction to Biometric Recognition", IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Image- and Video-Based Biometrics, Vol. 14, No. 1, 2004
- [3] Trends in Japan, "Place Your Hand On The Scanner", <http://web-japan.org/trends/science/sci050510.html>, 10 May 2005 accessed 29 October 2011
- [4] ATR Systems, "HandPunch series", <http://www.handpunch.com/>, accessed 25 October 2011
- [5] Fujitsu Australia, "Fujitsu launches new compact, high-performance palm vein biometric authentication device", <http://www.fujitsu.com/au/news/pr/archives/2006/20060308-02.html>, 9 March 2006 accessed 29 October 2011
- [6] Nicolae Duta, Anil K. Jain, Kanti V. Mardia, "Matching of Palmprints", Pattern Recognition Letters, Vol. 23, Issue 2, February 2002
- [7] Árendás Csaba, Bachraty Gergely, Jeges Ernő, Körmöczi Csaba, Molnár Roland, "Integrált Biometrikus Azonosító Rendszerek", BME Méréstechnika és Információs Rendszerek Tanszék, 2005
- [8] Lin Hong, Anil K. Jain, Sharath Pankanti, Ruud M. Bolle, "Identity Authentication Using Fingerprints", AVBPA, 1997
- [9] Chris Roberts, "Palm and Hand", Biometric Technologies, 2006
- [10] Anil K. Jain, Nicolae Duta, "Deformable Matching of Hand Shapes for Verification", IEEE Computer Society, Vol. 2, 1999

- [11] Alvy Ray Smith, "Color Gamut Transform Pairs ", Computer Graphics, Vol. 12, No. 3, August 1978
- [12] http://en.wikipedia.org/wiki/HSL_and_HSV, "HSL and HSV", Wikipedia, accessed 02 January 2012
- [13] Nobuyuki Otsu, "A Threshold Selection Method from Gray-Level Histograms", IEEE Transactions on Systems, Man and Cybernetics, Vol. 9, No. 1, pp. 62-66, January 1979
- [14] W. Niblack, "An Introduction to Digital Image Processing", Prentice Hall, pp. 115-116, 1986
- [15] H. Samet, M. Tamminen, "Efficient Component Labeling of Images of Arbitrary Dimension Represented by Linear Bintrees", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10, Issue 4, pp. 579-586, July 1988
- [16] I. T. Jolliffe, "Principal Component Analysis, Second Edition", Springer, 2002
- [17] Craig Lee, "ADO.NET Entity Framework Unleashed", Sams, 2010
- [18] Julie Lerman, "Programming Entity Framework", O'Reilly Media, 2009
- [19] Elena Deza, Michel Marie Deza, "Encyclopedia of Distances", Springer, 2009
- [20] FBI, "Integrated Automated Fingerprint Identification System", http://www.fbi.gov/about-us/cjis/fingerprints_biometrics/iafis/iafis, accessed 06 November 2011

8 Melléklet

8.1 Az adatbázis montázsja

