Doktori (PhD) értekezés

Ember-számítógép interakciók: az emberi arc modellezése

Bertók Kornél

Témavezető: Dr. Fazekas Attila

DEBRECENI EGYETEM
Informatikai Tudományok Doktori Iskola

Debrecen, 2018

Ezen értekezést a Debreceni Egyetem Természettudományi Doktori Tanács Informatikai Tudományok Doktori Iskola „*Diszkrét matematika, képfeldolgozás és komputergeometria*” programja keretében készítettem a Debreceni Egyetem természettudományi doktori (Ph.D.) fokozatának elnyerése céljából.

Debrecen, 2018.

.....
Bertók Kornél
jelölt

Tanúsítom, hogy Bertók Kornél doktorjelölt 2010-2018. között a fent megnevezett Doktori Iskola „*Diszkrét matematika, képfeldolgozás és komputergeometria*” programja keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Az értekezés elfogadását javaslom.

Debrecen, 2018.

.....
Dr. Fazekas Attila
témavezető

Ember-számítógép interakciók: az emberi arc modellezése

Értekezés a doktori (PhD) fokozat megszerzése érdekében az informatika tudományágban

Írta: Bertók Kornél okleveles programtervező matematikus

Készült a Debreceni Egyetem Informatikai Tudományok Doktori Iskolája Diszkrét matematika, képfeldolgozás és komputergeometria programja keretében.

Témavezető: Dr. Fazekas Attila

A doktori szigorlati bizottság:

elnök:	Dr. Sztrik János	.....
tagok:	Dr. Hajdu András	.....
	Dr. Nyúl László	.....

A doktori szigorlat időpontja: 2017. május 10.

Az értekezés bírálói:

	Dr.	.....
	Dr.	.....

A bírálóbizottság:

elnök:	Dr.	.....
tagok:	Dr.	.....
	Dr.	.....
	Dr.	.....
	Dr.	.....

Az értekezés védésének időpontja: 2018. .....

Tartalomjegyzék

Köszönetnyilvánítás

Köszönöm **feleségem Gréta** és **kisfiam Milán** soha véget nem érő bíztatását és támogatását, ami újra és újra energiát adott az évek során mialatt az értekezésemhez kapcsolódó feladatokkal foglalkoztam.

Ugyanígy köszönöm **szüleim**, **testvérem** és **nagymamám** belém vetett töretlen hitét és sokrétű támogatását, mely végig kísérte teljes egyetemi pályafutásomat.

Köszönettel tartozom a **témavezetőmnek**, **Dr. Fazekas Attilának**, aki tanácsaival folyamatosan támogatta a tudományos fejlődésemet és munkámat és amiért felkeltette az érdeklődésemet a képfeldolgozás és gépi látás iránt.

Végül, de nem utolsó sorban köszönettel tartozom a **Komputergrafika és Képfeldolgozás Tanszék**, illetve az **Informatikai Tudományok Doktori Iskola** valamennyi volt és jelenlegi oktatójának. Megtisztelő volt, hogy tagja lehettem az előbbi intézményeknek és hogy bekapcsolódhattam számos érdekes kutatómunkába.

1. Bevezetés

Napjainkban a számítógépek térnyerése és a technológia fejlődése eljutott arra a szintre, hogy a hétköznapi felhasználók kezdenek úgy tekinteni a gépre, mint az emberi létezésük része. Egyre inkább azzal a természetes elvárással élnek, hogy a számítógép, a működésében és gondolkodásában is legyen olyan, mint mi emberek. A rohamos fejlődésnek köszönhetően mára külön tudományág alakult, mely az ember és számítógép közötti kapcsolatokkal foglalkozik: az *ember-számítógép interakció*. Az elnevezés az angol terminológiának megfelelő *human-computer interaction* kifejezésből ered, jelen értekezésben gyakran találkozhat majd az olvasó az ennek megfelelő rövidítéssel, a HCI-vel. A HCI elsődleges kutatási-, valamint fejlesztési feladata, hogy olyan új, esetlegesen alternatív kommunikációs eszközöket fejlesszen, melyek segítik az ember-számítógép interakciókat az emberi fél számára természetesebbé, illetve magától értetődővé tenni. Ily módon a HCI célja egyrészt a kapcsolatban szereplő emberi fél vizsgálata, másrészt pedig a szükséges hardveres és szoftveres interfészek biztosítása.

Tetszőleges ember-számítógép interakciót célszerű visszavezetni a neki megfelelő ember-ember interakcióra, ehhez azonban szükséges az emberi kommunikáció szabályainak az ismerete. E szabályrendszer pontos definiálása és következetes alkalmazása nélkül minden bizonnyal sosem jöhet létre interakció ember és számítógép között. Ez azonban korántsem kézenfekvő. Gondoljunk csak bele, például egy autóban lévő számítógép, mely folyamatosan monitorozza a sofőrt, mi alapján ismerje fel azt, hogy belenézett a visszapillantóba sávváltás előtt, vagy azt, ha megijed, vagy éppenséggel elálmosodik. Kövesse a tekintetünket? Vagy éppenséggel a mozdulatainkat? Csak úgy, mint az ember-ember interakciókban, itt is célszerű a rendelkezésre álló információ halmazból a lehető legtöbb, releváns atomi építőkockát meghatározni a további feldolgozás céljából.

A kutatásaim azon a feltevésen alapulnak, hogy a különböző szoftverek kizárólagosan billentyűzettel, egérrel, vagy érintéssel történő vezérlése ma már nem elégítik ki a felhasználói igényeket. Egyre inkább növekszik a kereslet a természetesebb, vagy eszköz-független interakcióra ember és számítógép között. A kommunikáció egyszerűsítésére irányuló terveimet a mindennapi életből merítettem. A szóbeliség (mint *verbális jelek* halmaza) a legtipikusabb és egyben a legnagyobb információ tartalommal bíró módja az ember-ember

kommunikációnak. Azt azonban, hogy egy társalgás során milyen jelentést tulajdonítunk a mondanivalónknak, a szóbeliség mellett számos más tényező is befolyásolja. A szóbeli információk kiegészítésére, ellenőrzésére vagy éppen hangsúlyozására a nem szóbeli, úgynevezett *nonverbális jelrendszert* alkalmazzuk. A nonverbális jelek tipikus megnyilvánulásai a mimika, a tekintet – szemkontaktus – szemmozgás; illetve az úgynevezett vokális jelek (pl. hangnem és hangerő); a gesztusok, a testtartás és a távolságtartás-térközszabályozás.

1.1. Célkitűzések

A PhD tanulmányaim során HCI kutatásokkal foglalkoztam, azzal a céllal, hogy az emberi fejből és arcból kinyerhető nonverbális jelek és jellegzetességek kinyerésére, reprezentálására, valamint felismerésére hozzak létre új képfeldolgozási módszereket. Céljaim középpontjában mindvégig az HCI technológiájának a továbbfejlesztése állt. Mivel a HCI valamennyi aspektusa, különösen az arcból kinyerhető információkon (pl. arckifejezések) alapuló megközelítések iránti igény egyre inkább növekszik, ezért a jövőben egyre több olyan megoldásra lesz igény, mely az ember-számítógép interakciókban folyamatosan tanulmányozza az emberi fél arcát. Az eredményeimet külön alfejezetekben foglalom össze:

- A rendszer alapját képező *arcdetektáló, követő és azonosító alrendszer* felépítését és működését,
- Több lehetséges eljárás tanulmányozásával és összehasonlító elemzésével egy 2-D pixel térbeli reprezentációt az *arci karakterisztikus pontokra*,
- Egy eljárást, mely segítségével nagy pontossággal meghatároztam az *arc térbeli elhelyezkedését és irányultságát*, a karakterisztikus pontok 2-D pixel térbeli helyzetének és az arc 3-D geometriájának ismeretében,
- Az előbbiekre támaszkodva egy eljárást változatos fejmozgások tér-, és időbeli reprezentálására és felismerésére.

2. Arcdetektálás, követés és azonosítás

Az arcdetektálás folyamata során a számítógép egy tetszőleges, de számára ismeretlen kép birtokában hoz döntést arról, hogy látható-e azon arc vagy sem. Amennyiben igen, úgy meg kell határozni, hogy hány arc található a képen és hogy azok hol találhatók a képen. Ez nekünk, embereknek egyszerűnek tűnhet, de a számítógép számára az arcdetektálás egy kihívást jelentő problémakör olyannyira, hogy mára az egyik legaktívabban kutatott területté nőtte ki magát a HCI tudományágon belül.

A HCI rendszerünk szempontjából is alapvető fontosságú a felhasználó arcának detektálása, mivel annak a legkisebb hibája is akkumulálódhat a feldolgozás későbbi lépéseiben, így közvetve azok is hibássá válhatnak. Tovább nehezíti a feladatot a kritikus futásidő, hiszen az interakciónak (ember és számítógép között) valós időben kell történnie, ráadásul az algoritmusoknak gyakran korlátozott tár-, és számításkapacitású eszközön kell futniuk. Az arcdetektálás tekintetében tehát meghatározó a skálázhatóság kritériuma, továbbá a módszernek ugyanúgy invariánsak kell lennie az arc megjelenésére, orientációjára, valamint a változatos fényviszonyok okozta hatásokra.

PhD tanulmányaim során létrehoztam egy arc lokalizálással (detektálás és követés) és felhasználói azonosítással foglalkozó alrendszert. A feladat magában foglalta a szükséges tanító-, és teszt adatbázisok összeállítását; az eredmények méréséhez szükséges tesztkörnyezet kialakítását; tesztesetek meghatározását; metrikák készítését az eredmények összehasonlítására; illetve a képfeldolgozási algoritmusok áttekintését.

2.1. Irodalmi áttekintés

Már az utóbbi két évtizedben is szép számmal jelentek meg publikációk a témában, több áttekintő munka is készült az időszak eredményeiről lásd [1, 2]. A tanulmányokból kivehető, hogy már az ezekben az években megjelenő eljárások is kategorizálhatók aszerint, hogy hogyan definiálják az arc tulajdonságait. Az eltérés főként abból adódik, hogy mely tulajdonságokat tekintjük irányadónak, és hogy hogyan reprezentáljuk ezeket. Ha egy durva kategorizálást szeretnénk adni a módszerekre, akkor azt mondhatnánk, hogy az eljárások négy nagy csoportba oszthatók: *tudás-alapú* módszerek, *jellemző-alapú* módszerek, *sablon-alapú* módszerek, illetve *megjelenés-alapú* módszerek. A

tudás-alapú módszerek – emberi megfigyelések alapján – előre definiált szabályok halmazát használják a detektálás során; a *jellemző-alapú módszerek* olyan strukturális jellemzőket használnak, melyek invariánsak az arc pozíciójára és megvilágítására; a *sablon-alapú eljárások* előzetes letárolt arc-sablonok alapján próbálnak döntést hozni; továbbá a *megjelenés-alapú módszerek* pedig egy reprezentatív tanító arcadatbázis alapján pusztán az arc külső jellegzetességeiből és megjelenéséből határoznak meg modelleket, melyeket később detektálásra használnak.

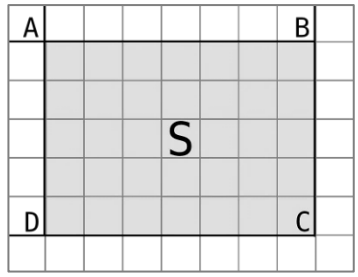
Az elmúlt évtizedben az arcdetektálás területe gyökeres változáson ment keresztül köszönhetően Viola és Jones munkásságának [3]. Az ő megoldásuk volt ugyanis az első, ami szélesebb felhasználói réteghez tudott eljutni azáltal, hogy a kidolgozott módszerük magas pontosság és találati arány mellett volt képes korlátozott kapacitású hardver eszközökön (pl. digitális kamerán) futni.

2.2. Viola-Jones arcdetektor

Jelen fejezetben röviden áttekintjük Viola és Jones munkáját [3] a *boosting* alapú arcdetektálás aspektusából – tesszük mindezt azért, mert a boosting algoritmus lényegében de-facto szabvánnyá nőtte ki magát az arcdetektáló algoritmusok területén Viola és Jones munkáját követően és annak köszönhetően. Az arcdetektáló algoritmus alapötlete három új megközelítést foglal egységbe a valós idejű működés érdekében, ezek rendre: az *integrál képreprezentáció* bevezetése; az osztályozó létrehozásának folyamata *AdaBoost* segítségével; illetve az osztályozók *vízesés/kaszkád struktúrába* való szervezése.

2.2.1. Az integrál képreprezentáció

Az integrálkép reprezentáció (angol terminológiában *summed area table*, röviden SAT) segítségével gyorsan és hatékonyan összegezhetjük egy rácsszerkezet téglalapalakú tartományában található rácspontok intenzitásértékeit. Az eljárás nem új keletű, a komputergrafikában régóta alkalmazzák [4], bár az elv már ezt megelőzően is ismert volt a matematikában, a többdimenziós eloszlásfüggvények területén. Viola és Jones felismerték, hogy az osztályozás során célszerűbb képrégiók jellemzőivel foglalkozni (különálló képpontok helyett). Választásuk a *Haar-féle jellemzőkre* esett és e jellemzők többféle méret melletti gyors kiszámítása érdekében vezették be az integrált képet.



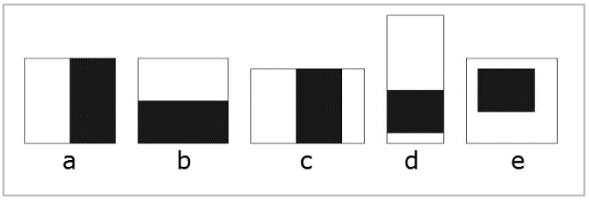
1. ábra. Az integrál képreprezentáció meghatározása.

Az integrálkép előállítása az alábbi módon lehetséges:

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y'), \tag{2.1}$$

ahol $ii(x,y)$ az integrál kép értéke az (x,y) pixel pozícióban, $i(x',y')$ pedig az (x',y') pixel intenzitása az eredeti képen. Az integrál kép használatával tehát rendkívül hatékonyan összegezhetők egy téglalapalakú tartományban elhelyezkedő pixelek intenzitásértékei, úgy ahogy az 1. ábra is mutatja, az ABCD téglalapba eső pixelek intenzitásértékeinek az összege négy tömbhivatkozással számítható:

$$\sum_{(x,y) \in ABCD} i(x,y) = ii(A) + ii(C) - ii(B) - ii(D). \tag{2.2}$$



2. ábra. Az integrál képreprezentáció és a Haar jellemzők (alsó sor: *a*, *b*, *c*, *d* és *e* ábra) illusztrációja.

A Haar jellemzők tekintetében – lásd 2. ábra – az integrálképet egy (súlyozott) különbség előállításához használjuk fel, például az (a) jellemző értéke a sötét-, és világos téglalapok alatti intenzitásértékek különbsége. A jellemzőket tehát az ii kép felett értékeljük ki. Egy jellemző, kettő-, vagy több szomszédos téglalapból épül fel, tehát egy jellemzőn belül a téglalapoknak vannak közös oldalaik és csúcspontjaik. Az i képen bármely téglalap alapú területen lévő pixelek összege az ii kép négy intenzitás értékének kiolvasásával megkapható. Ez a gyakorlatban azt jelenti, hogy az 2. ábrán látható (a) jellemző

értékének meghatározásához az integrálkép hat tömbhivatkozásra van szükség a fekete és fehér téglalapok közös éleihez tartozó csúcspontok miatt.

2.2.2. AdaBoost algoritmus

A *boosting* egy algoritmuscsaládot foglal magában, melynek tagjai több, szolid pontosságú (úgynevezett gyenge) hipotézis kombinációját felhasználva határoznak meg egy magas pontosságú (úgynevezett erős) hipotézist. A technika felvezetéséhez az olvasónak célszerű elolvasnia az [5] könyvfejezetet. A továbbiakban röviden ismertetjük az arcdetektáláshoz használt *adaptive boosting* (röviden *AdaBoost*) algoritmust [6]. Az AdaBoost iterációnként egy *gyenge osztályozót* határoz meg. Az adaptivitása abban rejlik, hogy egy gyenge osztályozó definiálása során nagyobb súllyal veszi számításba azokat a tanító mintákat, melyeket az általa már előállított gyenge osztályozók hibásan osztályoztak. A gyenge osztályozó elnevezés a gyenge osztályozási képességből fakad. Azonban, ha a gyenge osztályozók pontossága kicsivel több mint 50%, akkor a boosting algoritmus minden újabb iterációs lépésben egy pontosabb *erős osztályozót* épít fel a rendelkezésre álló gyenge osztályozókból [6].

Legyen adva a tanítóminták halmaza: $S = \{(x_i, z_i), i = 1 \dots N\}$, ahol az egyes $x_i \in X$ változók (esetünkben az arcok) az osztályozó értelmezési tartományának az elemei; valamint a $z_i \in Z$ célváltozók a címke tér elemei. Bináris osztályozási problémák esetén $Z = \{-1, +1\}$, a pozitív és negatív mintáknak megfelelően. Az AdaBoost kimenetét a szakirodalom erős (boosted) osztályozóként hivatkozza és az alábbi formában definiálja:

$$F^T(x) = \sum_{t=1}^{T} f_t(x), \tag{2.3}$$

ahol az $F^T: X \to \mathbb{R}$ additív modell, a T darab f_t bázisfüggvény, mint alacsony komplexitású gyenge osztályozó lineáris kombinációjaként áll elő (a bázis függvényeket később ismertetem). Az osztályozás során a bemenő x minták címkéjét tisztán az F^T előjele alapján határozzuk meg. Továbbá megjegyezzük, hogy egyes implementációkban az F^T abszolút értéke az osztályozás megbízhatóságával lehet arányos.

Az AdaBoost, a $(t+1)$-edik iterációban a legjobb tulajdonságokkal rendelkező f_{t+1} bázisfüggvényt próbálja meghatározni, az előző iterációban meghatározott F^T ismeretében. A bázis függvények – mint gyenge osztályozók – egyetlen attribútum érték mentén osztályozzák a

tanítómintákat: pozitív-, illetve negatív mintáknak. Az osztályozás egy $h_t: X \to \mathbb{R}$ függvénnyel történik, mely tulajdonképpen a gyenge hipotézis: esetünkben az integrál kép alapján számított Haar jellemző, lásd 2. ábra. A gyenge osztályozáshoz szükséges még egy θ_t döntési küszöb, mely a h_t függvény értékeit sorolja két halmazba. Az f_t bázis függvény az előbbi döntési küszöböt és gyenge hipotézist felhasználva az alábbi formában definiálható:

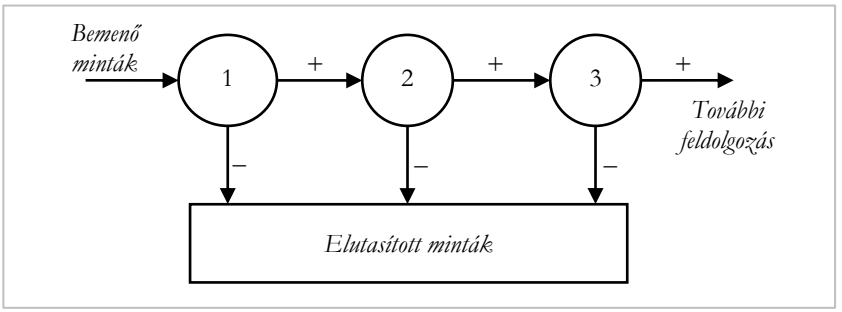
$$f_t(x) = \begin{cases} c_1, & ha\ h_t(x) < \theta_t; \\ c_2 & különben. \end{cases} \tag{2.4}$$

A fenti egyenletben szereplő c_1 és c_2 mennyiségek az osztályozás kimenetéhez tartozó konfidenciák. Az optimális c_1 és c_2 értékek meghatározását nem ismertetjük, de lényegében egy súlyozott költségfüggvény minimalizálásának eredményeként állnak elő [7]. A súlyokat arra használja az algoritmus, hogy a soron következő iterációs lépésben csökkentse a hibás osztályozás hatását. Ezt úgy éri el, hogy a megelőző iterációs lépésben rosszul osztályozott mintákat felsúlyozza, így helyezve nagyobb hangsúlyt a nehezen osztályozható mintákra.

2.2.3. Kaszkád struktúra

Már említettem, hogy a Viola-Jones algoritmus *kaszkád struktúrába* rendezi a boosted osztályozókat. A kaszkád struktúra kritikus pontja az algoritmusnak, hiszen ezzel a lépéssel kisebb-, kompaktabb felépítésű és hatékonyabb osztályozók készítésére nyílik lehetőségünk. Az alapötlet az, hogy viszonylag könnyen készíthetünk olyan boosted osztályozókat, amelyek a negatív példák nagy részét már az algoritmus első néhány lépéseiben visszautasítják, viszont a pozitívokat elfogadják. A kaszkád struktúra lényegében egy olyan speciális döntési fa, melyben minden elemnek egy rákövetkezője lehet, kivéve az utolsó szinten elhelyezkedő elemet, mely egy levélelem. Viola és Jones 4667 darab jellemző felhasználásával épített fel egy 32 darab boosted osztályozóból álló kaszkád struktúrát. A kaszkád struktúra csúcsai rendre $2, 5, 20, 20, 20, 50, 50, \underbrace{100, \ldots, 100}_{5\text{ db}}, \underbrace{200, \ldots, 200}_{20\text{ db}}$ darab gyenge osztályozót tartalmaztak. Ez részben intuíción alapul, másrészt viszont a struktúra lényege az, hogy a lehető legkorábbi szakaszban, a lehető legkevesebb gyenge osztályozóval minél több negatív mintát utasítson el. A pozitív mintákra mindegyik osztályozó pozitív választ ad, így a bonyolult számítások csak az ígéretes részekre futnak le. Ez egyben azt is jelenti, hogy még az ígéretes részeken növekszik a detektor

futásideje, addig mindenhol máshol jelentősen csökken. A detektálás futásideje a teljes bemenő képre drasztikusan lecsökkenhet, ha az kevés ígéretes részt tartalmaz. Ahogy azt a 3. ábra is mutatja, a detektálás során minden egyes bemenő minta a boosted osztályozóknak egy sorozatán halad végig. A boosted osztályozók kimenete bináris: ha pozitív választ ad, akkor elindítja a rákövetkező osztályozót; ha negatív választ ad akkor megszakítjuk a kiértékelést és a mintát elutasítjuk (mondván, hogy nem arcot ábrázol).



3. ábra. A kaszkád struktúra. Minden egyes csúcs egy boosted osztályozót reprezentál (lásd 2.3. egyenlet).

A kaszkád struktúra a tanítási folyamatra is hatással van. Feltételezhetjük, hogy az arcdetektálás lényegében egy ritka esemény bekövetkezésének a detektálása. Következésképpen, milliárdos nagyságrendű negatív mintára lenne szükség ahhoz, hogy magas teljesítményű detektort lehessen tanítani. A nagy mennyiségű negatív minta kezelésére egy úgynevezett bootstrap folyamatot lehet bevezetni, mely során minden boosted osztályozóra manuálisan meghatározunk egy döntési küszöböt. A következő szinten elhelyezkedő boosted osztályozó elkészítéséhez az előbbi küszöböt és az eddig létrejött részleges osztályozót felhasználva olyan negatív mintákat keresünk, amelyek a tanítás során még nem lettek elutasítva. Ezt leszámítva azonban, a boosted osztályozókat egymástól függetlenül tanítjuk. Az előbb ismertetett döntési küszöb megválasztása kritikus, ugyanis általa túlságosan elutasítóvá vagy elfogadóvá válhat az osztályozás. Még az előbbi esetben a detektorunk kellően gyors lesz, de ugyanakkor sok hamis negatív találatot fog adni, addig a másik esetben lesz egy kellően pontos, de lassú detektorunk. Egy jó tulajdonságokkal rendelkező arcdetektor készítése, a paraméterek finomhangolása miatt nagyon sok időt emészthet fel.

2.3. Detektor tanítások eredményei

Az arcdetektálásnak alapvető szerepe van a HCI rendszeremben; ez képezi a rendszer alapját, így a detektálás során bekövetkező hibák felszivároghatnak a magasabb szintű komponensekbe is. Fontos leszögezni, hogy ugyan léteznek szabadon elérhető arcdetektorok, de jónak láttam saját arcdetektorok elkészítését, melyek jobban igazodnak az elvárásaimhoz: arcok detektálása változatos pozíciókban, orientációval és valós időben [8]. Mivel a detektor néhány eleme – például a jellemzőkinyerő, vagy a tanuló algoritmus – lecserélhető, így ki kell választani egy olyan kombinációt, amely a legjobban igazodik a céljaimhoz. Itt két szempont igazán fontos, a futásidő és a pontosság. A futásidő viszonylag egyértelmű, jól mérhető mennyiség, és mint korábban említettem, talán az egyik legfontosabb mérőszám korlátozott kapacitású eszközök esetén. A futási időre vonatkozó teszteket egy Intel (R) Core(TM)2 Quad CPU Q9550 @ 2.83 GHz processzorral felszerelt számítógépen végeztem el.

A pontosság nehezebben definiálható. A találati arányok számításánál nem csak helyes és hibás eredményt különböztethetünk meg. El kell különíteni azt az esetet, amikor a detektor arcot talált, ott, ahol nem volt, attól az esettől, amikor nem detektált egy létező arcot. Az előbbit *hamis pozitív* találatnak, utóbbit *hamis negatívnak* nevezzük. Ennek analógiájára beszélhetünk *valós pozitív* és *valós negatív* találatokról is. A detektálás folyamatának sajátossága (lásd 2.4. fejezet) miatt a valós negatív találatokat nem fogom számszerűsíteni, hiszen a csúszóablakos eljárás miatt az ablak számos pozíciójában keletkezne efféle találat, így nem lenne kifejező ereje ennek a mérőszámnak. Az 1. táblázatban bemutatott konfúziós mátrix foglalja össze a fent említett fogalmakat.

		Aktuális osztály ***annotáció eredménye***	
		Pozitív	Negatív
Becsült osztály ***teszt eredménye***	Pozitív	**TP** *Valós pozitív*	**FP** *Hamis pozitív*
	Negatív	**FN** *Hamis negatív*	**TN** *Valós negatív*

1. táblázat. Az eredmények kvantitatív kiértékelésénél használt mérőszámok.

Az osztályozóim jóságának számszerűsítésére a *precizitás* (precision) és *felidézés* (recall) mérőszámokat használtam: $precision = \frac{TP}{TP+FP}$, valamint $recall = \frac{TP}{TP+FN}$. Egyéb mérőszámokat a valós negatív találatok nehézkes értelmezése miatt nem származtattam a konfúziós mátrixból. A gyakorlatban az volt a célom, hogy mind a precizitás, mind az felidézés értékét maximalizáljam a detektor tanítások során. Ehhez az F_1 mérőszámot is meghatároztam, mely a precizitás és a felidézés összefoglalására szolgál. Az F_1 mérőszám a harmonikus középértéket reprezentálja a két mutató között. Az alábbiakban a saját arcdetektoraim elkészítése során szerzett tapasztalataimat összegzem.

2.3.1. Tanító és teszt adatbázisok

A tanítás első lépése a tanító adatbázis létrehozása és felcímkézése, vagyis annotálása. A tanító adatbázis létrehozásának egyik legfontosabb szempontja, hogy az minél reprezentatívabb legyen. Részben igaz az a megállapítás, hogy minél több tanítóminta áll rendelkezésünkre, annál jobban teljesít majd az osztályozó. Arra azonban figyelni kell, hogy a tanuló algoritmusok hajlamosak túltanulni, vagyis, ha bizonyos típusú tanítóképek felülreprezentáltak, akkor az osztályozó statisztikai modellje az adott kategóriában található változatosságokat fogja leírni, nem pedig az általános jellegzetességeket. A tanító adatbázis képeit két, egymással diszjunkt halmazba kell sorolni: a pozitív-, és a negatív minták halmazába. A pozitív minták egy-egy kivágott arcot ábrázoltak; a negatívok pedig értelemszerűen az arcot nem tartalmazó képek voltak. Viola és Jones 4916 darab pozitív mintát és 9500 darab negatív mintát használtak az arcdetektoruk létrehozásához (melyből 350 millió darab arcot nem tartalmazó komponenst nyertek ki a tanításhoz). Ez a nagyságrend és arány az, amit érdemes követni a tervezésnél.

A tanításokhoz két pozitív adatbázist hoztam létre az 2. táblázatban szereplő adatbázisokból, melyek 6500 db mintát tartalmaztak külön-külön. Az egyik pozitív adatbázis csak frontális arcokat tartalmazott, még a másik a frontális orientáció mellett 45°-ban a függőleges tengely körül (pozitív és negatív irányban) elforgatott arcot is. A csak frontális arcokat tartalmazó adatbázist a továbbiakban *eredeti adatbázisként* fogom hivatkozni, még a másikat *kiterjesztett adatbázisként*. A negatív adatbázist tekintve 15000 db mintát gyűjtöttem és alkalmaztam egy internetes keresőrobot volt segítségével [9].

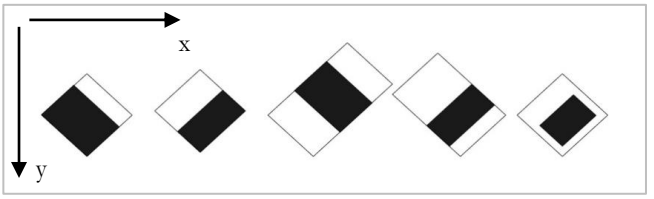
Adatbázis	Személyek száma	Képek darabszáma	Arc pozíció
HuComTech Multimodal Database [10]	113	2260 (50 órányi felvételből)	5
The Extended Yale Face Database B [11]	28	16128	9
The Extended Cohn-Kanade Dataset (CK+) [12]	123	10558	2

2. táblázat. A pozitív mintákhoz felhasznált adatbázisok.

Az elkészült osztályozók tesztelésére, az előbb hivatkozott keresőrobotot felhasználva hoztam létre két pozitív-, és egy negatív tesztadatbázist (a tanításnak megfelelően): az egyik pozitív adatbázis 1000 db csak frontális arcot tartalmazott, még a másik 1000 db a frontális orientáció mellett 45°-ban a függőleges tengely körül (pozitív és negatív irányban) elforgatott arcot is. A negatív tesztadatbázist tekintve 1000 db mintát használtam fel. A tesztadatbázisnál felhasznált minták felbontása 640×480 pixel volt.

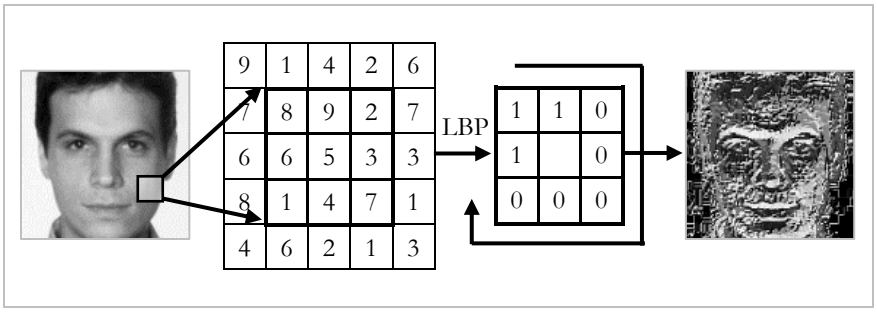
2.3.2. Jellemőkinyerő algoritmusok változatai

A Haar jellemzők (lásd 2. ábra), az integrál képnek köszönhetően gyorsan és egyszerűen számíthatók, azonban a tengelyállású alap jellemzők felhasználási köre erősen korlátozott. Ezek a jellemzők önmagukban nem igazán alkalmasak a frontálistól eltérő orientációjú arcok detektálására, vagy a megvilágításban bekövetkező hirtelen változások kezelésére. A szakirodalomban gyakran használják a Haar jellemzőknek egy kiterjesztett halmazát [13], melyet az eredeti tengelyállású jellemzőknek a 45°-ban történő elforgatásával kapnak, lásd 4. ábra.



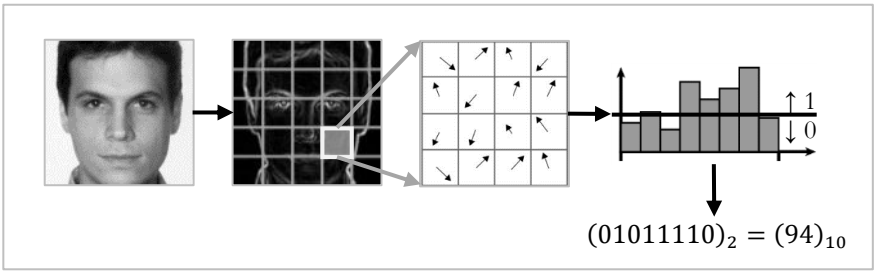
4. ábra. A 45°-kal elforgatott Haar jellemzők.

A Haar jellemzők egyik legnagyobb gyengesége, hogy nem robusztusak az extrém megvilágítási viszonyokra. A *lokális bináris minta* (*local binary patterns*, röviden LBP) elnevezésű eljárást [14] gyakran használják a képfeldolgozásban az eltérő megvilágításból fakadó problémák kiszűrésére. Az eljárás egy kép pixeleit címkézi fel aszerint, hogy milyen az egyes pixelek szomszédságában lévő pixelek intenzitása (lásd 5. ábra). Ezekből a címkékből az algoritmus bináris, úgynevezett LBP kódokat hoz létre, melyek felhasználásával a kép adott régióinak a textúráját tudjuk leírni – csak úgy, mint a Haar jellemzők esetén. Az LBP kód létrehozásának folyamata elolvasható a következő publikációban: [14]. Továbbá részletesen olvashatunk arról, hogy miként kombinálható hatékonyan az LBP osztályozó az AdaBoost algoritmussal a [15] publikációban.



5. ábra. Textúrák leírása lokális bináris minták segítségével. Pl. a jelölt régióhoz számított LBP kód: $(11000001)_2 = (193)_{10}$

Textúrák leírására gyakran használatos az élhisztogram leíró, mely a képen található lokális élek iránya alapján épít hisztogramot. A módszer jelentősége abban rejlik, hogy hisztogram normalizálás után lehetőségünk nyílik különböző méretű képek összehasonlítására. A Haar jellemzőkkel ellentétben, az élek sokkal inkább invariánsak a megvilágításban bekövetkező változásokra és képesek a bonyolultabb geometriai jellemzők kódolására is. Az élhisztogramhoz hasonló módszer, a *gradiens irány hisztogram* [16] (*histogram of oriented gradients*, röviden HoG), mely az élhisztogram továbbfejlesztésének is tekinthető (lásd 6. ábra). A HoG is hatékonyan kombinálható az AdaBoost algoritmussal olyannyira, hogy mára szabvány lett a gyalogos detektálás területén, lásd [17, 18, 19].

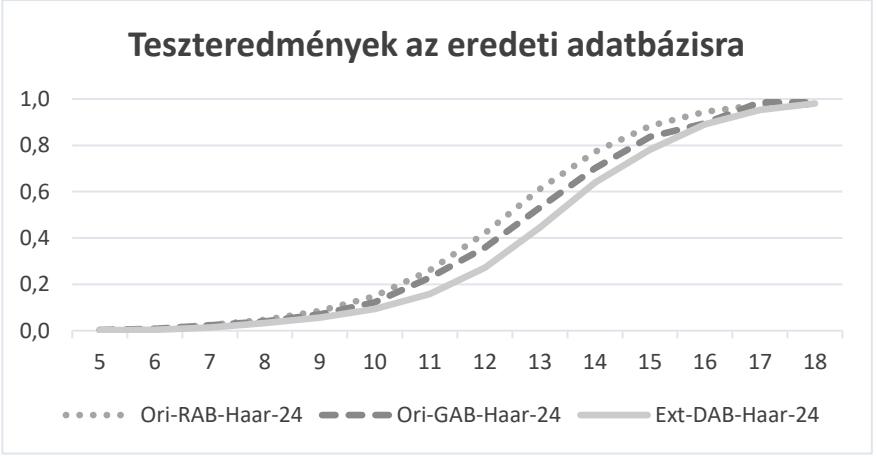


6. ábra. Textúrák leírása *gradiens irány hisztogram* segítségével. A jelölt téglalaphoz számolt HoG leíró: $(01011110)_2 = (94)_{10}$.

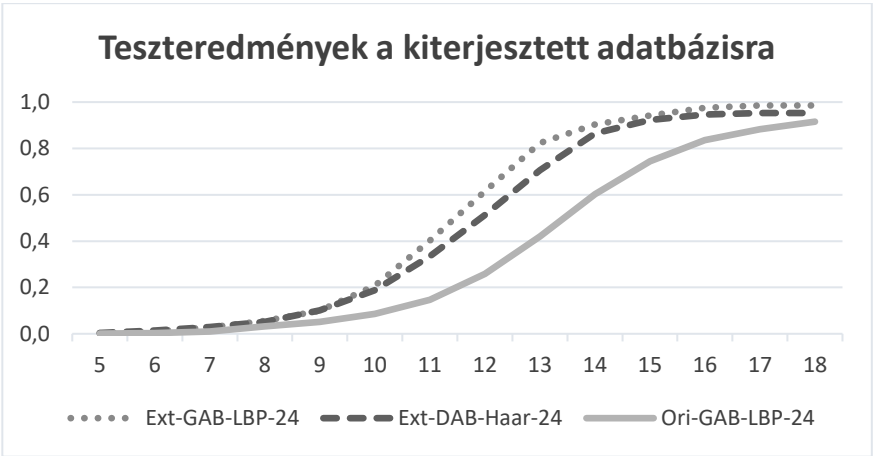
Az optimális jellemzőtípus kiválasztása során az volt a célom, hogy a detektor a lehető leginkább invariáns legyen a fej pozíciójára és az eltérő megvilágításból fakadó fényviszonyokra. A fejpozíció kapcsán azonban elég szűk a volt mozgásterem, hiszen a pozitív adatbázis megalkotásakor ügyelnem kellett arra, hogy az egyes minták megfeleltethetők legyenek egymásnak. Az eredmények számszerűsítését tekintve az eredeti-, és a kiterjesztett adatbázisra tanítva és tesztelve végeztem el több mérést. Az osztályozókat a két adatbázisra külön-külön tanítottam úgy, hogy a tanítás összes többi paramétere megegyezett. Az alábbi felsorolásban összegzem a tapasztalataimat:

- Általánosságban elmondható, hogy az eredeti adatbázisra tanított osztályozók jobban teljesítenek a csak frontális arcokat tartalmazó tesztadatbázisokon; és ugyanígy jobbak a kiterjesztett adatbázison tanított osztályozók a kiterjesztett tesztadatbázison.
- Az eredeti adatbázis esetén a Haar-féle jellemzők jobban teljesítenek, mint a másik kettő típus (lásd 7. ábra).
- Az előző megállapítás azonban nem igaz a kiterjesztett adatbázisra. Itt az LBP jellemzők sokkal jobban teljesítenek a gyakorlatban, azaz kevésbé érzékenyek az arc orientációjára (lásd 8. ábra).
- Érdekességképpen megemlíthető, hogy az LBP akkor is jól teljesít a kiterjesztett adatbázison tesztelve, ha a tanítás az eredeti adatbázison történt, (lásd 8. ábra, *Ori-GAB-LBP-24*).
- Megemlítendő, hogy a HoG meglehetősen rosszul teljesített minden esetben (lásd 0), ami többek közt betudható annak, hogy a tanító minták alacsony felbontása miatt kevésbé karakterisztikusak az élek az arcon belül.

Összességében az mondható el, hogy ha a végfelhasználásra szánt környezetben túlnyomórészt frontális arcok fordulhatnak csak elő, akkor célszerű Haar-féle jellemzőt választani és csak frontális arcokra tanítani. Másrészt, célszerű az LBP használata és a kiterjesztett adatbázisra történő tanítás, ha változatos orientációk mellett akarunk arcokat detektálni.



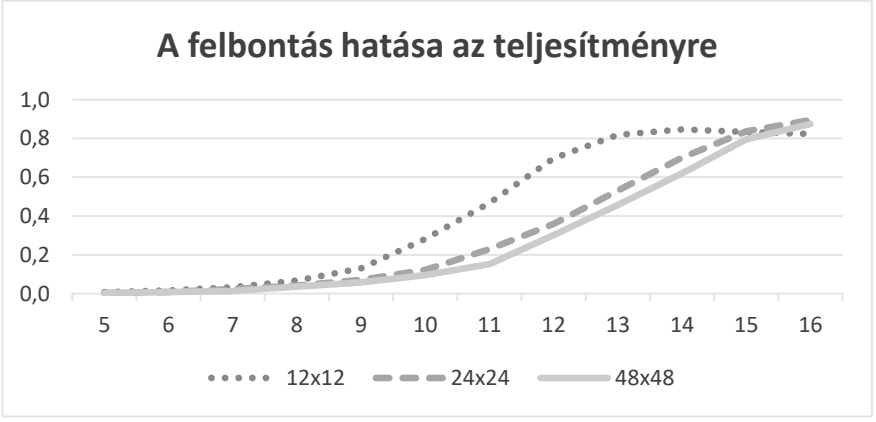
7. ábra. A három legjobb osztályozó teljesítménye az eredeti tesztadatbázisra. A függőleges tengelyen az osztályozók F_1 mérőszámát ábrázoltam, a vízszintes tengelyen pedig a kaszkád struktúra szintjeit. Rövidítések: *Ori*: Eredeti adatbázisra tanítva; *Ext*: kiterjesztett adatbázisra tanítva; *DAB/GAB/RAB*: boosting algoritmus típusa; *24*: tanítóminták horizontális és vertikális pixel mérete.



8. ábra. A három legjobb osztályozó teljesítménye a kiterjesztett teszt adatbázisra. A függőleges tengelyen az osztályozók F_1 mérőszámát ábrázoltam, a vízszintes tengelyen pedig a kaszkád struktúra szintjeit.

2.3.3. Tanító minták mérete

A tesztek során a pozitív minták méretének hatását a pontosságra három különböző felbontás mellett vizsgáltam úgy, hogy minden más paraméter megegyezett: 12×12 , 24×24 és 48×48 pixel méretekkel. Ebben a tesztesetben, az előzőek során legjobbnak talált Haar-féle jellemzőket használtam fel. A tanítás során azt tapasztaltam, hogy a tanító minták mérete nem igazán van hatással a tanítás futásidejére, így a méretet érdemes egyéb a modell jóságára vonatkozó kritériumok szerint megválasztani. Az 9. ábra mutatja a három detektor teljesítményére vonatkozó tapasztalatainkat.



9. ábra. A három detektor teljesítménye, a pozitív minták eltérő felbontása mellett. A vízszintes tengelyen a kaszkád struktúra szintjeit ábrázoltam, még a függőleges tengelyen a detektorok F_1 mérőszámát. Mindhárom detektor tanításához Haar-féle jellemzőket használtam.

Azt tapasztaltam, hogy a precizitás karakterisztikája független a pozitív minták méretétől, azonban a felidézés értéke a kaszkád egy bizonyos szintje felett elkezd csökkenni, ami a legkisebb felbontás mellett volt a legszembetűnőbb. Külön érdekesség, hogy a legkisebb felbontáson intenzív emelkedés figyelhető meg az F_1 mérőszám karakterisztikájának második harmadában, ami a harmadik harmadra stagnálni kezd. A 24×24-es és 48×48-as felbontásokhoz tartozó F_1 mérőszámok szinte egybeesnek és a 15. szint felett jóságban megelőzik a 12×12-es felbontást. Az egyetlen ok, amiért felesleges tovább tanítani a detektorokat, és amiért a gyakorlatban ajánlott a 24×24 -es felbontást választani az a következő: szignifikáns eltérés mutatkozik a futási időben. Tesztjeim alapján a felbontás négyszerezésével a futási idő megduplázódik (lásd 3. táblázat). A pozitív minták magasabb

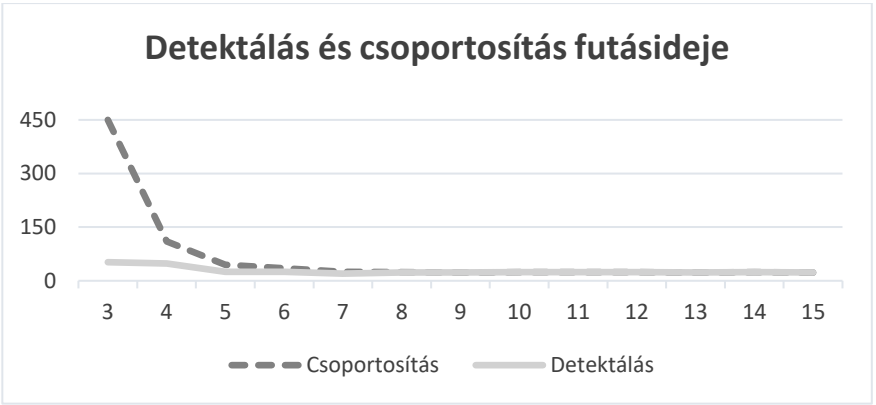
felbontása tehát ronthatja a hatásfokot, a felbontást úgy kell megválasztani, hogy elérjük a megfelelő pontosságot, de ne növeljük meg túlságosan a detektor futási idejét.

12×12	24×24	48×48
44 ms	84 ms	157 ms

3. táblázat. A pozitív minták felbontásának, illetve a futási időnek a kapcsolata.

2.3.4. Kaszkád struktúra szintjei

A kaszkád struktúra a gyakorlatban úgy működik, hogy minden szint kiszűri a minták egy részét, és csak azt fogja pozitívnak értékelni, amelyik az összes szinten átmegy. Így a szintek számának növelésével általában csökken a felidézés értéke, és nő a precizitásé. Előzetes várakozásaim alapján feltételeztem, hogy a kaszkád struktúra szintjeinek növekedésével a futásidő is nőni fog mondván, hogy minél több szint van, annál több kiértékelést kell elvégezni. Meglepő módon azonban ennek pont az ellenkezőjét tapasztaltam, hiszen a kevesebb szintből álló osztályozók esetén magasabb futásidőket mértem. Ez azzal magyarázható, hogy a kevés szintből álló osztályozók több pozitív találatot adnak, így az ezt követő csoportostást is több elemre kell lefuttatni. A csoportosításról bővebben a 2.4. fejezetben írok. A 10. ábrán ábrázoltam a detektálás és csoportosítás futásidejét a szintek számának függvényében. Kijelenthető, hogy a szintek számának szerepe elenyésző tisztán a detektálás futásidejét tekintve. Az osztályozó tanítása arányaiban ugyan több időt igényel, de a detektálás futásideje lényegében nem függ a szintek számától.



10. ábra. A függőleges tengely a futásidőt jelenti milliszekundumban, a vízszintes pedig a szintek darabszámát a kaszkád struktúrában.

2.3.5. Boosting algoritmus változatai

A különböző jellemzőkinyerő eljárások kipróbálása mellett különböző típusú boosting algoritmusok is hatással lehetnek a detektor pontosságára. Viola és Jones az eredeti munkájukban [3] a *diszkrét AdaBoost* algoritmust (DAB) használták [6], ahol a gyenge osztályozóknak megfelelő bázisfüggvények az $f_t(x): X \to \{-1; +1\}$ formában definiálhatók. A diszkrét AdaBoost egy általánosítása a *RealBoost* algoritmus (RAB), ami a DAB-tól annyiban tér el, hogy $f_t(x)$ bázisfüggvények valósértékű függvények. A RAB az $f_t(x)$ előjele alapján osztályozza a mintákat pozitívnak, vagy negatívnak, továbbá $f_t(x)$ abszolút értéke arányos a becslés megbízhatóságával. A RAB azért is lehet érdekes számunkra, mert több kutatás látott napvilágot azzal kapcsolatban, hogy hatékonyan működik akkor is, amikor több nézőpontban ábrázolt arcokat akarunk detektálni [20, 21, 22].

Az előző kettő boosting variáns az iterációs lépésekben mohón határozza meg az $f_t(x)$ bázisfüggvény értékeit. A mohó algoritmus azonban nem mindig ad optimális megoldást és meglehetősen érzékeny a hibás kezdő lépésekre. Az AdaBoost algoritmus egyik jó tulajdonsága, hogy képes azonosítani a kiugró értékeket (outlier-eket), azaz a nehezen kategorizálható mintákat. Az AdaBoost algoritmus felsúlyozza ezeket a nehezen osztályozható mintákat. Azonban könnyen lehet, hogy a magas súllyal rendelkező minta valójában egy kiugró érték. Sok kiugró érték esetén pedig túl nagy hangsúlyt helyezhet a boosting algoritmus ez ezekhez tartozó tanító mintákra. Az AdaBoost algoritmus egyik variánsa, a *gentle AdaBoost* (GAB) ezt a problémát próbálja orvosolni azáltal, hogy a kiugró értékekhez kisebb súlyt rendel, ezzel korlátozva a mohó algoritmust. Ugyan a GAB algoritmus néhány esetben alig teljesít jobban a RAB-nál, de bizonyos esetekben jobban teljesít zajos környezetekben [23]. A teszteredményeimből azt tapasztaltam, hogy az eredeti-, vagy kiterjesztett adatbázistól függetlenül már az első öt legjobb detektor esetében is megtalálható mind a három boosting típus. Ebből arra következtettem, hogy a detektor teljesítménye nem függ jelentősen a boosting algoritmus típusától, de azt azért megjegyzem, hogy mind az eredeti-, mind a kiterjesztett adatbázisra a GAB algoritmus teljesített a legjobban.

2.3.6. Tanítás optimális paraméterei

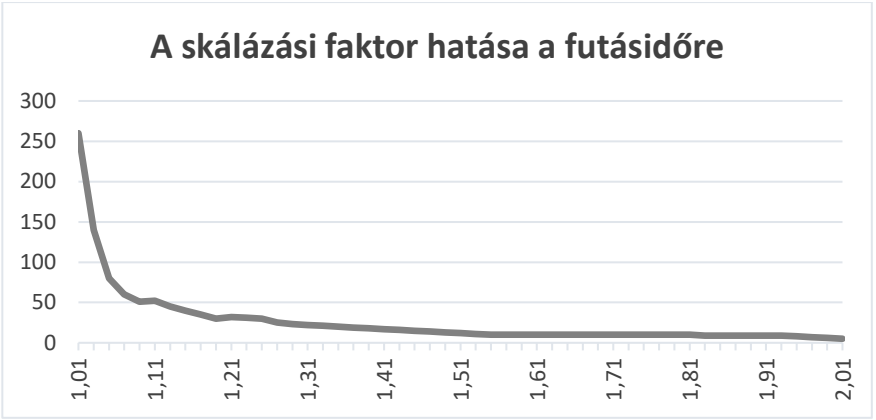
Jelen pontban a detektorok tanítása során szerzett tapasztalataimat összegzem. A tanítás során 6500 db pozitív-, és 15000 db negatív mintát

használtam fel. Az elkészült osztályozók tesztelésére, ugyanezen keresőrobotot felhasználva létrehoztam egy 1000 db arcot és ugyanennyi arcot nem tartalmazó képből álló adatbázist. A *tanítóminták méretének* tekintetében azt tapasztaltam, hogy a precizitás karakterisztikája közel független a mérettől, a felidézés értéke azonban a kaszkád egy bizonyos szintjétől kezdve csökkenni kezd. A gyakorlatban ajánlott a 24×24 pixeles felbontást választani mivel a felbontás négyszerezésével a futási idő megduplázódik. A *kaszkád struktúra szintjeinek számának* növelésével mindig csökken a felidézés értéke, és nő a precizitásé. Tesztjeim alapján a kaszkád struktúra szintjeinek száma nincs jelentős hatással a futásidőre és egy 17-18 szintből álló kaszkád segítségével elérhető 90% feletti precizitás és felidézés. A *jellemzőkinyerő algoritmusokról* az mondható el, hogy ha a végfelhasználásra szánt környezetben túlnyomórészt frontális arcok fordulhatnak csak elő, akkor célszerű Haar-féle jellemzőt választani az osztályozáshoz és csak frontális arcokra tanítani, ellenkező esetben megfontolandó az LBP jellemzők használata és a kiterjesztett adatbázisra történő tanítás. A *boosting algoritmusnál* arra a következtetésre jutottam, hogy a detektor teljesítménye nem függ jelentősen a boosting algoritmus típusától, de célszerű a GAB algoritmus használata.

2.4. Arcdetektálás folyamata

A detektálás során egy rögzített méretű ablakkal pásztázzuk végig sorfolytonosan a képet. A detektor csak egy bizonyos méreten képes detektálni az arcokat, ami eldől a tanításkor – esetünkben 24×24 pixel felbontású arcokat lehet detektálni. A gyakorlatban egy képpiramist szoktak készíteni a bemenő képből, hogy a detektor a keresett objektumot több méretben is képes legyen detektálni. A képpiramis egy olyan képreprezentáció, mely egy adott kép különböző méretű variánsaiból épül fel. Tehát a képpiramis egy változó felbontású képstruktúra, melynek létrehozásához egy skálázási faktort használunk. A skálázási faktor segítségével néhány iteráción keresztül csökkentjük a kép méretét addig ameddig az nagyobb, mint a detektor tanításakor használt méret. A kép skálázása a képpiramis egymást követő szintjei között biliniáris interpolációval történik. Értelemszerűen a skálázást követően az integrálképet is meghatározzuk (a képpiramis minden szintjén). A skálázás jelentős hatással van mind a futásidőre, mind a pontosságra. Ha a faktor alacsony, például 1,05, akkor az azt jelenti, hogy a képpiramis minden szintjén 5%-kal csökkentjük a kép méretét az előző

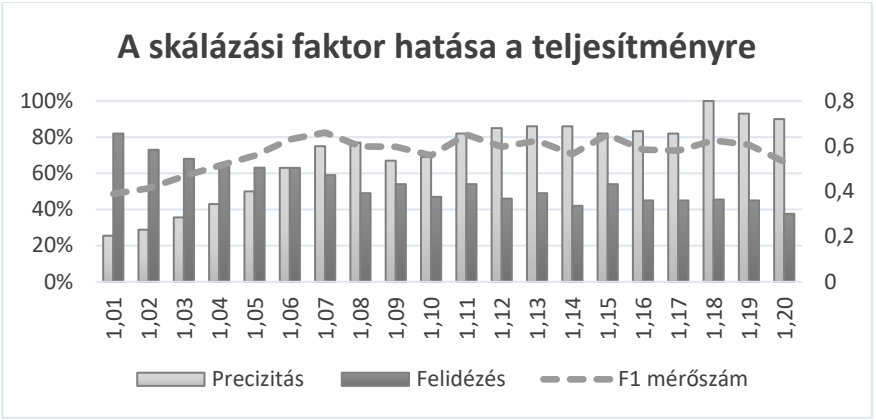
szinthez képest, így kétszer annyi szinten keresünk, mint 1,1-es skálázási faktor esetén. Általánosságban az mondható el, hogy a futásidő exponenciálisan csökken a skálázási faktor növelésével, lásd 11. ábra. Így például 1,01-es skálázási faktor mellett a futási idő 260 ms egy képre, azonban 1,1-es faktor esetén lecsökken 50 ms-ra.



11. ábra. A függőleges tengely a futásidőt jelenti milliszekundumban értelmezve, a vízszintes pedig a képpiramishoz tartozó skálázási faktort.

Jelentősen gyorsítható a rendszer, ha előzetes információval rendelkezünk az objektum lehetséges pixelméretéről. Ez esetben ugyanis csökkenthető a képpiramis azon szintjeinek a száma, melyeken a detektornak le kell futnia. Az arc méretére vonatkozó előzetes ismeret hiányában a detektor le fog futni a képpiramis összes szintjére: a tanító adatbázisbeli pozitív képek felbontásától, egészen a teljes kép felbontásáig. Összességében a skálázási faktorról elmondható, hogy minél nagyobbnak választjuk, annál kevesebb lesz a képpiramis szintjeinek a száma és lesz gyorsabb az algoritmus ezzel egyidőben.

Egy idő után nem gyorsul annyit a detektálás, hogy a minőségbeli romlás miatt az megérje tovább növelni a skálázási faktort. Tesztjeim során azt tapasztaltam, hogy a skálázási faktor növelésével elkezd növekedni a hamis negatívok száma (csökken a felidézés) és csökkenni az hamis pozitívoké (nő a precizitás) lásd 12. ábra. Ez annak köszönhető, hogy a skálázási faktor növelésével csökken a képpiramis szintjeinek a száma, azaz csökken az esélye annak, hogy egy valós arcot detektáljunk a fix méretre tanított osztályozóval. Másrészt a szintek számának csökkenésével annak az esélye is csökken, hogy egy nem létező arcot detektáljunk.



12. ábra. A *precizitás*, *felidézés* és F_1 *mérőszám* értékeinek alakulása a skálázási faktor függvényében. Az elsődleges függőleges tengely a *precizitás* és *felidézés* értékeit reprezentálja, a másodlagos pedig az F_1 *mérőszámét*.

A képpiramis fölött értelmezett csúszóablakos detektálás jellegéből fakadóan tovább javítható az osztályozás pontossága. A detektálás során az első letapogatás után néhány pixel pozícióval eltoljuk az ablakot – az eltolás mértéke függ az adott szinthez tartozó skálázási faktortól. Így kapjuk a következő pozíciót, melyre újból lefuttatjuk az osztályozást és tovább haladunk. A detektor azonban érzéketlen az eltolás és skálázás kis változásaira, ezért a gyakorlatban többszörös detektálás jelenhet meg az arc körüli régiókban. A gyakorlatban viszont egyértelmű találatokat szeretnénk, így célszerű szűrni a találatokat. A találatokat első körben diszjunkt halmazokba rendezzük, majd összevonjuk azokat a halmazokat, melyeknek vannak egymással átfedésben lévő találataik. Az összevonások végeztével kiátlagoljuk a közös halmazba eső téglalapok csúcsainak koordinátáit és az így kapott téglalap lesz a végső, egyértelmű találat. Az előbbi halmazok számosságát *topográfiai csoportszámnak* is szokás nevezni. A gyakorlatban célszerű elvetni azokat a halmazokat, melyek topográfiai csoportszáma kisebb, mint egy előre megadott küszöbszám. Ezzel csak azokat a pozitív találatokat hagyjuk meg, melyeket a küszöbszámtól több átfedő skálán, vagy pozícióban találatunk meg. Így, a küszöbszám növelésével egyre nagyobb lesz a precizitás értéke, mert fokozatosan kiszűrjük a hamis pozitív találatokat. Viszont ezzel egyszerre csökkenni fog a felidézés értéke – mert elkezdjük a valós pozitív találatokat is kiszűrni. Az összevonások nem csak az osztályozó pontosságára, hanem annak a futásidejére is hatással vannak, lásd 2.3.4. fejezet.

2.5. Arckövetés folyamata

A Viola és Jones algoritmus segítségével relatíve egyszerűen és gyorsan detektálhatunk arcokat, azonban a futásideje (lásd 0) nem teszi lehetővé a valósidejű HCI alkalmazások létrehozását, mivel a detektálás ideje még az optimális esetben is 100 ms körül alakul. A detektálás másik nagy problémája, hogy sok hibás negatív találatot eredményez akkor amikor a felhasználó erőteljesen elmozdítja, vagy elforgatja a fejét. Ez utóbbi korlátozás a tanítás jellegéből fakad, hiszen ahogy azt az előző fejezetben ismertettem, a magas pontosság érdekében célszerű a tanító adatbázist úgy összeállítani, hogy a pozitív képek hasonló elrendezésben (pl. frontális nézet) ábrázolják az arcokat. A gyakorlatban – hatékonysági okokból kifolyólag – az arcdetektáló eljárásokat gyakran kombinálják *arckövető eljárásokkal*. A gyakorlati megvalósítás során a következő sémát alakítottam ki: az arcdetektort egy előzetesen meghatározott frekvenciával futtatom a bemenő képsorozaton (pl. minden ötödik másodpercben), két detektálás között pedig egy alkalmas követő eljárással próbálom megjósolni az arc pozícióját az egymást követő képkockák felett.

A követő eljárások általában gyorsabbak, mint az arcdetektálás. Az ok emögött egyszerű: számos információ válik elérhetővé az arcról a detektálás pillanatában, többek közt az arc elhelyezkedése a képkocka fölött, mozgásának iránya és sebessége, megjelenése, textúrája, stb. A következő képkockán ez az extra információ felhasználható ahhoz, hogy megbecsüljük az arc helyzetét. Egy ideális követő eljárás minden elérhető információt felhasznál az objektumról, ezzel szemben a detektornak egy ismeretlen környezetben kell dolgoznia. Ugyanakkor nem tekinthetünk a követő eljárásokra úgy mint, ha azok univerzális eljárások lennének. A követő eljárások egyik legnagyobb hátránya, hogy a becslés hibája – még ha alacsony is – akkumulálódhat a feldolgozás során és ez a magasabb szintű komponensekre is hatással lehet. Esetünkben, a követés hibája abban jelentkezhet, hogy az arcot befoglaló téglalap fokozatosan távolodik az arc valós pozíciójától. Így a detektor rendszeres futtatása szükségszerű, hiszen általa újra inícializálható a követő eljárás. Az objektum követés egyszerűen hangozhat, de mára külön szakterületté nőtte ki magát a gépi látáson belül [24]. Széles területet fednek le a követő eljárások (pl. dense vagy sparse optical flow, meanshift, camshift) melyek ismertetése nem fér bele jelen értekezés keretébe. A tervezett HCI rendszerünk konfigurációja lehetővé teszi egy egyszerű, de hatékony követési

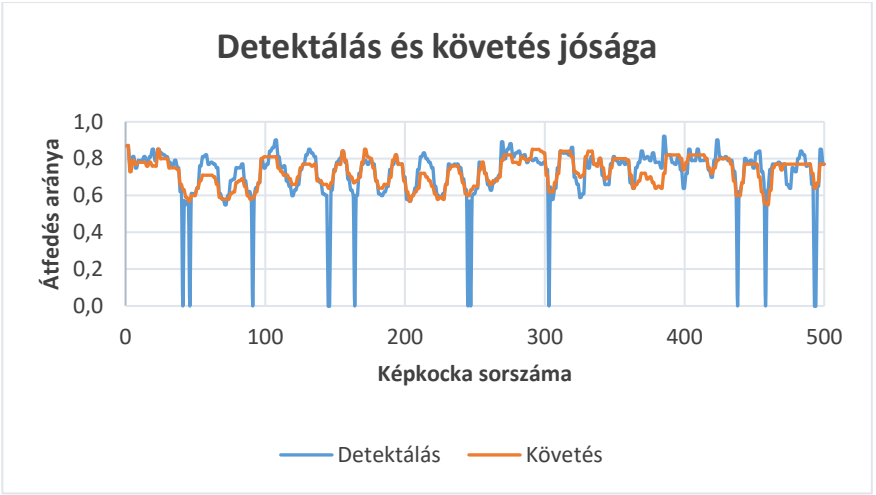
algoritmus adaptálását. Mivel a felhasználók a monitor előtt, a kamerával szemben ülnek, így az arc megjelenése és térbeli lokációja nem változik szignifikánsan két egymást követő képkocka között. Egy egyszerű sablon alapú követő eljárásra esett a választásom, mely az arc már ismert mintázata és az aktuális képkocka tartományai között számol normalizált kereszt-korrelációt egy mozgóablakos eljárás segítségével:

$$R(x,y) = \frac{\Sigma_{x',y'}\left(T(x',y')\cdot I(x+x',y+y')\right)}{\sqrt{T(x',y')^2\cdot I(x+x',y+y')^2}}, \tag{2.5}$$

ahol T egy $w \times h$ méretű sablonja az arcnak, I az aktuális képkocka, melyen meg akarjuk becsülni az arc pozícióját a sablon alapján, továbbá $x' = 1 \dots w-1$ és $y' = 1 \dots h-1$. A nevezőben lévő normalizációs együttható az eltérő fényviszonyokból eredő hibák hatását igyekszik csökkenteni. Az algoritmus kimenete egy $R \in \mathbb{R}^{(W-w+1)\times(H-h+1)}$ mátrix, feltételezve, hogy az I felbontása: W× H. Az arc pozíciója az I bemenő képen az R maximális értékének megfelelő pozícióján lesz.

2.5.1. Detektálás és követés összehasonlító eredményei

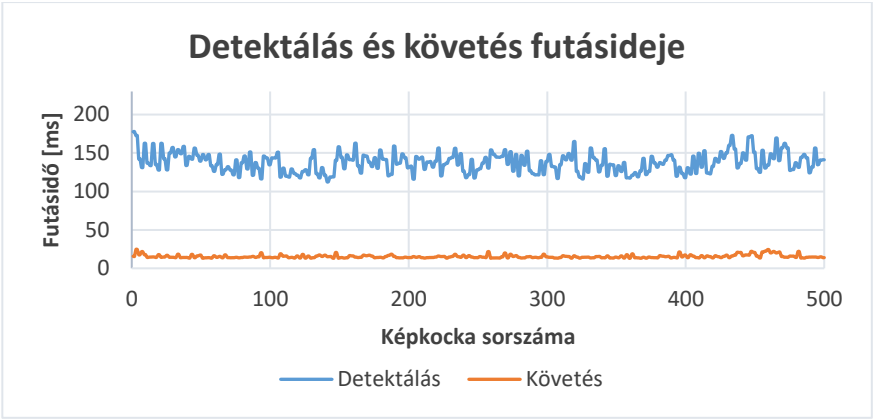
A detektálás és követés pontosságának számszerűsítése nem teljesen kézenfekvő feladat, ha nem csak a találat igaz, vagy hamis voltát akarjuk mérni. A követés hibájának a méréséhez szükségünk lesz egy módszerre, amivel két téglalap hasonlóságát tudjuk mérni az elhelyezkedésük és kiterjedésük tekintetében. A két téglalap közötti hasonlóságot definiálhatjuk az átfedésük mértékében. Legyen adva egy A és B téglalap: metszetük területét jelöljük S_I-vel, az uniójukét pedig S_U-val. Könnyen belátható, hogy az S_I/S_U arány értéke 1, ha a két téglalap teljes átfedésben van egymással és 0, ha egyáltalán nincsenek átfedésben egymással. Az előbb bevezetett arányszám segítségével a találat igaz vagy hamis volta mellett, visszacsatolást adhatunk a detektálás és követés jóságának a mértékére is. Az eredmények összevetéséhez lefuttattam egy tesztet, egy 30 másodperc körüli, 640×480 pixel felbontású videóra 15 FPS mintavételezési sebesség mellett. A videón előzetesen annotáltam az arc pozícióját. Megjegyzem, hogy a teszteredmények valamennyire szubjektívek, hiszen az arc annotációja egy intuitív folyamat eredménye. A videón változatos fejmozgásokat végeztem úgy, hogy a fej mindvégig frontális és ahhoz közeli pozícióban látszódott a kamerához képest.



13. ábra. Az S_I/S_U arányszámok külön-külön a detektálásra és a követésre meghatározva. Az arányszám 1, ha a detektálás, vagy követés téglalapja teljes átfedésben van az előzetesen annotált arccal és 0, ha egyáltalán nincsenek átfedésben egymással.

Külön-külön meghatároztam az S_I/S_U arányszámokat a detektor kimenetére és az annotált arcpozícióra; valamint a követés kimenetére és az annotált arcpozícióra (lásd 13. ábra). Az eredményeket tekintve mindössze 13 darab képkockán nem talált arcot a detektor, főleg akkor amikor erőteljesebben elfordítottam a fejem a kamerától. A hamis pozitív találatokkal nem foglalkoztam, viszont a felidézés értéke az előbbiek ismeretében: 0.97, ami meglehetősen jónak számít. A grafikonon az is jól látszik, hogy a követő algoritmus által becsült arcpozíció együtt mozog a detektoréval és rajta marad az arcon akkor is, amikor a detektor hibázik. Az S_I/S_U arányszámokra átlagosan 0.7 körüli érték adódik, ami jó átfedésnek tekinthető.

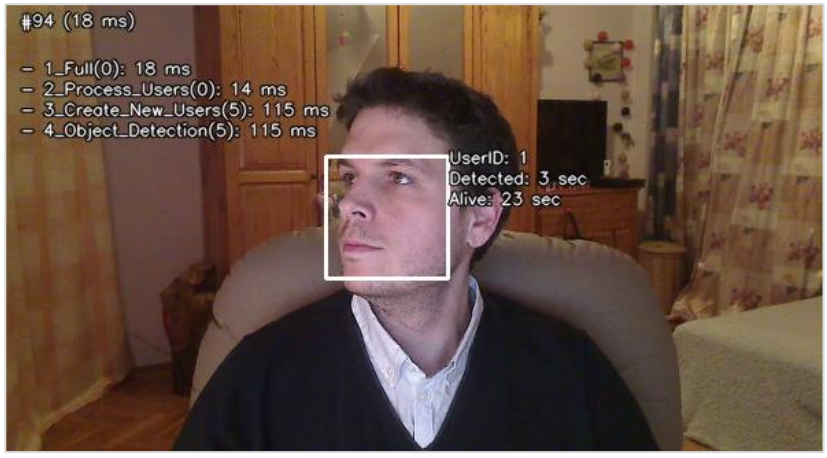
A teszt során 5 másodpercenként újra inicializáltam a követő algoritmust a detektor találatával, így a követés hibája maximum ennyi ideig halmozódhatott fel. A detektálási frekvencia a későbbiekben felhasználás függő lehet, de a futási idők (lásd 14. ábra) ismeretében bíztató, hogy nem kell gyakran detektálni a bemenő képfolyamon. Az arcdetektor átlagos futásideje 137 ms, ezzel szemben a követés lefut 15 ms alatt, tehát közel tízszer gyorsabban. Ez utóbbi futásidő kielégíti a valós idejűség kritériumát és lehetővé teszi a későbbi bonyolultabb számítások elvégzését is.



14. ábra. A detektálás és követés futásideje. Jól látszik, hogy a követés egy nagyságrenddel gyorsabb a detektálásnál.

2.6. Felhasználói azonosítás folyamata

Egy másik nagy előnye a követő eljárásoknak, hogy segítségükkel azonosságot tudunk rendelni az objektumokhoz. Ameddig a detektorok csak egy téglalap halmazzal térnek vissza eredményként, addig a követő eljárások – két egymást követő képkockát tekintve – egy kapcsolatot is felépítenek a téglalapok között. A követő algoritmus segítségével meta adatokat rendelhetünk a felhasználókhoz, mely magában foglalhatja az utolsó detektálás idejét, az aktuális befoglaló téglalapját az arcnak, vagy akár annak a textúráját is. Néhány meta adatot a 15. ábra is szemléltet ezek közül.



15. ábra. Az arcdetektálás, követés és azonosítás kimenete. Az utolsó sikeres detektálás futási ideje 115 ms volt, a követés viszont jelentősen gyorsabb, 15-20 ms alatt lefut.

A 15. ábrán az látható, hogy a felhasználót 23 másodperce tartjuk nyilván az alkalmazásban és az utolsó sikeres detektálása az arcának 3 másodperccel ezelőtt történt. A felhasználóhoz tartozó meta adatok eltárolásra kerülnek és a későbbiekben bonyolultabb döntések meghozatalához is felhasználjuk azokat. A felhasználó azonosítása az alábbi módon történik:

- (Felhasználói) azonosságot csak az arcdetektorok hozhatnak létre:
  - Ha az arcdetektor egyik kimenete legalább 80%-ban átfedésben van egy már meglévő azonosság befoglaló téglalapjával, akkor frissítjük a meta adatok körét a detektálás eredményével,
  - Különben egy új azonosságot hozunk létre a rendszerben a detektálás kimenetének megfelelően.
- A meglévő azonosságoknak megfelelő felhasználókra folyamatosan futtatjuk a követő algoritmusunkat a bejövő képszekvencia elemein:
  - Ha a követés sikeres, akkor frissítjük a meta adatok körét,
  - Különben töröljük mind az adott azonosságot, mind a hozzá kapcsolódó összes meta adatot.
  - Akkor is töröljük az azonosságot, ha több mint 10 másodperce egyetlen detektor találat sem adódott hozzá. Tesszük mindezt azért, mert egy 10 másodpercnél hosszabb követés esetén a hiba erőteljesen felhalmozódhat a rendszerben.

2.7. Összefoglalás

Jelen fejezetben ismertettem, hogy egy kameraképből kiindulva hogyan valósítható meg a HCI rendszerem alapját képező arc lokalizációja. Ismertettem az egyik legkorszerűbb objektumdetektor (Viola-Jones) működését és korlátait az emberi arcok lokalizációja kapcsán. Tanítások és tesztek hosszú sorozatán át javaslatot tettem az objektumdetektor optimális paraméterezésére. Vizsgáltam a tanító adatbázisok összetételét, a tanításhoz szükséges minták darabszámát, a tanítóminták felbontását, a kaszkád struktúra szintjeinek számát, a jellemzőkinyerő algoritmusok lehetséges körét és a boosting algoritmusok típusát. Egyszerre vizsgáltam ezek hatását a teljes rendszer pontosságára, valamint futásidejére vonatkozóan. Úgy találtam, hogy az általam tanított legjobb detektor segítségével képes lehetek változatos orientációjú arcokat detektálni viszonylag alacsony futásidő mellett. Ugyanebben a fejezetben ismertettem az általam létrehozott arckövető és

azonosító rendszeremet, mely segítségével jelentősen lecsökkentettem az arc lokalizációjához szükséges futásidőt. Továbbá a rendszer segítségével metaadatokat is rendeltem a kamera képeken látható arcokhoz további feldolgozások céljából. Az elvégzett teszteket egy Intel (R) Core(TM)2 Quad CPU Q9550 @ 2.83 GHz processzorral felszerelt számítógépen végeztem el. A rendszert C++-ban implementáltam. Az implementáció egyszerre több befogadó platformon is fordítható és futtatható, például Android operációs rendszert használó telefonokon is [25].

3. Karakterisztikus pontok az arcon

3.1. Irodalmi áttekintés

A további vizsgálatokhoz meg kell keresni az arc azon részeit, melyek releváns információt kódolnak. Például, a felhasználó gesztusainak felismeréséhez meg kell keresnünk az arcnak azon karakterisztikus (vagy jellemző) pontjait, amelyek részt vesznek a mimika, az arcmozgás vagy akár az érzelmek kifejezésében. E jellemzőpontok helyzetéből, mozgásából, vagy egyéb származtatott mennyiségeiből további következtetéseket lehet levonni, vagy akár osztályozni is tudjuk őket. PhD tanulmányaim során, meglévő módszerek tanulmányozása és felhasználása által létrehoztam egy alrendszert az arc karakterisztikus pontjainak reprezentálására. A feladat magában foglalta a szükséges tanító-, és teszt adatbázisok összeállítását; az eredmények méréséhez szükséges tesztkörnyezet kialakítását; tesztesetek meghatározását; metrikák készítését az eredmények összehasonlítására.

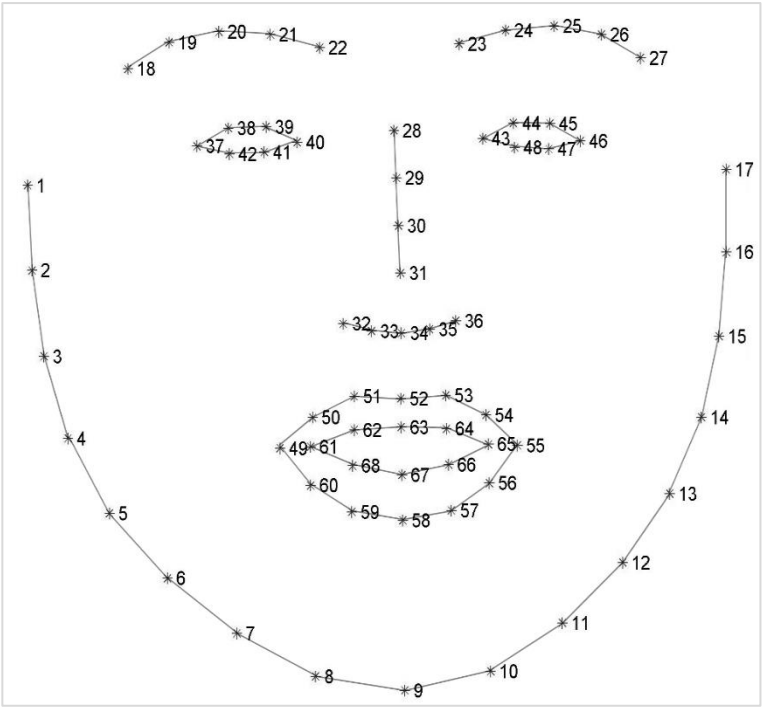
Az arci jellemzők követésének módszereit a szakirodalom két nagy kategóriába sorolja. Egyrészt beszélhetünk általános célú jellemző detektorokról, melyek általános célú pontdetektáló és követő eljárások. Többségére igaz, hogy nem rendelkeznek előzetes információval a modellezni kívánt objektum egészéről, hanem csak az arc néhány fontos jellemzőjét (pl. szemek és a száj középpontjai) detektálják. Egyes módszerek megszorításokat is bevezetnek a karakterisztikus pontokra vonatkozóan, hogy helyes konfigurációt kapjanak eredményül, lásd [26, 27, 28, 29]. A kutatásaim során én is próbálkoztam hasonló megoldásokkal: [30, 31]. A módszerek egyik előnyeke, hogy kevésbé érzékenyek a kezdeti pozícióra, nem úgy, mint a például takarásra, mely korlátozza a sikeres detektálást. A másik nagy kategóriába a statisztikai modell alapú módszerek tartoznak, melyek, a modellezni kívánt objektum (továbbiakban alakzat) körvonalának geometriai tulajdonságait, illetve a körvonalat övező megjelenést vizsgálják különböző statisztikai eljárások segítségével. E módszerek általában definiálnak egy mérőszámot, mely segítségével az alakzatok körvonalai, vagy azok megjelenése, nézőponttól és megvilágítástól függetlenül összehasonlíthatók. Továbbá meghatároznak egy átlag alakzatot, vagy esetenként megjelenést is – ez utóbbit többnyire véletlenszerűen generált mintákból.

A statisztikai modell alapú eljárások működése általában két lépésből áll: első lépésben egy modellt szokás építeni. Majd a kapott modellt illesztjük egy új képre: ahol egy ismeretlen archoz tartozó marker pontokat (más néven határpontokat) határozunk meg. A modellezni kívánt alakzatot a markerek és a közöttük fennálló kapcsolatok sorozataként szokás definiálni. Az alakzat geometriájának reprezentálása egy úgynevezett *pont-eloszlási modellt* (*point distribution model*, röviden *PDM*) szokás használni, mely az alakzat, mint egy nem-merev test lehetséges deformitásait, azaz az alakzatot alkotó markerek elmozdulásának lehetséges irányát és nagyságát írja le. A korszerű eljárások általában ezt az információt kombinálják a határpontok környezetében található textúra-információval. Számos példa létezik a szakirodalomban a deformálható objektumok modellezésére, pl. *aktív kontúr modellek*, *active shape models* (röviden *ASM*), *active appearance models* (röviden *AAM*), vagy *constrained local models* (röviden *CLM*). A PhD tanulmányaim alatt, magam is számos esetben használtam statisztikai modelleket arci jellemzők detektálására és követésére egyaránt.

A statisztikai modellezés során általában a karakterisztikus pontok elrendezésében és megjelenésében jelentkező kettős szemléletmód együttes hatása jelenik meg egy kombinált modellben: mely így egy alakzat-, és textúra modellből áll. Az irodalomban lévő eljárások többnyire csak a textúra modell definiálásában térnek el egymástól, az alakzatmodellt ugyanúgy állítják elő. A textúra modell szerepe a kombinált modell illesztése során mutatkozik meg. A modell illesztése egy optimalizációs feladat megoldásaként értelmezhető, és modell tanítás során meghatározott paraméterek változtatásával érhető el. Lényegében a határpontok környezetéből kinyerhető textúra információk alapján történik az illesztés, melyeket a modellalkotás folyamán „tanul meg" a rendszer.

3.2. Active Shape Models

Az ASM [32] alakzatmodellezési technikája az alapja majdnem az összes modern és hatékony statisztikai modellező eljárásnak. Az alábbiakban ezt módszert ismertetem röviden.



16. ábra. Egy lehetséges alakzatmodellje az arcnak. A modell 68 db határpontból áll, jelen értekezés alapjául is ezt az alakzatmodellt használtam.

Definíció (alakzat): az alakzat fogalmát a geometriában a ponthalmaz és a mértani hely szinonimájaként szokás használni. Az alakzatokat reprezentálhatjuk a körvonaluk mentén mintavételezett véges számú úgynevezett határpontjaikkal. Esetünkben a határpontok az arc határvonalai mentén, intenzitásváltozások formájában jelentkeznek (lásd 16. ábra). Az alakzatok megfeleltethetők egymásnak a határpontjaik segítségével. A modell tanítása során a határpontokat egyesével kell felvenni a tanító adatbázis minden egyes képén. A határpontok összekötésével egy kontúrt kapunk. Így tetszőleges s alakzat reprezentálható a határpontjainak sorozatával:

$$s = \{b_i \in \mathbb{R}^n : i = 1 \dots N\} \tag{3.1}$$

A kontúrt, a b_i határpontokhoz tartozó helyvektorok konkatenálásával kapjuk. Síkbeli alakzatok eseten $n = 2$, így az egyes b_i határpontok megadhatók az (x_i, y_i) formában és az $s \in \mathbb{R}^{2N}$ alakzat ábrázolható egy $2N$ komponensű vektorként:

$$s = (x_1 \dots x_N, y_1 \dots y_N)^T. \tag{3.2}$$

3.2.1. Alakzatok közös koordináta-rendszere

Adva van tehát arcadatbázis, mely az arcokat tartalmazó képekből és a hozzájuk tartozó alakzatmodellekből áll. A különböző arcok alakzatmodelljei ugyan hasonlítanak egymáshoz, de mégis különböző konfigurációkat alkotnak. Ugyanakkor csak akkor lehet összefüggéseket megállapítani az adatbázisból, ha az egyes alakzatok összehasonlíthatók egymással. Ehhez az alakzatokat közös koordináta-rendszerben kell megadnunk, mely lényegében az alakzatok egymásra való optimális illesztését jelenti. Erre alkalmas módszer a *Prokrusztész analízis* (*generalised Procrustes analysis*, vagy röviden *GPA*) [33]. A GPA a ponthalmazokat egy közös ekvivalencia-osztályba sorolja, a közöttük fennálló transzformációk (eltolás, forgatás és uniform skálázás) eltávolításával. Az eljárás bevezet egy távolság fogalmat: a Prokrusztész távolságot. Az illesztés során ezt felhasználva minimalizáljuk az egyes alakzatok közötti különbséget. Az alakzatokat közös koordináta-rendszerben írjuk fel egy $\bar{s}$ átlagalakzat alapján (lásd 3.3-as egyenlet), majd minimalizáljuk az egyes alakzatoknak az $\bar{s}$-től való négyzetes Prokrusztész távolságát.

3.2.2. Statisztikai alakzatmodell

Adva van tehát k darab alakzat: $\{s_i | s_i \in \mathbb{R}^{2N}\}_{i=1\dots k}$, melyet a GPA segítségével egy közös koordináta-rendszerben vettünk fel. Az s_i-ik alapján szeretnénk egy paraméteres statisztikai modellt definiálni, mely segítségével képesek lehetünk hasonló elrendezésű új alakzatokat generálni. A k darab alakzat egy-egy valószínűségi változó sűrűségfüggvényeként is értelmezhető a 2-D térben. Azonban az adatok dimenziója ($2N$) túl magas ahhoz, hogy a modellalkotás és illesztés folyamata hatékony legyen, így célszerű egy *főkomponens-analízis* (*principal component analysis*, röviden *PCA*), segítségével az adatok dimenziószámát lecsökkenteni kezelhető méretűre. Első lépésben, meghatározzuk az alakzatok átlagát:

$$\bar{s} = \frac{1}{k}\sum_{i=1}^{k} s_i, \tag{3.3}$$

majd az egyes s_i alakzatoknak az $\bar{s}$-tól vett kovariancia mátrixát:

$$C_s = \frac{1}{k-1}\sum_{i=1}^{k}(s_i - \bar{s})\,(s_i - \bar{s})^T. \tag{3.4}$$

Számítsuk ki a C_s mátrix φ_i sajátvektorait és a hozzájuk tartozó λ_i sajátértékeket ($i = 1 \dots k$). Rendezzük a φ_i-ket a sajátérték nagysága szerinti

csökkenő sorrendbe. A sajátvektorok egy k dimenziós ellipszoid tengelyeit jelölik ki, a sajátértékek pedig azt mondják meg, hogy mekkora a tengelyek mentén a variancia. A dimenziószám csökkentéséhez hagyjuk el azokat a főkomponenseket, melyek mentén kicsi a variancia. Ezzel arányosan csak kis adatmennyiséget vesztünk azaz modell kifejezőereje nem változik jelentősen, de az komplexitása érezhetően csökken. Válasszuk ki a t darab ($1 \leq t \leq k$) legnagyobb sajátértékhez tartozó sajátvektort és képezzünk belőlük egy $\Phi \in \mathbb{R}^{2N \times t}$ mátrixot:

$$\Phi = [\varphi_1 \quad \varphi_2 \quad \dots \quad \varphi_t]. \tag{3.5}$$

A gyakorlatban, a Φ mátrix segítségével lehetséges a tanítóhalmazban lévő alakzatok közelítése az alábbi formában (alakzatmodell):

$$s \approx \bar{s} + \Phi b, \tag{3.6}$$

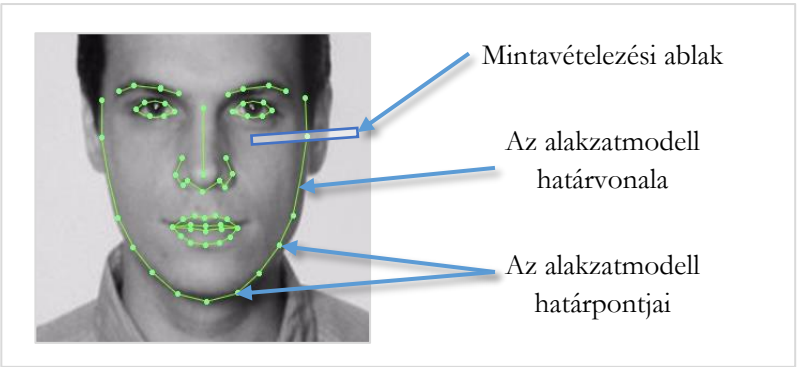
ahol $b \in \mathbb{R}^t$ oszlopvektor a főkomponensek skálázási faktora, más néven a deformálható modell paraméterhalmaza. A fenti egyenletből következően $b = \Phi^T(s - \bar{s})$. A $b \in \mathbb{R}^t$ oszlopvektor megválasztásával új, a tanítóhalmazban lévő alakzatokhoz hasonló elrendezésű alakzatokat tudunk generálni. Az egyes $b_i \in \mathbb{R}$ komponensekkel lehet pl. a száj nyitottságának mértékét, vagy az arc formáját kontrollálni. Értelemszerűen csak azoknak az arci jellemzőknek változtatható a geometriája a b_i komponensekkel, melyek kellő hangsúllyal vannak reprezentálva a tanító adatbázisban. A gyakorlatban a b_i komponenseket általában a $\pm 3\sqrt{\lambda_i}$ intervallumból mintavételezik (lásd 17. ábra).

17. ábra. A b paramétervektor változtatásával generált alakzatok. Az egyes $b_i \in \mathbb{R}$ komponensek $\pm 3\sqrt{\lambda_i}$ intervallum felett voltak értelmezve.

3.2.3. Illesztés menete

Az illesztésről csak röviden írok, mivel az alakzatmodellel ellentétben nem használom fel közvetlenül a további fejezetekben. Az illesztés során kifeszítjük az $\bar{s}$ átlagalakzatot az arc fölé, majd oly módon transzformáljuk azt, hogy a legjobban illeszkedjen az ismeretlen archoz. Tehát azt az s alakzatmodellt keressük, amely a legideálisabban írja le egy bejövő képen látható ismeretlen arc karakterisztikus pontjait. A modell illesztése egy költségfüggvény minimalizálását jelenti – ügyelve arra, hogy az alakzatra vonatkozó korlátozások érvényben maradjanak. A $b \in \mathbb{R}^t$ paramétervektor biztosítja azokat a korlátozásokat, amelyek megakadályozzák, hogy az alakzatmodell hamisan torzuljon az illesztés során. Ha nem rendelkezünk pontos információval arra vonatkozóan, hogy az illeszteni kívánt alakzatmodellt a minimalizációs eljárás első lépésében hova kell kifeszíteni, akkor egy meglehetősen nehéz optimalizálási feladattal találjuk szembe magunkat.

Az ASM, a illesztés során az alakzat körvonalára merőlegesen keresi a határpontok ideális helyét a bemenő képen. Ehhez a tanító adatbázisból származtatott pixelmintákat is felhasznál, melyek egyfajta kezdetleges textúra modellként is értelmezhetők. Az ASM, a k darab alakzat összes határpontjára merőlegesen egy egységnyi hosszúságú mintavételezési ablakot illeszt, úgy, hogy az ablak középpontjában az adott határpont legyen. A mintavételezi ablak lényegében egy függvény, mely minden határponthoz egy $(2w + 1)$ hosszúságú pixelmintát (1-D tömböt) rendel, ahol w a mintavételezési ablak hosszúságának a fele (lásd 18. ábra).



18. ábra. Lokális pixelminták mintavételezése. Az egyes ablakokat (*kék körvonalú téglalap*) merőlegesen illesztjük az alakzatmodell határvonalán (*zöld szakaszok*) elhelyezkedő határpontokra (*zöld pontok*).

3.3. Active Appearance Models

Az AAM technika [34] egy az ASM-hez hasonló és részben azon alapuló statisztikai modellező eljárás. Az alábbiakban megpróbáljuk röviden összehasonlítani az ASM-mel, hogy rögtön a fejezet elején felhívjuk a figyelmet a fontosabb különbségekre és a konstrukció alapelveire:

- Az ASM közel sem használja ki az összes rendelkezésre álló információt a modellépítés és illesztés során. A kezdetleges textúra modellbe határpontonként csak a körvonalra merőlegesen irányban elhelyezkedő pixelek intenzitásértékeit veszi bele. Ezzel szemben az AAM az alakzat teljes megjelenését felhasználja.
- Az ASM algoritmus az illesztés minden iterációjában csak a határpontok aktuális pozíciójában keresi azok új és jobb helyét. Ezzel szemben az AAM minden egyes iterációban próbál az adott arcnak jobban megfelelő textúrát elállítani és illeszteni.
- Az ASM, az alakzatmodell határpontjai és az arcon az azoknak megfelelő pontok közötti távolságot minimalizálja. Ezzel szemben az AAM bemenő kép és egy hasonló megjelenésű, szintetikusan generált kép megjelenésbeli különbségét minimalizálja.

3.3.1. Statisztikai textúra modell

Az AAM egy alakzat-, és egy textúra modell kombinációja. Az s alakzatmodell egy az egyben megegyezik az ASM-nél ismertetettnél (lásd 3.2.2. fejezet). A textúra modell előállítása során az k darab arcmintát kifeszítjük az átlag $\bar{s}$ alakzat felé, úgy, hogy az alakzatmodelleket alkotó határpontok ugyanarra a pixel koordinátára essenek. Az így kapott arcokat *alakzatnormalizált foltnak* (*shape-free*, vagy *shape-normalized patch*) hivatkozza a szakirodalom. Az foltok mindegyike ugyanannyi szürkeskálás pixelt tartalmaz, így leképezhetők k darab M hosszúságú textúra vektorra, jelöljük ezeket a következő módon: $\{g_i | g_i \in \mathbb{R}^M\}_{i=1\ldots k}$. A textúra g_i vektorokat az átlaguk (μ_{g_i}) és szórásuk (σ_{g_i}) alapján normalizáljuk az eltérő megvilágításból fakadó hatások kiszűrése végett:

$$g_i' = \frac{g_i - \mu_{g_i}}{\sigma_{g_i}}. \tag{3.7}$$

Az alakzatmodellhez hasonlóan egy PCA-t alkalmazunk a g_i' textúra vektorokra annak érdekében, hogy a kapott textúramodell kezelhető méretű legyen:

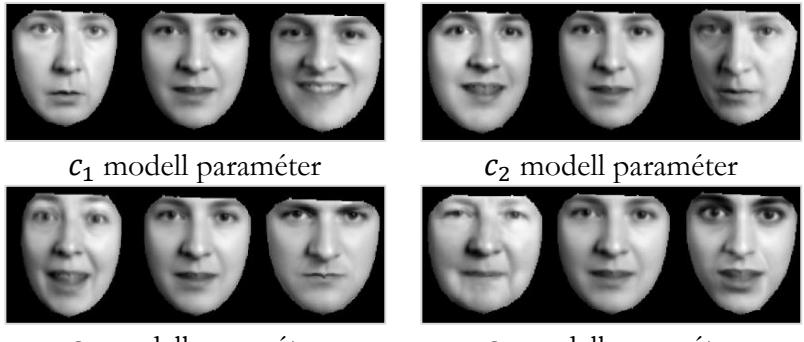
$$g \approx \bar{g} + \Phi_g b_g, \tag{3.8}$$

ahol a $\bar{g}$ vektor a normalizált átlagtextúra (lásd 19. ábra); $\Phi_g \in \mathbb{R}^{M \times t}$ a textúra vektorokhoz számolt kovariancia mátrix t darab legnagyobb sajátértéhez tartozó sajátvektorait tartalmazza; $b_g \in \mathbb{R}^t$ oszlopvektor pedig a textúra paraméterek sorozata. A $b_g \in \mathbb{R}^t$ paramétervektorral jelen esetben a világosság-, és kontrasztbeli megjelenést manipulálhatjuk.

19. ábra. Az átlag textúra az alakzatnormalizált koordináta-rendszerben (*bal oldali ábra*); illetve az alakzatnormalizált koordináta-rendszerben az egyes pixelek intenzitásértékeinek a szórása (*jobb oldali ábra*).

3.3.2. Kombinált modell

Az alakzat-, és textúra modell paraméterei közötti korrelációk feltárásához vegyítsük e két modellt, majd alkalmazzunk egy újabb PCA-t az így létrejött kombinált modellre, az alábbiakban szerint.



20. ábra. A c paramétervektor első négy paraméterének – c_1, c_2 c_3 és c_4 – változtatásának a hatása. A középső képek az átlagértékkel készültek; a bal-, és jobb oldali képek pedig rendre ±3 szórással az átlagtól.

Első lépésben állítsuk elő az alábbi b vektort, a b_s alakzat-, és a b_g textúra modell paramétervektorokból:

$$b = \begin{pmatrix} W_s b_s \\ b_g \end{pmatrix} = \begin{pmatrix} W_s \Phi_s^T (s - \bar{s}) \\ \Phi_g^T (g - \bar{g}) \end{pmatrix}, \tag{3.9}$$

ahol W_s az alakzat paraméterekhez rendelt diagonális súlymátrix, az alakzat paraméterek változásának hatását fejezi ki a textúra paraméterekre vonatkozóan. Az alakzat és textúra paraméterek eltérő mérték-egységeiből fakadó hatást kompenzáljuk a segítségével. A tanítás során perturbáljuk a b_s alakzatparamétereket, majd mintavételezzük a tanítómintákat a perturbáció eredményeként kapott alakzatpontok felett. A W_s mátrix elemei a perturbációt megelőző-, és az azt követő intenzitásértékek közötti eltérést kódolja. A vektorokra egy újabb PCA-t alkalmazva kapjuk az alábbi modellt:

$$b \approx Qc, \tag{3.10}$$

ahol a Q mátrix a b paraméter mátrix sajátvektorait tartalmazza, a c vektor pedig a modell megjelenési paramétereit. Ez utóbbi segítségével lehet a modell formáját (b_s komponens) és színezettségét (b_g komponens) egyszerre változtatni. A 20. ábra a c paramétervektor első négy paraméterének változtatásának hatását mutatja, ahol a c_1 paraméter a kovariancia mátrix legnagyobb sajátértékéhez tartozik. Látható, hogy a változások, mind a megjelenésben, a pozícióban és az arckifejezésekben is megjelennek. Mind az alakzat-, mind a textúramodell felírható a c vektor segítségével:

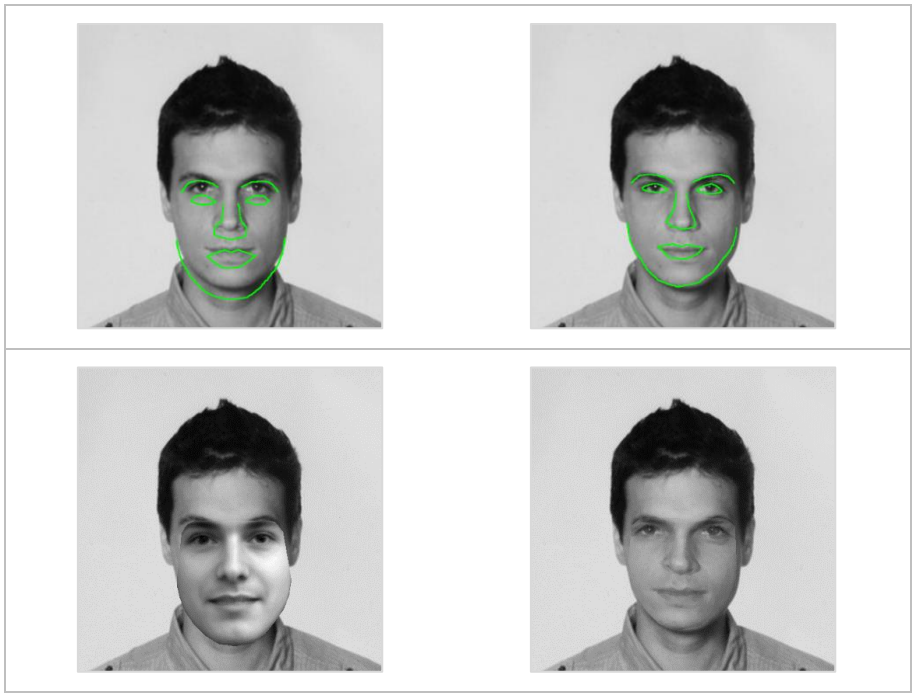
$$s \approx \bar{s} + \Phi_s W_s Q_s c, \text{ valamint } g \approx \bar{g} + \Phi_g Q_g c \tag{3.11}$$

Így a c paramétervektor komponenseinek a változtatásával, egyrészt megvalósítható egy textúra szintézis a g textúra modell segítségével, másrészt az így kapott alakzatnormalizált folt kifeszíthető az s alakzatmodell által meghatározott határpontok felé. A 3.11. egyenletben megadott modellek együttesét a szokás *megjelenési modellnek* vagy *kombinált modellnek* hivatkozni.

3.3.3. Illesztés menete

Csak úgy, mint az ASM-nél, terjedelem hiányában itt sem foglalkozok részletesebben az illesztési algoritmussal. Az AAM illesztési algoritmusa egy bemeneti kép, és egy mesterségesen generált kép (paraméteres megjelenési modell) közötti különbséget minimalizálja. Az illesztés első lépésében

feltételezzük, hogy ismert az a pozíció, orientáció és skála, mellyel a mesterségesen előállított modell durván a bemenő arc felé helyezhető. Az optimalizálási feladat pedig megfogalmazható úgy, mint a c paramétervektor iteratív finomhangolása, addig ameddig egy olyan megjelenési modellt nem generálunk, amely a lehető legjobban illeszkedik a képen látható archoz. A 21. ábra szemlélteti az illesztés folyamatát a kezdőpozícióból kiindulva egészen a konvergencia eléréséig.



21. ábra. Az AAM illesztés kimenete egy ideális kezdőpozícióból indítva az algoritmust. A *felső sor* az alakzatmodell változását demonstrálja kezdő pozíciótól, egészen a konvergencia eléréséig. Az *alsó sor* pedig a textúra modell változását reprezentálja az előzőnek megfelelően.

3.4. Constrained Local Models

A CLM is egy statisztikai modellező eljárás [35, 36, 37], mely az előzőekben ismertetett ASM és AAM módszerek előnyös tulajdonságait vegyíti és növeli ezáltal a modellezés kifejezőerejét: az AAM-hez hasonlóan, a CLM is az alakzat-, és a képi megjelenés (textúra) együttes modellezésén alapul, azonban a *textúra modellezése* nem globálisan az arc egészén, hanem mint ahogy a nevében is benne van, lokálisan történik. Az illesztés során természetesen

szükség van az arc pozíciójának az ismeretére, mellyel megbecsülhető a modellkezdeti helye is a bemenő kép felett. Az alakzat modellezése megegyezik az eddig ismertetettnél (lásd 3.2.2. fejezet), azonban a textúra modellezése eltér az AAM-nél ismertetetttől (lásd 3.3.1. fejezet). Nem a teljes arc textúráját használjuk, hanem az alakzatmodell határpontjai körül mintavételezett foltokból állítjuk elő a textúramodellt. A CLM illesztése tehát lokális képi információn alapul, így kombinálva a lokális jellemződetektorokat, a megjelenési-, illetve az alakzatmegszorításos modelleket használó módszerek előnyeit. A CLM technika alapvetően az alakzatmodell határpontjait körülvevő megjelenés modellezésén alapul. Modellillesztés során a következő iteratív algoritmus fut le:

- A határpontok környezetére minden iterációban új *sablonokat generálunk*, melyek az alakzatmodell határpontjaihoz tartozó detektorokként is értelmezhetők.
- Mindeközben figyelembe vesszük az arc geometriájára vonatkozó megszorításokat.

A sablonok generálása egy alakzat-, illetve textúra modell kombinációjaként előálló *kombinált modell* felhasználásával történik. Az illesztés során hasonlóan járunk el, mint az AAM esetén, amikor is egy mesterségesen generált kép és egy bemenő kép között határozzuk meg a hasonlóságot. Mintavételezzük a bemenő kép azon régióit melyek az alakzatmodell határpontjainak a környezetébe esnek, majd normalizált korrelációt számolunk közöttük. A korreláció eredményeként úgynevezett válaszfelületeket lapunk. A modellillesztés egy optimalizációs eljárás, mely során úgy változtatjuk az alakzatmodell paramétereit, hogy közben maximalizáljuk az egyes határpontokban kapott válaszok összegét. Minden iteráció végén új jelölteket kapunk a jellemzőpontok helyére vonatkozóan, majd néhány iterációs lépésen keresztül folytatjuk az eljárást. Az illesztés során a konvergenciát egyrészt az befolyásolja, hogy a tanító adatbázis mennyire változatos és reprezentatív; másrészt pedig az, hogy mekkora pontossággal detektáltuk az arcot, amely fölé az első iterációs lépésben kifeszítjük a modellt.

3.4.1. Kombinált modell

Az s alakzatmodell előállítása az eddieknek megfelelően történik. A g textúra modellt, a határpontoknak megfelelően N darab *textúra folt* függvényében definiáljuk. A textúra foltokat leképezzük egy vektorra, íz így kapott N darab

vektort összefűzzük egy szürkeskálás értékeket tartalmazó, közös vektorba, melyekből adott számú tanítóminta esetén megkonstruálható a g textúra modell. Az alakzat-, és textúra modell paraméterei közötti korrelációk feltárásához a CLM ugyanúgy vegyíti az s és g modelleket egy közös kombinált modellbe, mint az AAM, majd b_s alakzat-, és a b_g textúra modell paramétervektorok együttesére alkalmaz egy újabb PCA-t (lásd 3.10. egyenlet).

A 22. ábra a c paramétervektor első két paraméterének változtatásának hatását reprezentálja, ahol a c_1 paraméter a kovariancia mátrix legnagyobb sajátértékéhez tartozik. Látható, hogy a változások, mind a megjelenésben, a pozícióban és az arckifejezésekben is megjelennek. Tehát a c paramétervektor komponenseinek a változtatásával egyrészt megvalósítható egy textúra szintézis a g textúra modell segítségével, másrészt az így kapott alakzatnormalizált folt kifeszíthető az s alakzatmodell által meghatározott határpontok felé, azaz a segítségével egyszerre lehet a modell formáját, mind a megjelenését variálni.

c_1 modell paraméter c_2 modell paraméter

22. ábra. A c paramétervektor első két paraméterének – c_1 és c_2 – változtatásának a hatása. A középső képek az átlagértékkel készültek; a bal-, és jobb oldali képek pedig rendre ±3 szórással az átlagtól.

3.4.2. Alakzatmegszorításos keresés

Az illesztés menetét a CLM esetében sem ismertetem részletesen. A modell illesztését a szerzők alakzatmegszorításos keresésnek hivatkozzák, mivel az optimalizációs feladat költségfüggvénye függ az alakzatmodell paramétereitől. Az optimalizáció során meghatározzuk a korrelációt a sablonok és a bemenő képnek a modell határpontjai körül mintavételezett régiói között. Ennek eredményeként N darab válaszfüggvényt (2-D felületet) kapunk, melynek megkeressük a maximum helyeit úgy, hogy közben figyelembe vesszük az alakzatra vonatkozó megszorításokat. A modell illesztése egy célfüggvény optimalizációját jelenti, melyhez a szerzők a *Nelder-Mead algoritmus*t használták fel, lásd [38, 39].

3.5. Összehasonlító eredmények

Az előző fejezetekben ismertetett arcmodellező eljárások összevetésére teszteket végeztem, három darab, kutatási célokra szabadon felhasználható, interneten elérhető implementációra:

- ASM: Active Shape Models with Stasm [40, 41],
- AAM: The AAM-API [42, 43],
- CLM: Face Analysis SDK [36].

Mind a tanítás, mind a tesztelés során az arcdetektálás fejezetben felhasznált adatbázisokból válogattam ki képeket (lásd [10, 11, 12]) a 4. és 5. táblázatnak megfelelően. A tanítást és a tesztelést itt is a 2.3-as fejezetben ismertetett számítógépen futtattam le. A képek felbontása mind a tanítás, mind a tesztelés során 640×480 pixel volt. Viszonyításképpen egy arcon belül a szem, illetve száj felbontása hozzávetőlegesen 45×25 és 80×35 pixel volt. A képek annotálását az AAM-API-ban közzétett annotációs szoftverrel végeztem el: [42, 43]. Az alakzatmodellemnek a széleskörben elterjedt CMU Multi-PIE adatbázis alakzatmodelljét használtam: [44, 45]. Az alakzatmodell 68 darab pontból áll és a geometriáját a 16. ábra mutatja. A tanítóadatbázis elrendezését a 4. táblázat tartalmazza.

Arc pozíció	Képek darabszáma	Személyek száma
Frontális	200	100
Frontális és bal profil átmenet	200	100
Frontális és jobb profil átmenet	200	100

4. táblázat. A tanítóadatbázis elrendezése.

A három módszert ugyanarra a 600 db képre tanítottam. 100 emberről válogattam ki (személyenként) 2 db frontális-, 2 db bal-, és 2 db jobb profil orientációjú képet. A tesztadatbázist hasonló módon állítottam össze, mint a tanítóadatbázist, azonban olyan személyekről szerepeltettem benne képeket, akik a tanítóadatbázisban nem szerepeltek. A tesztadatbázis felépítését az 5. táblázat illusztrálja.

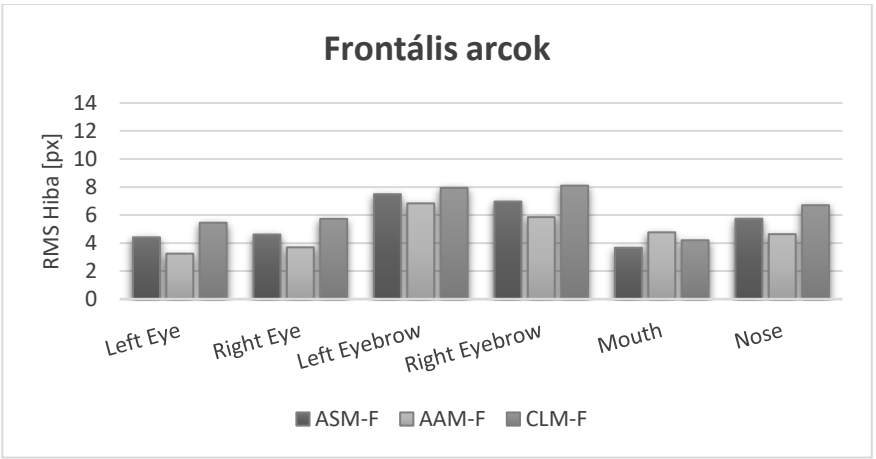
	Képek darabszáma	Személyek száma
Frontális	100	50
Frontális és bal profil átmenet	100	50
Frontális és jobb profil átmenet	100	50

5. táblázat. A tesztadatbázis elrendezése.

A három módszert ugyanarra 300 db képre teszteltem: 50 emberről válogattam ki (személyenként) 2 db frontális-, 2 db bal-, és 2 db jobb profil orientációjú képet. A tesztelés során a modellillesztés pontosságát akartam mérni. Erre vonatkozóan gyakran használt mérőszám a *root-mean-square* (röviden RMS) hiba. Az RMS hiba azt mutatja meg, hogy az annotálás során rögzített határpontok, valamint a modell illesztése során kapott határpontok között mekkora az eltérés. A teszteseteket úgy definiáltam, hogy visszacsatolást kapjak a RMS hiba és a fej irányultságának kapcsolatára. E célból kifolyólag a tesztadatbázist két részre osztottam: egyre mely csak frontális orientációjú arcokat tartalmazott, és egy másikra mely csak 45°-ban oldalra néző arcokat tartalmazott. A továbbiakban ez utóbbit *frontális-profil átmenetnek* fogom hivatkozni.

3.5.1. Eredmények frontális arcokra

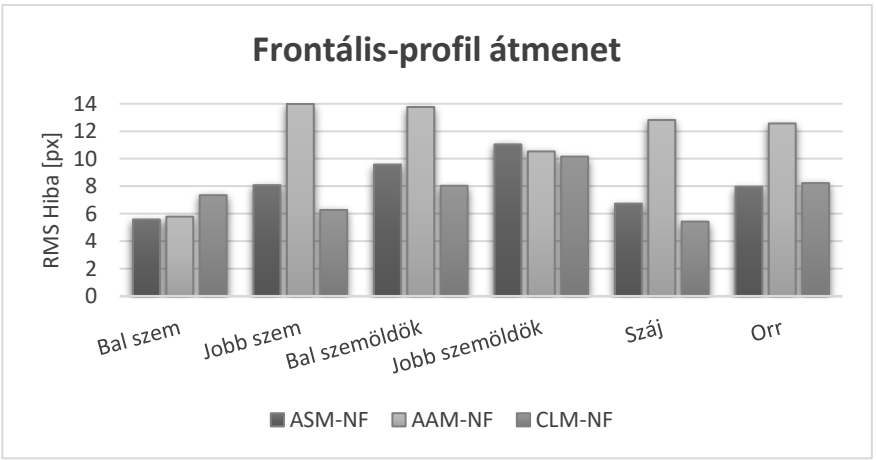
A kiértékeléshez aggregáltam az adatbázisokra kapott eredményeket aszerint, hogy az alakzatmodell pontjai mely arcrész modellezésében vesznek szerepet. Összesítettem a bal szemet, jobb szemet, bal szemöldököt, jobb szemöldököt, orrot és szájat alkotó határpontok RMS hibáit, és az így kapott eredményeket vetettem össze egymással. Frontális esetben (lásd 23. ábra) nem tapasztaltam túl nagy eltérést a három módszer jósága között, igaz megjegyzem, hogy az esetek többségében az AAM volt a legpontosabb – igaz csak 1-2 százalékkal. Az adatokból tisztán látszik, hogy a szemöldökök hibája közel kétszer akkora, mint az összes többi arci jellemzőé. Ezt a további kutatások során mindenképpen figyelembe kell vennünk, hiszen a 8 pixel környéki hiba már meghatározó lehet pl. egy gesztus felismerésénél.



23. ábra. Tesztelés csak frontális orientációjú arcokra.

3.5.2. Eredmények frontális-profil átmenetre

Rosszabb eredményekkel szembesültem akkor, amikor nem tisztán frontális orientációjú arcokra teszteltem (lásd 24. ábra). Az AAM lényegében megbukott a frontális-profil átmenetre végzett tesztek során: közel 2,5-szer rosszabb eredményeket produkált, mint a frontális esetben. Ezzel szemben, a másik két módszernél másfélszeres romlás figyelhető meg (lásd 6. táblázat).



24. ábra. Tesztelés csak frontális-profil átmenet orientációjú arcokra.

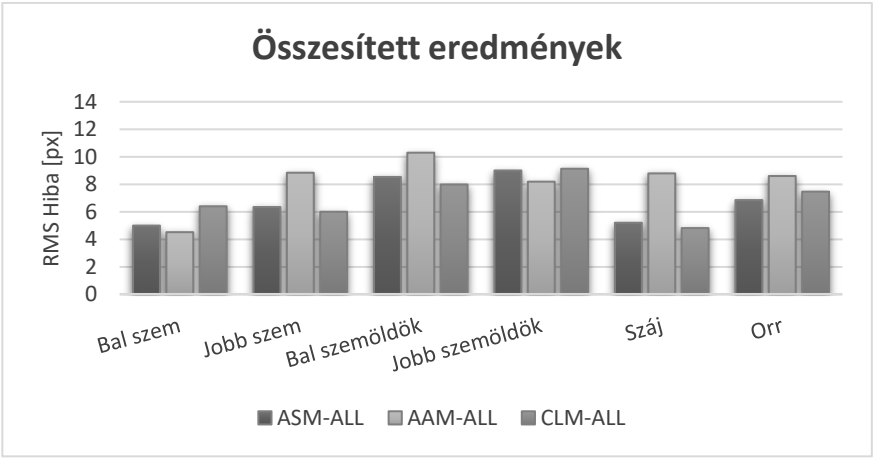
Az AAM kapcsán lehetett volna újabb modelltanítással, esetleg másik implementációval kísérletezni és javítani az eredményeket, de a másik két implementáció hatékonysága miatt ezeket az ötleteket elvetettem. Az ASM és CLM esetén hasonló tendencia figyelhető meg, mint frontális esetben: a

szemöldökök hibája közel kétszer akkora, mint az összes többi arci jellemző hibája. További érdekesség, hogy a jobboldali arcpontokra, átlag 15%-kal rosszabb az illesztés, mint a baloldaliakra. Ennek egy lehetséges magyarázata az, hogy a frontális-profil átmenet tesztadatbázis összeállítása valamelyest szubjektív volt az orientációk tekintetében. Elképzelhető tehát, hogy a „frontális és jobb profil átmenet" kategóriához tartozó képeken az arc pár fokkal jobban el van forgatva, mint a másik esetben. A következő fejezetben bővebben írok róla, de a teljes profil orientációhoz közeledve exponenciálisan nő az illesztés hibája, ami magyarázhatja a 15%-os különbséget.

	Frontális [px]	**Frontális-profil Átmenet [px]**	**Összesített eredmények [px]**
ASM	5,02	7,66	6,34
AAM	4,55	11,51	8,03
CLM	5,88	7,13	6,50

6. táblázat. RMS hibák összegzése módszerenként és adatbázisonként. A kisebb hiba jobban közelíti az annotáció során kapott valós értéket. Pl. a CLM, a frontális arcokat tartalmazó tesztadatbázisra átlagosan 5,88 pixel RMS hibával dolgozott.

A frontális-profil átmenetek esetén, az esetek közel 80%-ban a CLM jobb eredményt ad, mint az ASM, de az eltérés nem szignifikáns, pár pixelen belüli. Az összesített eredmények (25. ábra) alapján is kiegyenlített a két módszer pontossága.



25. ábra. Összesített eredmények, mindkét adatbázisra.

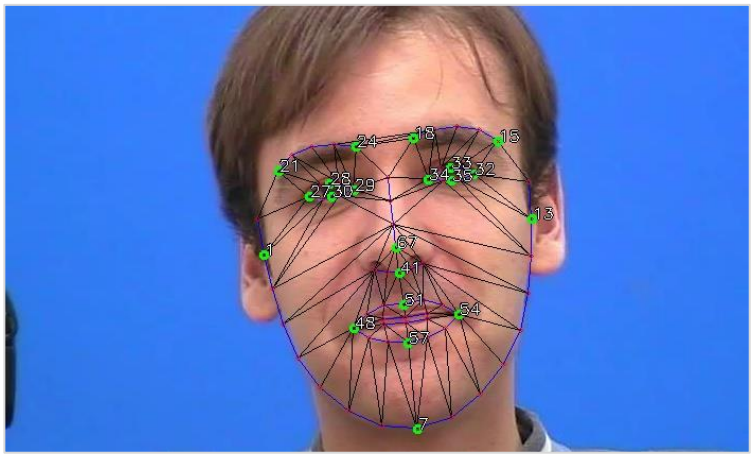
A futásidő (7. táblázat) tekintetében sem tapasztaltam jelentős különbséget a módszerek között, így további teszteket futtattam le.

	Frontális [ms]	Frontális-Profil Átmenet [ms]	Összesített eredmények [ms]
ASM	68	63	65
AAM	398	350	374
CLM	96	73	84

7. táblázat. A futási idők alakulása módszerenként és adatbázisonként. Pl. az ASM, orientációtól függetlenül átlagosan 65 ms alatt végezte el a modellillesztést egy arcra.

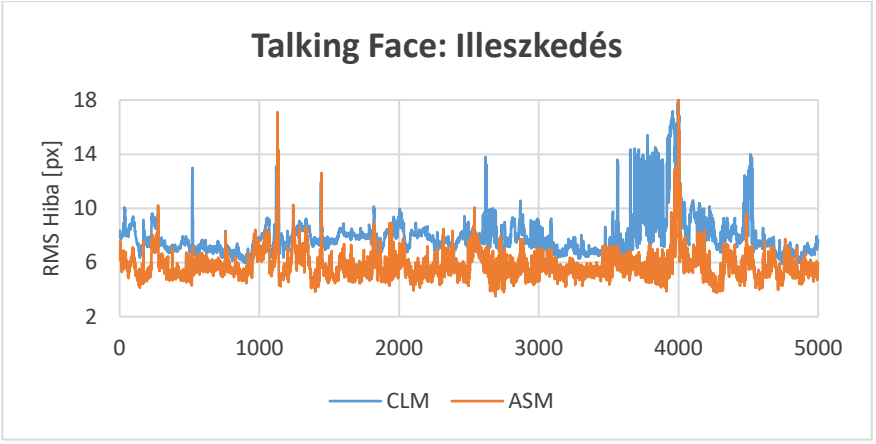
3.5.3. Eredmények videó folyamra

Az előzőekben ismertetett adatbázisokon történő tesztelést nem tartottam elég kimerítőnek, így elvégeztem egy újabb tesztet egy előzetesen annotált spontán beszédhez tartozó videó felvételre. A felvételt az INRIA FGnet kutatócsoport rögzítette és kutatási célokra szabadon felhasználhatóvá tette [46]. A felvétel során egy felhasználó arcát rögzítették közel 3,5 percen keresztül (5000 darab képkocka). A felvételen annotálták az arc karakterisztikus pontjait az XM2VTS adatbázis formátumának megfelelően [47]. Ez a formátum ugyan ugyanúgy 68 db arcpontot tartalmaz, mint az általam is használt, de a pontok elrendezése eltérő volt. Bizonyos pontok azonban megfeleltethetők voltak egymásnak: 4-4 db pont a két szemen, 2-2 db a két szemöldökön, kettő az orron, négy a szájon, illetve három az arc körvonalán (lásd 26. ábra).



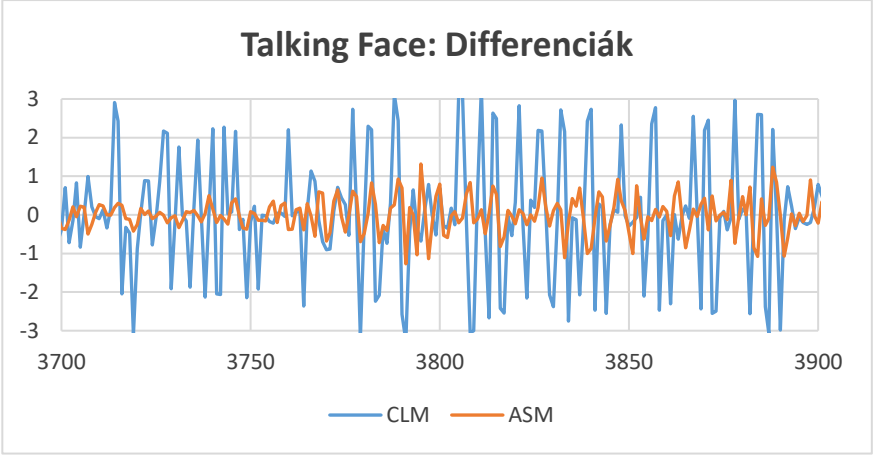
26. ábra. Pontmegfeleltetések az XM2VTS és CMU Multi-PIE annotációs formátuma között. Zöld körrel jelöltem azt a 21 db pontot, melyekre az RMS hibákat meghatároztam (az annotáció és az illesztés között). A képre az adatbázishoz mellékelt archálót rajzoltam ki. A háromszögelésnek itt is csak a megjelenítésben van szerepe.

Az AAM-et előzetesen elvetettem, így csak az ASM-et és CLM-et futtattam le az 5000 képkockára. A hibamérés pontosságának érdekében, a bemeneti videófolyamon nem követtem az arcokat, hanem minden képkockára lefuttattam az arcdetektort. Interpoláltam továbbá az RMS-t azokra a képkockákra, ahol az arcdetektor nem talált arcot, vagy ahol a találat nem volt legalább 70%-os átfedésben az annotációs adatok által kijelölt arccal. Megjegyzem, hogy kevesebb mint 100 db képkockára kellett interpolálni az RMS-t, az összes többire közvetlenül számoltam a hiba értékét az annotációs-, és a modellillesztés által meghatározott 21-21 db pont között. A 27. ábra mutatja a 21 db pont átlagos RMS hibáját a két módszerre.



27. ábra. Az átlagos RMS hibája a két módszernek. A vízszintes tengely a képkocka sorszáma, a függőleges pedig a 21 db pontra számított RMS hiba.

A grafikonból kitűnik, hogy az ASM hibája szinte végig alacsonyabb, mint a CLM-é. Az ASM átlag 2,24 pixellel pontosabb illesztést végez, mint a CLM, ráadásul a videó felvétel teljes hossza alatt. A különbség nem teljesen szignifikáns, azonban a 3700-4000 sorszámú képkockák között mégis kicsúcsosodik: itt a CLM esetén az átlagos RMS hiba intenzíven ingadozik a $[7; 14]$ pixel intervallumban. Ebben az időintervallumban a felhasználó erőteljesebben elneveti magát és ezt a gesztust a CLM kevésbé tudja sikeresen lekövetni, mint az ASM. Maga a jelenség jobban látszik, ha kinagyítjuk a kérdéses szakaszhoz tartozó RMS hiba deriváltját (28. ábra). Látható, hogy CLM esetén az RMS deriváltja viszonylag magas frekvenciával ingadozik a $[-3; 3]$ intervallumban.



28. ábra. A rossz CLM illesztés szakaszához tartozó RMS deriváltja

A 28. ábrán látható átlagos RMS hiba tulajdonképpen megkétszereződik az egymást követő képkockák között, ami komoly hatással lehet a feldolgozás további lépéseiben. Ugyan az ASM-re ez a fajta kilengés nem jellemző[1], de az eredmények alapján úgy döntöttem, hogy a rendszer magasabb szintű komponenseiben indokolt lehet egy zajszűrő eljárás beépítése.

A futási idő szintén az ASM-nek kedvez, a 8. táblázatból kiolvasható, hogy közel kétszer gyorsabb az ASM, mint a CLM. Bár mind az ASM, mind a CLM ebben tekintetben tovább optimalizálható futásidő tekintetében, úgy találtam, hogy a további számításokhoz elégséges lesz az ASM pontossága és gyorsasága. Az általam tanított és a végfelhasználásra szánt ASM alakzatmodelljében 10 db különböző paramétert használtam fel (azaz $b \in \mathbb{R}^{10}$ és $\Phi \in \mathbb{R}^{132\times10}$). Emlékezzünk vissza, hogy a b paraméter változtatásával vagyunk képesek az eredeti modellhez hasonló, de egy kicsit másabb elrendezésű új alakzatmodelleket létrehozni. A kovariancia mátrix 10 legnagyobb sajátértékéhez tartozó sajátvektora szolgáltatta a variancia 85%-át, ezek rendre a varianca 35%, 26%, 8%, 5%, 3%, 3%, 2%, 1%, 1% és 1%-ért voltak felelősek.

	Átlagos futási idő [ms]
ASM	31
CLM	60

8. táblázat. A modellillesztés átlagos futási ideje a tesztfelvétel egy képkockájára.

[1] Az esetek *2,18%*-ában nagyobb a derivált abszolút értéke, mint 1 pixel.

A gyakorlati megvalósításban az ASM modellillesztés futásidejét egy egyszerű módszerrel tovább optimalizáltam. Jelölje s_k azt a statisztikai modellt, melyet egy bemenő képszekvencia k-adik elemére illesztettünk. Ekkor, a képszekvencia $(k+1)$-edik elemére célszerű az s_k-t kifeszíteni az arc felé az $\bar{s}$ átlagalakzat helyett. Tesszük mindezt azért, mert feltételezhető, hogy az s_k konfiguráció jobban leírja az arcot a soron következő képkockán, így az algoritmus gyorsabban konvergál abból kiindulva.

Összegezvén a fejezetben leírtakat, az arc lokalizálását követően az *arc belső modellezésével*, azaz a *karakterisztikus pontok kinyerésével* foglalkoztam. Három népszerű statisztikai alak-, és textúra modellező eljárást ismertettem: ASM, AAM és CLM. A fejezet kimenete és eredményterméke egy 2-D archáló lett, mely határpontjai az arcnak egyes meghatározó pontjait reprezentálták, mint pl. szemek-, vagy szemöldökök sarkainak pontjai, szemöldökök és száj körvonalának bizonyos pontjai. Átfogó tesztesetek során értékeltem ki a módszereket a gyakorlatban. Céljaim itt is hasonlóak voltak, mint az arc lokalizálása során: *magas pontosság* mellett akartam kinyerni a karakterisztikus pontokat, *valós időben*, *változatos fejtartások* és *arckifejezések* mellett. Az átfogó tesztek során sikeresen előállítottam egy konfigurációt, mely az eredményeim alapján kiszolgálja az előbb megfogalmazott elvárásokat és magát a rendszert a gyakorlatban.

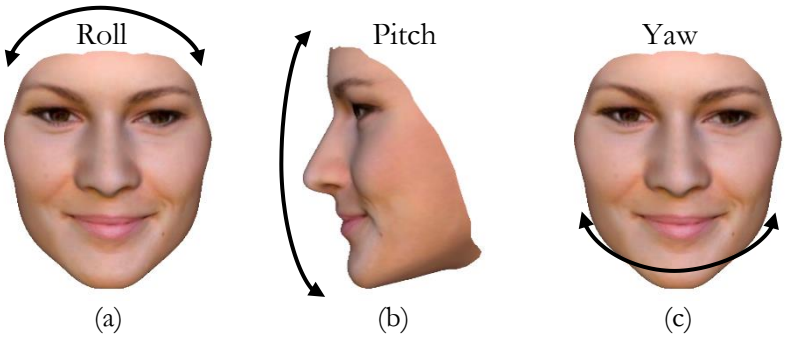
4. Arc elhelyezése a térben

A digitális képfeldolgozás és gépi látás tudományterületén belül *pose estimation*-nek hivatkozzuk azt a folyamatot, amikor egy bemeneti képen egy objektum térbeli pozícióját és orientációját kívánjuk meghatározni. A *pose* angol szó, magyar fordításban hasonló jelentéssel bír, mint a pozíció, ezért a továbbiakban *fejtartásként* fogok hivatkozni a fej térbeli pozíciójának és orientációjának az együttesére. A *fejpozíció* ennek megfelelően a fej tömegközéppontjának koordinátákkal megadott térbeli helyzetét jelenti a továbbiakban. A fejtartás ismerete kulcsfontosságú az emberi viselkedés tanulmányozása során. Az ember-gép kommunikációban az emberi fejtartás ismerete azért lényeges a számítógép számára, mivel a gép ezáltal kaphat képet a kommunikációs partnere figyelmének középpontjáról, kommunikációra való hajlandóságáról, illetve annak egyéb belső állapotáról, viselkedésmódjáról.

Általánosságban kijelenthető, hogy a fejtartás gépi meghatározása önmagában is hasznos lehet, de általa további magasszintű vizsgálatok, kutatások elvégzésére nyílik lehetőségünk. PhD tanulmányaim során, az eddig ismertetett eredményeimre támaszkodva létrehoztam egy rendszert, mely képfeldolgozási algoritmusokat felhasználva határozza meg az emberi fél fejtartását 2-D képeken. A feladat magában foglalta a fejtartás reprezentációját és kiszámítását; a szükséges teszt adatbázisok összeállítását; az eredmények méréséhez szükséges tesztkörnyezet kialakítását; tesztesetek meghatározását. Ezekről fogok írni ebben a fejezetben.

Első körben nézzük a fejtartás jellemzőit. A kinematikában, a mechanizmust úgy szokták definiálni, mint több egymással mozgásbeli kényszerkapcsolatban álló merev testekből felépített mozgó szerkezetet. Bizonyos feltételek mentén az emberi fejet is tekinthetjük egy mechanizmus kapcsolódó merev testének. A mechanizmus két-két szomszédos tagja egy mechanikai párt képez. Egy tagnak a másikhoz képest, ha nincs kényszerítve, *hat szabadságfoka* van (*6 degrees of freedom*, röviden *6DoF*), vagyis hatféle elemi mozgást képes végezni, ezek:

- Három egymásra merőleges irányba történő elmozdulás és,
- Három egymásra merőleges tengely körüli elfordulás.



29. ábra. A fejpozíció 3 szabadsági foka. (a) *Roll*: θ_z szöggel való forgatás a hosszirányú z tengely körül. (b) *Pitch*: θ_x szöggel való forgatás az oldalirányú x tengely körül. (c) *Yaw*: θ_y szöggel való forgatás a függőleges y tengely körül.

Esetünkben a fej szabadsági fokát hatnak fogjuk tekinteni, ami az Olvasó részére ellentmondásosnak tűnhet, hiszen a hatos érték azt jelenti, hogy a fej és a kinematikai párja (felsőtest) között egyáltalán nincs semmilyen kapcsolat (azaz a két tag egymáshoz képest szabadon elmozdulhat), ami nyilván nem igaz, de a modell alkotáshoz szükségünk van erre a kezdeti feltételre. Így a fejtartást egy 6DoF modell segítségével fogom megadni a 3-D euklideszi térben. Ehhez két darab 3-komponensű vektorra lesz szükség: egyrészt a *fejpozíciót leíró vektorra*, mely megadja, hogy az x, y és z tengelyek mentén hol helyezkedik el a világ koordináta-rendszerben a fej; illetve a *fej orientációját leíró vektorra*, amely megadja, hogy a világ koordináta-rendszerben milyen irányban áll a fej. Az orientációnál maradva, egy merev test orientációjának a leírására a 3-D térben több lehetőség is létezik, az egyik legegyszerűbb az *Euler-szögek módszere*. Az Euler-szögek segítségével, egy merev test helyzetét három darab skalármennyiség segítségével lehet leírni az euklideszi térben rögzített 3-D koordináta-rendszerhez képest. Lényegében arról van szó, hogy egy rotációs mátrix felbontható három elemi forgatási mátrix szorzatára. Az Euler-szögek egy tetszőleges koordináta-rendszer térbeli helyzetét jelölik egy rögzített koordináta-rendszerhez képest, attól függően, hogy a tengelyek mentén milyen sorrendben történik a forgatás különböző konvenciókról beszélhetünk. E konvenciók közül néhányat *roll-pitch-yaw* (röviden *RPY*) szögeknek hivatkozza a szakirodalom, pl. a z-x-y irányú forgatásokat is (lásd 29. ábra).

4.1. Irodalmi áttekintés

A fejtartás meghatározás az elmúlt évek technológiai fejlődésének köszönhetően egy intenzíven kutatott területté nőtte ki magát a HCI-n belül. Egy sok részletre kiterjedő áttekintést olvashatunk a témáról a következő művekben: [48, 49]. Helyhiány miatt nem kívánom ennyire részletesen ismertetni a témát, csak egy alapvető osztályozást adok meg az irodalmi módszerekre. Jelen fejezetben csak a 2-D képeken megjelenített fejpozíció meghatározással kívánok foglalkozni, melynek módszereit két nagy csoportba lehet sorolni: *megjelenés alapú* módszerek és *jellemző alapú* módszerek.

A *megjelenés alapú módszerek* az arc, illetve az arci jellemzők megjelenéséből határozzák meg a fejtartást. Ezek közül a legáltalánosabb módszerek diszjunkt halmazokra bontják a felismerni kívánt fejtartásokat és az egyes halmazokba olyan képeket sorolnak, melyek az adott halmaz által reprezentált fejtartásnak megfelelnek. Innentől pedig a felismerési problémát egy osztályozási problémaként értelmezik, azaz osztályozókat tanítanak az egyes halmazokra. Sok módszer létezik az irodalomban, melyek a magas dimenziós képtérben értelmezett probléma helyett egy alacsonyabb dimenziós bázisban próbálják felírni a problémát. Találunk módszereket, melyek PCA-t használnak [50], vagy egyéb nem-lineáris dimenzió csökkentő eljárásokat pl. [51, 52]. Ugyanebbe a kategóriába soroljuk azokat a módszereket, melyek többosztályos osztályozókat használnak, mint pl. tartóvektor gépeket valamilyen képi jellemzővel, pl. [53, 54]. Az osztályozási probléma kapcsán találunk megoldást a problémára neurális hálók alkalmazása mellett is [55]. Ugyanebbe a kategóriába tartoznak azok az eljárások, melyek több egy-egy konkrét fejtartásnak megfelelő arcdetektálás együtteseként értelmezik a fejtartás felismerését, pl. [56, 57].

Ezzel szemben a *jellemző alapú módszerek* az arc és az arci jellemzők geometriájára építenek, ugyanis az alakzatmodell határpontjaiból ugyanúgy lehetőségünk nyílik a fejtartás felismerésére [58], mint a megjelenés alapú módszerek esetén. Csak úgy, mint a modell alapú módszerek esetén, itt is találunk olyan eljárásokat, melyek többosztályos osztályozókat használnak a fejtartás felismerésére [59] – annyi különbséggel, hogy a bemenetet itt az egyes arci jellemzők koordinátái jelentik. Léteznek az irodalomban hasonló elven működő és kiemelkedő teljesítményt produkáló diszkriminatív eljárások is a fejtartás meghatározására [60]. Szintén népszerűek azok az eljárások, amelyek különféle deformálható modellek alapján határozzák meg a fejtartást. Találunk

ezek között valószínűségi döntéshozó rendszereket, pl. amelyek adaptív nézet-alapú AAM-ek segítségével határozzák meg a fejtartást [61, 62]. Ugyanakkor számos eljárás létezik, melyek 2-D és 3-D pontmegfeleltetések alapján próbálják megbecsülni a fejtartást [61]. Az én választásom is egy jellemző alapú módszerre esett, mely egy előre definiált 3-D fejmodell-, és a kameraképen detektált arci jellemzők 2-D pontjai közötti megfeleltetések alapján közelíti a fejtartást, egy iteratív algoritmus segítségével.

4.2. Kamera modell

Mivel 3-D objektumok és azok 2-D vetülete alapján kívánom meghatározni a fejtartást, így szükség lesz egy kamera modellre [63]. A kamera modell segítségével megadható, hogy a világ 3-D teréből, hogy tudunk a kamera képterébe áttérni. Ideálisan ezt úgy képzelhetjük el, mint egy projektív leképzést a 3-D térből a 2-D térbe. A leképezés megadható homogén koordinátás transzformációs mátrixokkal, de a gyakorlatban több okból kifolyólag máshogy szoktak eljárni:

- A nagy látószögű lencsék torzítják a képet: a valóságos egyenesek a képen görbék lehetnek, azaz a leképzés nem projektív. Tehát a lencsetorzítás figyelembe vételéhez nem használhatunk csak egy projektív leképzést, figyelembe kell venni a nemlineáris torzítást is.
- A kamera modellt a gyakorlatban két jól elkülönülő részre szokás választani. Egyrészt beszélhetünk a modell *külső paramétereiről* (extrinsic parameters), melyek kamera térbeli helyét és irányát írják le. Illetve beszélhetünk *belső paraméterekről* (intrinsic parameters), melyek nem változnak meg, ha a kamerát fizikailag elmozgatjuk[2]. A külső-, és belső paramétereket másképp határozzuk meg, és előfordulhat, hogy csak az egyiket szeretnénk megváltoztatni (ezért szokás külön-külön mátrixban reprezentálni).
- A kameramodellek általában nem engednek meg tetszőleges projektív transzformációt, hanem megszorítják azok körét. Pl. kikötik, hogy a képen látható függőleges és vízszintes egyenesek a szenzoron merőlegesek; vagy, hogy a képen minden irányban felmért azonos távolságok a szenzoron azonos távolságoknak felelnek meg.

[2] Megváltozhatnak, ha pl. a lencsét elcsavarjuk, de ha csak önmagában a kamera vázát mozgatjuk, akkor nem.

A kameramodell három részből áll: *külső paraméterek*, *nemlineáris torzítás*, és a *lineáris belső paraméterek*. Ezt a sorrendiséget kell követni az egyes transzformációk során, hogy egy térbeli pontnak megkapjuk a helyzetét a képen. A modell úgy van megadva, hogy a nemlineáris torzítás hatását alkalmazzuk előbb a pontokra és csak utána következnek a lineáris hatások. Ennek az az előnye, hogy ha a képet eltérő felbontásban használjuk, mint korábban, vagy pedig a kép egy részét levágjuk, akkor csak a belső paramétereken kell módosítani, a nemlineáris torzítás paraméterein nem. Így ugyanazt a kalibrációt könnyen módosíthatjuk úgy, hogy kisebb felbontású átméretezett képeken számoljunk vele, mint amiből a kalibrációt készítettük. A leképezés paramétereinek a meghatározását a kamera kalibrációjának nevezzük [64].

4.2.1. Külső kalibráció

A *külső kalibráció* a teret olyan helyzetbe transzformálja, hogy az origó a kamerába kerüljön. A külső kalibráció segítségével tudunk átjárást biztosítani a világ-, és kamera koordináta-rendszerek között. Ismeretes, hogy a kamera a z tengely (pozitív) irányába néz; a kép sorai az x tengellyel, az oszlopai pedig az y tengellyel párhuzamosan állnak. Tehát a külső kalibráció megadható a következő térbeli eltolás és forgatás segítségével: $[R \mid t]$, ahol t egy tetszőleges 3×1 -es eltolási vektor, R pedig egy 3×3 -as ortogonális forgatási mátrix.

4.2.2. Belső kalibráció

A kamera koordináta-rendszerbeli pontokat az origóból középpontosan le kell vetíteni a szenzor síkjára. A *belső kalibráció* lényegében azt mondja meg, hogy a képsíkon hogyan transzformáljuk a pontot annak érdekében, hogy megkapjuk annak a pixel koordinátait. A belső kalibráció egy nagyításból és egy eltolásból áll. A nagyítás általában külön vízszintes és függőleges nagyítási arányokból áll. A paramétereket az alábbi C kameramátrix írja le:

$$C = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.1}$$

ahol a c_x és c_y eltolás megadja, hogy a képen hova esik az optikai középpont. Az f_x és f_y nagyítások pedig azt fejezik ki, hogy ha az optikai középpont

környékén egy pixellel arrébb megyünk a képen, akkor ez mekkora látószögnek felel meg a térben. A leképezés az alábbi formában adható meg:

$$[x \quad y \quad w]^T = C \cdot (t + R \cdot [X_W \;\; Y_W \;\; Z_W]^T), \tag{4.2}$$

ahol $[X_W \;\; Y_W \;\; Z_W]^T$ térbeli koordináták, $[x \quad y \quad w]^T$ pedig homogén síkbeli koordináták.

4.2.3. Nemlineáris torzítás

A *nemlineáris torzítás* kezelésére azért van szükség, mert problémába ütközünk, ha egy nagyobb[3] látószögű képet projektív módon képezünk le. Mivel a szenzor mindenütt egyenletes felbontású így az ugyanakkora szögben látott tárgyak ilyenkor a kép szélén túl nagynak, a kép közepén pedig túl kicsinek látszanának. A nemlineáris torzítás modellje abból a feltevésből indul ki, hogy az optika (így a torzítás is) szimmetrikus az egyik tengelye körüli forgatásra. A szakirodalomban leggyakrabban használt modellben nyolc valós paraméter van, de ebből nem kell mindet használni. A legegyszerűbb esetben csak az első két paramétert (K_1 és K_2) használjuk. Ezek az optikai középponttól való távolságot módosítják úgy, hogy az irányt helybenhagyják. Ekkor egy új (x_d, y_d) radiálisan torzított képpont koordinátája a régi (x_u, y_u) torzítatlan koordinátának egy polinom függvénye lesz:

$$x_d = x_u(1 + K_1 r^2 + K_2 r^4 + \cdots), \tag{4.3}$$

$$y_d = y_u(1 + K_1 r^2 + K_2 r^4 + \cdots)\,. \tag{4.4}$$

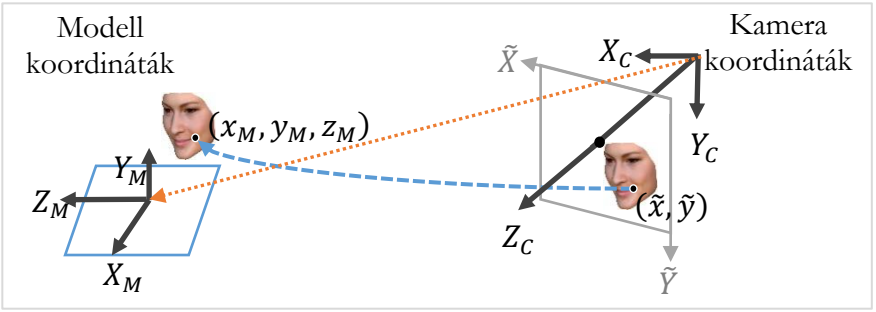
Ez egy páratlan függvény (az origóban sima a transzformáció) és az elsőfokú együtthatója rögzítve van, mert az a kép nagyítását adná meg. A radiális torzítás mellett szokás tangenciális torzításról is beszélni[4], melyek paramétereit P_n-nel szokás jelölni. Ezekre azért van szükség, mert ugyan a torzítás forgás-szimmetrikus, de ennek a szimmetriatengelye nem feltétlenül merőleges a szenzor síkjára, ha a szenzor a lencséhez képest ferde.

[3] Kis látószögű lencsék esetén elhanyagolható a nemlineáris torzítás mértéke.

[4] A tangenciális torzítás együtthatóit az egyszerűség kedvért nem szerepeltettük a nemlineáris torzítás modelljében.

4.3. Perspective n-Point probléma

A *perspective n-Point probléma* (PnP probléma) definiálása során abból indulunk ki, hogy ismert néhány 2-D koordinátájú pont a kameraképen – ezek lehetnek pl. egy archáló pontjai –, melyek egy hasonló geometriájú 3-D modell kontrollpontjaihoz tartoznak [65]. A PnP probléma megoldása során n darab 2-D és 3-D összetartozó pontpár alapján határozzuk meg a modell pozícióját és orientációját a 3-D térben. A pontmegfeleltetésekre mutat példát a 30. ábra, ahol 3-D modell (x_M, y_M, z_M) pontjának feleltetjük meg a képsík $(\tilde{x}, \tilde{y})$ pontját.



30. ábra. A fejtartás meghatározásához használt kameramodell koordináta-rendszerei, illetve a probléma vizualizációja. A fejtartás meghatározása tulajdonképpen a modell koordináta-rendszer pozíciójának és orientációjának a meghatározása a kamera koordináta-rendszerhez képest.

Tehát a modell és kamera koordináta-rendszerek között tesszük lehetővé átjárást (úgy, mint a külső kalibráció esetén). PnP problémák esetén a legegyszerűbb esetben az $n = 3$, vagy $n = 4$ és ekkor rendre P3P és P4P problémáról beszélünk. P3P probléma esetén maximum 8 darab megoldása lehet a problémának [66]. *P3P probléma* esetén speciálisan egy 3-D háromszög és annak a 2-D vetülete alapján határozzuk meg egy térbeli háromszög pozícióját és orientációját. Ezzel szemben a *P4P probléma* esetén két eset lehetséges:

- Ha mind a 4 darab pont egy síkba esik, akkor létezik egyértelmű megoldása a problémának,
- Ellenben két különböző megoldás is létezhet.

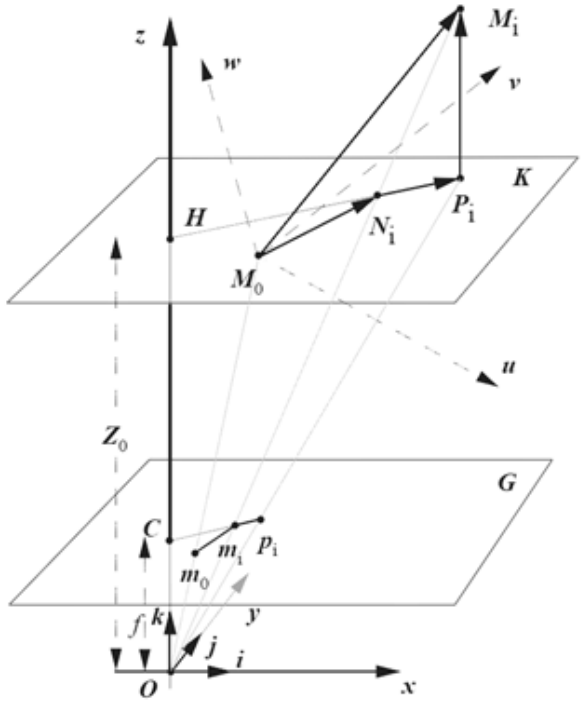
A P4P problémának tehát létezhet egyértelmű megoldása. Ennek megfelelően óriási a kereslet a stabil P4P megoldásokra. A meglévő eljárások két nagy csoportba oszthatók:

- *Modell alapú eljárások*, melyek a kamera, mint leképezési modell közelítésén alapulnak.
- *Geometriai eljárások*, melyek a képsík és az objektum koordináta-rendszere közötti kapcsolatból vezetik le a megoldást.

Az általános probléma megoldó eljárásokra vonatkozóan számos tanulmány látott napvilágot. Pl. az EPnP (*effcient PnP camera pose estimation*) eljárás, mellyel egy nem-iteratív módszer segítségével $O(n)$ idő alatt kaphatjuk meg a probléma egyértelmű megoldását négy, vagy több összetartozó pontpár esetén [67]. Az RPnP (*robust PnP camera pose estimation*), mely az egyik legelső olyan stabil, nem-iteratív eljárás volt, amelyik kevesebb, mint 5 darab pontmegfeleltetés esetén határozta meg az optimális megoldást $O(n)$ idő alatt [68]. Az LHM (*Lu-Hager-Mjolsness*), a PnP probléma egyik legjobb iteratív megoldása az irodalomban, de eléggé instabil a közös síkba eső pontok esetén [69]. Az eljárások többségére igaz, hogy esetenként csak több tucat pont-megfeleltetés esetén érnek el pontos eredményeket. Ez a gyakorlatban nem célravezető, hiszen a probléma jellegéből fakadóan csak nehézkeseken lehet megtalálni a megfeleltetéseket. Léteznek ugyan eljárások, pl. DLS (*direct least-squares*), melyek $n \leq 7$ esetén is hatékonyak, de cserében erőteljesen bekorlátozzák a mozgásteret [70]. A fejtartás meghatározására irányuló választásom a POSIT (*POS with ITerations*) [71] nevet viselő, modell alapú eljárásra esett, mely egyben az egyik legnépszerűbb, nem egy síkba eső P4P probléma megoldása is.

4.4. POSIT algoritmus

A POSIT algoritmussal (minimum) 4 darab 2-D és 3-D összetartozó pontpár alapján határozhatjuk meg a fejtartást. Az algoritmus két részből áll: az első részre POS-ként (*pose from orthography and scaling*) szoktak hivatkozni. Ekkor skálázott ortografikus vetítéssel (*scaled ortographic projection*, *SOP*) közelítjük a perspektív vetítést és határozzuk meg a modell térbeli pozícióját és orientációját. A következő lépésben, egy iteratív algoritmussal javítjuk a vetítést, addig ameddig konvergenciát nem tapasztalunk – a képsíkon detektált 2-D pontok és a 3-D pontok vetületei között.



31. ábra. POSIT koordináta-rendszerei. Az ábra a skálázott ortografikus és perspektív vetítését reprezentálja az M_i és M_0 3-D modellpontoknak.

4.4.1. Skálázott ortografikus vetítés

A SOP lényegében a perspektív vetítést közelíti és akkor célszerű használni, amikor a 3-D modell (esetünkben a fej) mélysége jelentősen kisebb, mint a kamerától vett távolsága. A továbbiakban, jelöljük a 3-D világbeli arcmodell pontjait $M_i = (X_i, Y_i, Z_i)$ -vel, illetve a 2-D képsíkbeli arcmodell pontjait $m_i = (x_i, y_i)$-vel, ahol $i = 1 \dots N$ (lásd 31. ábra). A SOP tehát azt feltételezi, hogy a 3-D modellpontok Z_i koordinátái egy szűk intervallumból veszik fel az értékeiket. Így a perspektíva f/Z_i skála tényezőjét az $s = f/Z_o$ állandóval közelítjük az összes többi modellpontra, ahol Z_o egy referencia pontként kiválasztott M_0 pont mélysége; f pedig a G képsík távolsága a vetítés középpontjától, azaz a fókusztávolság. A perspektív vetítés alapján egy $M_i = (X_i, Y_i, Z_i)$ pont képe a G képsíkon az $m_i = (x_i, y_i)$ pont, ahol:

$$x_i = f\frac{X_i}{Z_i},\ y_i = f\frac{Y_i}{Z_i}. \tag{4.5}$$

Másrészt SOP esetén ugyanennek a M_i pontnak a képe a G képsíkon pedig a $p_i = (x_i', y_i')$ pont, ahol:

$$x_i' = f\frac{X_i}{Z_0},\ y_i' = f\frac{Y_i}{Z_0}. \tag{4.6}$$

Vegyük észre, hogy az M_0 referenciapont perspektív és SOP vetülete egybeesik. SOP esetén tehát meghatározunk egy K síkot, mely párhuzamos a G képsíkkal és Z_0 távolságra van a vetítés középpontjától O-tól, majd:

- Merőlegesen vetítjük az összes M_i modell pontot a K síkra, ezek lesznek a P_i pontok,
- A K síkra merőlegesen vetített pontokat perspektív vetítéssel a G képsíkra vetítjük, melynek eredményeként a p_i pontokat kapjuk[5].

4.4.2. Fejtartás közelítése

A SOP segítségével tehát a valós fejtartást közelítjük. A kamera koordináta-rendszer $\vec{\imath}$, $\vec{\jmath}$ és $\vec{k}$ bázisvektorait írjuk fel a 3-D modell koordináta-rendszerében, mely az $\vec{u}$, $\vec{v}$ és $\vec{w}$ bázisokkal rendelkezik. A kamera-, és modell koordináta-rendszer közötti átjárás a kameramodellnél ismertetett térbeli eltolás és forgatás segítségével lehetséges: $[R \mid t]$. A forgatási mátrix az alábbi alakban adható meg:

$$R = \begin{bmatrix} i_u & i_v & i_w \\ j_u & j_v & j_w \\ k_u & k_v & k_w \end{bmatrix}, \tag{4.7}$$

ahol i_u kamera koordináta-rendszer $\vec{\imath}$ bázisvektorának u koordinátáját jelenti a modell koordináta-rendszerében[6]. A forgatási mátrix felírásához elegendő a kamera koordináta-rendszer $\vec{\imath}$ és $\vec{\jmath}$ bázisvektorait transzformálni a modell koordináta-rendszerbe, hiszen a harmadik vektort a másik kettő vektoriális szorzataként kapjuk. Az eltolási vektor értelemszerűen a kamera koordináta-rendszer origójából mutat a modell koordináta-rendszer origójába: $t = \overrightarrow{OM_0}$. Az eltolási vektor megegyezik $s^{-1}\overrightarrow{Om_0}$ vektorral, ahol s a SOP skálázási

[5] Minél kisebb intervallumból veszik fel a Z_i koordináták az értékeiket a K és G sík távolságához képest, annál jobban közelíti a SOP a perspektív vetítést.

[6] Így az R forgatási mátrix feladata az, hogy közvetlen átjárást biztosítson modell koordináta-rendszerből a kamera koordináta-rendszerbe. A fordított irány az R invertálásával érhető el – jelen esetben $R^{-1} = R^T$.

faktora. Tehát a t vektor felírásához az M_0 referenciapont Z_o koordinátájának a meghatározása szükséges. Mindent összegezve, a fejtartás kiszámításához az $\vec{\imath}$ és $\vec{\jmath}$ vektorok felírása a modell koordináta-rendszerében, valamint a Z_o koordináta meghatározása szükséges. Az egyes M_i modellpontok SOP vetülete a G képsíkon felírható az alábbi formában is (4.6. egyenlet alapján):

$$x_i' = x_0 + s \cdot (X_i - X_0),\ y_i' = y_0 + s \cdot (Y_i - Y_0). \tag{4.8}$$

A perspektív vetítéshez tartozó 4.5. egyenlet alapján felírható az alábbi összefüggés az ismert és ismeretlen paraméterek között:

$$\overrightarrow{M_0M_i} \cdot \frac{f}{Z_0} \cdot \vec{\imath} = x_i \cdot (1 + \varepsilon_i) - x_0, \tag{4.9}$$

$$\overrightarrow{M_0M_i} \cdot \frac{f}{Z_0} \cdot \vec{\jmath} = y_i \cdot (1 + \varepsilon_i) - y_0, \tag{4.10}$$

ahol $\vec{\imath}$ és $\vec{\jmath}$ modell koordináta-rendszerben felírandó ismeretlen vektorok és a Z_o szintén ismeretlen. A fenti két egyenlet minden más tagja ismert, így $\overrightarrow{M_0M_i}$ a modell koordináta-rendszer vektora, valamint az m_0 és m_i képpontok koordinátái is. Az ε_i tag pedig az alábbi formában definiálható:

$$\varepsilon_i = \frac{1}{Z_0} \cdot \overrightarrow{M_0M_i} \cdot \vec{k}, \tag{4.11}$$

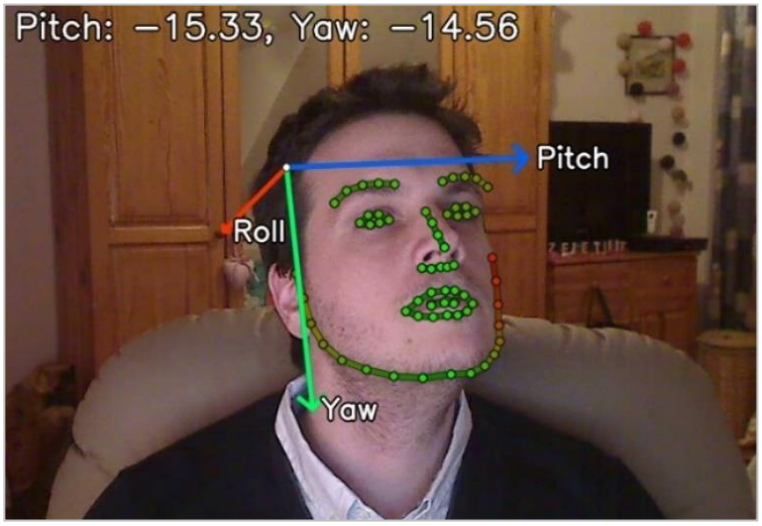
ahol $\vec{k} = \vec{\imath} \times \vec{\jmath}$, ahogy már az előzőekben is említettük. A továbbiakhoz megmutatható, hogy az 4.9. és 4.10. egyenletek jobb oldalán lévő $x_i \cdot (1 + \varepsilon_i)$ és $y_i \cdot (1 + \varepsilon_i)$ mennyiségek valójában p_i pont – azaz a 3-D M_i modellpont SOP vetületének x_i' és y_i' koordinátái. Továbbá az 4.9. és 4.10. egyenletek átírhatók az alábbi alakra:

$$\overrightarrow{M_0M_i} \cdot I = x_i \cdot (1 + \varepsilon_i) - x_0, \tag{4.12}$$

$$\overrightarrow{M_0M_i} \cdot J = y_i \cdot (1 + \varepsilon_i) - y_0, \tag{4.13}$$

ahol az ε_i-nek konstans értékeket adva egy lineáris egyenletrendszert kapunk, ahol ismeretlenek az $I = f/Z_0 \cdot \vec{\imath}$ és $J = f/Z_0 \cdot \vec{\jmath}$ vektorok lesznek. Modellpontonként két egyenletet kapunk és összesen hat ismeretlenünk van, így a lineáris egyenletrendszer megoldásához minimum négy pont-megfeleltetésre van szükség. Az egyenletrendszer megoldásával, az $\vec{\imath}$ és $\vec{\jmath}$ vektorok az I és J normalizálásával kaphatók meg; Z_0 pedig az I, vagy a J

vektor euklideszi normájával egyenlő. A szerzők ezt az algoritmust nevezték el POS-nak. Mivel kezdetben nem ismerjük az ε_i pontos értékét, így a POS által meghatározott fejtartás csak közelíti a képen látható tényleges fejtartást. Azonban a POS során kiszámolt $\vec{\imath}$ és $\vec{\jmath}$ vektorok ismeretében jobb becslés adható ε_i tényleges értékére és ezáltal jobb fejtartás határozható meg egy újabb iteráció során. A szerzők az ε_i iteratív frissítését POSIT-nak nevezték el. Az első iterációban az $\varepsilon_i = 0$, ami megfelel annak, hogy a 3-D M_i modellpontnak az m_i perspektív-, és p_i SOP vetülete egybeesik. A PnP probléma iteratív megoldásából jelentős előnyt lehet kovácsolni a fej követése során, hiszen az egymást követő képkockák esetén, a megelőző képkockára számolt végleges ε_i érték megadható az aktuális képkocka kezdő ε_i értékének – ezzel csökkentve az iterációk számát. A fejtartás meghatározására a 32. ábra mutat egy kimenetet. A képre rajzolt koordináta-rendszer a POSIT bemeneteként megadott 3-D modell koordináta-rendszere az aktuális képkockára számolt fejtartásnak megfelelően eltolva és elforgatva. A POSIT másik bemenetét az ASM által szolgáltatott 2-D pontokat jelentik, melyeket egy piros-zöld skála mentén színeztem a következőnek megfelelően: minél zöldebb egy pont annál közelebb helyezkedik el a neki megfelelő 3-D modellpont kamera koordináta-rendszer origójához. Demonstrációs célzattal csak az x és y tengelyek körüli elforgatások értékeit írtam ki a képekre.



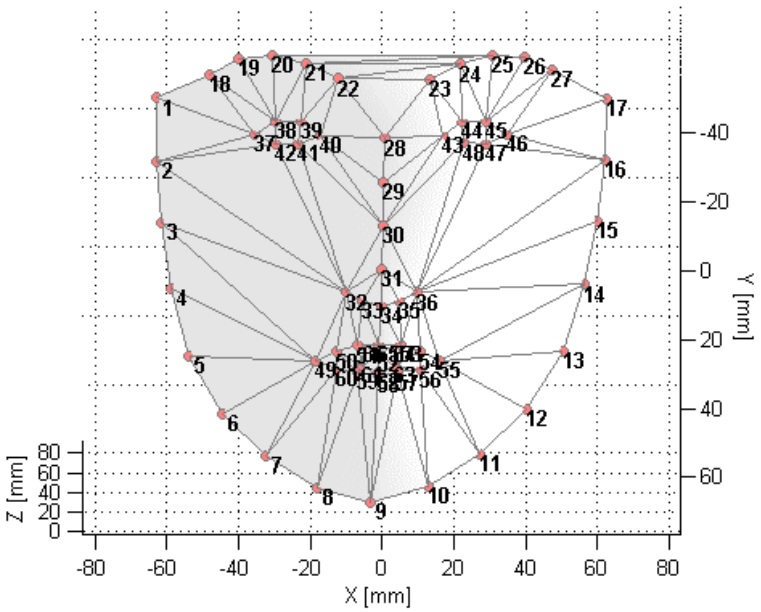
32. ábra. A fejtartás meghatározás kimenete. A képre rajzolt koordináta-rendszer megegyezik a 3-D fejmodell koordináta-rendszerével. Az x tengely körül -15,33° fokkal van elforgatva a modell, az y tengely körül pedig -14,56° fokkal.

4.5. Eredmények

A POSIT tehát azt a transzformációs mátrixot határozza meg, mely a modell koordináta-rendszeréből visz át a kamera koordináta-rendszerébe. A fordított irányra nem lesz szükségünk, de a forgatási mátrix és eltolási vektor invertálásával könnyedén előállíthatjuk azt is. Az egyértelműség kedvéért gyorsan áttekintjük a tesztelés során használt koordináta-rendszereket és modelleket.

4.5.1. Koordináta-rendszerek és modellek

A *kamera koordináta-rendszer* jobbsodrású és megegyezik a 4.2.1. fejezetben leírtakkal: a kamera a z tengely pozitív irányába néz; a kép sorai az x tengellyel párhuzamosan, az oszlopai pedig az y tengellyel párhuzamosan állnak. A *modell koordináta-rendszerre* vonatkozóan nincs megszorítás, de célszerű ugyanúgy jobbsodrásúnak választani, mint a kamera koordináta-rendszert; és ügyeljünk rá, hogy a 3-D modell pontokat, egy a kamera koordináta-rendszerrel együttálló modell koordináta-rendszerben vegyük fel[7].



33. ábra. A 3-D fejmodellem. A fejtartás meghatározása során a piros pontoknak feleltetem meg a kameraképre illesztett ASM 2-D pontjait. A képre rajzolt hálónak csak a modell megjelenítésében van szerepe.

[7] Ez azért szükséges, mert a POSIT a kamerához képest adja meg a modell helyzetét.

A 3-D modell pontjainak meghatározásához egy internetes adatbázis volt a segítségemre [72], ahol a szerzők egy sztereó kamera segítségével készítettek felvételeket emberi arcokról, majd publikálták a színes és a mélységképeket egyaránt. A mélységtérkép pontossága 0,32 mm mind x, y és z irányban egyaránt, ami megfelelt az elvárásaimnak. A korábbi fejezetben ismertetett ASM-et illesztettem pár színes képre az adatbázisban, majd a színes képhez tartozó mélységtérképből kinyertem az egyes határpontokhoz tartozó mélységeket és kiátlagoltam azokat. Az így kapott 3-D modellpontokat eltoltam, úgy, hogy az orrhegy essen a modell koordináta-rendszer origójába (lásd 33. ábra, 31. indexű pontja). Az így kapott pontokat normalizáltam a $[-1, +1]$ intervallumra. A 33. ábra a normalizálás előtti állapotot mutatja így pl. látható, hogy az arc átlagos szélessége 12 cm körüli.

4.5.2. Tesztadatbázis

A tesztelés elején problémába ütköztem, mivel csak korlátozott számban álltak rendelkezésre megfelelő minőségű tesztadatbázisok. A problémát úgy hidaltam át, hogy a szükségesnek ítélt teszteket két csoportba osztottam és ennek megfelelően két adatbázison mértem az algoritmus jóságát (lásd 9. táblázat). A tesztjeim során kiemelt figyelmet fordítottam a fej x (pitch) és y (yaw) tengely körüli elforgatásaira, mivel a további kutatásaim során főként ezeket használtam fel.

Adatbázis	Személyek száma	Képek darabszáma	Fejtartás
Biwi Kinect Head Pose Database [73]	20	15.000	$\pm 75°$ yaw és $\pm 60°$ pitch
GI4E Head Pose Database [74]	10	36.000	$\pm 30°$ yaw és $\pm 30°$ pitch

9. táblázat. A pozitív mintákhoz felhasznált adatbázisok.

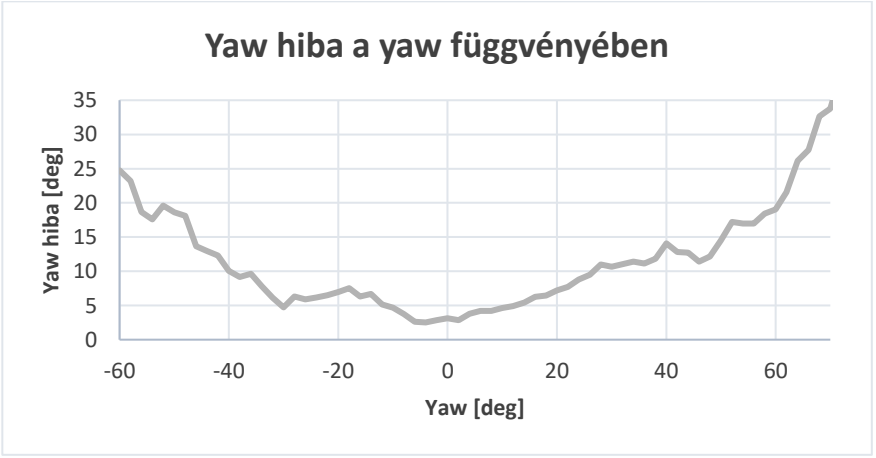
A tesztek egy részét a Biwi adatbázison [73] futtattam le, melyet egy Kinect mozgásérzékelő szenzor segítségével rögzítettek a szerzők. Az adatbázis képeinek a felbontása 640×480 pixel. Az adatbázis annotációját emberi erő nélkül, teljesen automatikusan végezték el[8]. Tehát nincs információ a mérések pontosságára vonatkozóan, ami megnehezíti a tesztek kiértékelését. Az annotációs adatok tekintetében csak a fej középpontjának 3-D koordinátáját

[8] Egy Faceshift nevezetű motion capture szoftver segítségével, lásd faceshift.com.

és a három tengely körüli elforgatás szögét publikálták. Ezzel szemben a GI4E adatbázishoz [74] sokkal több metaadatot publikáltak a szerzők, pl. egy 54 pontból álló alakzatmodell 2-D és 3-D valós referencia koordinátáit (továbbiakban *ground truth*). A képek felbontása is magasabb, 1280×720 pixel. A fejtartást, egy erre a feladatra dedikált szenzor segítségével határozták meg, melynek ismert a mérési hibája, így 0.83º roll, 0.86º yaw és 1.05º pitch pontosság feltételezhető az annotációs adatokat tekintve. Az adatbázis legnagyobb problémája az, hogy mind a pitch és mind a yaw tekintetében csak egy szűk tartományt fed le – fele akkorát, mint a Biwi.

4.5.3. Tesztek a Biwi adatbázison

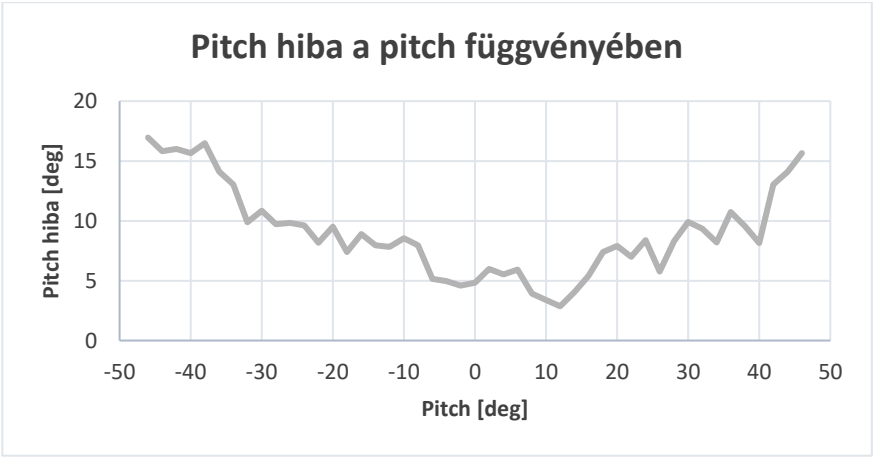
A Biwi adatbázis kellően nagy tartományt fed le, mind a pitch, mind a yaw tekintetében A tesztjeim során, az általam számolt fejtartás pontosságára voltam kíváncsi a mérésekhez tartozó intervallum teljes hosszában. Előzetesen feltételeztem, hogy a fejtartás hibája növekszik a fej oldalra fordításával, így azt az intervallumot akartam kijelölni, ahol a hiba kellően alacsony a további feldolgozásokhoz. A 34. ábra mutatja yaw pontosságának alakulását az adatbázis teljes tartománya felett. A negatív értékek jelentik a balra irányt, a pozitívok pedig a jobbra irányt. Méréseim alapján a $[-40°, +40°]$ szögtartomány az, amikor stabilan 10° fok alatt mozog az orientáció abszolút hibája, a minimumát pedig a $[-12°, +12°]$ tartományban éri el.



34. ábra. A függőleges tengely körüli elforgatás pontossága. Az ábra vízszintes tengelye a ground truth-hoz tartozó yaw; a függőleges pedig az általam meghatározott yaw abszolút hibája. Minden adat fokban értendő.

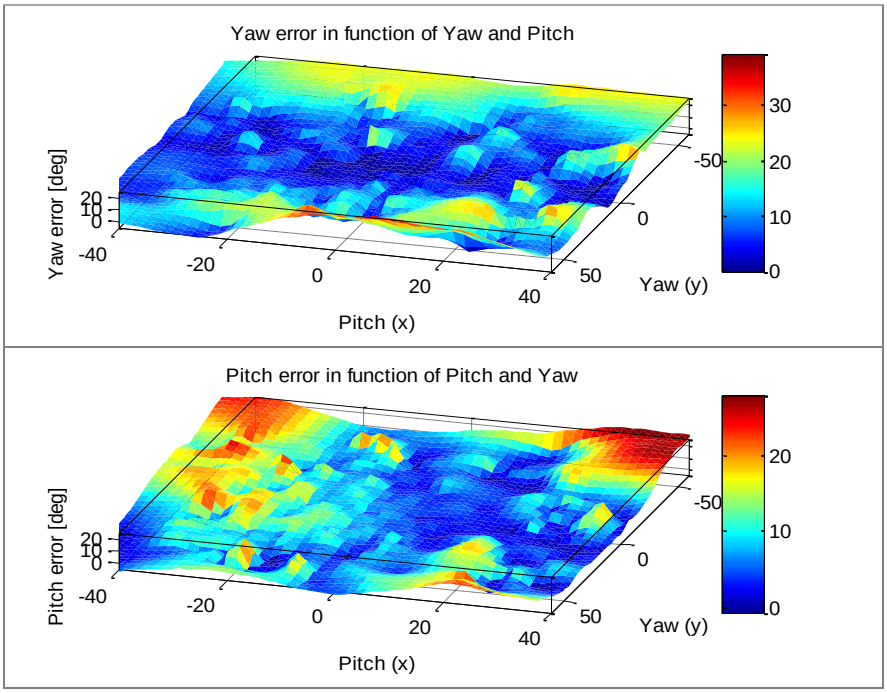
A 34. ábrán látható, hogy a szélső értékekhez közeledve exponenciálisan emelkedik a hiba, amiben komoly szerepet játszhat az ASM modellillesztés pontossága. A 3.5. fejezetben ismertettem, hogy frontális-profil adatbázisra alacsonyabb volt a modellillesztés pontossága, mint arra az adatbázisra, amely csak frontális arcokat tartalmazott. Emlékezzünk vissza, hogy a frontális-profil átmenet a ±45°-os yaw orientációnak felelt meg. A fejtartás meghatározás tesztelése során úgy tapasztaltam, hogy ez az a tartomány, ahol 10°-nál kisebb lehet az algoritmus abszolút hibája (a függőleges tengelyt tekintve). De még ezzel együtt is a gyakorlatban stabilan működhet az eljárásom egy 80-100°-os tartomány felett.

A 35. ábra mutatja a pitch pontosságának alakulását az adatbázis teljes tartománya felett. A negatív értékek a felfelé-, még a pozitívok a lefelé irányuló nézést jelentik. Pitch esetén egy kicsit összetettebb a helyzet, itt a $[-32°, +40°]$ szögtartomány az, amikor stabilan 10° fok alatt mozog az orientáció abszolút hibája, a minimumát pedig a $[-6°, +16°]$ tartományban éri el. Külön érdekesség, hogy a hibafüggvény karakterisztikája nem szimmetrikus a 0°-ra, ami azt jelenti, hogy a felfelé néző emberekre nagyobb hibával dolgozik az algoritmus. Mivel a kamera pont szemből nézte a tesztalanyokat, így a jelenség feltárásához további tesztek elvégzését tartottam szükségesnek.



35. ábra. Az oldalirányú tengely körüli elforgatás pontossága. Az ábra vízszintes tengelye a ground truth-hoz tartozó pitch; a függőleges pedig az általam meghatározott pitch abszolút hibája. Minden adat fokban értendő.

Kézenfekvő megoldásnak tűnt az abszolút hibák elemi felületként történő ábrázolása. A yaw-, és pitch hibákat külön-külön ábrázoltam a yaw és pitch orientációk függvényében (lásd 36. ábra). Egy $N \times M$-es szabályos rácsot definiáltam a ground truth méréshez tartozó yaw-pitch koordináta-rendszerben. Az abszolút hibákat két fokonként összesítettem és átlagoltam a yaw-pitch sík felett, hogy a megjelenített felület kevésbé legyen zajos. Az apróbb lyukakat a szomszédos rácspontok alapján interpoláltam. Nézzünk egy konkrét példát, pl. 12° pitch és -40° yaw esetén az átlagos pitch hiba 7,44° valamint az átlagos yaw hiba 10,06°. Az ábrán látszik, hogy a pitch hibája valóban drasztikusan növekszik a felfelé nézéssel, ugyanakkor némi korreláció fedezhető fel a fej függőleges tengelye körüli elforgatásával is, hiszen a balra-lent és balra-fent irányokban mértem a legnagyobb hibákat. Ez fakadhat az adatbázis jellegéből is, hiszen minden tesztalany a bal oldalról kezdi, majd ugyanabban a sorrendben végzi ugyanazt a mozdulatsort. A yaw-ra számolt hibákra is ugyanez a jelenség látszik: nagyobb a hiba mértéke a baloldalon, mint a jobbon.



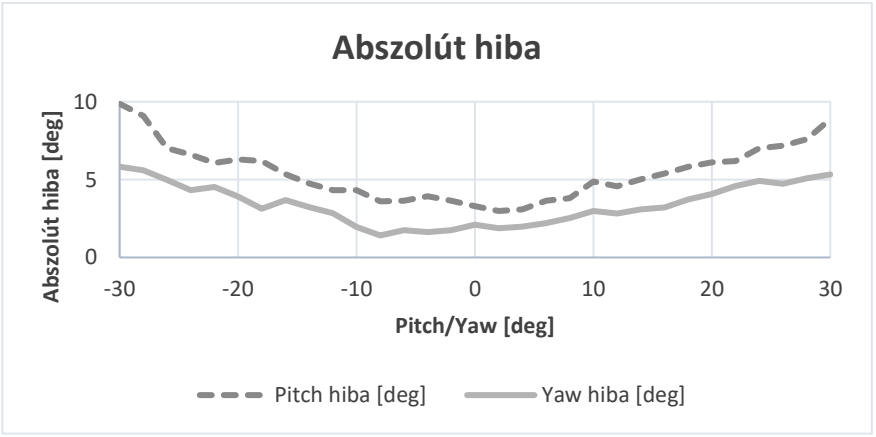
36. ábra. A függőleges-, és oldalirányú tengelyek körüli elforgatás abszolút hibája. A z tengely mentén ábrázoltam a méréseim abszolút hibáit; az x és y tengelyek pedig a ground truth-hoz tartozó pitch és yaw értékeket jelentik. Minden adat fokban értendő.

4.5.4. Tesztek a GI4E adatbázison

A Biwi adatbázisra kapott eredmények alapján úgy gondoltam, hogy egy automatikusan annotált adatbázison – ahol nem ismert a mérések hibája – nem elegendő kiértékelni a módszert, ahhoz, hogy átfogó képet kapjak a pontosságról. Így további teszteket végeztem el egy másik adatbázison is: GI4E [74]. Előzetesen már említettem, hogy ehhez az adatbázishoz sokkal több metaadatot publikáltak a szerzők, így például:

- Egy 54 darab pontból álló alakzatmodell 2-D és 3-D koordinátáit, mellyel könnyedén verifikálható az általam meghatározott fejtartás.
- Továbbá, egy erre a feladatra dedikált szenzor segítségével a fej pozícióját és orientációját a 3-D térben (a mérések hibája 1° körüli).

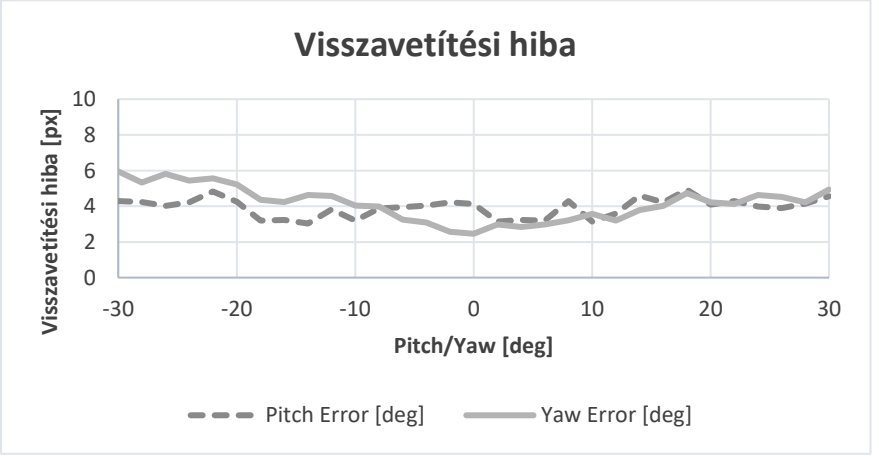
Az adatbázis ugyan egy kicsit szűkebb tartományt fed le a fejtartások tekintetében, de több hasznos tesztet is le tudtam futtatni rá. A 37. ábrán szemléltettem az adatbázisra kapott abszolút hibákat, melyeket a saját ASM modellem és 3-D fejmodellem alapján határoztam meg. A pitch erre az adatbázisra is pontatlanabb, mint a yaw, de a két függvény karakterisztikája jobban megfelel egymásnak, mint a Biwi adatbázis esetén. A függőleges tengely körüli fejmozgás hibája a $[-25°, +25°]$ tartományban van 5° alatt, ami sokkal jobb eredmény, mint amit a Biwi esetén mértem. Ezzel szemben az oldalirányú fejmozgás egy szűkebb intervallumban van 5° alatt: $[-15°, +15°]$, de itt is jobb eredményeket kaptam, mint a Biwi esetén.



37. ábra. A pitch-, és a yaw abszolút hibája. Az ábra vízszintes tengelye a ground truth-hoz tartozó tényleges pitch és yaw értékeket reprezentálja, még a függőleges az általam meghatározott pitch és yaw hibáját (az ASM modellillesztés hibájával terhelve). Minden adat fokban értendő.

Az adatbázist felhasználva ki tudtam mérni tisztán a fejtartás meghatározás hibáját is, úgy, hogy alkalmaztam az eljárásomat a szerzők által publikált 54 db pontot tartalmazó 2-D és 3-D arcmodellekre – ezek a mérések nem voltak terhelve egyéb hibával, mint pl. az ASM modellillesztés. A teszteset eredményeként visszacsatolást kaptam az ASM modellillesztés hatására a fejtartás meghatározást illetően. A ground truth-hoz tartozó 2-D és 3-D pontmegfeleltetések alapján számolt abszolút hibákat nem ábrázolom külön grafikonban, ugyanis azt tapasztaltam, hogy a teljes intervallum hosszában közel egyenletesek, yaw esetén 1,45° az abszolút hiba, pitch esetén pedig 1,77° – ami összeadódhat egyrészt a szenzor mérési hibájából, másrészt az szerzők által választott 3-D modellből is.

Megvizsgáltam azt is, hogy hogyan alakul a visszavetítés hibája az oldalirányú-, és függőleges fejmozgások tekintetében. A visszavetítés hibáját a következő módon kaphatjuk meg: vetítsük le a 3-D fejmodellt a 6 DoF fejtartáshoz tartozó transzformációs mátrix segítségével a képsíkra, majd számítsuk ki az RMS hibát a vetítés során kapott 2-D pontok és a ground truth (vagy az ASM által számított) 2-D pontok között. A visszavetítés hibáját a 38. ábra mutatja.



38. ábra. A visszavetítés hibája. Az ábra vízszintes tengelye a ground truth-hoz tartozó pitch és yaw értékeket reprezentálja fokban, még a függőleges a fejtartás ismeretében, a 3-D fejmodellnek a képsíkra levetített pontjai és az ASM által meghatározott határpontok RMS hibáját pixelben.

Megfigyelhető, hogy a visszavetítés hibája sokkalta egyenletesebb, mint a fejtartás abszolút hibája: 2-6 pixel között mozog az intervallum teljes hosszában. Ennek egy lehetséges magyarázata az, hogy az általam használt 3-D fejmodell és a GI4E adatbázisban szereplő tesztalanyok fejének az arányai némiképp eltérőek – a módszer ezen függőségén azonban nem igazán lehet változtatni. Megjegyzem, hogy a ground truth-hoz csatolt 3-D fejmodell vetítése során 0,42 pixel pitch és 0,39 pixel yaw visszavetítési hibát mértem – tehát subpixel pontos. A méréseket nem ábrázolom külön grafikonon, de azok mindegyike megtekinthető a *Függelék C* fejezetben.

4.5.5. Összegzés

A fenti tapasztalatokat összegezve elmondható, hogy ha a bemenő adatok (2-D és 3-D mérések) kevésbé zajterheltek, akkor a fejtartás, mint P4P probléma kellő pontossággal megoldható a fentebb ismertetett POSIT algoritmussal. Ekkor az abszolút hiba várható értéke 1,5° körül alakul, illetve a visszavetítés hibája is 1 pixel alatti. Ismeretlen környezetben azonban meghatározó az ASM modellillesztés hibája és az a tény, hogy az eljáráshoz egy előzetesen létrehozott 3-D fejmodellt használok, melynek arányai eltérhetnek a végfelhasználó fejének arányaitól. Ezek a bizonytalanságok egyben azt is jelentik, hogy a függőleges és a vízszintes tengely mentén értelmezett orientációt csak egy 80-100°-os tartományban van értelme felhasználni a további kutatásokhoz. Ez a korlátozás nem számottevő a vízszintes tengelyt tekintve, hiszen fel-le irányban egy átlagos felnőtt -60,4°-tól egészen 69,6° fokban képes elmozdítani a fejét [75]. Ellenben a függőleges tengely körüli irányt tekintve meghatározóbb lehet a korlátozás, hiszen ebben az irányban -79,8°-tól 75,3°-ig mozoghat a fej.

A módszer összevetése más nyilvánosan elérhető és egységesen használt irodalmi megoldásokkal adatbázis hiányában elég nehézkes. A szakirodalomban előforduló módszerek túlnyomó részét főként belső használatra szánt, rendkívül drága[9] adatbázisokon tesztelik, melyek pár száz, vagy ezer képet tartalmaznak 5-10 tesztalanyról. A szakirodalomban használt módszerek túlnyomó többsége az általunk is használt abszolút hibával jellemzi az orientációszámítás pontosságát. A [48]-ben leírt tanulmányban látszik, hogy a módszerem pontossága nem marad el a vetélytársakétól és teljesíti is az elvárásaim túlnyomó részét egy jövőbeli fejtartás meghatározó algoritmusra

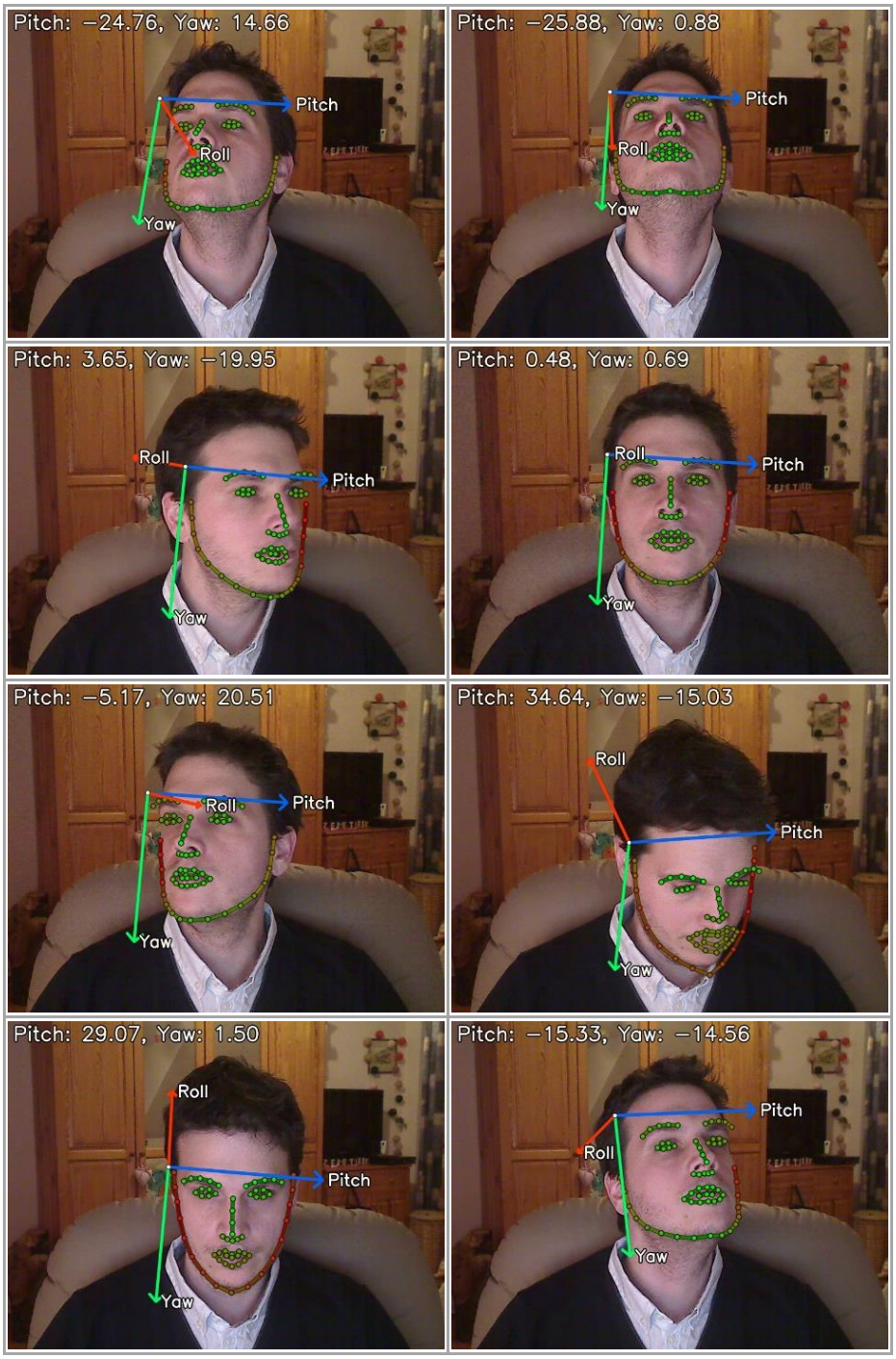
[9] Feltehetően az adatgyűjtés nehézkes és költséges volta miatt.

vonatkozóan: egykamerás rendszeren működik, nincs szükség kézi beavatkozásra, több ember fejtartása meghatározható egy képen belül, többé-kevésbé megvilágítás független, széles tartományt lefed az egyes tengelyen mentén, és valós idejű számítást tesz lehetővé. A 10. táblázatban összegeztem a módszerek pontosságait a két tesztadatbázisra. Hangsúlyozom, hogy a Biwi adatbázis esetén a $\pm 75°$ yaw és $\pm 60°$ pitch tartományban összegeztem az értékeket, GI4E esetén pedig egységesen $\pm 30°$ yaw és pitch tartományban tudtam mérni.

Adatbázis	Pitch pontossága	Yaw pontossága
Biwi Kinect Head Pose Database [72]	9,02°	12,03°
GI4E Head Pose Database [73]	5,49°	3,41°

10. táblázat. A vízszintes és függőleges tengelyek körüli orientáció átlagos pontossága a két tesztadatbázisra.

Összegezvén a munkámat, jelen fejezetben egy 2-D archáló ismeretében kísérletet tettem a *fej térbeli helyzetének felismerésére*. 6DoF modellként értelmeztem az emberi fejet a 3-D valós világnak megfelelő térben. Ebben a 3-D térben határoztam meg a fejnek a három tengely mentén vett eltolását és a három tengely körül értelmezett elforgatását. A fejtartás felismerését egy PnP probléma definiálásával és levezetésével ültettem át a gyakorlatba. Definiáltam egy geometriailag helyes 3-D fejmodellt és a 3-D-2-D pontmegfeleltetések alapján számítottam ki a fejtartást. Maga az eljárás a perspektív vetítés közelítésének is tekinthető. A módszer pontosságát több adatbázison teszteltem. Úgy találtam, hogy *kellően nagy tartományban*, *kellően nagy pontosság* érhető el a fejtartás felismerése során, így a módszer alapját képezheti a további feldolgozásaimnak. A 39. ábrán a teljesség igénye nélkül néhány kimenetet mutatok a fejtartás meghatározására vonatkozóan. Élő kameraképről mentettem ki néhány képkockát a feldolgozások után. A képekre rajzolt koordináta-rendszer megegyezik a 3-D fejmodell koordináta-rendszerével, illetve a fejmodell pontjait egy piros-zöld skála alapján színeztem, ahol minél zöldebb egy pont színe, annál közelebb helyezkedik el a kamerához – ezt a fejpozíció Z komponense, illetve az orientációhoz tartozó forgatási mátrix alapján határoztam meg. Demonstrációs célzattal minden esetben feltüntettem az x és y tengelyek közötti elforgatás szögét (fokban).



39. ábra. A fejtartás meghatározás néhány kimenete.

5. Fejmozgások felismerése

Az értekezés elején, a bevezető fejezetben ismertettem a HCI céljait és kihívásait. Azzal a megállapítással éltem, hogy a *HCI* elsődleges célja a kapcsolatban szereplő emberi fél vizsgálata és ezáltal az ember-számítógép interakciók természetesebbé tétele az emberi fél számára. Ehhez azonban a rendelkezésünkre álló információ halmazból – esetünkben főként digitális képek közül – ki kell választani, vagy éppenséggel elő kell állítani az interakciók releváns atomi építőelemeit – csak úgy, mint az arc pozíciója, karakterisztikus pontjai, vagy térbeli orientációja – a további feldolgozás céljából. A kommunikáció egyszerűsítésére irányuló terveimet a mindennapi életből merítettem. A szóbeliség (mint *verbális jelek*, hangok halmaza) a legtipikusabb és egyben a legnagyobb információ tartalommal bíró módja az ember-ember kommunikációnak. De az egyes kommunikációs helyzeteket célszerű ennél alaposabban is megvizsgálni, hiszen a verbális jelek mellett a szóbeli információk kiegészítésére, ellenőrzésére vagy éppen hangsúlyozására a nem-szóbeli, úgynevezett *nonverbális jelrendszert* alkalmazzuk. A nonverbális jelek tipikus megnyilvánulásai a mimika, a tekintet, szemkontaktus, vagy éppenséggel a fej-, és szemmozgás.

Az értekezés utolsó fejezetében az emberi fejből és arcból kinyerhető nonverbális jelek és jellegzetességek kinyerésével, reprezentálásával, valamint felismerésével foglalkozok. Az arcdetektálás, karakterisztikus pontok kinyerése és a fejtartás meghatározása során elért eredményeimre támaszkodva javaslatot teszek egy fejmozgás alapú gesztusok (mint nonverbális jelek) felismerését megcélzó rendszerre, mely alkalmas lehet kognitív infokommunikációs rendszerek irányítására – amelyben a gesztusok egyfajta parancsként is értelmezhetők (a CogInfoCom terminológiájában ezeket *bodicon*oknak nevezi [76]). A fejmozgások felismerése tulajdonképpen emberi tevékenységek felismerését jelenti, az angol szakirodalom ezt a folyamatot *human activity recognition*ként hivatkozza, melyet a továbbiakban gyakran rövidítve *HAR*-ként fogok használni. Tehát a HAR során, néhány előzetes megfigyelés és mérés alapján a felhasználó által végzett gesztusokat kívánjuk felismerni. A HAR segítségével tehát új irányokat nyithatunk a személyre szabott interfészek piacán, hiszen számos akár már meglévő alkalmazás funkcionalitása terjeszthető ki könnyedén új vezérlési eszközrendszerek alapján. Megjegyzem, hogy a HAR interfészek

emberközelisége miatt egyre több megoldás jelenik meg a piacon, melyek a mozgásukban korlátozott emberek számára kíván segítséget nyújtani a mindennapi feladataik megkönnyítésében. A HAR fejlesztések tehát segíthetnek a mozgáskorlátozott emberek életminőségét javítani és akár segíthet nekik visszatérni a munkaerőpiacra. Friss statisztikák nem léteznek, de az Európai Közösségek Statisztikai Hivatala (Eurostat) 2003-as jelentéséből kitűnik, hogy több mint 80 millió európai állampolgár rendelkezik valamiféle fogyatékossággal [77]. E széles réteg számára hozhat tehát közvetlen változást a számítógépes HAR.

5.1. Irodalmi áttekintés

Kijelenthető, hogy a fejmozgásból kinyerhető információ meghatározó erejű a kommunikációban; mindez jól látszik a szakirodalomban is, hiszen a fejmozgás felismerésre jelentős figyelem összpontosul a HCI szakterületén belül. Napjainkban is számos kutatás és működő alkalmazás, megoldás lát napvilágot a témakörben, beleértve a biztonsági rendszereket, beteg monitorozó rendszereket és egyéb más megoldásokat, melyek magukban foglalják az ember és számítógép közötti kölcsönhatásokat. Az alkalmazások nagy többsége magasszinten ismeri fel az emberi tevékenységeket, melyek gyakran több művelet összekapcsolása által jönnek létre és gyakran nem is egy, hanem több ember végzi el azokat.

Az eljárások csoportosítása a szenzor elhelyezése alapján tehető meg. Beszélhetünk *relatív* és *abszolút* elhelyezkedésű eljárásokról, ahol a viszonyt azt határozza meg, hogy a HAR szenzor hogyan kapcsolódik a felhasználóhoz. Ha a szenzor közvetlenül az felhasználóra van helyezve, akkor *relatív eljárásokról* beszélünk. A relatív kifejezés itt a *szenzor elhelyezkedésére* utal, mivel ekkor az adatgyűjtő szenzor az felhasználóval együtt mozog a komplex gesztusok kivitelezése és felismerése során. A szenzor által mért jelek több csoportba oszthatók, de a szakirodalomban leggyakrabban valamilyen mozgáshoz kötődő mennyiséget (pl. *pozíció*, *orientáció*, *sebesség* vagy *gyorsulás*) mérnek egy arra alkalmas hardverrel. Manapság, az olcsó IMU szenzorok (*gyorsulásmérők*, *giroszkópok*, *magnetométerek*) térnyerésével egyenes arányban megnőtt a relatív eljárások alkalmazása. A megoldások többsége a szenzoros mérésekből kinyert jellemzők osztályozásán alapul pl. [78, 79, 80, 81].

A másik nagy csoportba az *abszolút eljárások* tartoznak, ekkor a szenzor pozíciója nem változik a gesztusok kivitelezése és felismerése során, az mindig

egy helyben marad. Ebben az esetben nem szükséges, hogy a felhasználó kellemetlen, általa kevésbé elfogadott eszközöket viseljen, illetve a mozgását sem korlátozza pl. egy vezetékes kapcsolat a szenzor és a számítógép között. Azonban a HAR tere lekorlátozódik a környezetnek egy szűk területére (oda ahová a szenzort elhelyeztük). Az abszolút eljárások főként gépi látás alapú eljárások és ezen belül tovább kategorizálhatók annak függvényében, hogy az időt, mint dimenziót hogyan kezelik. Beszélhetünk *közvetlen osztályozási problémákról*, melyek pusztán képi jellemzőkön alapuló osztályozókat használnak, így az idő, mint együttható nem jelenik meg a modellben. Ebben az esetben, a tevékenységek felismerése képkockáról-képkockára haladva történik az osztályozó segítségével. Továbbá beszélhetünk *időbeli állapottér modellekről*, ahol az idő külön dimenzióként jelenik meg a modellben. Ez esetben minden megfigyelés megfeleltethető egy jellemző vektornak, melyet egy adott időpillanatban készült képkockából nyerünk ki. Pár példát láthatunk az előbbi megoldásokra a következő tanulmányokban: [82, 83, 84, 85].

Azonban mind a relatív-, mind az abszolút eljárások többségére igaz, hogy a megfigyelések során előálló tulajdonságokat az idő függvényében indexelik. Ez a fajta indexelés teszi lehetővé, hogy megfogalmazzuk a *HAR problémát*[10] az alábbi módon [86]:

Definíció (HAR probléma): legyen adva k darab idősornak egy $S = \{s_0, \dots, s_{k-1}\}$ halmaza. Az s_i idősor egy valószínűségi változó realizációja, mely egy adott tulajdonság (attribútum) időbeli megfigyeléseiből épül fel. A k darab idősor az $I = [t_\alpha, t_\omega]$ időintervallum felett van értelmezve. A HAR célja: az I időintervallum felosztása I_j partíciókra az S halmaz alapján, úgy, hogy minden I_j intervallum egy adott tevékenységhez (pl. fejrázáshoz, vagy bólintáshoz) tartozzon. Az I_j időintervallum során elvégzett tevékenységet jelöljük az $l_j \in L$ címkével. Feltételezhetjük, hogy az I_j intervallumok diszjunktak és nem üresek. A HAR probléma megadható a (S, I, L) rendezett hármassal.

A fejmozgások felismerése során egyrészt meghatározzuk és megfigyeljük azokat az attribútumokat, amelyekkel egy-egy fejmozgást reprezentálható, másrészt az egyes fejmozgásokat időben is be kell határolni – tudni kell mikor

[10] Az egyszerűség kedvéért feltesszük, hogy egyszerre csak egyfajta mozdulatot végezhet a felhasználó. Pl. nem bólogathat és rázhatja a fejét egyszerre.

kezdődött és végződött egy-egy mozdulat. Az általam javasolt eljárás, egy *gépi látás alapú* eljárás, ahol a szenzor *elhelyezkedése abszolút* a felhasználóhoz képest. A fejmozgásokat fix helyen, a képernyőhöz közeli térben kívánom kellően magas pontossággal felismerni. A következő fejezetekben bevezetem a fejmozgásnak egy alkalmas tér-, és időbeli reprezentációját, majd egy dinamikus idővetemítésen (*dynamic time warping*, röviden *DTW*) alapuló modell segítségével kísérletet teszek a változatos fejmozgások felismerésére.

5.2. Fejmozgás

A fejtartás felismerésével és követésével képesek lehetünk bizonyos tevékenységek felismerésére. Az elmúlt néhány évben számos tanulmány látott napvilágot ezt illetően [87, 88, 89, 90]. E tanulmányok mindegyike a fejtartás követésére összpontosít, és bár definiálnak néhány egyszerű jellemzőt a felismerni kívánt tevékenységekhez mégis azt kell, hogy mondjuk ezek egyike alkalmas komplex tevékenységek felismerésére. A komplex mozdulatok (pl. egy írott 'a' betű kontúrjának előállítása a fej mozgásával) felismeréséhez először tanulmányoztam a fejmozgás jellemzőit. Egy-egy fejmozgás tipikus hossza három-öt másodperc, de nem kívánok felső korlátot adni a lehetséges mozgások időbeliségét illetően. A felhasználótól nem várom, hogy tudatában legyen annak, hogy egy-egy mozgást pontosan hogyan és mennyi idő alatt kell elvégezni. Így két fejmozgás – még ha ugyanaz a felhasználó is végzi el – sosem lesz teljesen megegyező[11]. A fejmozgáshoz tartozó idősorok tehát változnak az idő vagy sebesség függvényében. Az összehasonlításukhoz az idő dimenzió vonatkozásában nem-lineárisan kell illeszteni őket egymáshoz.

A következő fejezetekben először bevezetem a fejmozgásoknak egy lehetséges reprezentációját. A reprezentáció definiálásakor arra törekedtem, hogy változatos és komplex fejmozgásokat tudjak felismerni valós időben. A reprezentáció során a fejtartásból nyertem ki bizonyos jellemzőket, majd határoztam meg a fejmozgás trajektóriáját. Természetesen a fejmozgásokat időben is be kellett határolnom. A fejmozgás kezdetét és végét is a fejtartásból nyertem ki. Így az általam javasolt modell szintetizálja a térbeli és időbeli dimenzióit a fejmozgásoknak.

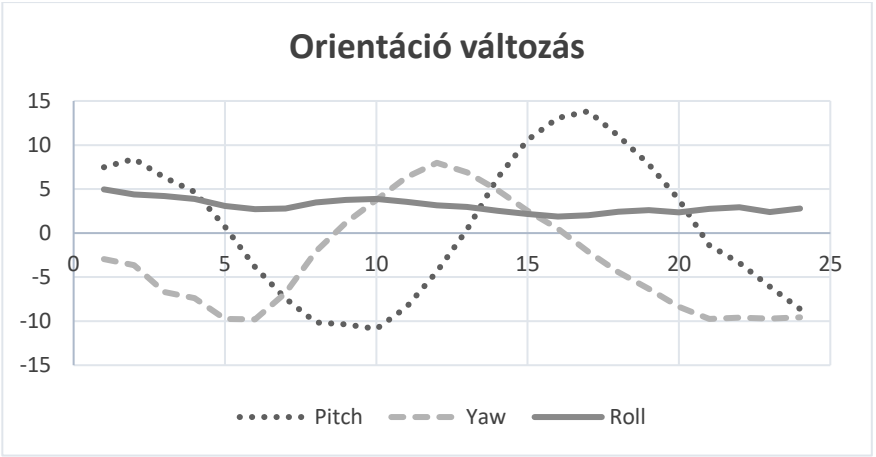
[11] Nincs olyan lineáris transzformáció, amely az egyiket átvinné a másikba, azaz nem-lineárisan illeszthetők egymáshoz.

5.2.1. Fejmozgás reprezentációja

Ebben a fejezetben ismertetem a fejmozgás térbeli reprezentációját és időbeli behatárolását. A mozgást leíró jellemzők meghatározásából indulok ki. Az előző fejezetekben számos metaadatot nyertem ki az arcról, pl. pozíciója és kiterjedése, bizonyos karakterisztikus pontjainak koordinátái, illetve a térbeli elhelyezkedése és orientációja. A fejmozgások felismeréséhez meg kellett határoznom azokat a metaadatokat, melyekkel a legpontosabb és legstabilabb működést kaptam. A [8, 91] publikációimban azt tapasztaltam, hogy az arc bizonyos pontjainak a követésével stabil működést lehet elérni. Azonban a módszer átfogó tesztelése során kiderült, hogy a megbízhatóság javításra szorul. Akkor jelentkeztek problémák, amikor gyorsabb mozgások esetén az optikai folyam algoritmus [92] elvesztette a pontokat, illetve túl nagy változások álltak be a gradiens alapú irány meghatározásban, melyek szűrése nehézkes volt. Így magasabb szinten a fejtartás alapján próbáltam megoldani a problémát. Ehhez azonban alaposabban meg kellett vizsgálni a fej térbeli pozíciójának és orientációjának a változását az idő függvényében.

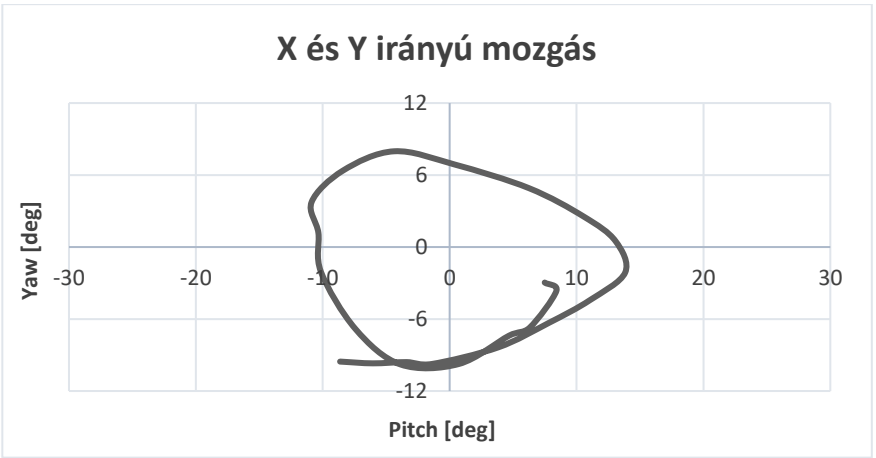
Feltételezhető, hogy a felhasználó a kamera előtt ülve mozgatja a fejét, úgy, hogy annak jórészt csak az orientációja változik. Kutatásaim kezdetén úgy döntöttem, hogy a rendszeremben csökkentem a modellem komplexitását a fej térbeli pozíciójára irányuló dimenziók elhagyásával. Úgy véltem, hogy a fejmozgás reprezentációjában elegendő lehet csak az x és y tengely körüli elforgatásokat szerepeltetni (pitch és yaw). Így a többi metaadat elhagyható a felismerés során. Később azonban az eredményeim ismeretében mégis kiegészítettem a metaadatok körét a felismerés során – melynek az okát lentebb ismertetem.

A 40. ábrán láthatjuk a fej térbeli orientációjának időbeli változását egy körkörös fejmozgás elvégzése során (lényegében egy 'O' betűt formáztam meg a fejemmel). Látható, hogy a z tengely körüli fejmozgás varianciája elhanyagolható ebben az esetben, a másik két tengely mentén azonban jelentős kilengés látható. A pozíció változást külön nem ábrázoltam, de ott is hasonlóan alacsony varianciát mértem az egyes komponensekben, mint a z tengely körüli orientációváltozás esetében.



40. ábra. A fej orientációjának változása egy 'O' betű leírása közben. A vízszintes tengely a feldolgozott képkockák sorszámai, a függőleges pedig a tengelyek körüli elforgatások szögei fokban.

A 41. ábrán látható, hogy 24 db képkockára számított pitch és yaw értékek hűen visszaadják a körkörös fejmozgás trajektóriáját. A méréseket itt értelemszerűen a pitch-yaw koordináta-rendszerben ábrázoltam.



41. ábra. A fej orientációjának változása egy 'O' betű leírása közben. A vízszintes tengely az x tengely körüli elforgatás szögeit ábrázolja, a függőleges pedig az y körülit.

5.2.2. Hibás fejtartások kiszűrése

A mérési hibákból fakadóan előállhatnak olyan esetek, amikor a fejtartás nem, vagy csak hibásan határozható meg – elegendő arra gondolni, hogy az ASM illesztés során rossz helyre konvergál, vagy arra, hogy a P4P problémánknak

sincs minden esetben egyértelmű megoldása. A hibás fejtartásokat így ajánlott kiszűrni a fejmozgás felismerése előtt. Jelen fejezetben egy lineáris Kálmán szűrőt ismertetek, melyet a fejtartás „nyers" kimenetére alkalmazok. A Kálmán-szűrés elméletét [93] nem áll módomban részletesen ismertetni, de röviden összefoglalva a Kálmán szűrő egy olyan algoritmus, mely sorozatos mérések által ad optimális becslést változó rendszerek állapotáról, figyelembe véve az állapotméréseket és a zavaró tényezőket (pl. zajok és mérési bizonytalanságok). Az állapotátmenet modell egy Wiener-folyamat modelljének tekinthető [94], mely a mozgó objektumok dinamikájának modellezését tekintve jó kompromisszumot biztosít az összetettség és a teljesítmény között. Az egyes $x_k \in \mathbb{R}^9$ állapotváltozók rendre a pozíciót, a sebességet és a gyorsulást reprezentálják a modell koordináta-rendszerének az északi, keleti és lefelé mutató irányába, tehát:

$$x_k = (x, y, z, \dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z})^T, \tag{5.1}$$

ahol x, y és z komponensek a fej térbeli pozíciójához tartoznak k-adik időpillanatban (az első és második deriváltak pedig rendre a sebesség és a gyorsulás). Az állapotátmenet modell teremt kapcsolatot az aktuális k-adik és a megelőző $(k-1)$-edik időpillanat között:

$$x_k = F_k x_{k-1} + B_k u_k + w_k, \tag{5.2}$$

ahol F_k az állapotátmenet mátrix, u_k egy ismert bemeneti vektor, w_k pedig a folyamat normális eloszlású zaját reprezentálja, ez utóbbinak a kovariancia mátrixát jelöljük Q_k-val. Bővebben kifejtve ($u_k = 0$ esetén):

$$x_k = \begin{pmatrix} E_3 & \Delta t E_3 & \frac{1}{2}\Delta t^2 E_3 \\ Z_3 & E_3 & \Delta t E_3 \\ Z_3 & Z_3 & E_3 \end{pmatrix} x_{k-1} + w_k, \tag{5.3}$$

ahol E_3 a 3×3-as egységmátrix, Z_3 a 3×3-as zérusmátrix, a Δt skalármennyiség pedig k-adik és a $(k-1)$-edik időpillanat között eltelt idő. A Kálmán-szűrő megfigyelési modellje teremt kapcsolatot az aktuális x_k állapot és az aktuális $z_k = (x, y, z)^T$ mérés között:

$$z_k = H_k x_k + v_k, \tag{5.4}$$

ahol a H_k mátrix alapján történik a leképezés az x_k állapotról a z_k mérésre, továbbá v_k mérés normális eloszlású zaja, melynek a kovariancia mátrixát

jelöljük R_k-val. Az egyszerűség kedvéért tegyük fel, hogy csak 3-D pozíciók, mint megfigyelések állnak a rendelkezésünkre, így a fenti egyenlet átírható az alábbi formában:

$$z_k = (E_3 \;\; Z_3 \;\; Z_3)x_k + v_k. \tag{5.5}$$

Működésében a Kálmán-szűrő két részre osztható: *jóslás fázis* és egy *javítás fázis*. Azaz kicsit bővebben, a Kálmán-szűrő átlagolja a rendszer állapotainak becsült adatait, majd egy újonnan bejövő méréssel súlyozott átlagolást végez. A súlyozást a Q_k kovarianciából számolja, mely a rendszer állapotainak becsléséből fakadó bizonytalanságokból származik. A súlyozott átlag eredményeként egy új állapotbecslést kapunk, mely a becsült és mért állapot között van. A folyamat lépésenként ismétlődik egy iterációs eljárással. A kovariancia mátrixok hatása a folyamatra:

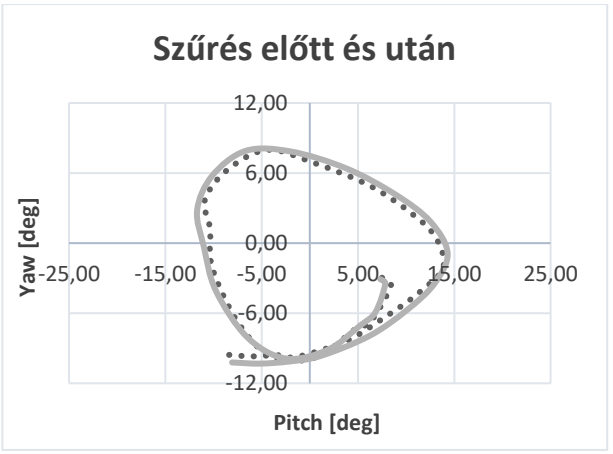
- Q_k a folyamat zajának kovariancia mátrixa, mely a teljes bizonytalansághoz járul hozzá. Nagy Q_k esetén a szűrő jobban fogja követni a nagy és hirtelen változásokat a mérésekben.
- R_k a megfigyelések zaja, amely azt határozza meg, hogy mekkora mennyiségű információ kerüljön felhasználásra a mérésekből. Ha R_k túl nagy, akkor a szűrő a méréseket kevésbé gondolja pontosnak, ellenben kis R_k esetén jobban követi a méréseket.

A fentebb definiált modell könnyedén kiterjeszthető a 3-D pozíció és orientáció együttes szűrésére[12]. Ehhez az állapot változókat egészítsük ki 9 darab másik komponenssel, melyek rendre a roll (ψ), pitch (ϕ) és yaw (θ) szögek, valamint ezek első és második deriváltjai (szögsebesség, szöggyorsulás):

$$\mathrm{x} = \left(x, y, z, \dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z}, \psi, \theta, \phi, \dot{\psi}, \dot{\theta}, \dot{\phi}, \ddot{\psi}, \ddot{\theta}, \ddot{\phi}\right)^T. \tag{5.6}$$

Az állapotátmenet $F_k \in \mathbb{R}^{18\times18}$ és a megfigyelési modell $H_k \in \mathbb{R}^{18\times6}$ mátrixát terjedelem hiányában nem ismertetetem. A 42. ábrán mutatok egy példát a fejmozgás meghatározás során felhasznált adatok szűrés előtti úgynevezett nyers értékeire és a szűrést követő értékeikre. Kivehető, hogy a szűrt értékek sokkal simább és egyenletesebb karakterisztikával rendelkeznek.

[12] Mivel orientáció kapcsán 3 db Euler szög áll a rendelkezésünkre, így a modell lineáris jellege megmarad.



42. ábra. A pitch és yaw értékek ábrázolása az idő függvényében. *Pontozott* vonallal jelöltem az eredeti, nyers méréseket. *Folytonos* vonallal pedig a Kálmán-szűrővel simított méréseket.

Jelen értekezésben a folyamat-, és a megfigyelések zajának kovariancia mátrixai rendre a $Q_k = 10^{-4} \cdot E_{18}$ valamint $R_k = 10^{-4} \cdot E_6$ mátrixok voltak, ahol E_{18} a 18×18-as és E_6 a 6×6-os egységmátrix.

5.2.3. Trajektória

Az előző fejezetben ismertetett Kálmán-szűrő segítségével kiszűrtem azokat az eseteket, amelyek a fejmodell illesztése, vagy a fejtartás meghatározása során kiugró/hibás értékkel vettek volna részt felismerés során. A szűrő alkalmazása tehát az alábbiak miatt szükséges:

- Egyrészt a fejmozgások időbeli behatárolását a fejtartás alapján kívánom meghatározni, 1-1 kiugró értékből pedig könnyen hihetnénk azt, hogy éppen egy gyors mozgás történik a képen.
- Másrészt pedig sok zajos mérés komoly hatással lehet a mozgás karakterisztikájára, ami megnehezíteni a modellillesztést a fejmozgás felismerése során.

A fejmozgások felismerése során tehát csak az x és y tengelyek körüli elforgatásokat kívántam felhasználni. Későbbi tesztjeim során azonban úgy találtam, hogy jobb eredményeket érhetek el, ha a z tengely körüli elforgatást is figyelembe veszem. Első körben a fejmozgás térbeli reprezentációját adom meg. A bevezető fejezetben definiáltam a HAR problémát. Legyen adva egy kép szekvencia, melyről sejtjük, hogy az elemei valamilyen mozgásformát

írnak le. A képszekvencián látható fejmozgáshoz egy $s \in S$ idősort szeretnék hozzárendelni. Az előzőekben ismertettem, hogy s egy adott tulajdonság időbeli megfigyeléseiből épül fel, tehát s egy statisztikai minta, mely álljon az $a_\tau, a_{\tau+1}, \dots, a_{\tau+n}$ megfigyelésekből. Az egyes a_i megfigyelések $\mathcal{F}$-fel jelölt jellemzőtér elemei: $a_i \in \mathcal{F}$, ahol $i \in [\tau : \tau + n]$. Tehát összesen n darab megfigyelésünk van és az első megfigyelés időbélyege, $\tau \geq 0$. Definiáljuk az egyes $m \in \mathcal{M}$ fejmozgásokat az alábbi formában:

$$m = (l, a_\tau, a_{\tau+1}, \dots, a_{\tau+n}) = (l, s), \tag{5.7}$$

ahol $l \in L$ az elvégzett tevékenység címkéje (pl. bólintás, vagy fejrázás). A térbeli reprezentáció az a_i attribútumok előállítását jelenti. Az attribútumok mérése során vegyük a 3-D fejmodell koordináta-rendszerének z irányú bázisvektorát: $\underline{k} = (0,0,1)$[13]. Forgassuk el $\underline{k}$-t a pillanatnyi fejtartásnak megfelelően és az eredményül kapott vektort jelöljük $\underline{k_i}$-vel. Az így kapott irányvektorok ismeretében definiáljuk az attribútumokat az alábbi módon:

$$a_i = \underline{k_i} - \underline{k_{i-1}}, \tag{5.8}$$

azaz az a_i attribútum az aktuális-, és a megelőző irányvektor különbsége, ami esetünkben a fej orientációjának változását jelenti az előző állapotához képest. Tehát a_i a két szomszédos képkocka közötti fejmozgás orientációváltozásának irányába mutat. Kevésbé intenzív fejmozgások esetén a_i normája zérushoz közeli, még intenzív fejmozgások esetén a norma szignifikánsan eltér a zérustól. A felismerés során növelhetjük a robusztusságot, ha az egyes a_i méréseket eltoljuk a_τ-val (így a mérések függetlenek lesznek a kezdő pozíciótól), tehát a fejmozgásokat az alábbi, formában definiáltam:

$$m = (l, 0, a_{\tau+1} - a_\tau, \dots, a_{\tau+k} - a_\tau) = (l, s'). \tag{5.9}$$

A 43. ábra a kép szekvencia két-két szomszédos tagja közötti orientáció változását mutatja. Az orr környékéről kiinduló nyíl jelzi az orientáció változás irányát, a nyíl nagysága pedig a mozgás intenzitásával arányos.

[13] A koordináta-rendszer origója a fej középpontja a z tengely pedig a kifelé mutat a fejből – a fej közepéből az orr felé.

43. ábra. A fejmozgás megváltozásának grafikus reprezentációja.

Az időbeli orientáció változás alapján a fejmozgások időbeli behatárolása is lehetséges. Az alábbi listában definiált egyszerű állapotátmenet gépet használtam az időbeli behatároláshoz:

- **Mozog** állapotba megy: amikor **nem-mozog** állapotban van és az a_i normája egy előre rögzített küszöb szám felé ugrik.
- **Mozog** állapotban marad: addig ameddig az a_i normája az előre rögzített küszöb felett van.
- **Mozgás-vége** állapotba megy: amikor **mozog** állapotban van és az a_i normája az előre rögzített küszöb szám alá esik.
- **Nem-mozog** állapotba megy és ott marad: amikor **mozgás-vége** állapotban van.

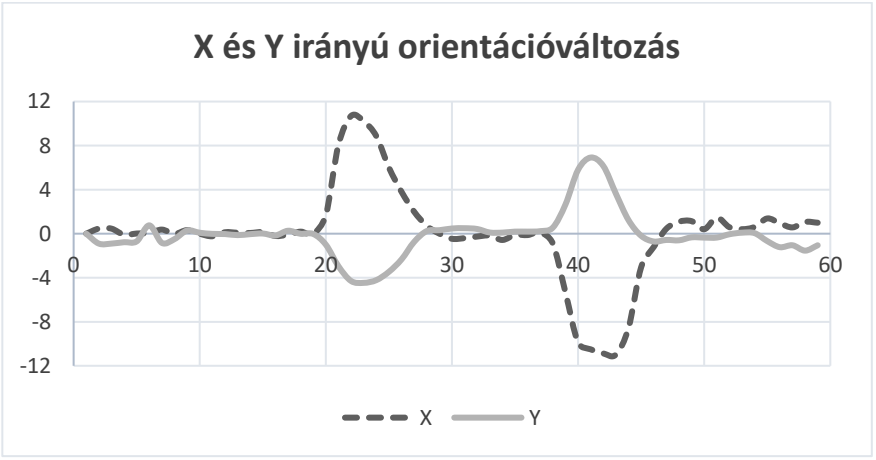
Az a_i normáját egy pár milliszekundomus ablak felett átlagoltam és hasonlítottam egy előre rögzített küszöbszámhoz – így kiszűrve a zajok okozta hirtelen kiugrásokat. Nyilvánvaló, hogy a fejmozgás akkor kezdődik amikor az állapotátmenet gép **mozog** állapotba megy és akkor ér véget, amikor **mozgás-vége** állapotot vesz fel. Az egyes a_i attribútumokat az így kijelölt intervallum felett rögzítjük a további feldolgozások céljából. A fejmozgások térbeli reprezentációjának és időbeli behatárolásának algoritmusát a 11. táblázatban definiált állapotátmenet géppel összegeztem.

1. Orientáció változásának meghatározása a megelőző időpillanathoz képest.
2. *Fejmozgás időbeli behatárolása* az orientáció változás alapján:
   - Ha a fejmozgás elindult, akkor GOTO 3. lépés,
   - Ha a fejmozgás véget ért, akkor GOTO 4. lépés,
   - Különben GOTO 1. lépés.
3. *Fejmozgás térbeli reprezentációja*: orientáció változás értékének tárolása egy alkalmas konténerben, majd GOTO 1. lépés.
4. *Hasonlóság meghatározása* az aktuális fejmozgás és néhány előzetesen rögzített fejmozgás között, majd GOTO 1. lépés.

11. táblázat. Fejmozgások behatárolása térben és időben

A 44. ábra demonstrálja az x és y irányú orientációváltozást az idő függvényében. A fejmozgás az alábbi komponensekből épül fel:

- A 20-as időbélyegig végig mozdulatlanul, egy helyben állt a fejem,
- 20-30 intervallum: átlós fejmozgást végeztem jobbra-fel irányba,
- 30-36 intervallum: egy helyben állt a fejem,
- 36-47 intervallum: átlós fejmozgást végeztem balra-le irányba,
- A 47-es időbélyeget követően újból mozdulatlanul állt a fejem.



44. ábra. Az *x* és *y* irányú orientációváltozás egy fejmozgás elvégzése során. A függőleges tengely reprezentálja az irányváltozást fokban, a vízszintes pedig az időt másodpercben.

5.3. Felismerés folyamata

Az előző fejezetben ismertettem a fejmozgások definícióját (5.9. egyenlet), de azok felismeréséről eddig még nem esett szó. Az alábbiakban ismertetem azt az eljárást mellyel két fejmozgás hasonlóságát határoztam meg. Ehhez egy DTW alapú eljárást dolgoztam ki [95, 96], továbbá egy saját készítésű fejmozgás adatbázist is létrehoztam és felhasználtam az egyes mozgásformák jellegzetességeinek kinyeréséhez és a robusztus felismeréshez.

5.3.1. Dinamikus idővetemítés

A DTW eljárást két eltérő ütemben kivitelezett mozgás (gyakorlatban idősor) hasonlóságának mérésére szokás használni. Lényegében két idősor lehető legjobb illeszkedését határozza meg. A DTW egy generatív modell: a $p(s, l)$ együttes eloszlást modellezi, ahol $s \in S$ a mozgás trajektóriájához tartozó idősor és $l \in L$ a mozgást leíró címke. Az együttes eloszlás ismeretében új mintákat is képes generálni, de ami fontosabb, hogy adott s esetén az l címke jósolható. Tehát egy $m \in \mathcal{M}$ fejmozgás felismerése során néhány ismert címkéjű idősor alapján szeretnénk egy ismeretlen s idősorhoz tartozó l címkét meghatározni.

A DTW segítségével két idősor között határozzuk meg az optimális illesztést, az idő tengely transzformálásával. Egy transzformációt definiálunk, amellyel a legoptimálisabban képezhető le az egyik idősor (*teszt idősor*) a másik idősornak (*referencia idősor*) egy részére, vagy egészére. A DTW egyfajta távolságfüggvényként is értelmezhető, mely ugyan *pozitív szemidefinit* és *szimmetrikus*, de nem feltétlenül teljesíti a háromszög egyenlőtlenséget, tehát egy *szemi-metrika*. Az illesztés pont-pont megfeleltetéseken alapul, melyhez szükségünk lesz egy alkalmas távolságfüggvényre az $\mathcal{F}$ a jellemzőtér felett. A továbbiakhoz tegyük fel, hogy van két idősorunk:

$$\begin{aligned} X &:= (x_1, x_2, \dots, x_N) \\ Y &:= (y_1, y_2, \dots, y_M)' \end{aligned} \tag{5.10}$$

ahol $X, Y \in S$ idősorok, továbbá az $x_i, y_j \in \mathcal{F}$ a jellemzőtér elemei és az egyszerűség kedvéért az $i \in [1{:}N]$ indexet fogjuk használni az X idősor esetén és a $j \in [1{:}M]$ indexeket az Y esetén, valamint $M, N \in \mathbb{N}$. Ahogy

említettem, szükséges egy $c: \mathcal{F} \times \mathcal{F} \rightarrow \mathbb{R}_{\geq 0}$ távolságfüggvény definiálása az x_i és y_j megfigyelések közötti hasonlóság mérésére[14]:

$$C(i,j) = c(x_i, y_j) \geq 0. \tag{5.11}$$

A $c(x_i, y_j)$ függvény értéke annál alacsonyabb, minél inkább hasonlít x_i az y_j-hez. A két idősor optimális illesztését az x_i és y_j megfigyelések közötti c távolságfüggvény (mint költségfüggvény) minimalizálásával határozzuk meg. Az x_i és y_j mennyiségek csak a c függvény paramétereiként vesznek részt közvetlenül a számításban, tehát az eljárás alkalmazható azokban az esetekben is, amikor az X és Y idősorok egy-, vagy többváltozósak, folytonosak stb. Az egyetlen feltétel az, hogy definiálható legyen közöttük egy c távolságfüggvény a megfigyelések közötti hasonlóság mérésére. A c távolságfüggvény alapján, az x_i és y_j megfigyelléspárokra meghatározzuk a $C \in \mathbb{R}^{N \times M}$ távolság-, vagy költségmátrixot. A továbbiakban ennek a C mátrixnak kívánjuk előállítani egy minimális költségű útját, mely egyben a két idősor optimális illesztését is jelenti. A minimális költségű út előállítása egy $w(t)$ *vetemítő görbe* meghatározását jelenti. A görbét *vetemítő függvények* által definiáljuk, melyek az idősorok n és m indexei között teremtenek kapcsolatot úgy, hogy egy közös, normalizált időtengelyre vetítik azokat. A vetemítő függvények tehát az eredeti idősorokat transzformálják át oly módon, hogy az eredményül kapott, úgynevezett vetemített idősorok annyira hasonlóak legyenek, amennyire az csak lehetséges

Megjegyzem, hogy $w(t)$ előállítása lehetséges lineáris illesztéssel (pl. összetartozó időbeli események megfeleltetésével), sőt akár korreláció maximalizálással is. A DTW ezzel szemben egy optimalizálási probléma megoldásának tekinti azt. A fentebb említett $w_x(t)$ és $w_y(t)$ vetemítő függvények az alábbi formában definiálhatók:

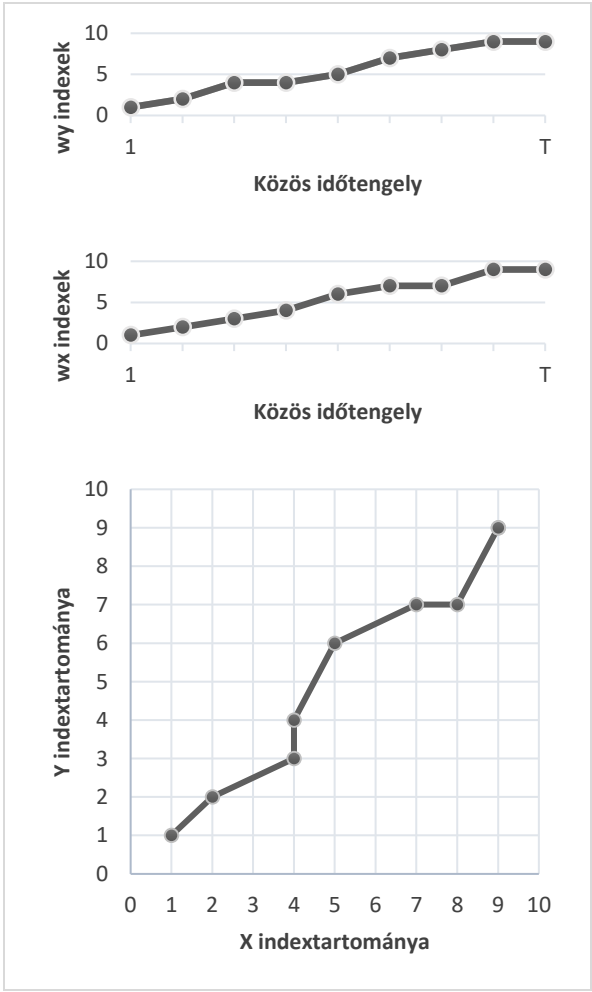
$$\begin{aligned} i &= w_x(t) \in [1:N] \\ j &= w_y(t) \in [1:M]\text{'} \end{aligned} \tag{5.12}$$

[14] A c függvény általában az euklideszi távolság, de más metrika is használható. Gyakran költségfüggvénynek is hivatkozzák.

ahol, $t \in [1:T]$ a közös tengely indextartománya. A $w(t)$ vetemítő görbe definíció szerint legyen a fenti egyenletben megadott függvények rendezett párja:

$$w(t) = \Big(w_x(t), w_y(t)\Big). \tag{5.13}$$

A 45. ábra szemlélteti az eddig leírtakat. A felső két grafikonon a $w_x(t)$ és $w_y(t)$ vetemítő függvényekkel közös időtengelyre transzformált idősorok láthatók, továbbá az alsó ábrán látható rácson egy vetemítő görbét vettem fel a két idősor között.



45. ábra. Két idősor normalizálása közös időtengelyre. A felső két grafikon a két idősort ábrázolja a közös időtengelyen. A lenti grafikonon pedig egy vetemítő görbe látható.

A vetemítő görbéhez tartozó rácspontok felcímkézhetők a $t \in [1:T]$ index-szel. T tehát a normalizált hossza a két idősornak (a normalizált időtengelyen). Mivel a vetemítő függvények közös időtengelyre transzformálják az idősorokat, így a használatukkal képesek vagyunk definiálni a hasonlóság mértékét az X és Y idősorok között (mely egyfajta akkumált torzulásként is felfogható a két idősor között):

$$d_w(X, \mathrm{Y}) = \frac{1}{M_w} \sum_{t=1}^{T} C\left(w_x(t), w_y(t)\right) m_w(t). \tag{5.14}$$

A fenti egyenletben szereplő a $C\left(w_x(t), w_y(t)\right)$ kifejezés az $x_{w_x(t)}$ és $y_{w_y(t)}$ megfigyelések távolsága a közös időtengely t indexénél; az $m_w(t)$ mennyiség egy nem-negatív súlyozási együttható; és az M_w pedig egy normalizációs konstans (általában $M_w = N + M$). A 45. ábra alsó grafikonján, a vetemítő görbe rácspontjaira számoljuk ki a $C\left(w_x(t), w_y(t)\right)$ kifejezés értékét. Az $m_w(t)$ és M_w mennyiségekre azért van szükség, mert több vetemítő görbe is létezhet a két idősor között, nekünk viszont azt kell előállítani, amelyiknek minimális a költsége, így az egyes vetemítő görbéket súlyozni és normalizálni kell ahhoz, hogy azokat össze tudjuk hasonlítani egymással. Említettem, hogy több vetemítő görbe lehetséges két idősor között. A nem-valódi vetemítő görbék kiszűrése érdekében néhány peremfeltételt szokás alkalmazni a vetemítő függvényekre, pl. elvárjuk, hogy legyenek monoton növekvők:

$$\begin{aligned} w_x(t+1) &\geq w_x(t) \\ w_y(t+1) &\geq w_y(t) \end{aligned}. \tag{5.15}$$

A minimális költségű vetemítő görbe, a két idősor optimális illesztését jelenti. Legyen w az optimális vetemítő görbe, ekkor igaz rá az, hogy az összes vetemítő görbe közel w rendelkezik a minimális akkumulált torzulással (lásd 5.14. egyenlet) az X és Y idősorok között:

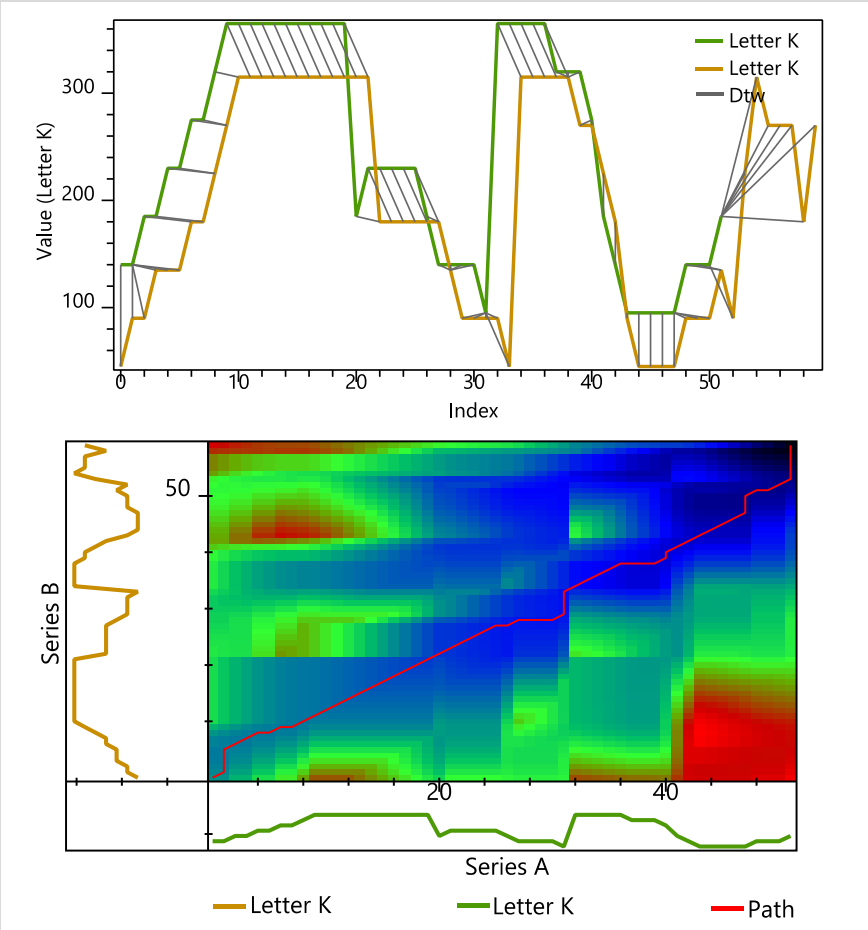
$$D(X, Y) = \min_{w} d_w(X, Y). \tag{5.16}$$

Tehát ha az optimális vetemítő görbéhez tartozó vetemítő függvényekkel közös időtengelyre vetítjük az X és Y idősorokat, akkor a vetített idősorok annyira hasonlóak lesznek, amennyire az csak lehetséges. Az 5.16 egyenletben megfogalmazott optimalizálási feladat keresési tere meglehetősen nagy, de a

DTW segítségével mégis megoldható $O(N \cdot M)$ idő alatt, ehhez azonban szükség van bizonyos optimalizációra. Én most csak röviden foglalom össze a technikákat, a bővebb kifejtésük megtalálható a következő tanulmányban: [96]. Számos optimalizálása létezik a DTW-nek, de többnyire mindegyik azt a kérdést válaszolja meg, hogy: *Milyen tulajdonságokkal kell rendelkeznie egy optimális vetemítő görbének*? A legnépszerűbb eljárások az alábbi feltételezésekből és korlátozásokból indulnak ki a vetemítő görbét illetően [97]:

- *Monotonitás*: a görbe nem „fordulhat vissza" önmagába, nem keletkezhetnek hurkok stb.
- *Folytonosság*: mind függőleges, mind vízszintes irányban egyet léphetünk a görbe előállítása során.
- *Korlátosság*: a vetemítő görbének a bal alsó sarokból kell indulnia és a jobb felső sarokban kell végződnie.
- *Szabályozó ablak*: egy optimális vetemítő görbe az átlóhoz közel helyezkedik el, így a keresési tér korlátozható az átlóhoz közeli területre.
- *Lejtés*: a vetemítő görbe nem lehet túl meredek, vagy túl lapos. Ha ez a korlátozás nem teljesül, akkor előállhat az az eset, hogy egy nagyon rövid idősor hasonlít egy nagyon hosszúra.

Összegezve a leírtakat, a DTW bemenete egyedül az x_i és y_j megfigyeléspárokra kiszámolt a $C \in \mathbb{R}^{N \times M}$ költségmátrix. A kimenete abból a szempontból előnyös a számunkra, hogy a $D(X, Y)$ függvény (melyet gyakran DTW távolságnak is neveznek) segítségével egy alkalmas sebesség/tempó invariáns hasonlósági mértéket definiálhatunk az egyes idősorok között. Ez a hasonlósági mérték az, ami lehetővé teszi az idősorok klaszterezését és osztályozását – ami esetünkben a fejmozgások felismerését jelenti. A 46. ábra két fejmozgás optimális illesztését illusztrálja. A fejmozgás reprezentációja egy korábbi cikkem alapján történt [8], melyet könnyebb grafikonon szemléltetni, mint a jelen értekezés alapjául vett reprezentációt.



46. ábra. Két idősor egymáshoz illesztése. A két idősor, két k betű trajektóriájához tartozik, melyeket *zöld* és *sárgás-barna* színnel jelöltem az ábrákon. Az ábra alsó felében illusztráltam a költség mátrixot, *piros* színnel jelöltem az optimális vetemítő görbét.

A 46. ábra két darab k betűhöz tartozó idősor optimális vetemítét demonstrálja. A felső ábrán szürke vonallal kötöttem össze a két idősor azon indexeit, melyeket a vetemítő függvények kiválasztanak az optimális vetemítő görbe előállítása során. Az ábra lenti grafikonja illusztrálja az illesztéshez számolt $C \in \mathbb{R}^{N \times M}$ költség mátrixot. A mátrixban futó piros görbe a két idősor között fennálló optimális költségű w vetemítő görbe. A költség mátrixban kék-zöld-piros átmenettel színeztem a távolságokat, ahol minél pirosabb egy cella, annál nagyobb a hozzájuk tartozó x_i és y_j megfigyelésekre számolt távolság értéke. Látható, hogy a w minden esetben a legkékebb cellák

felett fut, azaz ez valóban ez a legkisebb költségű út. Mivel w nagyon közel helyezkedik el a mátrix főátlójához, így az mondható el, hogy a két idősor nagyon hasonlít egymáshoz.

5.3.2. Fejmozgás adatbázis és felismerés

A DTW segítségével két idősor között definiálhatunk hasonlósági mértéket, én viszont ennél jóval több fejmozgást szerettem volna felismerni. A probléma megoldását egy DTW-n alapuló osztályozó elkészítése jelentette. A HAR rendszer alapos vizsgálatához egy 26 osztályos osztályozót készítettem. Az osztályokat jelöljük $M_i \subset \mathcal{M}$ -vel, ahol $\mathcal{M}$ a fejmozgások halmaza és $i \in [1:26]$. Az M_i osztályaink az angol ABC fejjel leírt betűinek trajektóriái voltak, tehát az M_i osztályok olyan fejmozgásokat tartalmaznak, mint pl. egy p, vagy h betű fejjel történő leírása:

$$M = \{M_i\}_{i\in[1:26]}. \tag{5.17}$$

Az egyes osztályok diszjunktak és nem üresek. Az osztályok magas száma és változatossága komoly kihívás elé állított. Egyrészt általános elvárás az osztályozó felé, hogy az jól általánosítson, így osztályonként több tíz, esetleg száz fejmozgásra van szükség a tanítás során. Szükség van tehát egy fejmozgás adatbázisra. Másrészt, mivel változó hosszúságú idősorokat akarok osztályozni, így eléggé be van határolva az alkalmas osztályozó algoritmusok köre. A DTW távolság miatt kialakítható egy térbeli rendezettség az idősorok között. Feltételeztem, hogy a hasonló fejmozgások DTW távolság alapján közel helyezkednek el egymáshoz, így akár egy konvencionális k-NN (*k-nearest neighbors*) osztályozó is illik a problémához. Ezzel azonban az a probléma, hogy a többségi szavazás miatt minden bejövő teszt mintára elő kell állítani a hozzá legközelebb álló k darab többi mintát (a már rendelkezésre álló adatbázis elemei közül). Majd többségi szavazás alapján határozzuk meg a bemenő minta címkéjét, feltételezve, hogy a bemenő minta tulajdonságai közel azonosak a hozzá legközelebb álló k darab mintáéval. Mivel a térbeli rendezettség a DTW távolságon alapul, így a bejövő teszt mintára ki kell számolni a DTW távolságot az adatbázis összes többi mintájával. Ha így járunk el, akkor egy teszt minta osztályozásához több 100, esetleg 1000 darab költségmátrixot kellene kiszámítani, ami rendkívül költséges feladat és ellentmond a valósidejű felismerés kritériumának. A felismerés folyamatát megpróbáltam annyira leegyszerűsíteni, amennyire az csak lehetséges.

Bizonyos elvek mentén lecsökkentettem az M_i osztályok számosságát 1-re, majd egy 1-NN osztályozót hoztam létre. Így a bejövő fejmozgásokra annyi költségmátrixot kell kiszámolni, ahány osztályunk van az adatbázisban – esetünkben 26 darabot.

Az 1-NN osztályozó azt jelenti, hogy bármely bemenő fejmozgás tulajdonságai hasonlóak a környezetében hozzá legközelebb eső fejmozgás tulajdonságaihoz. Az osztályok számosságának csökkentéséhez lehetséges lett volna osztályonként egy-egy átlagos idősort meghatározni, pl. [98, 99]. Én azonban inkább a meglévő fejmozgásokból választottam ki minden osztályból pontosan egyet: azt, amelyik a legjobban reprezentálja az adott osztály összes elemét, jelöljük ezt az elemet $m_i^r \in M_i$-vel. Tehát 26 darab m_i^r fejmozgás meghatározása a cél. Az adatbázis összeállítása során a mintákat 10 különböző embertől gyűjtöttem össze, mindenki 20 darab fejmozgást végzett osztályonként, így 200 db fejmozgást kaptam minden egyes betűre. Osztályonként 100 darab fejmozgást használtam fel arra [15], hogy meghatározzam az m_i^r fejmozgást, így az $r \in [1:100]$ index a fejmozgás M_i osztályon belüli sorszámát jelenti. Az m_i^r elemtől azt vártam, hogy a legjobban szeparálja a hozzá tartozó M_i osztályt a többitől. Az m_i^r elem meghatározása során kiszámoltam az M_i osztályon belül az összes lehetséges elempár DTW távolságát és azt az elemet választottam m_i^r-nek, amelyik minimalizálta az osztályon belüli távolságot:

$$m_i^r = \min_D D\left(m_i^a, m_i^b\right), \tag{5.18}$$

ahol $m_i^a, m_i^b \in M_i$. Értelemszerűen a fejmozgások címkéit kihagytam a DTW távolság kiszámításából, valamint $a, b \in [1:100]$ és $a \neq b$. A gyakorlatban m_i^r egyfajta klaszter középpontként is értelmezhető – gyakran én is így fogok rá hivatkozni az alábbiakban. Az m_i^r elemek megtekinthetők az 53. ábra (lásd Függelék D).

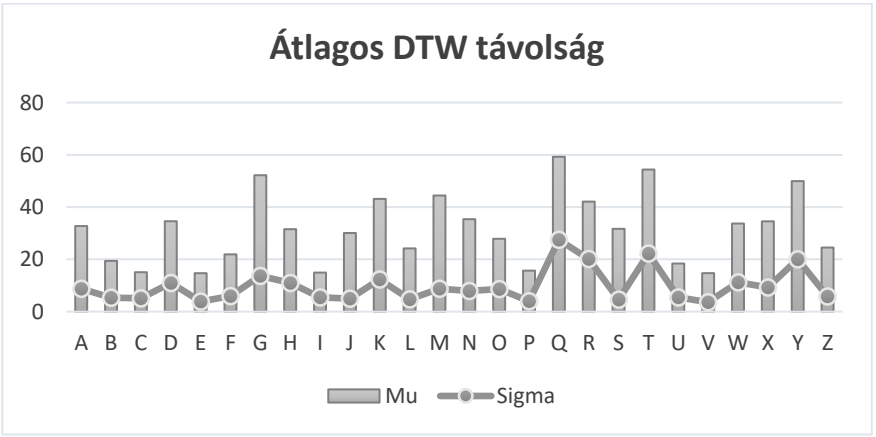
[15] A maradék fejmozgásokat kizárólag tesztelési célokra használtam fel.

5.4. Gyakorlati eredmények

A teszteredmények részletes ismertetése előtt célszerű megnézni az egyes klaszter középpontok tulajdonságait, hiszen ezek azok az elemek, amelyek minimalizálják az M_i osztályokon belüli DTW távolságokat az elemek között. A 47. ábra illusztrálja a klaszter középpontok átlagos távolságát (μ_i) és korrigált tapasztalati szórását (σ_i) a többi elemhez képest M_i-n belül, azaz definíció szerint:

$$\mu_i = \frac{1}{100}\sum_{j=1}^{100} D\left(m_i^r, m_i^j\right)$$
$$\sigma_i = \sqrt{\frac{1}{99}\sum_{j=1}^{100}\left(D\left(m_i^r, m_i^j\right) - \mu_i\right)^2}. \tag{5.19}$$

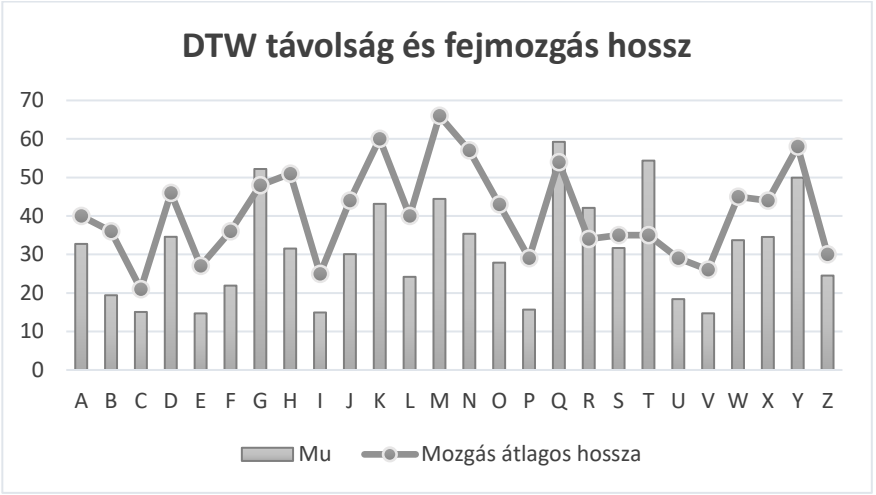
A 47. ábra alapján azonban csak közvetett feltevéseink lehetnek a rendszer pontosságára, mivel a lentebbi ábrából közvetlenül csak annyi látszik, hogy az osztályok térbeli kiterjedése elég változatos[16]. Például az e, vagy v betűk esetén a klaszterközéppontok nagyon közel helyezkednek el a többi ponthoz. Tehát ezeknek a klaszteréknek az összes pontja egy kis sugarú körben helyezkedik el. Ezzel szemben például a t, vagy q betűk esetén sokkal nagyobb DTW távolságokat kaptam a többi elemhez képest, így ezeknek az osztályoknak az elemei közel háromszor akkora sugarú körben helyezkednek el, mint az előzőek.



47. ábra. Átlagos DTW távolság és szórás az osztályokat reprezentáló elem és a többi elem között.

[16] Az ábrán az osztályok neveit nagybetűvel jelöltem az olvashatóság kedvéért.

Ebből akár már a felismerés pontosságára is lehetne következtetni: hihetnénk azt, hogy kevésbé lesz pontos a fejmozgások felismerése, azon osztályok esetén, amelyek magas μ_i és σ_i értékkel rendelkeznek, hiszen ezen osztályok a nagy térbeli kiterjedésük miatt akár átfedésben is lehetnének más osztályokkal. További vizsgálataim során azonban a következőt tapasztaltam: a μ_i és σ_i értékek az osztályon belüli idősorok átlagos hosszával korrelálnak. Az idősorok átlagos hossza alatt az azokat alkotó megfigyelések darabszámának átlagát értjük. A felismerés során a fejtartás mintavételezése 50 ms-ként történik, így a fejmozgások időbeli hossza: 1,5 – 3,5 másodperc. A 48. ábra alapján a magas μ_i és σ_i érték inkább a DTW jellegéből fakad. Azaz nagyobb μ_i és σ_i érték várható olyan osztályok esetén, melyek hosszabb idősorokat tartalmaznak, de korántsem biztos, hogy a felismerés pontossága rosszabb lesz ezen osztályok esetén.

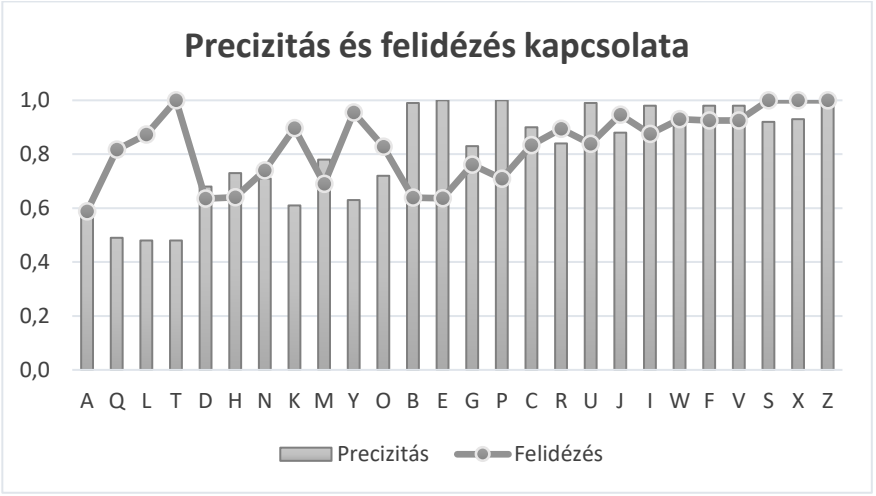


48. ábra. Az átlagos DTW távolság és az átlagos mintahossz korrelációja. Látható, hogy minél hosszabbak a minták egy osztályon belül, annál nagyobb az m_i^r elem átlagos DTW távolsága a többi elemhez képest.

A módszer részletes tesztelése egy független adatbázison történt, ahol az egyes M_i osztályok ugyanúgy 100 darab elemet tartalmaztak, mint az m_i^r elem meghatározása során használt adatbázis. A tesztadatbázis összes m elemére ($i \in [1:26]$ és $j \in [1:100]$) meghatároztam a $D(m_i^r, m)$ távolságot, majd az m fejmozgást abba az osztályba soroltam be, amelyre a minimális D távolság adódott. Mivel az m valódi címkéje ismert – az adatbázis annotált –, így a

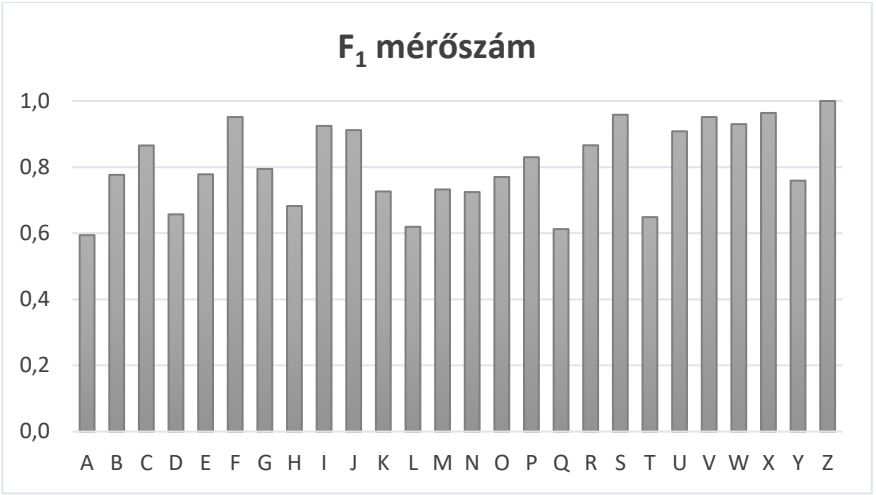
hipotézisvizsgálat során osztályonként mérhető az első-, és másodfajú hiba mértéke. A hipotézisvizsgálatról már beszéltem az előzőkben, így most nem részletezem. Ugyanakkor megjegyzem, hogy a magas felidézéssel rendelkező osztályozók kevés pozitív esetet osztályoznak tévesen. Ezzel szemben a magas precizitású osztályozók kevesebb a hamis találatok eredményeznek.

A 49. ábra illusztrálja az osztályonként meghatározott precizitás és felidézés értékeket. Az ábra alapján, a 26 darab osztályra számolt precizitás átlagos értéke 0,81 és az átlagos felidézés pedig 0,83, amelyek garanciát jelentenek mind a hamis pozitívok, mind a hamis negatívok alacsony darabszámára a felismerés során. Látható azonban, hogy az a, k, l, q, t és y osztályok esetén relatív sok hamis pozitív találatra kell számítani, ezeknél az osztályoknál a precizitás értéke 0,56. Ennek analógiájára sok hamis negatív találatra kell számítani az a, d, e és h osztályok esetén, ezeknél a felidézés értéke 0,63.



49. ábra. A precizitás és felidézés kapcsolata az egyes osztályok esetén.

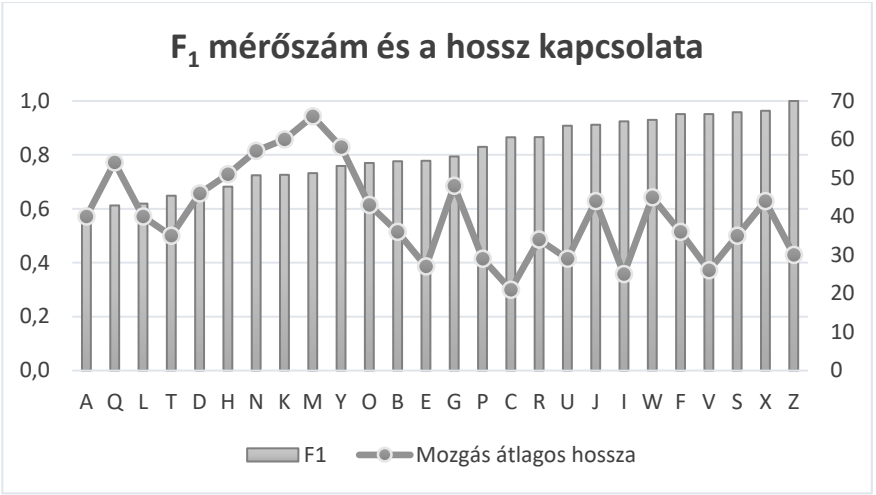
A gyakorlatban azonban az a cél, hogy mind a precizitás, mind az felidézés értékét maximalizáljuk. Az F_1 mérőszám a precizitás és a felidézés összefoglalására szolgál. Az F_1 mérőszám a harmonikus középértéket reprezentálja a két mutató között.



50. ábra. A tesztadatbázis osztályaira számolt F_1 mérőszám. Minél magasabb az adott osztályra számolt érték, annál pontosabb az osztályozó.

A 50. ábra illusztrálja a tesztadatbázis osztályaira számolt F_1 metrikát. Az F_1 mérőszám alapján a 10 legkisebb hibával felismerhető fejmozgásra az r, u, j, i, w, f, v, s, x és z osztályok esetén számíthatunk. Külön érdekesség, hogy a z osztály esetén mind a 100 db fejmozgás helyesen lett felismerve. Ellenben a 10 legnagyobb hibával felismerhető fejmozgásra az a, q, l, t, d, h, n, k, m és y osztályok esetén számíthatunk. A legnagyobb hibával felismerhető fejmozgásoknál a probléma okát a más osztályok mozgásformáival való hasonlóságokban kell keresni. Pl. a betűt hasonlóan írjuk, mint d-t, de ugyanígy hasonlóság mutatkozik pl. az $m - n$, vagy $l - e$ betűk esetén.

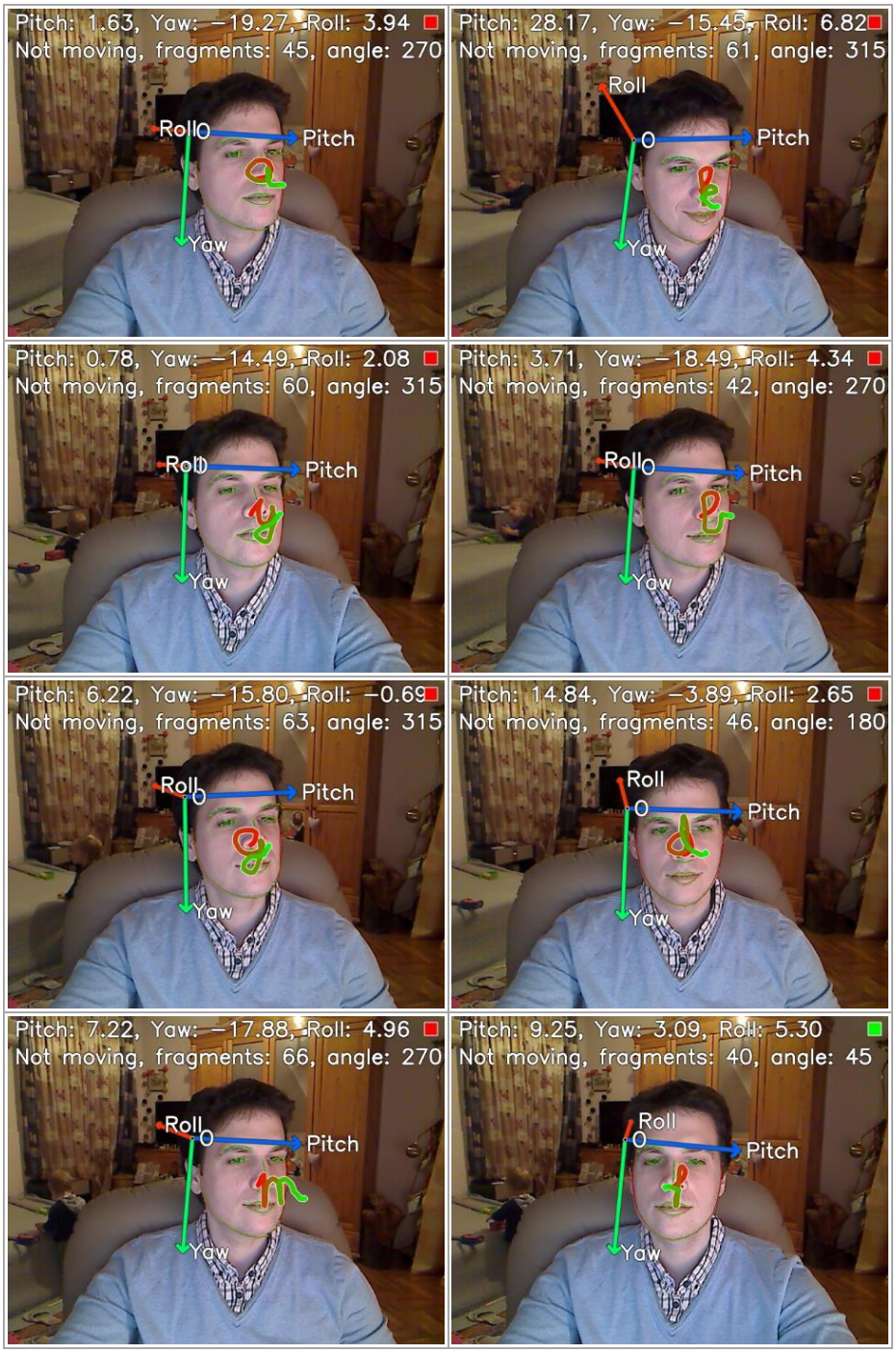
Megvizsgáltam továbbá, hogy van-e korreláció a fejmozgások átlagos hossza és az F_1 mérőszámuk között. A 51. ábra alapján kijelenthető, hogy nem függ össze a két mérés, rövidebb minták felismerése is történhet magas pontossággal (pl. i, v és z), illetve hosszabbaké is alacsonyabb pontossággal (pl. q, h és m) és vice versa. Mindent egybevetve láthatjuk, hogy a rendszer pontossága az osztályok felére több mint 80%-on felüli (mind a precizitást, mind a felidézést tekintve), ami mindenféleképpen jó eredménynek számít és bíztató a jövőre nézve.



51. ábra. A tesztadatbázis osztályaira számolt F_1 mérőszám és az átlagos mintahossz kapcsolata. Látható, hogy a két mérés nem korrelál egymással.

Kevesebb osztály, esetleg más reprezentáció segítségével tovább növelhető a pontosság, továbbá érdemes lehet bevezetni egy null-osztályt is, melynek a feladata az egyik osztályba sem tartozó gesztusok elkülönítése lenne. A null-osztály bevezetése a klaszterközéppontokra számított μ_i és σ_i érték birtokában viszonylag könnyen megvalósítható. Ha egy újonnan bejövő minta és a hozzá legközelebb eső klaszterközéppont DTW távolsága nagyobb $\mu_i + c\sigma_i$, akkor nem fogadjuk el a mintát. Az előző egyenletben a c egy alkalmas konstans szorzó. Jelen értekezés keretein belül nem végeztem részletesebb vizsgálatokat ebben az irányban, mivel eredményeim alapján úgy ítéltem meg, hogy a kifejlesztett rendszer alkalmas arra, hogy igazolja az eredeti célkitűzéseimet – azaz, hogy képes legyek *komplex fejmozgásokat*, *nagy pontossággal*, *valós időben felismerni*.

A fejezet zárszavaként, a 52. ábrán néhány példával illusztrálnám a fejmozgások felismerését. A fejmozgások, mint idősorok grafikus megjelenését demonstráltam, lényegében ezek azok az adatok, amiből a felismerés dolgozik. Piros-zöld átmenettel ábrázoltam a mozgásokat, ahol minél pirosabb a trajektória színe, annál régebbi a mozgás a trajektória azon részén. Ismét hangsúlyozom, hogy a trajektória előállítása, csak a fej irányultságának pillanatnyi állapotától függ.



52. ábra. A fejmozgás felismerés néhány pillanata.

6. Összefoglalás

Jelen értekezésben az ember és számítógép közötti kapcsolatokkal foglalkoztam. Elsődleges célom a kommunikációban szereplő emberi fél vizsgálata volt, olyan módszerek gyakorlati megvalósítását helyeztem fókuszba, melyek alapot teremthetnek a HCI változatos fajtáinak, és amelyek természetesebbé tehetik a HCI-t az emberi fél számára. Az értekezés elején az emberi fejből és arcból kinyerhető nonverbális jelek és jellegzetességek kinyeréséből, reprezentálásából, valamint ezek felismeréséből indultam ki. Céljaim középpontjában mindvégig az ember-számítógép kommunikáció technológiájának a továbbfejlesztése állt.

Az értekezés elején ismertettem, hogy egy kameraképből kiindulva hogyan valósítható meg a HCI rendszerem alapját képező *arc lokalizációja*. Ismertettem az egyik legkorszerűbb objektumdetektor (Viola-Jones) működését és korlátait az emberi arcok lokalizációja kapcsán. Tanítások és tesztek hosszú sorozatán át javaslatot tettem az objektumdetektor optimális paraméterezésére. Vizsgáltam a *tanító adatbázisok összetételét*, a *tanításhoz szükséges minták darabszámát*, *a tanítóminták felbontását*, *a kaszkád struktúra szintjeinek számát*, *a jellemzőkinyerő algoritmusok lehetséges körét* (Haar, LBP és HoG) és a *boosting algoritmusok típusát* (DAB, RAB és GAB). Egyszerre vizsgáltam ezek hatását a teljes rendszer pontosságára, valamint futásidejére vonatkozóan. Úgy találtam, hogy az általam tanított legjobb detektor segítségével képes lehetek változatos orientációjú arcokat detektálni viszonylag alacsony futásidő mellett. Ismertettem az általam létrehozott *arckövető és azonosító rendszeremet*, mely segítségével jelentősen lecsökkentettem az arc lokalizációjához szükséges futásidőt. Továbbá a rendszer segítségével metaadatokat is rendeltem a kamera képeken látható arcokhoz további feldolgozások céljából.

Az arc lokalizálását követően az *arc belső modellezésével*, azaz a *karakterisztikus pontok kinyerésével* foglalkoztam. Három népszerű statisztikai alak-, és textúra modellező eljárást ismertettem: ASM, AAM és CLM. A fejezet kimenete és eredményterméke egy 2-D archáló lett, mely határpontjai az arcnak egyes meghatározó pontjait reprezentálták, mint pl. szemek-, vagy szemöldökök sarkainak pontjai, szemöldökök és száj körvonalának bizonyos pontjai. Átfogó tesztesetek során értékeltem ki a módszereket a gyakorlatban. Céljaim itt is hasonlóak voltak, mint az arc lokalizálása során: *magas pontosság* mellett akartam kinyerni a karakterisztikus pontokat, *valós időben*, *változatos fejtartások* és

arckifejezések mellett. Az átfogó tesztek során sikeresen előállítottam egy konfigurációt, mely az eredményeim alapján kiszolgálja az előbb megfogalmazott elvárásokat és magát a rendszert a gyakorlatban.

A 2-D archáló ismeretében kísérletet tettem a *fej térbeli helyzetének felismerésére*. Egy 6DoF modellként értelmeztem az emberi fejet a 3-D valós világnak megfelelő térben. Ebben a 3-D térben határoztam meg a fejnek a három tengely mentén vett eltolását és a három tengely körül értelmezett elforgatását. A problémára együttesen, mint fejtartás felismerés hivatkoztam. A fejtartás felismerését egy PnP probléma definiálásával és levezetésével ültettem át a gyakorlatba. Definiáltam egy geometriailag helyes 3-D fejmodellt és a 3-D-2-D pontmegfeleltetések alapján számítottam ki a fejtartást. Maga az eljárás a perspektív vetítés közelítésének is tekinthető. A módszer pontosságának mérése közel sem volt kézenfekvő. Több adatbázison, több tesztet futtattam le, hogy körül határoljam a felismerés pontosságát. Végül úgy találtam, hogy *kellően nagy tartományban*, *kellően nagy pontosság* érhető el a fejtartás felismerése, így a módszer alapját képezheti a további feldolgozásaimnak.

Végezetül, az utolsó fejezetben nonverbális jelek felismerésével foglalkoztam. E jelek tipikus megnyilvánulása számomra a fejmozgás volt. Az előzetes eredményeimre támaszkodva tettem javaslatot a fejmozgások felismerését megcélzó HAR rendszeremre. A fejmozgások felismerése alatt emberi tevékenységek felismerését kell érteni. Első körben megvizsgáltam, hogy miből célszerű építkezni a felismerés során. Javaslatot tettem a fejmozgások térbeli reprezentációjára, mint az arc függőleges és vízszintes tengelyei körüli elforgatás függvényére. Ismertettem a fejtartásban bekövetkező hibás, kiugró értékek kiszűrésének algoritmusát, továbbá a fejmozgások időbeli behatárolását is. Végezetül egy *dinamikus idővetemítésen* (DTW) és egy *fejmozgás adatbázison* alapulva ismertettem a fejmozgások osztályozását. A módszer pontosságának igazolására egy 26 osztályos osztályozót hoztam létre, ahol az egyes osztályok az angol ABC kisbetűinek fejjel leírt trajektóriái voltak. Az átfogó tesztek alapján úgy találtam, a javasolt módszer alkalmas a változatos és komplex fejmozgások felismerésére, kellően magas pontosság mellett, valós időben.

7. Summary (összefoglaló angolul)

In this dissertation, I elaborated on some aspects of human-computer interactions (hereinafter HCI). I focused on the practical implementation of HCI that could provide a basis for making the interactions for the human party more natural as it was. At the beginning of the dissertation, I originated from the extraction, representation and recognition of nonverbal signs – extracted from the human head or face. The focal point of my goals was the development of human-computer communication technology.

At the beginning of my dissertation, I explained the way of *face localization* as the basis of my HCI system from camera image. I described the operation and limitations of the state-of-the art object detection system (Viola-Jones) in relation to the localization of human faces. Through a long series of trainings and tests, I proposed the optimal parameterization of the object detector. I carried out an investigation related to the *composition of training database*, the *number of training samples*, the *resolution of training samples*, the *number of stages in cascade hierarchy*, the effect of *feature extraction* (Haar, LBP and HoG) and *boosting* (DAB, RAB and GAB) methods. I analyzed the effect of the previous parameters in together for the system's accuracy and runtime. I found that faces could be detected with various orientations and low runtime with the best detector trained by myself. I also introduced a subsystem for *face tracking* and *user identification* in order to make face localization real-time. Additionally, with this subsystem, I assigned metadata to faces for further processing.

After the face localization, I proposed a system for modeling the *internal structure of human faces*, in other words the *extraction of facial feature points*. I described three popular statistical shape and texture modeling methods: ASM, AAM and CLM. The output and end product of this chapter was a 2-D shape of the face, whose boundary points represented certain characteristic points of the faces such that the corner points of the eyes or eyebrows, some contour points of the mouth, etc. During a comprehensive test case, I evaluated the methods in practice. My goals were similar here as during the face localization: I wanted to obtain the feature points in high accuracy, real-time and with varied head poses and facial expressions. During the comprehensive tests, I have successfully created a configuration (based on my results) that serves my expectations and the system itself in practice.

In the knowledge of the 2-D face shape, I attempted to *recognize the head pose*. I interpreted the human head in the 3-D space as a 6DoF model. In this space, I determined the shift of the head along the three axes and the rotation of the axis around the three axes. I referred to the problem as head pose recognition. Recognizing head pose was implemented by defining a PnP problem. I defined a geometrically correct 3-D head model and calculated the head pose based on 3-D-2-D point correspondences. The process itself can be considered as the approximation of perspective projection. Measuring accuracy of the method was not as easy as it was. I ran several tests on multiple databases to delimit the recognition's accuracy. Finally, I found that the head pose can be recognized in a sufficiently large range with high precision, so my method can be the basis of my further developments.

Finally, in the last chapter I elaborated on the recognition of nonverbal signs. The typical manifestation of these signs were head movements. Based on my preliminary results, I proposed a HAR system for *detecting head movements*. The recognition of head movements was interpreted as the recognition of human activities. First of all, I determined the set of atomics where the recognition was worth building from. I proposed a spatial representation of head movements in the function of rotations around vertical and horizontal axes of the face. I described an algorithm for filtering outliers as wrong head poses before the recognition. I also described the temporal localization of head movements. Finally, I provided a method for the classification of head movements based on *dynamic time warping* (DTW) and a *head movement database*. To prove the accuracy of the method, I created a classifier with 26 classes, where each class contained several head-drawn trajectories of the English alphabets lowercases. Based on comprehensive tests, I found that the proposed method is suitable for recognizing varied and complex head movements with high accuracy in real time.

Publikációk listája

Könyvfejezet (külföldi)

[P1] K. Bertók, A. Fazekas, "Gesture Recognition: Performance, Applications and Features," *Hauppauge, NY: Nova Science Publishers*, pp. 1-30, 2018, (megjelenés alatt).

Referált folyóiratcikk (külföldi)

[P2] K. Bertók, A. Fazekas, "Recognizing Complex Head Movements," *Australian Journal of Intelligent Information Processing Systems*, vol. 14, no. 4, pp. 3-17, 2016.

[P3] L. Sajó, K. Bertók, A. Fazekas, "Perceptual Color Representation of the Face: Extracting the Color of Skin, Hair and Eyes," *JPRR* vol. 6, no. 2, pp 166-174, 2011.

Konferenciakötetben megjelent cikk (külföldi)

[P4] K. Bertók, A. Fazekas, "Face Recognition on Mobile Platforms," *7th IEEE International Conference on Cognitive Infocommunications*, pp. 37-43, 2016.

[P5] K. Bertók, A. Fazekas, "Recognizing Human Activities Based on Head Movement Trajectories," *5th IEEE Conference on Cognitive Infocommunications*, pp. 273-278, 2014.

[P6] K. Bertók, A. Fazekas, "Gesture Recognition - Control of a Computer with Natural Head Movements," International Conference on Computer Graphics Theory and Applications and International Conference on Information Visualization Theory and Applications, pp. 527-530, 2012.

[P7] L. Hunyadi, K. Bertók, T.E. Németh, I. Szekrényes, Á. Abuczki, G. Nagy, N. Nagy, P. Németi, A. Bódog, "The outlines of a theory and technology of human-computer interaction as represented in the model of the HuComTech project," *2nd Cognitive Infocommunications International Conference*, pp. 249-255, 2011.

Folyóiratcikk (hazai)

[P8] K. Bertók, L. Sajó, A. Fazekas, "A robust head pose estimation method based on POSIT algorithm," *Argumentum* vol. 7, pp. 348-356, 2011.

Konferenciakötetben megjelent cikk (hazai)

[P9] K. Bertók, A. Fazekas, "Fejmozgás alapú gesztusok felismerése," *Képfeldolgozók és Alakfelismerők országos 9. konferenciája*, pp. 736-746, 2013.

[P10] K. Bertók, A. Fazekas, L. Sajó, "Az emberi fej térbeli helyzetének és a tekintet irányának meghatározása," *Képfeldolgozók és Alakfelismerők 8. konferenciája*, pp. 393-406, 2011.

[P11] Fazekas, S. Szeghalmy, K. Bertók, L. Sajó, "Multi-modális ember-gép kapcsolatok," *Képfeldolgozók és Alakfelismerők 8. konferenciája*, pp. 387-392, 2011.

Egyéb

[P12] T. Balogh, K. Bertók, "Közgazdász vagy informatikus? - A gazdaságinformatikus képzés hallgatói megítélése a Debreceni Egyetemen," *Informatika a felsőoktatásban 2011*, pp. 110-119, 2011.

Tárgymutató

Irodalomjegyzék

[1] E. Hjelmås and B. K. Low, "Face Detection: A Survey," *Computer Vision and Image Understanding,* vol. 83, p. 236–274, 2001.

[2] M. H. Yang, D. Kriegman and N. Ahuja, "Detecting faces in images: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 1, pp. 34-58, 2002.

[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* vol. 1, pp. 511-518, 2001.

[4] F. C. Crow, "Summed-area Tables for Texture Mapping," *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques,* vol. 18, no. 3, pp. 207-212, 1984.

[5] Z. H. Zhou, Ensemble Methods: Foundations and Algorithms, Chapman and Hall/CRC, 2012, pp. 23-47.

[6] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences,* vol. 55, no. 1, pp. 119-139, 1997.

[7] R. Schapire and Y. Singer, "Improved Boosting Algorithms Using Confidence-rated Predictions," *Machine Learning,* vol. 37, no. 3, pp. 297-336, 1999.

[8] K. Bertók and A. Fazekas, "Recognizing Human Activities Based on Head Movement Trajectories," *5th IEEE Conference on Cognitive Infocommunications (CogInfoCom), 2014,* pp. 273-278, 2014.

[9] Neowise Software, "NeoDownloader, Download and view pictures & videos from websites," 22 January 2014. [Online]. Available: http://www.neodownloader.com/. [Accessed 2015].

[10] L. Hunyadi, K. Bertók, I. Szekrényes, Á. Abuczki, G. Nagy, N. Nagy, P. Németi and A. Bódog, "The outlines of a theory and technology of human-computer interaction as represented in the model of the HuComTech project," *2nd International Conference on Cognitive Infocommunications,* pp. 1-5, 2011.

[11] A. S. Georghiades, P. N. Belhumeur and D. J. Kriegman, "From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 23, no. 6, pp. 643-660.

[12] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar and I. Matthews, "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and

emotion-specified expression," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops,* pp. 94-101, 2010.

[13] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," *International Conference on Image Processing,* pp. 900-903, 2002.

[14] T. Ojala, M. Pietikainen and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 7, pp. 971-987, 2002.

[15] S. Liao, X. Zhu, Z. Lei, L. Zhang and S. Z. Li, "Learning multi-scale block local binary patterns for face recognition," *Proceedings of the 2007 International Conference on Advances in Biometrics,* pp. 828-837, 2007.

[16] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* vol. 1, pp. 886-893, 2005.

[17] M. Enzweiler and D. M. Gavrila, "Monocular Pedestrian Detection: Survey and Experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 31, no. 12, pp. 2179-2195, 2008.

[18] F. Suard, A. Rakotomamonjy, A. Bensrhair and A. Broggi, "Pedestrian Detection using Infrared images and Histograms of Oriented Gradients," *IEEE Intelligent Vehicles Symposium,* pp. 206-212, 2006.

[19] Q. Zhu, M. C. Yeh, K. T. Cheng and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* vol. 2, pp. 1491-1498, 2006.

[20] S. Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang and H. Shum, "Statistical Learning of Multi-view Face Detection," *Proceedings of the 7th European Conference on Computer Vision-Part IV,* pp. 67-81, 2002.

[21] T. Mita, T. Kaneko and O. Hori, "Joint Haar-like Features for Face Detection," *Proceedings of the Tenth IEEE International Conference on Computer Vision,* vol. 2, pp. 1619-1626, 2005.

[22] B. Wu, H. Ai, C. Huang and S. Lao, "Fast Rotation Invariant Multi-view Face Detection Based on Real Adaboost," *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition,* pp. 79-84, 2004.

[23] A. Torralba, K. P. Murphy and W. T. Freeman, "Sharing features: efficient boosting procedures for multiclass object detection," *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* vol. 2, pp. 762-769, 2004.

[24] A. Yilmaz, O. Javed and M. Shah, "Object tracking: A survey," *ACM Computing Surveys,* vol. 38, no. 4, 2006.

[25] K. Bertók és A. Fazekas, „Face recognition on mobile platforms," *7th IEEE International Conference on Cognitive Infocommunications,* pp. 37-43, 2016.

[26] I. Fasel, B. Fortenberry and J. Movellan, "A generative framework for real time object detection and classification," *Computer Vision and Image Understanding,* vol. 98, no. 1, pp. 182-210, 2005.

[27] P. Zhenyun, T. Linmi, X. Guangyou and Z. Hongjiang, "Detecting Facial Features on Images with Multiple Faces, Advances in Multimodal Interfaces," *Advances in Multimodal Interfaces - ICMI 2000,* vol. 1948, pp. 191-198, 2000.

[28] E. B. Sudderth, A. T. Ihler, W. T. Freeman and A. S. Willsky, "Nonparametric Belief Propagation," *In Proceedings of the 2003 IEEE Conference on Computer Vision and Pattern Recognition,* vol. 1, pp. 605-612, 2003.

[29] V. Erukhimov and L. Kuang-chih, "A bottom-up framework for robust facial feature detection," *8th IEEE International Conference on Automatic Face & Gesture Recognition,* pp. 1-6, 2008.

[30] K. Bertók, L. Sajó and A. Fazekas, "A robust head pose estimation method based on POSIT algorithm," *Argumentum,* vol. 7, pp. 348-356, 2011.

[31] K. Bertók, A. Fazekas and L. Sajó, "Az emberi fej térbeli helyzetének és a tekintet irányának meghatározása," *Képfeldolgozók és Alakfelismerők Országos Konferenciája (KÉPAF VIII.),* pp. 393-406, 2011.

[32] T. F. Cootes, C. J. Taylor, D. H. Cooper and J. Graham, "Active shape models - Their training and application," *Computer Vision and Image Understanding,* vol. 61, no. 1, pp. 38-59, 1995.

[33] J. C. Gower, "Generalized procrustes analysis," *Psychometrika,* vol. 40, no. 1, pp. 33-51, 1975.

[34] T. F. Cootes, G. J. Edwards and C. J. Taylor, "Active appearance models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 23, no. 6, pp. 681-685, 2001.

[35] D. Cristinacce and T. Cootes, "Automatic Feature Localisation with Constrained Local Models," *Pattern Recognition,* vol. 41, no. 10, pp. 3054-3067, 2008.

[36] J. M. Saragih, S. Lucey and J. F. Cohn, "Deformable Model Fitting by Regularized Landmark Mean-Shift," *International Journal of Computer Vision,* vol. 91, no. 2, pp. 200-215, 2011.

[37] J. Saragih, S. Lucey and J. F. Cohn, "Face Alignment through Subspace Constrained Mean-Shifts," *IEEE International Conference on Computer Vision (ICCV),* 2009.

[38] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal,* vol. 7, no. 4, pp. 308-313, 1965.

[39] D. Cristinacce and T. F. Cootes, "A comparison of shape constrained facial feature detectors," *6th IEEE International Conference on Automatic Face and Gesture Recognition,* pp. 375-380, 2004.

[40] S. Milborrow and F. Nicolls, "Active Shape Models with SIFT Descriptors and MARS," *Proceedings of the 9th International Conference on Computer Vision Theory and Applications,* pp. 380-387, 2014.

[41] S. Milborrow, "Building Stasm 4 Models," 2014.

[42] M. B. Stegmann, B. K. Ersbll and R. Larsen, "A Flexible Appearance Modelling Environment," *IEEE Transactions on Medical Imaging,* vol. 22, no. 10, pp. 1319-1331, 2003.

[43] M. B. Stegmann, "Analysis and Segmentation of Face Images using Point Annotations and Linear Subspace Techniques," Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2002.

[44] R. Gross, I. Matthews, J. F. Cohn, T. Kanade and S. Baker, "Multi-PIE," *Proceedings of the Eighth IEEE International Conference on Automatic Face and Gesture Recognition,* 2008.

[45] R. Gross, I. Matthews, J. F. Cohn, T. Kanade and S. Baker, "Multi-PIE," *Image and Vision Computing,* 2010.

[46] I. FGnet, "Talking Face Video," [Online]. Available: http://www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/talking_face.html. [Accessed 2018].

[47] K. Messer, J. Matas, J. Kittler, J. Luettin and G. Maitre, "XM2VTSDB: The Extended M2VTS Database," *2nd International Conference on Audio and Video-based Biometric Person Authentication,* pp. 72-77, 1999.

[48] E. Murphy-Chutorian and M. M. Trivedi, "Head Pose Estimation in Computer Vision: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 31, no. 4, pp. 607-626, 2009.

[49] B. Czupryński and A. Strupczewski, "High Accuracy Head Pose Tracking Survey," in *Active Media Technology*, Springer International Publishing, 2014, pp. 407-420.

[50] L.-P. Morency, P. Sundberg and T. Darrell, "Pose Estimation Using 3D View-Based Eigenspaces," *Proceedings of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures,* pp. 45-52, 2003.

[51] V. Balasubramanian and S. Panchanathan, "Biased Manifold Embedding: A Framework for Person-Independent Head Pose Estimation," *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR '07,* pp. 1-7, 2007.

[52] L. Chen, L. Zhang, Y. Hu, M. Li and H. Zhang, "Head pose estimation using Fisher Manifold learning," *IEEE International Workshop on Analysis and Modeling of Faces and Gestures, 2003. AMFG 2003,* pp. 203-207, 2003.

[53] B. Ma, W. Zhang, S. Shan, X. Chen and W. Gao, "Robust Head Pose Estimation Using LGBP," *Proceedings of the 18th International Conference on Pattern Recognition,* vol. 2, pp. 512-515, 2006.

[54] Y. Cheng, Q. Fu, L. Gu, S. Z. Li, B. Schölkopf and H. Zhang, "Kernel Machine Based Learning for Multi-View Face Detection and Pose Estimation," Microsoft Research, 2001.

[55] M. Voit, K. Nickel and R. Stiefelhagen, "Neural Network-based Head Pose Estimation and Multi-view Fusion," *Proceedings of the 1st International Evaluation Conference on Classification of Events, Activities and Relationships,* pp. 291-298, 2007.

[56] M. J. Jones and P. Viola, "Fast Multi-view Face Detection TR2003-96," Mitsubishi Electric Research Laboratory, 2003.

[57] Z. Zhang, Y. Hu, M. Liu and T. Huang, "Head Pose Estimation in Seminar Room Using Multi View Face Detectors," in *Lecture Notes in Computer Science*, vol. 4122, 2007, pp. 299-304.

[58] D. Ramanan, "Face Detection, Pose Estimation, and Landmark Localization in the Wild," *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 2879-2886, 2012.

[59] T. Vatahska, M. Bennewitz and S. Behnke, "Feature-based head pose estimation from images," *7th IEEE-RAS International Conference on Humanoid Robots, 2007,* pp. 330-335, 2007.

[60] J. Whitehill and J. R. Movellan, "A discriminative approach to frame-by-frame head pose tracking," *8th IEEE International Conference on Automatic Face & Gesture Recognition, 2008. FG '08,* pp. 1-7, 2008.

[61] L.-P. Morency and J. M. J. Whitehill, "Monocular Head Pose Estimation Using Generalized Adaptive View-based Appearance Model," *Image and Vision Computing,* vol. 28, no. 5, p. 754–761, 2010.

[62] L.-P. Morency, J. Whitehill and J. Movellan, "Generalized adaptive view-based appearance model: Integrated framework for monocular head pose estimation,"

8th IEEE International Conference on Automatic Face & Gesture Recognition, 2008. FG '08, pp. 1-8, 2008.

[63] A. Kaehler and G. Bradski, Learning OpenCV 3, Computer Vision in C++ with the OpenCV Library, O'Reilly Media, 2015.

[64] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, no. 11, pp. 1330-1334, 2000.

[65] V. Lepetit and P. Fua, "Monocular Model-Based 3D Tracking of Rigid Objects: A Survey," *Foundations and Trends® in Computer Graphics and Vision,* vol. 1, no. 1, pp. 1-89, 2005.

[66] M. A. Fischler and R. C. Bolles, "The random sample consensus set: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM,* vol. 24, no. 6, pp. 381-395, 1981.

[67] V. Lepetit, F. Moreno-Noguer and P. Fua, "EPnP: An Accurate O(n) Solution to the PnP Problem," *International Journal of Computer Vision,* vol. 81, no. 2, pp. 155-166, 2009.

[68] S. Li, C. Xu and M. Xie, "A Robust O(n) Solution to the Perspective-n-Point Problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, no. 7, pp. 1444-1450, 2012.

[69] C.-P. Lu, G. D. Hager and E. Mjolsness, "Fast and globally convergent pose estimation from video images," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, no. 6, pp. 610-622, 2000.

[70] J. A. Hesch and S. I. Roumeliotis, "A Direct Least-Squares (DLS) method for PnP," *IEEE International Conference on Computer Vision (ICCV), 2011,* pp. 383-390, 2011.

[71] D. F. Dementhon and L. S. Davis, "Model-based object pose in 25 lines of code," *ernational Journal of Computer Vision - Special issue: image understanding research at the University of Maryland,* vol. 15, no. 1-2, pp. 123-141, 1995.

[72] S. Gupta, M. K. Markey and A. C. Bovik, "Anthropometric 3D Face Recognition," *International Journal of Computer Vision,* vol. 90, no. 3, pp. 331-349, 2010.

[73] G. Fanelli, M. Dantone, J. Gall, A. Fossati and L. Gool, "Random Forests for Real Time 3D Face Analysis," *International Journal of Computer Vision,* vol. 101, no. 3, pp. 437-458, 2013.

[74] Grupoinvest GI4E, "GI4E Head Pose Database," Public University of Navarra, Campus Arrosadía, 31006, Pampona Navarre (Spain), 2015.

[75] V. F. Ferrario, C. Sforza, G. Serrao, G. Grassi and E. Mossi, "Active range of motion of the head and cervical spine: a three-dimensional investigation in healthy young adults," *Journal of Orthopaedic Research,* vol. 20, no. 1, pp. 122-9, 2002.

[76] C. A. Baranyi P., "Definition and Synergies of Cognitive Infocommunications," *Acta Polytechnica Hungarica,* vol. 9, no. 1, pp. 67-83, 2012.

[77] D. Didier and K. Antti, "Employment of disabled people in Europe in 2002," *Statistics in focus,* vol. Theme 3, no. 26/2003, 2003.

[78] R. Nishkam, l. D. Nikhi, M. Preetham and L. L. Michael, "Activity recognition from accelerometer data," *In Proceedings of the Seventeenth Conference on Innovative Applications of Artificial Intelligence (IAAI),* pp. 1541-1546, 2005.

[79] B. Ling and S. I. Stephen, "Activity Recognition from User-Annotated Acceleration Data," *Lecture Notes in Computer Science,* vol. 3001, pp. 1-17, 2004.

[80] H. Ghasemzadeh, H. Dept. of Electr. Eng.Ghasemzadeh, J. Barnes, E. Guenterberg and R. Jafari, "A phonological expression for physical movement monitoring in body sensor networks," *5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008.,* pp. 58-68, 2008.

[81] V. J.P., P. Dario and W. T.A., "Human motion recognition using a wireless sensor-based wearable system," *Personal and Ubiquitous Computing,* vol. 16, no. 7, pp. 897-910, October 2012.

[82] L. Daw-Tung, "Facial Expression Classification Using PCA and Hierarchical Radial Basis Function Network," *Journal of Information Science and Engineering,* vol. 22, pp. 1033-1046, 2006.

[83] S. Kota, C. E. S. Digital Syst. Group, J. Raheja, A. Gupta, A. Rathi and S. Sharma, "Principal Component Analysis for Gesture Recognition Using SystemC," *International Conference on Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09.,* pp. 732-737, 2009.

[84] S. M. S. Abu and B. Prabir, "Classification of Facial Expressions Using K-Nearest Neighbor Classifier," *Lecture Notes in Computer Science,* vol. 4418, pp. 555-566, 2007.

[85] S. Alexandre and v. W. Aldo, "Comparative evaluation of static gesture recognition techniques based on nearest neighbor, neural networks and support vector machines," *Journal of the Brazilian Computer Society,* vol. 16, no. 2, pp. 147-162, 2010.

[86] O. Lara and M. Labrador, "A Survey on Human Activity Recognition using Wearable Sensors," *IEEE Communications Surveys & Tutorials,* vol. 15, no. 3, pp. 1192-1209, 2013.

[87] A. Al-Rahayfeh and M. Faezipour, "Eye Tracking and Head Movement Detection: A State-of-Art Survey," *IEEE Journal of Translational Engineering in Health and Medicine,* pp. 11-22, 2013.

[88] J.-z. Liu and Z. Zhao, "Head movement recognition based on LK algorithm and Gentleboost," *2011 7th International Conference on Networked Computing and Advanced Information Management (NCM),* pp. 232-236, 2011.

[89] K. Liu, Y. Luo, G. Tei and S. Yang, "Attention recognition of drivers based on head pose estimation," 2008.

[90] E. Murphy-Chutorian and M. Trivedi, "Head Pose Estimation and Augmented Reality Tracking: An Integrated System and Evaluation for Monitoring Driver Awareness," *IEEE Transactions on Intelligent Transportation Systems,* vol. 11, no. 2, pp. 300-311, 2010.

[91] K. Bertók and A. Fazekas, "Fejmozgás alapú gesztusok felismerése," *Képfeldolgozók és Alakfelismerők Országos Konferenciája (KÉPAF IX.),* pp. 736-746, 2013.

[92] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proceedings of the 7th International Joint Conference on Artificial intelligence,* vol. 2, pp. 674-679, 1981.

[93] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering,* vol. 82, pp. 35-45, 1960.

[94] Y. Bar-Shalom, R. X. Li and T. Kirubarajan, Estimation with applications to tracking and navigation, Wiley-Interscience, 2001.

[95] D. J. Bemdt and J. Clifford, "Using Dynamic Time Warping to Find Patterns in Time Series," AAA1-94 Workshop on Knowledge Discovery in Databases, 1994.

[96] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems,* vol. 7, no. 3, pp. 358-386, 2005.

[97] H. C. S. Sakoe, "Dynamic programming algorithm optimization for spoken word recognition," in *Readings in Speech Recognition*, Morgan Kaufmann Publishers Inc., 1990, pp. 159-165.

[98] D. L. M. R. T. P. G. S. L. Gupta, "Nonlinear alignment and averaging for estimating the evoked potential," *IEEE Transactions on Biomedical Engineering,* vol. 43, no. 4, pp. 348-356, 1996.

[99] A. K. P. G. F. O. Petitjean, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition,* vol. 44, no. 3, pp. 678-693, 2011.

Függelék

Függelék A. Arcdetektorok eredményei

Results for the frontal test database							
Training database	**Recall (%)**	**Precision (%)**	**Runtime (ms)**	**Best stage**	**Type (BT)**	**Feature type**	**Resolution**
Original	99	98	94,99	18	GAB	HAAR	24x24
Original	97	98	102,11	18	RAB	HAAR	24x24
Extended	98	98	121,80	18	DAB	HAAR	24x24
Original	98	95	129,05	17	DAB	HAAR	24x24
Original	98	91	113,16	18	GAB	HAAR	48x48
Original	96	97	150,47	17	DAB	LBP	24x24
Original	95	93	106,81	18	GAB	LBP	24x24
Extended	90	99	96,65	20	GAB	LBP	24x24
OpenCV Alt Tree	94	97	199,97	43	GAB	HAAR	20x20
Original	94	87	127,24	15	GAB	LBP	48x48
Original	71	98	43,70	16	GAB	HAAR	12x12
OpenCV Alt	99	76	247,08	21	GAB	HAAR	20x20
OpenCV LBP	94	69	78,78	20	GAB	LBP	24x24
Extended	96	68	90,50	20	RAB	HOG	24x24
Original	90	76	90,22	20	DAB	HOG	24x24
OpenCV Alt2	98	71	283,40	19	GAB	HAAR	20x20
Original	68	92	89,66	15	GAB	LBP	12x12
OpenCV Default	98	47	246,12	24	DAB	HAAR	24x24

Results for the extended test database							
Training database	**Recall (%)**	**Precision (%)**	**Runtime (ms)**	**Best stage**	**Type (BT)**	**Feature type**	**Resolution**
Extended	97	98	61,70	20	GAB	LBP	24x24
Extended	97	95	91,38	18	DAB	HAAR	24x24
Original	94	94	84,05	18	GAB	LBP	24x24
Original	94	93	91,96	18	RAB	HAAR	24x24
Original	94	92	103,90	18	DAB	HAAR	24x24
Original	94	91	98,17	18	GAB	HAAR	24x24
Original	96	86	106,00	18	GAB	HAAR	48x48
OpenCV Alt Tree	94	95	177,26	43	GAB	HAAR	20x20
Original	96	82	97,83	15	GAB	LBP	48x48
Original	93	89	102,14	17	DAB	LBP	24x24
OpenCV LBP	95	68	66,42	20	GAB	LBP	24x24
Original	79	96	42,63	16	GAB	HAAR	12x12
Extended	97	66	95,82	20	RAB	HOG	24x24
OpenCV Alt	96	74	219,35	21	GAB	HAAR	20x20
Original	91	68	90,23	20	DAB	HOG	24x24
Original	78	91	70,28	15	GAB	LBP	12x12
OpenCV Alt2	93	66	261,59	19	GAB	HAAR	20x20
OpenCV Default	96	46	213,80	24	DAB	HAAR	24x24

12. táblázat. Teszteredmények a két adatbázisra. A *training database* a detektor tanítása során használt adatbázist jelenti. A *best stage* a kaszkád struktúrának azt a szintjét jelenti, ahol a legjobb eredményeket sikerült elérni. A *type (BT)* a boosting algoritmus típusa. A *feature type* a jellemzőkinyerő algoritmus típusa. A *resolution* pedig a pozitív minták felbontása. Az eredményeket a *recall*, *precision* és *runtime* alapján rendeztük sorba, a következő súlyokkal az átlagban: $0{,}4$, $0{,}4$ és $0{,}2$. Az osztályozóimat összehasonlítottam az OpenCV képfeldolgozó könyvtárcsomagban közzétett osztályozókkal [62].

Függelék B. Karakterisztikus pontok kinyerése

Karakterisztikus pontok detektálásának eredményei

	ASM-F AVG	ASM-NF AVG	ASM-ALL AVG	AAM-F AVG	AAM-NF AVG	AAM-ALL AVG	CLM-F AVG	CLM-NF AVG	CLM-ALL AVG
LE1	5,90	10,09	8,00	3,88	8,22	6,05	5,03	9,04	7,03
LE2	4,85	6,08	5,47	3,51	4,54	4,02	5,88	7,61	6,74
LE3	3,14	3,58	3,36	3,64	5,50	4,57	4,98	7,37	6,18
LE4	3,99	3,55	3,77	2,88	6,89	4,89	4,98	5,27	5,12
LE5	3,07	3,79	3,43	3,23	5,00	4,11	5,12	6,96	6,04
LE6	5,50	6,41	5,96	2,33	4,60	3,47	6,72	7,85	7,28
RE1	4,14	5,23	4,69	4,63	13,39	9,01	4,48	5,66	5,07
RE2	3,28	7,38	5,33	3,72	13,39	8,55	4,95	4,38	4,66
RE3	5,65	8,47	7,06	3,22	13,97	8,60	6,20	5,37	5,78
RE4	6,24	9,27	7,75	3,77	14,15	8,96	5,97	7,93	6,95
RE5	4,61	10,22	7,41	3,48	14,15	8,82	6,70	7,12	6,91
RE6	3,76	7,91	5,83	3,36	14,86	9,11	6,05	7,21	6,63
LEB1	6,54	9,48	8,01	7,27	12,05	9,66	8,32	4,13	6,23
LEB2	6,24	4,81	5,53	5,27	13,52	9,40	5,86	9,17	7,52
LEB3	9,69	14,45	12,07	7,96	15,72	11,84	9,62	10,80	10,21
REB1	4,39	8,47	6,43	5,23	7,83	6,53	9,82	9,05	9,44
REB2	9,88	10,22	10,05	5,19	11,59	8,39	6,03	9,16	7,60
REB3	6,62	14,45	10,53	7,14	12,17	9,66	8,45	12,27	10,36
M1	3,85	5,70	4,77	5,23	9,37	7,30	3,42	4,33	3,87
M2	3,32	5,75	4,54	5,73	15,72	10,72	4,70	5,87	5,29
M3	3,73	5,78	4,75	3,78	15,31	9,54	4,01	6,00	5,00

	ASM-F AVG	ASM-NF AVG	ASM-ALL AVG	AAM-F AVG	AAM-NF AVG	AAM-ALL AVG	CLM-F AVG	CLM-NF AVG	CLM-ALL AVG
M4	2,34	8,36	5,35	4,49	14,55	9,52	4,37	5,35	4,86
M5	3,92	6,09	5,01	3,87	6,74	5,31	4,40	6,00	5,20
M6	3,66	7,15	5,40	4,72	13,52	9,12	4,04	4,90	4,47
M7	3,84	7,28	5,56	5,04	13,97	9,51	4,08	5,84	4,96
M8	4,66	7,81	6,23	5,28	13,35	9,31	4,57	5,15	4,86
N1	6,54	9,06	7,80	5,21	14,04	9,62	9,12	13,22	11,17
N2	5,19	9,69	7,44	4,41	14,84	9,62	6,20	10,59	8,40
N3	2,99	6,70	4,85	2,42	14,83	8,62	3,07	5,67	4,37
N4	6,28	6,30	6,29	5,45	11,49	8,47	5,92	4,76	5,34
N5	7,70	8,07	7,89	5,69	7,64	6,66	9,21	6,90	8,05

13. táblázat. A karakterisztikus pontok detektálásának eredményei. A táblázatban az ASM, AAM és CLM eredményeit foglaltam össze, mind a csak frontális arcokat tartalmazó adatbázis (pl. *ASM-F*), mind pedig a kiterjesztett adatbázis (pl. *AAM-NF*) esetén, valamint külön oszlopban szerepeltettem ezek átlagát (pl. *CLM-ALL*). A táblázat sorai az alakzatmodell egy-egy határpontjához tartoznak. A táblázat celláiban az annotált adatkhoz mért RMS pixel hibát szerepeltettem. Az első oszlop alapján: *LE#*: bal szem, *RE#*: jobb szem, *LEB#*: bal szemöldök, *REB#*: jobb szemöldök, *M#*: száj, *N#*: orr, ahol # egy számjegyet jelent.

Karakterisztikus pontok detektálásának eredményei

	ASM-F AVG	ASM-NF AVG	ASM-ALL AVG	AAM-F AVG	AAM-NF AVG	AAM-ALL AVG	CLM-F AVG	CLM-NF AVG	CLM-ALL AVG
Left Eye	4,41	5,58	5,00	3,24	5,79	4,52	5,45	7,35	6,40
Right Eye	4,61	8,08	6,35	3,70	13,99	8,84	5,73	6,28	6,00
Left Eyebrow	7,49	9,58	8,53	6,83	13,76	10,30	7,94	8,04	7,99
Right Eyebrow	6,96	11,05	9,01	5,85	10,53	8,19	8,10	10,16	9,13
Mouth	3,66	6,74	5,20	4,77	12,82	8,79	4,20	5,43	4,82
Nose	5,74	7,97	6,85	4,64	12,57	8,60	6,70	8,23	7,47

14. táblázat. A 13. táblázat eredményeinek összegzése az arc fontosabb testrészeire, azaz a szemekre, a szemöldökökre, a szájra és az orra.

Függelék C. Fejtartás felismerés

Pitch hiba a Biwi adatbázison

Pitch [deg]	-50,0	-45,0	-40,0	-35,0	-30,0	-25,0	-20,0	-15,0	-10,0	-5,0	0,0
Pitch Error [deg]	-	16,1	15,8	13,3	11,2	9,6	8,9	8,4	7,7	6,0	5,3
Pitch [deg]	5,00	10,00	15,00	20,00	25,00	30,00	35,00	40,00	45,00	50,00	-
Pitch Error [deg]	4,9	4,3	5,2	7,0	7,9	8,7	9,3	11,9	12,7	-	-

15. táblázat. A pitch hibája a pitch függvényében a Biwi adatbázis felett. Az eredményeket 5°-onként összegeztem.

Yaw [deg]	-50,0	-45,0	-40,0	-35,0	-30,0	-25,0	-20,0	-15,0	-10,0	-5,0	0,0
Pitch Error [deg]	10,0	8,6	8,6	9,3	8,7	7,9	7,9	8,2	7,4	7,2	6,5
Yaw [deg]	5,00	10,00	15,00	20,00	25,00	30,00	35,00	40,00	45,00	50,00	-
Pitch Error [deg]	6,4	6,5	6,7	7,1	7,1	7,4	7,6	7,1	6,8	9,5	-

16. táblázat. A pitch hibája a yaw függvényében a Biwi adatbázis felett. Az eredményeket 5°-onként összegeztem.

Yaw hiba a Biwi adatbázison

Yaw [deg]	-50,0	-45,0	-40,0	-35,0	-30,0	-25,0	-20,0	-15,0	-10,0	-5,0	0,0
Yaw Error [deg]	20,9	14,3	11,3	7,9	6,7	6,1	6,6	6,2	4,9	3,3	3,1
Yaw [deg]	5,00	10,00	15,00	20,00	25,00	30,00	35,00	40,00	45,00	50,00	-
Yaw Error [deg]	3,8	4,9	5,8	7,7	9,1	10,8	11,7	12,3	12,9	23,7	-

17. táblázat. A yaw hibája a yaw függvényében a Biwi adatbázis felett. Az eredményeket 5°-onként összegeztem.

Pitch [deg]	-50,0	-45,0	-40,0	-35,0	-30,0	-25,0	-20,0	-15,0	-10,0	-5,0	0,0
Yaw Error [deg]	0,0	6,3	6,2	7,0	7,5	9,3	10,8	9,2	8,7	7,7	8,2
Pitch [deg]	5,00	10,00	15,00	20,00	25,00	30,00	35,00	40,00	45,00	50,00	-
Yaw Error [deg]	9,3	10,8	9,4	9,5	9,0	10,0	10,9	14,2	12,9	-	-

18. táblázat. A yaw hibája a pitch függvényében a Biwi adatbázis felett. Az eredményeket 5°-onként összegeztem.

Pitch hiba a GI4E adatbázison

Pitch [deg]	-30,0	-28,0	-26,0	-24,0	-22,0	-20,0	-18,0	-16,0	-14,0	-12,0	-10,0	-8,0	-6,0	-4,0	-2,0	0,0
ASM pitch hiba [deg]	9,9	9,1	7,0	6,6	6,1	6,3	6,2	5,3	4,8	4,3	4,3	3,6	3,6	3,9	3,6	3,3
GT pitch hiba [deg]	2,1	2,2	1,4	1,6	1,1	2,1	2,2	1,5	1,4	1,1	1,3	1,0	1,9	1,0	1,4	1,9
Pitch [deg]	2,0	4,0	6,0	8,0	10,0	12,0	14,0	16,0	18,0	20,0	22,0	24,0	26,0	28,0	30,0	-
ASM pitch hiba [deg]	3,0	3,1	3,6	3,8	4,9	4,6	5,0	5,4	5,8	6,1	6,2	7,0	7,2	7,6	8,9	-
GT pitch hiba [deg]	1,4	2,0	1,5	2,0	1,8	2,1	2,4	1,4	2,1	1,8	2,5	2,2	2,0	2,0	2,5	-

19. táblázat. A pitch hibája a pitch függvényében a GI4E adatbázis felett. Az eredményeket 2°-onként összegeztem. Az *ASM pitch error* az általam illesztett modell alapján számolt pitch hibára vonatkozik. A *GT pitch error* pedig az adatbázishoz mellékelt 2-D ground truth annotációs adatokra számolt pitch hibára vonatkozik.

Yaw hiba a GI4E adatbázison

Yaw [deg]	-30,0	-28,0	-26,0	-24,0	-22,0	-20,0	-18,0	-16,0	-14,0	-12,0	-10,0	-8,0	-6,0	-4,0	-2,0	0,0
ASM yaw hiba [deg]	5,8	5,6	5,0	4,3	4,5	3,9	3,1	3,7	3,2	2,8	2,0	1,4	1,8	1,6	1,7	2,1
GT yaw hiba [deg]	1,3	1,3	1,0	2,1	2,0	2,1	1,2	1,6	1,2	0,9	2,3	2,3	1,2	1,9	1,7	1,3
Yaw [deg]	2,0	4,0	6,0	8,0	10,0	12,0	14,0	16,0	18,0	20,0	22,0	24,0	26,0	28,0	30,0	-
ASM yaw hiba [deg]	1,9	2,0	2,2	2,5	3,0	2,8	3,1	3,2	3,7	4,1	4,6	4,9	4,7	5,1	5,3	-
GT yaw hiba [deg]	1,6	1,6	1,1	0,9	0,6	0,6	0,7	1,0	1,1	1,6	1,7	1,9	2,1	1,4	1,6	-

20. táblázat. A yaw hibája a yaw függvényében a GI4E adatbázis felett. Az eredményeket 2°-onként összegeztem. Az *ASM yaw error* az általam illesztett modell alapján számolt yaw hibára vonatkozik. A *GT yaw error* pedig az adatbázishoz mellékelt 2-D ground truth annotációs adatokra számolt yaw hibára vonatkozik.

A visszavetítés hibája a GI4E adatbázison

Pitch [deg]	-30,0	-28,0	-26,0	-24,0	-22,0	-20,0	-18,0	-16,0	-14,0	-12,0	-10,0	-8,0	-6,0	-4,0	-2,0	0,0
ASM rep. error [deg]	4,3	4,2	4,0	4,2	4,8	4,2	3,2	3,2	3,0	3,8	3,2	3,9	3,9	4,0	4,2	4,1
GT rep. Error [deg]	0,6	0,5	0,5	0,5	0,5	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,3	0,3
Pitch [deg]	2,0	4,0	6,0	8,0	10,0	12,0	14,0	16,0	18,0	20,0	22,0	24,0	26,0	28,0	30,0	-
ASM rep. error [deg]	3,1	3,2	3,2	4,3	3,1	3,6	4,6	4,2	4,9	4,1	4,3	4,0	3,9	4,1	4,6	-
GT rep. Error [deg]	0,4	0,3	0,4	0,4	0,5	0,4	0,4	0,4	0,4	0,5	0,5	0,5	0,5	0,5	0,4	-

21. táblázat. A visszavetítés hibája a pitch függvényében a GI4E adatbázis felett. Az eredményeket 2°-onként összegeztem. Az *ASM rep. error* az általam illesztett modell alapján számolt visszavetítés hibájára vonatkozik. A *GT rep. error* pedig az adatbázishoz mellékelt 2-D ground truth annotációs adatokra számolt visszavetítés hibájára vonatkozik.

Yaw [deg]	-30,0	-28,0	-26,0	-24,0	-22,0	-20,0	-18,0	-16,0	-14,0	-12,0	-10,0	-8,0	-6,0	-4,0	-2,0	0,0
ASM rep. error [deg]	6,0	5,3	5,8	5,4	5,6	5,2	4,4	4,2	4,6	4,6	4,0	4,0	3,2	3,1	2,6	2,5
GT rep. Error [deg]	0,5	0,5	0,5	0,5	0,4	0,4	0,5	0,4	0,4	0,4	0,3	0,4	0,3	0,3	0,3	0,3
Yaw [deg]	2,0	4,0	6,0	8,0	10,0	12,0	14,0	16,0	18,0	20,0	22,0	24,0	26,0	28,0	30,0	-
ASM rep. error [deg]	3,0	2,8	3,0	3,2	3,6	3,2	3,8	4,0	4,7	4,2	4,1	4,6	4,5	4,2	4,9	-
GT rep. Error [deg]	0,3	0,3	0,3	0,3	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	0,4	-

22. táblázat. A visszavetítés hibája a yaw függvényében a GI4E adatbázis felett. Az eredményeket 2°-onként összegeztem. Az *ASM rep. error* az általam illesztett modell alapján számolt visszavetítés hibájára vonatkozik. A *GT rep. error* pedig az adatbázishoz mellékelt 2-D ground truth annotációs adatokra számolt visszavetítés hibájára vonatkozik.

Függelék D. Fejmozgások felismerése

Klaszterközéppontok és az átlagos mozgáshossz kapcsolata

	A	B	C	D	E	F	G	H	I	J	K	L	M
Mu	32,7	19,4	15,1	34,6	14,7	21,9	52,2	31,5	14,9	30,1	43,1	24,2	44,4
Sigma	8,8	5,4	5,1	10,9	3,8	5,9	13,7	10,9	5,4	5,0	12,3	4,7	8,7
AVG Movement Length	40	36	21	46	27	36	48	51	25	44	60	40	66
	N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
Mu	35,4	27,9	15,7	59,2	42,1	31,7	54,4	18,4	14,7	33,7	34,5	49,9	24,5
Sigma	7,9	8,6	4,0	27,5	20,2	4,6	22,3	5,4	3,6	11,2	9,3	20,1	5,8
AVG Movement Length	57	43	29	54	34	35	35	29	26	45	44	58	30

23. táblázat. Az osztályokat legjobban reprezentáló elem tulajdonságai, pl. átlagos DTW távolság az osztály többi elemétől (mu) és szórás (sigma). Valamint az osztályban lévő mozgások átlagos hossza – mérések darabszáma értelemben. 50 ms-ként mértem a fejtartást, így egy 40 db-ból álló mozgás időben 2 másodpercig tart.

Precizitás és felidézés értékei

	A	B	C	D	E	F	G	H	I	J	K	L	M
Precision	0,60	0,99	0,90	0,68	1,00	0,98	0,83	0,73	0,98	0,88	0,61	0,48	0,78
Recall	0,59	0,64	0,83	0,64	0,64	0,92	0,76	0,64	0,88	0,95	0,90	0,87	0,69
F_1 **score**	0,59	0,78	0,87	0,66	0,78	0,95	0,79	0,68	0,92	0,91	0,73	0,62	0,73
	N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
Precision	0,71	0,72	1,00	0,49	0,84	0,92	0,48	0,99	0,98	0,93	0,93	0,63	1,00
Recall	0,74	0,83	0,71	0,82	0,89	1,00	1,00	0,84	0,92	0,93	1,00	0,95	1,00
F_1 **score**	0,72	0,77	0,83	0,61	0,87	0,96	0,65	0,91	0,95	0,93	0,96	0,76	1,00

24. táblázat. A végleges tesztek során kapott precizitás, felidézés és az e kettőből meghatározott F_1 mérőszámok. Az osztályozás annál sikeresebb, minél magasabbak ezek a számok.

A fejmozgás felismerés tévesztési mátrixa

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	60			30											10											
B		99										1														
C			90						10																	
D	27			68			1								3		1									
E					100																					
F		2				98																				
G			7				83										7								3	
H		23				1		73			3															
I									98												2					
J		1						3	2	88		6														
K					1	1		31			61		6													
L		11			39				2			48														
M													78	22												
N								7		2			20	71												
O	7	14	2	4	1										72											
P																100										
Q	7			5			22								2		49				15					
R																16		84								
S	1		7																92							
T			2		10											23		10		48			7			

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
U																					99	1				
V																					2	98				
W																						7	93			
X					6					1														93		
Y		5				6	3			2	4		9	3		2	3								63	
Z																										100

25. táblázat. A fejmozgás felismerés tévesztési mátrixa. 100 db mozgásra teszteltem minden osztály esetén. Az egyes sorok jelentik a tényleges címkéket, az oszlopok pedig a jósolt címkéket. A mátrix celláiban darabszámok találhatók. Így pl. az a betűt hatvanszor jósoltam a-nak, harmincszor d-nek és tízszer o-nak. Az üres cellák a nulla darabra utalnak.

Átlagos DTW távolságok a módszer tesztelése során

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	30			36											30											
B		20										29														
C			16						22																	
D	37			32			69								58		70									
E					13																					
F		28				23																				
G			51				46										48								53	
H		37				48		25			36															
I									15												24					
J		41						41	31	29		39														
K					60	41		33			34		48													
L		27			22				28			21														
M									40				33	42												
N								45		31			37	31												
O	37	25	38	30	28										25											
P																17										
Q	45			48			45								48		43				164					
R																46		31								
S	58		29																36							
T			56		48											50		51		31			55			

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
U																					21	24				
V																					24	15				
W																						37	32			
X					47					35														29		
Y		57				55	64			65	54		66	48		80	58								49	
Z																										27

26. táblázat. Az átlagos DTW távolságok mátrixalakban reprezentálva. Osztályonként 100 db mozgásra teszteltem. Az egyes sorok jelentik a tényleges címkéket, az oszlopok pedig a jósolt címkéket. A mátrix celláiban DTW távolságok találhatók. Így pl. 30 az átlagos DTW távolsága azoknak az a betűknek, melyeket ténylegesen az a osztályba jósoltam. Ezzel szemben pl. 70 az átlagos DTW távolsága azoknak a d betűknek, melyeket a q osztályba jósolt az eljárás. Az üres cellák azt jelentik, hogy egy adott fejmozgást sosem soroltunk abba az osztályba, így átlagos DTW távolságot sem lehetett ezekre számolni.

53. ábra. Az egyes osztályokat legjobban reprezentáló $m_i^r \in M_i$ elemek ($i = 1 \dots 26$), mint fejmozgások grafikus reprezentációja.