Debreceni Egyetem
Informatikai Kar

VIZUÁLIS INFORMÁCIÓN ALAPULÓ KERESŐRENDSZER ARCI ADATBÁZISOKBAN

Témavezető:
Dr. Fazekas Attila
Egyetemi docens

Készítette:
Bertók Kornél
V. PTM

Debrecen
2010

Tartalomjegyzék

1 Bevezetés

Az emberiség történelme folyamán kevés új technológia fejlődött olyan hatalmas sebességgel és tört be mindennapjainkba olyan hirtelen, mint az informatika. Ma már nehéz olyan tevékenységet találni, amely műveléséhez ne lenne elengedhetetlenül szükséges, vagy legalábbis rendkívül hasznos a számítógépek használata. A kezdetben szűk szakmai közösségek által kifejlesztett és felhasznált eszközök egyre szűkösebbé váltak azáltal, hogy a felhasználók köre kiszélesedett. A különböző felhasználások a számítógéppel való kommunikáció újabb és újabb változatos formáit igénylik.

A HCI (ember-számítógép interakció) kutatási feladatai közé tartozik, hogy olyan új, alternatív kommunikációs (adat ki- és beviteli) eszközöket és módszereket fejlesszen, amelyek segítik az ember és gép közötti kapcsolatot az ember számára minél természetesebbé, magától értetődővé tenni.

Diplomamunkámban az ember-számítógép kommunikáció egy újszerű felhasználói eszközével, egy vizuális információn alapuló keresőrendszer kifejlesztésével foglalkozom. A rendszert az IPGD csoportban Dr. Fazekas Attila témavezetése mellett Sajó Levente doktorandusz hallgatóval fejlesztettük ki és implementáltuk C++ nyelven, operációs rendszertől független környezetben.

A dolgozat célja, egy olyan keresőrendszer ismertetése, mely segítségével lehetőségünk nyílik egy arcokat tartalmazó képi adatbázison összetett lekérdezéseket végrehajtani. A jelenlegi rendszer segítségével az adatbázisban szereplő emberek arc-, szem-, és haj színének lekérdezését tehetjük meg.

Fontos leszögezni, hogy a keresés kizárólag vizuális információkon alapul. Tehát nem állnak rendelkezésre úgynevezett metainformációk, melyek segítségével az érzékszervi információ, nyelvi információvá transzformálható. A keresés során a képeket nem foglaljuk gyűjteményekbe, hanem csak mint vizuális információt tároljuk metainformáció nélkül. Fogalmazhatunk úgy is, hogy a rendszernek nincs kiinduló információja arról, hogy valójában mit ábrázol a kép.

1.1 Keresőrendszerek

Napjaink információval túlterhelt társadalmában különösen fontos szerepet töltenek be a keresőrendszerek. E rendszerek segítségével vagyunk képesek megtalálni és elkülöníteni a számunkra hasznos információkat a lényegtelenektől.

Keresőrendszer alatt az informatikában olyan szolgáltatást értünk, ami adatbázisok rendszeres vagy egyéni kérésre történő rendezését, nyomon követését és kivonatolását biztosítja. Mindezek mellett lehetővé teszi a tartalomnak a felhasználó, és általában a szélesebb nyilvánosság részére történő rendelkezésre bocsátását. A keresést általában egy olyan kommunikációs helyzetnek minősítjük, melyben valaki, valahol, valamilyen módon rendelkezésre álló információt szeretne megtalálni. Jelen dolgozat kontextusában úgy érdemes szűkíteni a fogalom jelentéstartományát, hogy a kommunikáció általános modelljében a két kommunikáló fél közül az egyik egy természetes személy, a másik egy számítógép. Ez utóbbi fél digitális adatbázisokban rögzített információkat próbál meg visszakeresni és visszaadni kommunikációs partnere, az ember számára [1].

A megjelenítendő tartalom neve – amely szavakat, logikai operátorokat és egyéb attribútumokat tartalmazó összetett nyelvi kifejezés lehet – a keresőszó, melyet az ember ad meg a keresőrendszernek. Következő lépésben a rendszer megkeresi és kijelzi mindazt, amit ezzel kapcsolatban tud. Keresés alatt általában címszavas keresést értünk, vagyis a rendszer bekéri a keresendő címszót (pl. egy beviteli űrlapmezőben), majd megjeleníti azokat a tartalmi egységeket, amelyek a rendszer szerint a címszóhoz kapcsolódnak.

A keresés modelljének alaposabb kifejtése érdekében célszerű megválaszolni azt a kérdést, hogy „Mit lehet egyáltalán keresni?”. A rövid válasz erre nyilván az, hogy információt, de ebből azonnal következik az újabb kérdés, hogy „milyen információtípusok vannak?”. A válasz előtt rögzítenünk kell, hogy két szempont alapján, érzékszervi, illetve nyelvi szinten különíthetjük el egymástól a különböző információtípusokat. A digitális kommunikáció világán belül érzékszervi szinten a hallás és a látás érzékszerveire támaszkodó információtípusokkal érdemes foglalkozni. E dolgozat a keresés modelljének keresztmetszetét, a multimédiás tartalmakra azon belül is a látás érzékszerveire támaszkodó információtípusokra szűkíti le.

1.2 Tézisek

1. A Viola-Jones objektum detektorok robosztus valósidejű objektumdetektálásra képesek. Érzéketlenek, azaz robosztusak a beérkező kameraképre, gyors feldolgozásra képesek magas detektálási arány mellett. Egy 30 kép/másodperc teljesítményű 640x480 képpont felbontású webkamera képén, a Viola-Jones arcdetektor 30 kép/másodperc teljesítményt ér el egy 1.6 GHz-es Intel Pentium M processzorral rendelkező notebook-on, tehát képes az összes beérkező kép feldolgozására.
2. A távolság transzformáció egy hatékony eljárás lehet a színtér szegmentálására. Segítségével egész aritmetikát használva hatékonyan tudjuk közelíteni a színek és az egyes színklaszterek közötti euklideszi távolságot.
3. A színtér távolság transzformáltjának Voronoi felosztásával ki tudjuk alakítani azt a rácsszerkezetet, melynek minden belső pontja közelebb van az adott klaszter alappontjaihoz, mint az összes többi ponthoz.
4. Az arci jellemzők szegmentálása elvégezhető az azok szerkezeti információja alapján.
5. Az arci jellemzők színének kinyerése során sokkal gyorsabb eljárást és valósághűbb eredményt kapunk, ha humán megfigyelések alapján a színtér minden egyes elemét színkategóriákba soroljuk be.

1.3 A dolgozat felépítése

Dolgozatom 2. fejezetében keresőrendszerünk működéséhez szükséges elméleti hátteret fektetem le, különös figyelmet fordítva az arci jellemzők színének meghatározásának alapját jelentő objektumdetektálást (lásd 2.3. fejezet). A 2.4. és a 2.5. fejezetben, egy a színterünk hatékony és gyors szegmentálását megvalósító algoritmus alapköveit (távolság transzformáció és Voronoi diagram) ismertetem.

A 3. és 4. fejezetben a keresőrendszer tényleges működésével, az arci jellemzők színének meghatározásával foglalkozom. A jellemzők színét két lépésben határozzuk meg: először szerkezeti információ alapján szegmentáljuk az arcbőrt, a szemeket és hajat (lásd 3. fejezet). Majd második lépésben az eredeti képen a szegmentált terület alatti pixelek világosságkódjainak nagy elemszámú halmazát cseréljük le egy kisebb halmazra – melyet az adott jellemzők színének meghatározásához használunk fel. A felhasznált színmodellünket humán megfigyelések alapján készítettük el. Ezen a modellen alapszik a színmeghatározó metódusunk, melyet a 4. fejezetben részletesen ismertetek.

2 Elméleti háttér

Az elkészült keresőrendszer az IPGD csoportban eltöltött több mint kétéves kutató és fejlesztő munkám eredménye. A fejlesztés során a digitális képfeldolgozás számos szerteágazó területét tanulmányoztam és sajátítottam el.

Ebben a fejezetben a keresőrendszer felépítéséhez szükséges algoritmusokat, módszereket és elméleti alapokat, vagyis a rendszer alapköveit tekintjük át.

2.1 Színek használata

A szín voltaképpen a látható tartományba eső elektromágneses hullámok által kiváltott érzet [2], amely a hullámok spektrális eloszlásán (fizikai tulajdonságain) kívül döntő mértékben függ a szem és az agy működésétől, továbbá pszichológiai jelenségektől. A színek érzékelése tehát személyes élmény (nem mérhető objektivitás), vizsgálata emiatt a fizikától a biológián és a pszichológián át egészen a képzőművészetekig vezet. Ezzel a megállapítással eljutottunk a digitális képfeldolgozás legalapvetőbb és legnehezebben megválaszolható kérdéséhez: Mikor mondjuk egy színre, hogy az piros, vagy barna, stb.? Nyilvánvaló, hogy egy keresőrendszernek, mely például az arc színét hivatott meghatározni, annak az előző kérdésre kell helyest választ szolgáltatnia.

2.2 Színterek

Az előző kérdés megválaszolásához mindenekelőtt szükségünk lesz egy modellre, amely alkalmas a színek és a közöttük fennálló relációk matematikai leírására. Erre a modellre a továbbiakban színtérként fogok hivatkozni. A színterek a színek ábrázolására használható virtuális terek (koordinátarendszerek), melyben az egyes színek rögzített összefüggések alapján reprezentálhatók [3,4]. A színtér határozza meg, hogy milyen módon tárolunk egy színt. Általában egy–, kettő–, három–, vagy négydimenziós teret definiálnak, ahol egy–egy dimenzió rendre egy–egy jellemző számszerű kifejezésének felel meg, így az egyes színeket azok koordinátái fejezik ki. A színtérben az ábrázolható színek valamilyen rend szerint kerülnek elhelyezésre (pl. az alapján, hogy a színtér alapszíneinek milyen arányú keverésével állíthatók elő), és a pozíciójukat meghatározó koordinátákkal kerülnek azonosításra. A képfeldolgozásban leggyakrabban az RGB, HSV, HSI, CMYK, színtereket alkalmazzák, de ezeken kívül rengeteg színtér létezik még.

A színeknek rendkívüli jelentősége van az ember-számítógép interakciókban. De problémákat is jelentenek a feldolgozásban, hiszen az érzékelő számára a színek

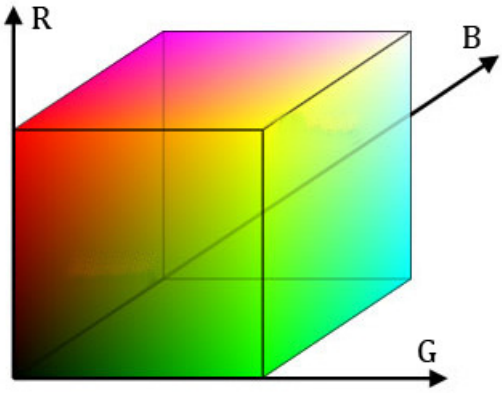
megváltozhatnak a felület orientációjával, a kamera nézőpontjának, a megvilágítás pozíciójának vagy spektrumának megváltozásával, valamint annak függvényében, hogy a fény hogyan hat az egyes objektumokra.

A számítástechnikában legismertebb RGB színtér nem az emberi szem érzékeléséhez igazodik, így használata csak akkor javasolt, ha a kameraképen minimális változások vannak. Az RGB térrel szemben léteznek invariáns tulajdonságokkal rendelkező színterek is. Olyanok, ahol a színárnyalat nem függ az objektum elhelyezkedésétől, ezért jobban használható a lényeges információk kinyerésére. E tulajdonságokkal rendelkező terekben a színek csoportosítása – magyarán a teljes színtér szegmentálása – egyszerűbben és átláthatóbban elvégezhető, mint más terek esetén.

2.2.1 RGB színtér

A bemeneti képünk RGB színtérben van definiálva, így az egyes pixelpozícióiban lévő intenzitás értékek, RGB koordinátáknak felelnek meg. Az RGB színrendszerben a színek a három alapszín, a vörös (R – Red), zöld (G – Green), kék (B – Blue) egymásra vetítésével állíthatók elő, tulajdonképpen ezt nevezik additív színkeverésnek (lásd 2.1. ábra). Ez a fajta színkeverési rendszer a kisugárzott, illetve az érzékelt fényen alapul, ezért csak fényt kibocsátó berendezésekkel hozható létre, illetve azokban alkalmazzák (pl. monitorokban).

Az RGB tér legnagyobb problémája, hogy a színcsoportok pozíciója és a csoport elemeit befoglaló alakzat formája a színtéren belül, egyáltalán nem kézenfekvő. Ezért a színtér szegmentálására érdemes, a színek elhelyezkedését illetően egy strukturáltabb színteret használni, mint pl. a HSI tér [5,6].



2.1. ábra. Az RGB színtér 3D-s reprezentációja. Az egyes tengelyek az alapszíneknek megfelelő címkékkel vannak ellátva.

2.2.2 HSI színtér

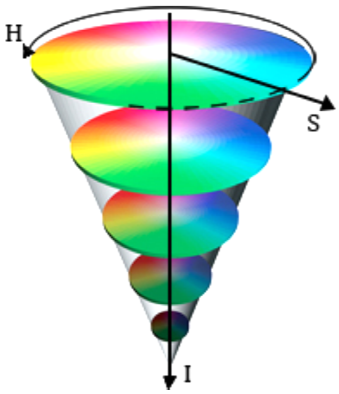
HSI színtér esetén, az RGB koordinátarendszer testátlója felől nézzük a színeket jelentő pontokat. Az R tengely irányát tekintjük *nulla* foknak, és ehhez képest határozzuk meg a pontok irányát, ami a HSI tér H (Hue – Színezeti szög) komponense lesz. A 2.1. táblázatban az RGB tér alap–, és a kiegészítő színeinek oldalszöge látható, mindazonáltal fontos megjegyezni, hogy a színezeti szög nem azonos a színnel, hiszen ezen túl van még két érték, amely befolyásolja a színérzetet. Úgy is fogalmazhatunk, hogy a színezeti szög, csak a fény hullámhosszúságának függvénye.

H (fok)	0°	60°	120°	180°	240°	300°
Szín	Piros	Sárga	Zöld	Kékeszöld	Kék	Bíbor

2.1. táblázat. Az alapszínek és kiegészítő színek jellegzetes színezete

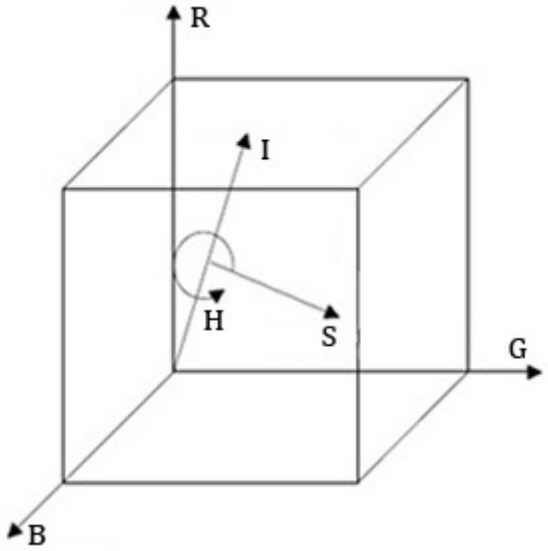
A HSI színtér második eleme az S koordináta (Saturation – Telítettség). Ez a komponens a szín élénkségét, vagyis a fehér összetevő mennyiségét fejezi ki. A spektrum-színek 100%-os telítettségűek, nincs fehér összetevőjük. Ugyanakkor például a rózsaszín néhány százalékban fehér összetevőt tartalmazó vörös. Egy átlagos szem húsz különböző telítettségi fokozatot tud elkülöníteni Az alapszínek és kiegészítő színek telítettsége maximális. A szürke szín árnyalatainak a telítettsége pedig nulla. Az 2.2. ábrán a kúp keresztmetszetét alkotó kör középpontjában a telítettség nulla, a kör kerületén pedig egy.

A színrendszer harmadik eleme az I koordináta (Intensity – Intenzitás), mely az adott szín sötétségét fejezi ki. Ez a fényforrás által kibocsátott fotonok mennyisége, illetve az egységnyi felületre beérkező fotonok száma. Például a barna szín spektrális eloszlása a sárgáéval azonos, de más a világosság értéke. Átlagosan mintegy ötszáz intenzitásfokozatot tudunk a szemünkkel megkülönböztetni.



2.2. ábra. A HSI színtér 3D-s reprezentációja.

A fenti fejezetből látható, hogy a HSI színtér használata a képfeldolgozásban rendkívül előnyös, mivel a feladatok egy része, egyetlen komponens módosításával elvégezhető. Ennek nem csak azért van jelentősége, mert kevesebb számítást igényel, hanem azért is, mert bizonyos jellemzők vizsgálatára (telítettség, színezeti szög, relatív világosság, világosság) az RGB térben nincs lehetőségünk – márpedig a vizuális információn alapuló keresőrendszerünk működéséhez e mennyiségek nélkülözhetetlenek.



2.3. ábra. A kapcsolat az RGB és a HSI színterek között. Az ábrán az RGB tér bonyolult struktúrája látható a színek elhelyezkedését illetően.

2.3 Objektumdetektálás

A keresőrendszerünk sikeres működéséhez elengedhetetlen egy nagy megbízhatóságú, robosztus objektumdetektáló rendszer. A robosztusságon azt értjük, hogy a rendszernek érzéketlennek kell lennie a beérkező kép minőségére.

Az IPGD csoportban 2008. első felében az Intel OpenCV [7] szabadon felhasználható képfeldolgozó könyvtár segítségével elkészítettünk egy olyan szoftver eszközt, amely a gyakorlati kísérletek során bebizonyította, hogy megfelel az előbbi bekezdésben megfogalmazott elvárásainknak.

Az alábbiakban ennek a rendszernek az elméleti hátterét tekintjük át a hozzá kapcsolódó algoritmusokkal együtt. Ezzel a komponenssel határozzuk meg a bőr-, szem-, és hajszín megállapításához szükséges arc és szem objektumok pozícióját a bemeneti képeken.

2.3.1 Viola-Jones detektorok

Paul Viola és Michael J. Jones egy olyan frontális arcdetektor rendszert alakítottak ki [8], mely eléri az előtte publikált legjobb eredmények [9,10,11,12,13] találati és hamis pozitív arányát (3. és 4. definíció). Az arcdetektor mintájára szem-, és szájdetektorokat készítettünk az Intel OpenCV szabadon felhasználható képfeldolgozó-könyvtár segítségével, melyben Viola és Jones közzétették arcdetektorukat és az implementált tanítási algoritmust.

Arcdetektáló rendszerük tisztán kiemelkedik az eddigi megközelítések közül gyorsaságában. Valós időben 30 kép/másodperc teljesítményt érhetünk el, 640x480 pixel felbontású kameraképen egy 1.6 GHz-es Intel Pentium M processzorral rendelkező notebookon. A Viola-Jones detektor szürkeskálás képekből kapott információk alapján képes magas képfeldolgozási arányt elérni. A következő definíciók a további részek érthetősége miatt kerültek ide.

1. Definíció. Hamis találat, vagy hamis pozitív

Ha a detektor a kép egy részletén igaz eredménnyel tér vissza, miközben a képrészleten nem szerepel a keresett objektum, akkor azt hamis találatnak vagy hamis pozitívnak nevezzük.

2. Definíció. Hamis negatív

A nem detektált objektumot hamis negatívnak, vagy hamis elutasításnak nevezzük.

3. Definíció. Találat

Ha a detektor a kép egy részletre igaz eredményt ad, és a képrészlet valóban a keresett objektumot ábrázolja, akkor ezt az eseményt találatnak nevezzük.

4. Definíció. Hamis pozitív arány, vagy hamis találati arány

Detektor tesztelése során a hamis találatok számát osztva a vizsgált képrészletek számával kapjuk a hamis pozitív-, vagy hamis találati arányt. Ez egy racionális szám a [0..1] intervallumból.

5. Definíció. Találati vagy detektálási arány

Detektor tesztelése során a találatok számát osztva a keresett objektumok számával kapjuk a találati vagy detektálási arányt. Ez egy racionális szám a [0..1] intervallumból.

Gondolatmenetünket három fő témakör köré csoportosíthatjuk:

1. Az integrált képreprezentáció bevezetése. A rendszer a kép intenzitásértékei helyett képrégiók jellemzőivel dolgozik [14]. Ezek a jellemzők a Haar-féle bázisfüggvényekre emlékeztetnek, bár néhány esetben ezeknél valamivel bonyolultabbak. A jellemzők többféle méret melletti gyors kiszámítása érdekében vezetjük be az integrált képet, amely az eredeti képből képpontonként néhány elemi művelettel – tehát rendkívül gyorsan és hatékonyan – előállítható. Az integrált képreprezentáció ismeretében a jellemzők értéke bármely pozícióban vagy skálázás mellett konstans időben meghatározható.

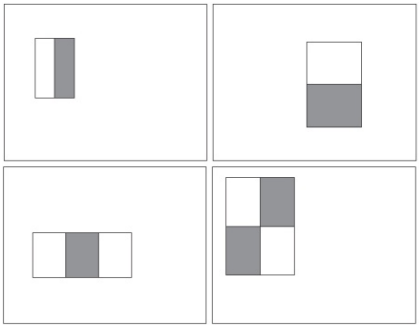
2. A második fontos tulajdonság egy osztályozó létrehozásának folyamata, a fontos jellemzők egy kis halmazának kiválasztásával, az AdaBoost-on alapuló tanuló algoritmust használva [15]. Egy kép részablakán belül az összes Haar jellemzők száma nagyon nagy, sokkal nagyobb, mint a pixelek száma. Azért, hogy a gyors osztályozást biztosítsuk, a tanuló folyamatnak ki kell zárnia az elérhető jellemzők nagy többségét, és a kritikus jellemzők kis halmazára kell koncentrálnia. Tieu és Viola munkája által motiváltan, a jellemző kiválasztás az AdaBoost egy egyszerű módosítása lett: minden gyenge osztályozót kényszerítenek, hogy annak eredménye csak egy jellemzőtől függjön [16]. Ennek eredményeként a Boost folyamat minden köre, amely új gyenge osztályozót választ, megfelel egy jellemző kiválasztó folyamatnak [17,18,14].

3. Az utolsó lépés, hogy az egyre bonyolultabb osztályozókat láncba, egy vízesésszerű sémába rendezzük. Az ötlet azon az észrevételen alapul, hogy gyakran egyszerű azt eldönteni, hogy egy objektum hol fordulhat elő, így a bonyolultabb feldolgozást csak ezeken a „bíztató” területeken kell elvégezni [19,20,13]. Az osztályozók teljesítményének fontos mértéke az úgynevezett „hamis negatív” találati arány, vagyis azon részablakok aránya, amelyeket elutasítunk, annak ellenére, hogy tartalmazzák a keresett objektumot. Ideális esetben a képen ténylegesen szereplő összes célobjektum régióját meg kell találni.

Egy gyors arcdetektornak széles alkalmazási köre lehet. A detektálás sebességének növekedése valós idejű arcdetekálást tesz lehetővé olyan rendszereken, melyeken ez korábban kivitelezhetetlen volt. Implementálható alacsony teljesítményű eszközök széles skáláján, beleértve például a kézikamerákat és integrált processzorokat is. A detektor készítői például egy Compaq iPaq kézikamerán implementálták rendszerüket, és azon 2 kép/másodperc mellett detektáltak arcot. (Az eszköznek egy alacsony teljesítményű 200 MIPS sebességű StrongArm processzora volt, mely nem volt képes lebegőpontos számításra.)

2.3.2 Jellemzők

A Viola-Jones objektumdetektáló rendszer egyszerű jellemzők értékei alapján osztályoz képeket. Sok érv szól a jellemzők használata mellett, a képpontok közvetlen használata helyett. A legáltalánosabb ok, hogy a jellemzők ad-hoc tudás területeket képesek kódolni, melyeket nehéz megtanulni véges mennyiségű tanító adathalmaz használatával. A rendszer egy másik fontos motivációja a jellemzők használatára: a jellemző-alapú rendszer sokkal gyorsabban dolgozik, mint a képpont-alapú.



2.4. ábra. Téglalap jellemzők a detektor ablakon belül. A fehér téglalapon belül fekvő pixelek összegét vonjuk ki a fekete téglalapon belül fekvő pixelek összegéből.

A használt egyszerű jellemzők emlékeztetnek a Haar-féle bázisfüggvényekre, melyeket Papageorgiou és társai használtak [14]. A Viola-Jones rendszer ezeknek a jellemzőknek három fajtáját használja. A *két-téglalap jellemző* értéke a pixelek összegének különbsége két téglalap terület között. A területeknek ugyanakkora a mérete és alakja, és függőlegesen vagy vízszintesen szomszédosak (lásd 2.4. ábra). A *három-téglalap jellemző* két szélső téglalap pixeleinek összegét vonja le a középső téglalap pixeleinek összegéből. Végül a *négy-téglalap jellemző* diagonális téglalap párok közötti különbséget számít.

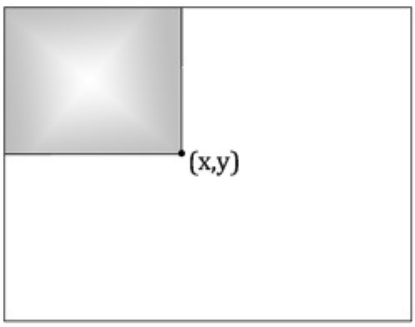
2.3.3 Integrál kép

Egy adott képet többfajta kódolással lehet tárolni. A Viola-Jones detektorok csak szürkeskálás képeket használnak. Ha színes képeket használunk, akkor az algoritmus először átalakítja a képeket szürkeárnyalatossá, majd ezeken dolgozik.

A szürkeárnyalatú képek egy lehetséges tárolási formája egy olyan mátrix, amelynek minden eleme a kép egy adott pixelének szürkeárnyalatát mutatja. Általában ez az elem egy [0,255]-ig terjedő egész szám. A képhez tartozó integrál kép egy ugyanolyan méretű mátrix, amelynek elemei az eredeti kép adott pixelétől balra és felfelé elhelyezkedő pixelértékek összegét tartalmazza. Tehát az integrál kép az x, y pontban a tőle balra fent levő téglalapban elhelyezkedő pixelek intenzitásértékeinek összege:

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

ahol *ii*(*x; y*) az integrál kép, és *i*(*x; y*) az eredeti kép (lásd 2.5. ábra).



2.5. ábra. Az integrál kép értéke az (x,y) pontban a balra fent levő téglalapban elhelyezkedő pixelek intenzitásértékeinek összege.

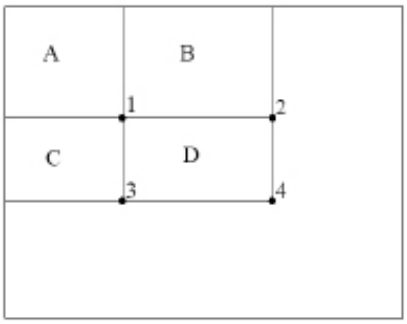
Jelölje $s(x,y)$ az intenzitásértékek területmérő függvényét, és definíció szerint legyen $s(x,-1) = 0$, továbbá $\mathrm{ii}(-1,\mathrm{y}) = 0$. Ekkor igazak az alábbi rekurzív összefüggések:

$$\mathrm{s}(\mathrm{x},\mathrm{y}) = \mathrm{s}(\mathrm{s},\mathrm{y}-1) + \mathrm{i}(\mathrm{x},\mathrm{y}), \tag{2.2}$$

$$\mathrm{ii}(\mathrm{x},\mathrm{y}) = \mathrm{ii}(\mathrm{x}-1,\mathrm{y}) + \mathrm{s}(\mathrm{x},\mathrm{y}). \tag{2.3}$$

A fentiekből látható, hogy az integrál kép az eredeti kép egyszeri végigolvasásával előállítható, majd a későbbiekben a jellemző kiértékelésben felhasználható.

Az integrál képet használva egy téglalap összeg négy tömbhivatkozással számítható (lásd 2.6. ábra). Két téglalap közötti különbség nyolc hivatkozással számítható. A két-téglalap jellemző mivel szomszédos téglalapok különbségét veszi, hat tömbhivatkozással számítható, a három-téglalap jellemző nyolc hivatkozással, a négy-téglalap jellemző kilenc hivatkozással.



2.6. ábra. A D-be eső képpontok összege az integrál kép négyszeri elérésével meghatározható. Az integrál kép értéke az 1-es pozícióban az A-ba eső képpontok intenzitásösszege. 2-ben az érték $\boldsymbol{A}+\boldsymbol{B}$, 3-ban $\boldsymbol{A}+\boldsymbol{C}$, 4-ben $\boldsymbol{A}+\boldsymbol{B}+\boldsymbol{C}+\boldsymbol{D}$. Így a D-be eső összeg a $\mathbf{4}+\mathbf{1}-(\mathbf{2}+\mathbf{3})$ összefüggés alapján számítható ki.

2.3.4 A jellemzők előnye a detektálás során

Annak érdekében, hogy – egyszerűségük és látszólagos rugalmatlanságuk ellenére – értékelni tudjuk a téglalap alakú jellemzők használhatóságát, vizsgáljunk meg hatékonysági szempontból egy hagyományosabb megközelítést. Több objektum detektorhoz hasonlóan a mi rendszerünk is többféle skálázás mellett olvassa végig a digitális képet. Kezdetben az objektumokat 24×24-es négyzetekben keressük, és a képet összesen 11 skálázás mellett olvassuk végig. Az ablak minden lépésben 1.25-ször nagyobb, mint az előző mérete volt.

A hagyományos megközelítés ezzel szemben az, hogy a képről egy 11 skálázás melletti piramist építünk fel, amelynél minden szint 1.25-ször kisebb, mint az előző. Majd egy rögzített skálájú detektort futtatunk le minden egyes szinten. Bár a piramis felépítése egyszerű, mégis igen költséges.

Hagyományos személyi számítógépeken még a 15 kép/másodperc sebesség elérése is nehézséget jelent. Ezzel szemben mi jellemzők egy halmazát definiáltuk, amely azzal a tulajdonsággal bír, hogy bármely eleme tetszőleges skálázás mellett néhány művelettel, rendkívül gyorsan meghatározható az értéke. Így akár a 15 kép/másodperc sebességű objektumdetektálás is könnyűszerrel elérhető, ami kevesebb időbe kerül, mint önmagában a

11 szintű piramis felépítése. Következésképpen minden olyan detektor, amely a hagyományos piramis módszert alkalmazza, várhatóan lassabb lesz az általunk tárgyalt megközelítésnél.

2.3.5 Az osztályozó függvények tanítása

Ha adott a jellemzőknek, továbbá pozitív és negatív tanítópéldáknak egy-egy halmaza, akkor a gépi tanulás bármely megközelítése használható egy osztályozó függvény tanulásához. Feltételezve, hogy a detektor alap felbontása 24×24-es, azt kapjuk, hogy a téglalap jellemzők halmazának számossága – a Haar-féle bázisfüggvényektől eltérően – igen nagy: 45.396. Ez a szám sokkal nagyobb, mint a képpontok száma. Habár minden jellemző hatékonyan számítható, a jellemzők teljes halmazának számítása megengedhetetlen. Ezeknek a jellemzőknek már igen kicsi halmazával is létrehozható hatékony osztályozó. Az igazi kihívás ezeknek az osztályozóknak a megtalálása.

A Viola-Jones rendszerben az AdaBoost egy változatát használják mind a jellemző kiválasztásra, mind pedig az osztályozó tanítására. Eredeti formájában az AdaBoost gyenge osztályozó függvények kombinálásával képes egy erős osztályozót előállítani, ezáltal tetszőleges tanuló algoritmus javítására használható.

Az alapgondolat a következő: a gyenge osztályozót lefuttatjuk egy tanítóhalmazon. A teljes tanítóhalmaz osztályozása után az előző osztályozó által hibásan osztályozott példákat újra súlyozzuk, és újra elvégezzük a tanítást. Így a végső erős osztályozó egy perceptron lesz, a gyenge osztályozók lineáris kombinációja, kiegészítve egy küszöbértékkel. Freund és Shapire bebizonyította, hogy az erős osztályozó tanulási hibája a körök számával párhuzamosan exponenciálisan tart nullához. Még fontosabbak az általános teljesítmény számeredményei, melyeket később bizonyítottak [17].

A hagyományos AdaBoost folyamatot könnyen értelmezhetjük egy mohó jellemző kiválasztó folyamatként. A fokozás általános problémáját figyelembe véve, amiben a jellemző osztályozó függvények egy nagy halmazát kombináljuk össze súlyozott többségi szavazás alapján, az igazi kihívás, hogy a jó osztályozókhoz nagy súlyt adjunk, míg a kevésbé jó függvényekhez kicsit. Az AdaBoost egy agresszív eljárás jó osztályozók kis halmazának kiválasztására, amik jelentősen eltérőek lehetnek. Egy analógiát rajzolva a gyenge osztályozók és a jellemzők közé, az AdaBoost egy hatékony eljárás jó jellemzők egy kis halmazának keresésére, melyek jelentősen eltérőek lehetnek.

Egy gyakorlati eljárás ennek az analógiának a kiegészítésére, hogy megszorítjuk a gyenge tanulót az osztályozó függvényeknek arra a halmazára, melyek csak egyszerű jellemzőktől függnek. Ennek eléréséhez a gyenge tanuló algoritmust úgy tervezték meg, hogy azt az egyszerű téglalap jellemzőt válassza ki, amely a legjobban választja szét a pozitív és negatív példákat Minden jellemzőhöz a gyenge tanuló meghatározza az optimális küszöbosztályozó függvényt, ami a példák minimális számú halmazát osztályozza rosszul.

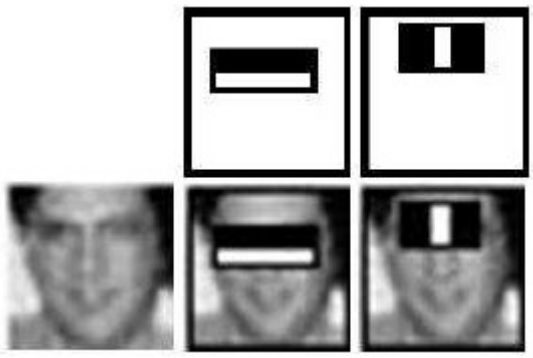
Így egy gyenge osztályozó ($h_j(x)$) egy jellemzőből (f_j), egy küszöbből (θ_j) és egy paritásból (p_j) áll, ahol a paritás az egyenlőtlenség jel irányát jelöli:

$$\mathrm{h_j(x)} = \begin{cases} 1, \text{ha } \mathrm{p_j f_j(x)} < \mathrm{p_j \theta_j}, \\ 0, \text{különben.} \end{cases} \tag{2.4}$$

Ahol x egy 24×24 pixel felbontású kép-alablak. Az AdaBoost algoritmus pszeudókódja, a függelék 9.1. fejezetében található.

2.3.6 A tanítás eredménye

Az implementált rendszerrel végzett kezdeti kísérletek azt mutatták, hogy már mindössze 200 jellemzőből konstruált osztályozó is elfogadható pontossággal működik. Számszerűsítve, 95%-os találati arány mellett a rendszer mindössze 1 hamis pozitív objektumot adott a 14.084-es teszthalmazból. Az arcdetektálás esetében az AdaBoost által kiválasztott jellemzők szemléletesek és könnyen értelmezhetők. Az első jellemző az arc azon tulajdonságára koncentrál, hogy a szemek környéke gyakran sötétebb, mint az orr és a szemek alatti rész.



2.7. ábra. Az AdaBoost által választott első és második jellemző. A két jellemző látható a felső sorban, az alsó sor pedig a jellemzőket egy tanító arcra helyezve mutatja meg.

Ez a jellemző az ablak méretéhez mérten meglehetősen nagy, következésképpen érzéketlen az arc méretére vagy elhelyezkedésére. A második kiválasztott jellemző azt írja le, hogy a szemek sötétebbek, mint az orrnyereg (lásd 2.7. ábra). Az eredmények bíztatóak, de

nem elegendőek a valós idejű alkalmazások készítéséhez. Sajnos a detektálási arány javításának legegyszerűbb módja – újabb jellemzők bevonása – közvetlenül növeli a számítási költségeket, így csökkenti a detektor sebességét.

2.3.7 A sebesség növelése

Ebben a részben röviden áttekintünk egy algoritmust, amellyel osztályozóinkat olyan „vízesés" struktúrába tudjuk rendezni, amely a detektálási arány növelése mellett a sebességet is rendkívüli mértékben megnöveli. Az alapötlet az, hogy könnyű olyan gyenge osztályozókat készíteni, amelyek a negatív példák nagy részét visszautasítják, viszont a pozitívokat elfogadják. Egyszerű osztályozókat használunk tehát a negatív példák elutasítására, és az összetettebb osztályozókat csak a biztató területeken futtatjuk le, hogy minél alacsonyabb hamis pozitív arányt érhessünk el.

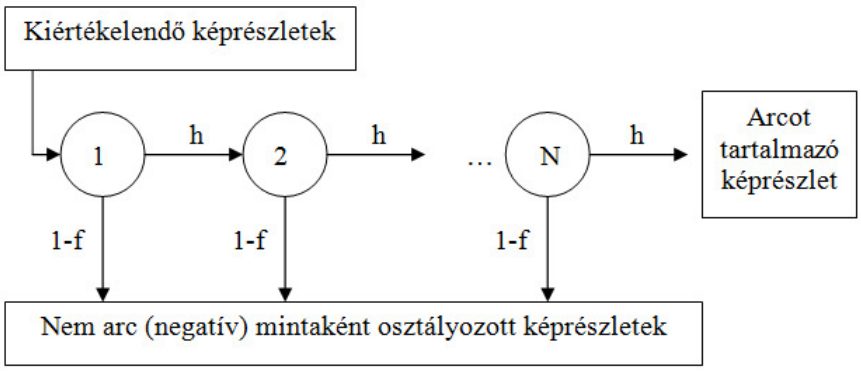
A vízesés egyes lépcsőiben szereplő osztályozókat az AdaBoost algoritmussal tanítjuk be. Az egy, két jellemzős gyenge osztályozó hatékony arcdetektor lehet, ha a küszöbértéket úgy hangoljuk, hogy a hamis pozitív arányt minimalizálja. Az AdaBoost kezdeti küszöbértéke, $\frac{1}{2}\sum_{t=1}^{T}\alpha_t$. Ha ezt csökkentjük, nő a detektálási, de sajnos a hamis pozitív arány is. A kísérletek azt mutatták, hogy a két jellemzős detektor képes az arcok 100%-át detektálni 40%-os hamis pozitív arány mellett. Ez természetesen elfogadhatatlan a gyakorlati alkalmazásokban. Mindamellett nagyon kevés művelettel képes lecsökkenteni azon ablakok számát, amelyek további feldolgozást igényelnek:

1. A téglalap jellemzők kiértékelése (jellemzőként 6-9 olvasás).
2. A gyenge osztályozó kiszámítása minden jellemzőre (jellemzőként 1 művelet).
3. A gyenge osztályozók kombinálása (jellemzőként egy szorzás, egy összeadás, és a végén egy küszöbérték ellenőrzés).

A végső osztályozó egy elfajuló döntési fa, amelyet mi „vízesésnek" nevezünk [21]. Az első osztályozó pozitív válasza elindítja a második osztályozót, amely szintén nagyon magas detektálási aránnyal rendelkezik. Ha ez pozitív választ ad, elindul a harmadik osztályozó, és így tovább. Ha bármely pontnál a válasz nemleges, az ablakot azonnal visszautasítjuk, ahogy ezt a 2.8. ábra is mutatja.

Az osztályozó szerkezete is azt tükrözi, hogy egy képen belül a részablakok túlnyomó többsége negatív példa. Ennek érdekében a vízesés a lehető legkorábbi szakaszban igyekszik elutasítani minél több negatív példát. Mivel a pozitív példára mindegyik osztályozó pozitív

választ ad, így egy ilyen megtalálása az összes ellenőrzéshez képest nagyon ritka esemény. A döntési fához hasonlóan egy adott osztályozót csak az előző osztályozók által pozitívnak ítélt példákkal tanítunk. Következésképpen, a második osztályozónak nehezebb dolga van, mint az elsőnek. Adott detektálási arány mellett a mélyebben elhelyezkedő osztályozók magasabb hamis pozitív aránnyal rendelkeznek. A vízesés struktúra tanítása a függelékben 9.2. fejezetében található.



2.8. ábra. Egy $\boldsymbol{N}$ fokozatú döntési fa szerkezeti felépítése. A fokozatok találati aránya $\boldsymbol{h}$, $\boldsymbol{f}$ pedig a hibásan osztályozott minták aránya. A teljes rendszerre vonatkozó találati arány $\boldsymbol{h^N}$, a hibás osztályozás arány pedig. $\boldsymbol{f^N}$.

2.4 Távolságmérés

2.4.1 Metrika

Metrikus tér alatt egy olyan (X, d) rendezett párt értünk, ahol X egy tetszőleges halmaz, $d: X^2 \to R^+$pedig olyan nemnegatív valós szám értékű függvény, melyre tetszőleges $x, y, z \in X$ esetén:

1. $d(x, y) = 0 \Leftrightarrow \mathrm{x} = \mathrm{y}$ (egyenlőségi tulajdonság) (2.5)
2. $d(x, y) = d(y, x)$ (szimmetria) (2.6)
3. $d(x, z) \leq d(x, y) + d(y, z)$ (háromszög–egyenlőtlenség) (2.7)

Ha (X, d) metrikus tér, X elemeit pontoknak, a $d(x, y)$ függvényt az X feletti metrikának vagy távolságfüggvénynek szokás nevezni.

2.4.2 Euklideszi távolság

A hagyományos euklideszi sík–, és térgeometria pontjai modellezhetőek valós számok rendezett n–eseivel, azaz n–dimenziós vektorokkal. Pl. a sík egy pontja megadható egy $P = (x, y)$ számpárral, a tér egy pontja egy $P = (x, y, z)$ számhármassal. Általában akárhány n-dimenziós térben is a pontok megadhatóak a $P = (p_1, p_2, \ldots, p_n)$ szám–n–essel, ahol a p_i számot a pont i–edik koordinátájának nevezzük. A valós szám–n–esek halmazát R^n–nel jelölve, értelmezhető a következő d_n–nel jelölt metrika:

$$d_n(P, Q) \coloneqq \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{2.8}$$

Az euklideszi távolságot definiálhatjuk pont és ponthalmaz között is:

$$\mathrm{d}(\mathrm{x}, \mathrm{A}) \coloneqq \min_{\mathrm{y} \in \mathrm{A}} \mathrm{d}(\mathrm{x},\mathrm{y}) \tag{2.9}$$

2.4.3 Probléma a színtér szegmentálása során

Az egyes objektumok színének meghatározásához célszerű a színterünk egyes színeit klaszterekbe foglalni. A művelet elvégzéséhez kiindulásként az egyes klaszterekbe alappontokat kell felvennünk, majd az RGB tér szegmentálását ezen alappontok mentén végezzük el. A szegmentálás során a tér minden egyes pontjára meg kell határozni az adott pont és az egyes klaszterek, mint ponthalmazok távolságát. Mivel az RGB tér nagyjából 16 millió pontot tartalmaz, így látható, hogy a 2.9-es képlet a színek száma miatt nem alkalmazható a gyakorlatban. Ez a fejezet ennek a problémának a megoldására kíván megoldást nyújtani.

2.4.4 Távolság transzformáció

Ebben a fejezetben egy az euklideszi távolság approximálására alkalmas technikát fogunk bemutatni [22,23]. A módszer neve távolság transzformáció, mely széles körben használható módszer, ugyanis a távolságmérés központi szerepet játszik a bináris képek összehasonlításában, az él-, illetve sarokdetektálásban, osztályozási feladatokban és térképészeti alkalmazásokban.

Alkalmazása azért előnyös számunkra, mert egész aritmetikát használva tudjuk a bináris képpontoknak a színklaszterektől mért távolságát közelíteni. Úgy is fogalmazhatunk, hogy a távolság transzformáció egy útkereső eljárás, amely meghatározza, hogy a színtér egyes pontjaiból kiindulva, melyik színklaszter érhető el a legrövidebb úton.

Legyen adva egy kétdimenziós bináris képünk $I(x,y)$, osszuk a képpontjait két halmazba (az objektumpontok-, illetve a háttérpontok halmazába).

$$\mathrm{I(x,y)} \in \{\mathrm{Ob, Bg}\} \tag{2.10}$$

Ennek a bináris képnek az $I_d(x,y)$ távolság távolságtranszformáltját úgy kapjuk, hogy minden egyes pixelét felcímkézzük a pixel és a hozzá legközelebb eső háttérpont távolságával. Matematikailag:

$$\mathrm{I_d(x,y)} = \begin{cases} 0, \text{ha } \mathrm{I(x,y)} \in \{\mathrm{Bg}\}, \\ \min(\|\mathrm{x}-\mathrm{x_0}, \mathrm{y}-\mathrm{y_0}\|, \forall \mathrm{I(x_0,y_0)} \in \mathrm{Bg}) \text{, ha } \mathrm{I(x,y)} \in \{\mathrm{Ob}\}, \end{cases} \tag{2.11}$$

ahol $\|x,y\|$ egy kétdimenziós metrika. Értelemszerűen különböző metrikák, különböző távolság transzformáltat eredményeznek. Az esetek többségében az 2.8-as képletben ismertetett euklideszi távolságot használjuk.

Ez egy izotróp metrika, melyben a mért távolságok függetlenek a tárgy helyzetétől, természetesen azzal a megkötéssel, hogy a tárgyak határai digitálisak így pontjaink csak diszkrét helyeken vannak értelmezve. A legnagyobb korlátozása az euklideszi metrikának az, hogy a digitális képfeldolgozás világában, komplex alakzatok esetében nehéz hatékonyan kiszámolni. Ennek érdekében több olyan közelítést is definiáltak, melyek a kétdimenziós digitális képek rácsszerkezetének köszönhetően könnyebben használhatók. Ezek közül az első a „city block”, vagy Manhattan távolság (lásd 2.9b. ábra), mely az L_1 normát használja:

$$\|\mathrm{x,y}\|_{\mathrm{L_1}} = |\mathrm{x}| + |\mathrm{y}|, \tag{2.12}$$

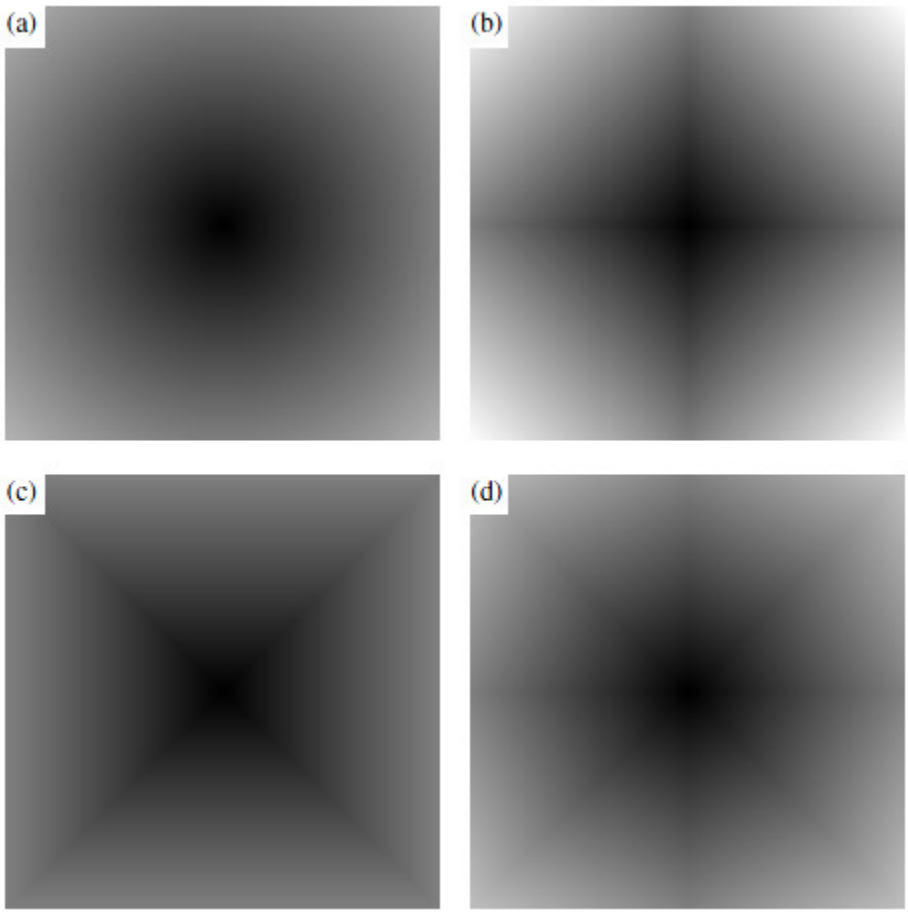
ahol a távolság az (x,y) pontba jutáshoz szükséges vízszintes és függőleges lépések számát jelenti. A technika egyik nagy hátránya, hogy csak a 4-szomszédok irányába tudunk lépni,

tehát egy átlós lépést, csak egy függőleges és vízszintes lépés segítségével lehet realizálni. Egy másik általánosan használt módszer, a sakktábla metrika (lásd 2.9c. ábra), mely a L_∞ normát használja:

$$\|\mathrm{x},\mathrm{y}\|_{\mathrm{L}_\infty} = \max(|\mathrm{x}|,|\mathrm{y}|) \qquad 2.13$$

Mely egy sakktáblán a királynak az (x, y) pontba jutáshoz szükséges lépéseinek darabszámát méri. Ezzel a módszerrel a pontok 8-szomszédait lehet elérni egy lépés során. Az euklideszi távolság közelítése érdekében különböző metrikák széles halmazát definiálták. Ezek közül több a Manhattan és a sakktábla távolság egyszerű számítási szabályain alapszik. Pl. az előző kettő átlagaként definiált hibrid metrika (lásd 2.9d. ábra):

$$\|\mathrm{x},\mathrm{y}\|_{\mathrm{Hibrid}} = \frac{1}{2}\left(|\mathrm{x}| + |\mathrm{y}| + \max(|\mathrm{x}|,|\mathrm{y}|)\right) \qquad 2.14$$



2.9. ábra. Grafikus összehasonlítása a különböző metrikáknak. Minden képen a képpontoknak a kép középpontjától mért távolságát szemléltetjük. (a) euklideszi távolság; (b) city block távolság; (c) sakktábla távolság; (d) hibrid metrika, az (b) és (c) keveréke.

2.4.5 Chamfer távolság transzformáció – kétmenetes algoritmus

A módszer segítségével a bináris képet kétszer bejárva – egyszer a bal felső sarokból indulva a jobb alsó felé, másodjára pedig a jobb alsótól indulva a bal felső felé - határozhatjuk meg annak a távolság transzformáltját. Ezzel a két bejárással határozzuk meg az egyes pixeleknek a tárgy felső-, és bal élétől mért, valamint a tárgy alsó-, és jobb élétől mért távolságát.

A bejárás során távolság maszkokat használunk. A maszk egy pixelnek és a hozzá legközelebbi szomszédjának távolságot adja vissza. Ezt két irányba futtatva egy olyan képet ad eredményül, ahol a tárgyak két széle között lévő legtávolabbi, vagyis középső pontok lokális maximumként jelentkeznek. Ha egy 3×3-as maszkot használunk, akkor az első bejárás az aktuális pixelnek a három felette lévő pixel és a tőle közvetlenül balra lévő pixel távolsága közül a legkisebbet adja vissza, képlettel:

$$\mathrm{I_d}(\mathrm{x},\mathrm{y}) = \min\begin{pmatrix}\mathrm{I_d}(\mathrm{x}-1,\mathrm{y}-1)+\mathrm{b}, & \mathrm{I_d}(\mathrm{x},\mathrm{y}-1)+\mathrm{a}, & \mathrm{I_d}(\mathrm{x}+1,\mathrm{y}-1)+\mathrm{b}, \\ \mathrm{I_d}(\mathrm{x}-1,\mathrm{y})+\mathrm{a} & & \end{pmatrix}, \quad 2.15$$

ahol az a és b növekmények értéke attól függ, hogy a képpontok 4-, vagy 8-szomszédossági relációban vannak-e egymással. A háttérpontoknak előzetesen 0 értéket adtunk.

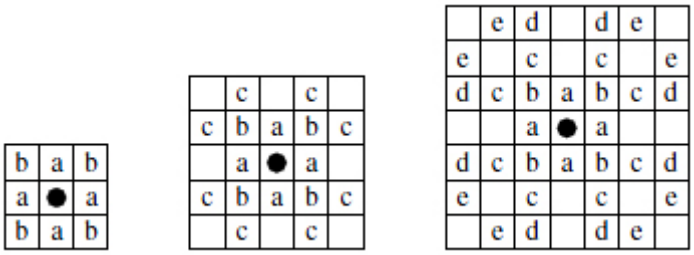
A második bejárás az aktuális pixelnek a három alatta lévő pixel és a tőle közvetlenül jobbra lévő pixel távolsága közül a legkisebbet adja vissza. A második bejárás csak akkor írja felül a távolság értéket, ha az kisebb az első körben számított értéknél. Ugyanis ebben az esetben a vizsgált pixel közelebb helyezkedik el a tárgy alsó-, vagy jobb széléhez:

$$\mathrm{I_d}(\mathrm{x},\mathrm{y}) = \min\begin{pmatrix}\mathrm{I_d}(\mathrm{x},\mathrm{y}), & \mathrm{I_d}(\mathrm{x}+1,\mathrm{y})+\mathrm{a}, & \mathrm{I_d}(\mathrm{x}-1,\mathrm{y}+1)+\mathrm{b}, \\ \mathrm{I_d}(\mathrm{x},\mathrm{y}+1)+\mathrm{a}, & \mathrm{I_d}(\mathrm{x}+1,\mathrm{y}+1)+\mathrm{b} & \end{pmatrix}. \quad 2.16$$

Az a és b növekmények különböző értékei, különböző metrikákat eredményeznek. A city block távolságot, az $a = 1$ és $b = 2$ értékekkel realizálják, a sakktábla távolság esetében $a = b = 1$. Az euklideszi távolságnak egy jobb közelítését kapjuk, ha az $a = 3$ és $b = 4$ növekményeket használjuk, és az eredményt elosztjuk 4-el. Ez egy a 2.9d. ábrán látható nyolcszög alakú mintát eredményez.

Általánosságban elmondható, hogy a növekmények optimalizálásával egyre pontosabb távolságokat kapunk, de a közelítés pontossága nagyban függ a maszk méretétől is. Ugyanis egy nagyobb maszk segítségével több utat tudunk összehasonlítani, 3×3-as esetben 4-et, 5×5-ös esetben 8-at, 7×7-es esetben pedig 16-ot (lásd 2.10. ábra). Egy 5×5-ös maszk

elfogadható kompromisszumot szolgáltat a számítás komplexitása és a közelítés pontossága között. A Chamfer távolság transzformáció műveleteinek száma minden egyes pixel esetében fix, továbbá elmondható még, hogy az időbonyolultsága a kép pixeleinek számával arányos.



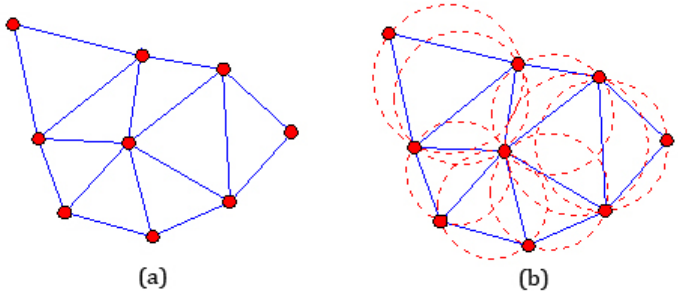
2.10. ábra. A növekmények helyei a 3×3-as, 5×5-ös, 7×7-es méretű maszkokon belül. Az üres helyek nem vesznek részt a számításban.

2.5 Voronoi felosztás

Színterünk szegmentálása során a klaszterek alappontjai köré egy olyan rácsszerkezetet szeretnénk kialakítani, melynek minden belső pontja közelebb van az adott klaszter alappontjaihoz, mint az összes többi ponthoz [24]. Ezáltal minden egyes klaszterhez egy konvex sokszöget rendelünk hozzá, melyek az azonos színűnek látott pixeleket foglalják magukban. Ez a fejezet az előbb felvázolt probléma megoldására ismertet egy Voronoi felosztásnak nevezett módszert.

2.5.1 Delaunay háromszögelés

Egy adott *P* ponthalmaz Delaunay-háromszögelése egy olyan egyenes szakaszokból álló vonalhálózat, aminek sokszögtartományai köré írt gömbjei csak határukon tartalmazzák a *P* ponthalmaz pontjait. A korlátos sokszögtartományok tehát húrsokszögek. Ha ezek a tartományok nem mind háromszögek, akkor a Delaunay-háromszögelés elfajuló, egyébként valódi (lásd 2.11. ábra). A Delaunay-háromszögelés akkor és csak akkor valódi, ha a *P* halmaz pontjai között semelyik három nincs egy egyenesen, és semelyik négy nincs egy körön. Az élek száma lineárisan függ a pontok számától. A *P* ponthalmaz valódi Delaunay-háromszögelésének fontos tulajdonsága, hogy a *P* halmaz összes háromszögelés között maximalizálja a háromszögek legkisebb szögét.



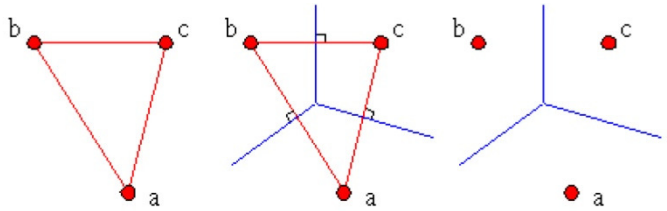
2.11. ábra. (a) Delaunay háromszögelés; a (b) ábrán látható, hogy a háromszögelés valódi Delaunay-háromszögelés.

2.5.2 Voronoi diagram

Legyenek a síkon (térben) szabálytalan elrendezésű pontjaink. Minden pont köré szerkeszthető egy olyan sokszög (poliéder), melynek belső pontjai (összes pontja a határát alkotó pontok kivételével) közelebb vannak a kérdéses ponthoz, mint az összes többi ponthoz. Az ilyen tulajdonsággal rendelkező sokszögek (poliéderek) konvexek és folytonosan töltik ki a síkot (teret). A meghatározásból következik, hogy a sokszög oldalai (a poliéder

oldallapjai) merőlegesek a körülvett pontot a többi ponttal összekötő egyenesekre és felezik azokat. Tulajdonképpen a Voronoi diagram egy pont halmaz közelségi diagramjaként fogható fel, mely segítségével a teret úgy oszthatjuk fel, hogy a tartomány minden pontja közelebb van a tartomány pontjához, mint bármely más ponthoz.

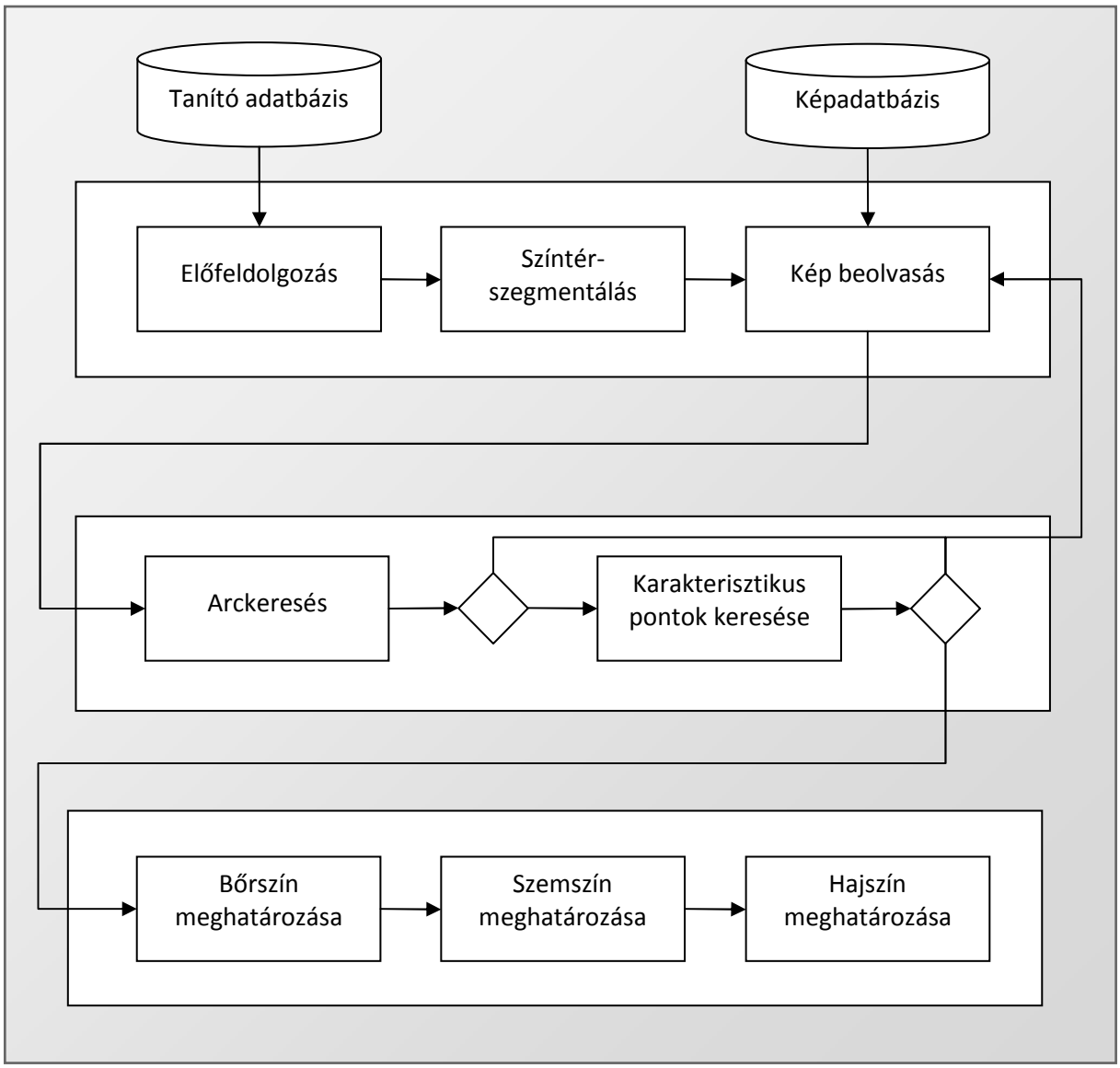
A Voronoi diagram valójában a Delaunay háromszögelés geometriai duálisa. Duális lévén egyik a másikból megkapható. A módszer az, hogy a Delaunay háromszögek éleinek felezőmerőlegeseit behúzzuk (lásd 2.12. ábra).



2.12. ábra. Delaunay háromszög Voronoi diagrammá alakítása.

3 Arci jellemzők szegmentálása

A rendszer tervezése és fejlesztése során az eddigiekben ismertetett módszerek közül több is megvalósításra került, ezek közül a végleges alkalmazás a valamely szempontból jobbnak bizonyult eljárásokat használja, amelyek e fejezetben kerülnek felhasználásra. Mivel a cél egy valós idejű rendszer tervezése, így az algoritmusok kiválasztásánál, illetve fejlesztésénél a fő szempont azok sebessége volt, de természetesen a hatékonyság is hasonló jelentőséggel bír. Ezen megfontolások alapján a rendszer vázát három fő modul képzi, amelyek az adatbázisból kapott kép alapján képesek arc–, illetve annak a karakterisztikus pontjainak a detektálására, valamint ezek alapján bizonyos adottságok meghatározására (lásd 3.1. ábra).



3.1. ábra. A rendszer szerkezeti felépítése.

3.1 Előkészületek

A dolgozat elsődleges célja az volt, hogy egy arcokat tartalmazó adatbázisból – meta információk használata nélkül – lekérdezhetők legyenek bizonyos arci paraméterek, mint pl. a bőrszín, a szemszín, és a hajszín. Az előbbi kérdések megválaszolásához, rendelkeznünk kell azokkal az ismeretekkel, hogy a bemeneti kép melyik területén kell vizsgálatokat folytatnunk.

Az arci jellemzők színének meghatározásához első lépésben arcot kell detektálni az adatbázisbeli képeken. Az arcdetektálásnak széles irodalma van, számos megoldást publikáltak ebben az irányban. Talán az egyik legsikeresebb eljárás, a Viola és Jones által kidolgozott módszer, mely hatékonyságát tekintve nem marad alul a többivel szemben, de a gyorsaságát tekintve felülmúlja az összes eljárást.

A 2.3-as fejezetben ismertetett Viola-Jones detektorok segítségével elég nagy találati arány mellett végezhető el az arc detektálása, de még így is születhetnek hamis találatok. Ezek azonban könnyen kiszűrhetők a következő algoritmussal: egy Viola-Jones szemdetektorral detektáljunk szemet az arcdetektor által visszaadott képrégió felső felében. Ha a szemdetektor nem ad vissza egyetlen találatot sem, akkor vessük el a detektált arcot hamis pozitívként, ellenkező esetben fogadjuk el pozitív találatként. Ezzel a módszerrel egyrészt a szivárványhártya színének meghatározásához szükséges régiót is megkapjuk, másrészt az arcdetektor egyfajta validálásának is felfogható a szemdetektálás.

A fentebb vázolt módszerrel tehát képesek vagyunk a keresett objektumok színének meghatározására. Az egyes jellemzők szegmentálásához további előkészületek szükségesek, melyeket az alábbi fejezetekben külön-külön ismertetek. A szegmentálás eredményét a színmeghatározó modulban fogjuk felhasználni (lásd 4. fejezet).

Habár a szegmentálás nagyrészt szerkezeti információkon alapszik, mégis néhány esetben rosszul szegmentált régiókat eredményezhet. Azonban jó végeredmény elérésére vagyunk képesek, ha ezt az eredményt a színtér szegmentálása során szerzett tapasztalatokkal vegyítjük.

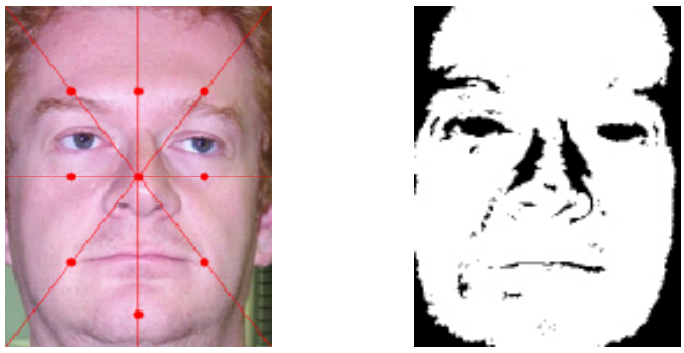
3.2 Az arcbőr szegmentálása

Az arcbőr szegmentálása során az a célunk, hogy a bőrt elkülönítsük a detektált arc többi részétől (pl. szem, haj, arcszőrzet), majd a színmeghatározó modul ezek közül a szegmentált pixelek közül fogja a leggyakrabban előfordulót bőrszínnek választani.

Abból a feltevésből indulunk ki, hogy az arcon az arcbőr alkotja a legnagyobb hasonló világosságkóddal rendelkező összefüggő régiót – olyan régió, melyben a színek közötti távolság kisebb, mint egy tapasztalati úton számított küszöbérték.

Az arcbőrrégió meghatározásához egy régiónövelő eljárást fogunk használni. Az algoritmus szétválogatja az arcbőrhöz tartozó pixeleket, azoktól a részektől ahol az arcbőr nem látható, vagy azoktól ahol annak a színe megváltozik (pl. rossz megvilágítás miatt). Az eljárás menete nagyon hasonló a számítógépes grafikában alkalmazott üregkitöltő algoritmusokéhoz, amikor egy homogén színű zárt tartományt töltünk ki egy adott színnel.

- A különböző megvilágítási hatások és az arc helyi gödrei miatt a régiónövelést több különböző pontból indítjuk (lásd 3.2a. ábra).
- Minden egyes lépésben a külső határpontok világosságkódját hasonlítjuk össze az addig elkészült régió átlagolt színével. Azokat a pontokat adjuk hozzá a régióhoz, melyek színének távolsága a régió átlag színétől, tíz százalék alá esik.
- A régiónövelések által kapott régiókat egyesítjük egy közös maszkba, amit a továbbiakban az arcbőr helyének tekintünk (lásd 3.2b. ábra).



(a) (b)

3.2. ábra. A bőrrégió meghatározása: (a) a régiónövelés kezdőpontjai, (b) a régiónövelés által eredményezett bináris maszk.

3.3 A szem szegmentálása

A szemszín meghatározása komplikáltabb, mint a bőrszíné. Ugyanis ebben az esetben nem járható út az, hogy a szemrégió színét számítjuk ki. Ebben az esetben előbb a szemrégión belül elhelyezkedő szivárványhártyát kell megkeresnünk, hiszen ténylegesen ez határozza meg a szem színét.

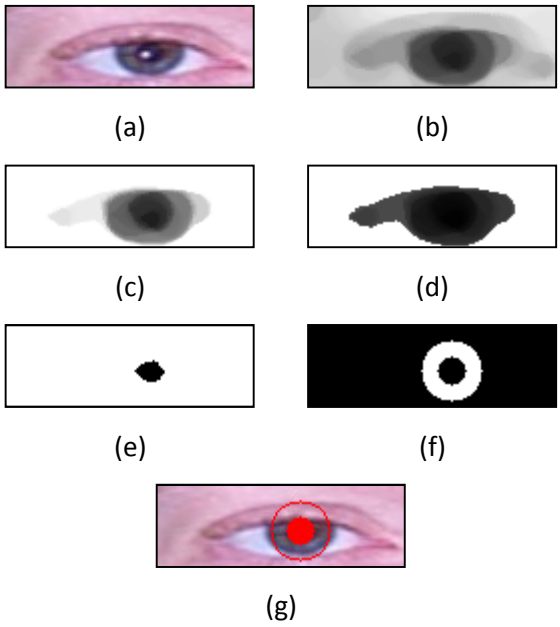
Az algoritmus első lépésében egy Viola-Jones detektor segítségével szemet detektálunk az arcon (lásd 3.3a. ábra). A hamis találatok kiszűrése érdekében a szemdetektálást az arc bal felső negyedébe korlátozzuk, ily módon csak a bal szemet keressük. Emlékeztetőül: ha a detektor nem talált szemet az arcon, akkor az arcot hamis találatként értelmezzük és egyetlen vizsgálatokat sem végzünk rajta. Következő lépésben a pupilla és a szivárványhártya régió pontos helyét keressük meg a szemdetektor által visszaadott szemen:

- A zajok (pl. szempilla, szemöldök) kiküszöbölése érdekében egy mediánszűrést (11×11-es maszkkal) végzünk a képen.
- Átkonvertáljuk a szemet tartalmazó képet CIE XYZ színtérbe, majd kimentjük a Z csatornát egy egycsatornás – magyarán szürkeskálás – képbe (lásd 3.3b. ábra). A „Z” csatorna használatával egy sokkal kontrasztosabb képet kapunk, mintha egyszerűen csak szürkeskálás képpé konvertálnánk a szemet tartalmazó képet [25,4]. Ez a lépés azért hasznos számunkra, mert a szemszín meghatározó algoritmusunk a szemrégiót alkotó viszonylag kontrasztos szivárványhártya–pupilla részek és a kevésbé kontrasztos szemhéj és környezete részek szétválasztásán alapul. Így azt szeretnénk elérni, hogy e részek között, a kontrasztosságot tekintve a lehető legnagyobb legyen a különbség.
- További kontrasztnövelést hajtunk végre a szemen (lásd 3.3c. ábra).
- A szürkeskálás képen a pupilla és szivárványhártya részek általában sötétebbek, mint a szem összes többi része. Egy hisztogram-kiegyenlítést végzünk ezen a képen, abból a célból, hogy pupilla és szivárványhártya részek az összes képen a legsötétebb intenzitásértékkel legyenek kigyújtva (lásd 3.3d. ábra).
- Majd az így kapott eredményképen egy küszöbölés segítségével szétválasztjuk a pupilla és szivárványhártya részeket a szem többi részétől (20-as szürkeségi küszöbértéket használtunk).

- A kisebb zajokat egy erózió morfológiai operátor segítségével eltávolítjuk a képről (11×11-es téglalapalakú maszkot használtunk) (lásd 3.3e. ábra).
- A szivárványhártya és a pupilla elkülönítésére egy kör alakú maszkot használunk (lásd 3.3f. ábra), melynek sugarát a Viola-Jones detektor által visszaadott ablak dimenziója alapján számítottuk ki. Abban az esetben, ha a visszaadott ablak $W \times H$ méretű, akkor a kör alakú maszk sugara:

$$R_{mask} = H * 0.8$$

A maszkot az előző lépés által visszaadott képen a fekete színű pixelek súlypontjára helyezzük rá (lásd 3.3g. ábra).

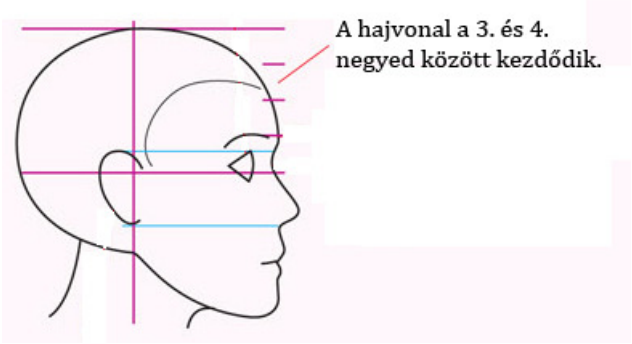


3.3. ábra. A szivárványhártya detektálása: (a) a detektált szem, (b) XYZ transzformáció, (c) kontrasztnövelt kép, (d) hisztogramkiegyenlített kép, (e) küszöbölés utáni eredmény, (f) a szivárványhártyához tartozó maszk, (g) maszkolt kép.

3.4 A haj szegmentálása

A számos hajviseletnek, stílusnak, hajszínnek és egyéb tényezőknek (pl. kopaszodás) köszönhetően, a hajszín meghatározása során koránt sincs annyira egyszerű dolgunk, mint az előző fejezetekben. Ebből kifolyólag lesznek olyan esetek, amikor a hajszín nem állapítható meg egyértelműen. A hajszín meghatározásához először a bőrrégiót kell megtalálni, mivel feltételezhető, hogy a bőrrégió feletti félhold alakú területben helyezkedik el a haj. A haj szegmentálásának lépései, a következők:

- Első lépésben átméretezzük a képeket úgy, hogy az arc mérete 400×400-as legyen. Majd az átméretezett képeken egy mediánszűrést végzünk (7×7-es maszkkal) az apró zajok eltávolításának érdekében.



3.4. ábra. A hajvonal kezdete a fejen. A hajvonal a szem alatt kezdődő, a fej tetejéig húzódó régió 3. és 4. negyede között helyezkedik el.

- Következő lépésben egy félhold alakú maszkot illesztünk az arcbőrt tartalmazó régió tetejére (lásd 3.5a. ábra), melynek a kiterjedését és centrumát az emberi fej biológiai arányai alapján számítjuk ki. Ha a Viola-Jones arcdetektor által visszaadott régió W hosszú és H magas, akkor a félhold alakú tartomány dimenzióit a következő formula alapján számíthatjuk ki:

$$W_{mask} = W * 0.8, H_{mask} = H * 0.4$$

- A félhold alakú régió ívének szintén az arc biológiai arányai miatt a fej felső felének 3. és 4. negyede között kell mennie (lásd 3.4. ábra).

- A hajat egy az arcszín szegmentálása során ismertetett régiónövelő eljárás segítségével szegmentáljuk. Első lépésben a félhold alakú régióban keressük meg a legnagyobb összefüggő komponenst, öt véletlenszerűen kiválasztott pont esetében. Második

lépésben elvégzünk egy újabb régiónövelést abból a pontból kiindulva, amelyikből a legnagyobb összefüggő komponenst kaptuk, de ezúttal nem korlátozzuk le az eljárást a félhold alakú maszkra (lásd 3.5b. ábra).

(a) (b)

3.5. ábra. A hajrégió megkeresése: (a) a félhold alakú maszk, (b) a régiónövelő eljárás eredménye.

4 Színinformáció kinyerése

A színterünk elméletileg végtelen, de a gyakorlatban is óriási mennyiségű szín reprezentálását teszi lehetővé. Azonban a legtöbb alkalmazás szempontjából elegendő lenne csak véges sok szín használata. Gondoljunk bele, hogy hány különböző hajszínt szoktak az emberek megnevezni? Ebből kifolyólag a gyakorlatban használható közel 16 millió színt tartalmazó színterünknek elegendő csak egy kis részhalmazát használni a lekérdezések során [26]. Ennek érdekében a színtéren olyan osztályozást kell definiálnunk, amelyeknek eredményeként létrejött osztályokat a lekérdezések során, az emberi érzékelés szempontjából egy közös címkével látunk el. Magyarán a színtér minden egyes elemét színkategóriákba soroljuk be. Ennek a leképezésnek a menetét a színmodellünkben definiáljuk.

A színmodellünkben a Broek által kifejlesztett szabályrendszert használjuk fel [27]. Az alapötlet az, hogy a folytonos színterünket humán megfigyelések alapján képezzük le diszkrét részhalmazokra. Vagyis az osztályozást pszichológiai kísérletek alapján végezzük el Emberek véleményét kértük ki, hogy milyennek látják a tanító adatbázisbeli képeken szereplő emberek bőr–, szem–, és hajszínét. Az eredmények összegzése és feldolgozása után az adatbázisbeli képek mindegyikéhez három címkét rendeltünk, melyek mindegyike az általunk kérdezett objektum színének nevét takarja. Azonban a színtér szegmentálásához nem szöveges információkra, hanem konkrét világosságkódokra van szükségünk, így a tanító adatbázisbeli képekhez rendelt szöveges címkéket le kell képeznünk a színterünk színeire.

A színtér szegmentálásának első lépésében tehát a tanító adatbázisbeli képek és a pszichológiai kísérletek alapján csoportosítjuk a színeket, majd a tényleges lekérdezések során ennek a csoportosításnak az eredményét használjuk fel a színinformáció kinyerésére.

4.1 Színek csoportosítása

A színtér szegmentálásának első lépésében színkategóriákat kell definiálnunk, a pszichológiai kísérletek elvégzéséhez. A kísérletek két fázisból állnak. Az első fázisban ki kell jelölnünk a lehetséges színkategóriákat. Minden egyes jellemzőhöz öt kategóriát rendeltünk [28,29], melyek az 4.1. táblázatban láthatók. A kategorizálás speciális jelleméből fakadóan a szóba jöhető színek nem fedik le teljesen a színterünket, így minden egyes jellemzőhöz definiáltunk még egy hatodik kategóriát is, melybe az összes többi színt soroltuk be.

Bőr	Szem	Haj
nagyon világos	kék	szőke
világos	szürke	aranybarna
átmeneti	zöld	barna
sötét vagy barna	barna	fekete vagy sötétbarna
nagyon sötét	sötétbarna	szürke vagy fehér
nem bőrszín	nem szemszín	nem hajszín

4.1. táblázat. A színkategóriák.

A második fázisban 40 emberrel töltettünk ki egy internetes kérdőívet (lásd 4.1. ábra). A kérdőív 40 oldalból állt és minden egyes oldal egy szemből fényképezett arcot (800×600-as felbontással) tartalmazott, melyről a kísérletben résztvevőknek el kellett dönteniük, hogy az arc egyes jellemzőinek színe az 4.1. táblázatban megadott kategóriák közül melyikbe tartozik.

A fényképeket több arcadatbázisból [30,31] és az internet segítségével a Google képkereső szolgáltatásával gyűjtöttük össze. A megkérdezettek véleményének összesítésével megkaptuk az egyes jellemzők színkategóriáinak alapelemeit. Magyarán minden adatbázisbeli képhez három címkét rendeltünk, melyek az adott jellemzőknek az emberek többsége által látott színét jelenti.

Bizonyos esetekben, a kísérletben résztvevő emberek egymásnak ellentmondó módon szavaztak a jellemzők színére. A válaszokat két csoportba osztottuk: külön csoportba kerültek azok a képek, melyek a szavazás során minden egyes jellemzőjük esetén a nyertes színkategória 50% feletti eredményt ért el. Itt ugyanis a többi színkategóriával vett különbség erősen szignifikáns, így a felcímkézett jellemzőt eredményesen használhatjuk fel a színtér szegmentálása során. A másik csoportba azok a képek kerültek, melyek jellemzőinek a színe

a megkérdezettek szavazatai alapján egyértelműen nem dönthető el. Ezeket a képeket kivettük az adatbázisból, így azok nem vesznek részt a további feldolgozások során.

Mint már említettem a színtér szegmentálásához nem szöveges információkra, hanem konkrét világosságkódokra van szükségünk, így a tanító adatbázisbeli képekhez rendelt szöveges címkéket le kell képeznünk a színterünk színeire. Magyarán a minden egyes jellemző, minden egyes címkéjéhez egy RGB értéket kell rendelnünk.

A leképezés első lépésében szegmentáljuk az adott jellemzőt, tehát meghatározzuk azt a régiót, amely a képen az jellemzőt tartalmazza. Majd következő lépésben az elkészült bináris maszkot felhasználva átlagoljuk a maszk alatti RGB értékeket. A címkékhez az átlagolt RGB értékeket rendeljük hozzá. Az azonos címkével rendelkező átlagolt RGB értékeket – jellemzőkként csoportosítva – a címke alapján klaszterekbe foglaljuk, majd a továbbiakban e klaszterek segítségével végezzük el a színtér szegmentálását.

4.1. ábra. Az internetes kérdőív.

4.2 Színtér szegmentálása

Az előző fejezet során előállltak azok a klaszterek, melyek alapján az RGB tér szegmentálása elvégezhető. Az RGB tér minden pontjára, egyesével fogjuk meghatározni, hogy az melyik klaszterhez tartozik. A kapott eredményeket egy CLUT (Color LookUp Table – Szín Visszakeresési Táblázat) nevezetű táblázatban tároljuk, hogy az időigényes szegmentálást csak egyszer, az inicializáló fázisban kelljen elvégezni. A keresés során ennek a CLUT–nak a közvetlen címzésével határozható meg az, hogy az adott intenzitásértéket az emberek többsége milyen színnek látja.

Az RGB tér szegmentálását végezhetnénk úgy, hogy kiszámoljuk minden egyes pontjának, a minden egyes klasztertől mért euklideszi távolságát (lásd 2.4.2 fejezet), majd ahhoz a klaszterhez soroljuk be a pontot, amelyik klaszterhez a legkisebb távolság adódott. Azonban ez az RGB tér elemszáma miatt meglehetősen lassú módszer, továbbá a másik probléma az, hogy az egyes csoportok között nem lesznek letisztultak a határok.

A szín–klaszter összerendelési problémára az előző bekezdésben ismertetett eljárás helyett, a szegmentálást a HSI sík két 2D-s vetületével végezzük el. A szegmentálásnak ez a típusa azért hasznos számunkra, mert a színeink két nagy csoportba oszthatók: vannak akromatikus színek (pl. fehér, szürke, fekete), illetve kromatikus színek (pl. a kék, zöld, piros). A HSI térben az akromatikus színek leírásában nem játszik jelentős szerepet a H koordináta, így az elhagyható az akromatikus–, és a kromatikus színcsoportok szeparálása során. A kromatikus színek valódi színét pedig elsősorban a H és az I koordináták határozzák meg (pl. az RGB tér alap–, és kiegészítő színeinek a telítettsége egyenlő eggyel), így az S koordináta minden további jelentős következmény nélkül elhagyható a kromatikus színek szegmentálása során [27]. Ennek megfelelően a három dimenzióban történő szegmentálás helyettesíthető, két kétdimenziós sík szegmentálásával. Magyarán a 3D-ban történő szeparáló hipersíkok keresése helyett, csak kétdimenziós szeparáló síkok egyenletét kell meghatároznunk [32].

Tehát a színtér szegmentálása az alábbi két lépés segítségével oldható meg:

- Először elvégezzük az akromatikus és kromatikus színek szeparálását azzal, hogy a 3D-s színterünket rávetítjük az SI síkra, majd az alappontok alapján elvégezzük az SI sík szegmentálását.

- Másodszor elvégezzük a kromatikus színek szegmentálását azzal, hogy rávetítjük a 3D-s színterünket a HI síkre, és az alappontok mentén elvégezzük az HI sík szegmentálását.

Mivel a szegmentálás a HSI színtérben történik, viszont a bemeneti képeink és a klasztereink alappontjai RGB színtérben vannak értelmezve, így első lépésben le kell képeznünk az RGB színterünk elemeit a HSI színtérre. A leképezést az alábbi képlet segítségével végeztük el [4]:

$$H = \begin{cases} Nincs\ értelmezve, ha\ S = 0 \\ 360° - cos^{-1}\left(\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)*(G-B)]^{1/2}}\right), ha\ \frac{B}{I} > \frac{G}{I}, \\ cos^{-1}\left(\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)*(G-B)]^{1/2}}\right), egyébként. \end{cases}$$

$$S = \begin{cases} Nincs\ definiálva, ha\ I = 0, \\ 1 - \frac{3}{R+G+B} * \min(R,G,B)\ , különben, ahol \end{cases}$$

$$I = \frac{R+G+B}{3}.$$

4.2. táblázat. Az RGB-HSI konverzió

A 2D-s HSI síkok szegmentálása az alább ismertetett algoritmus segítségével történik:

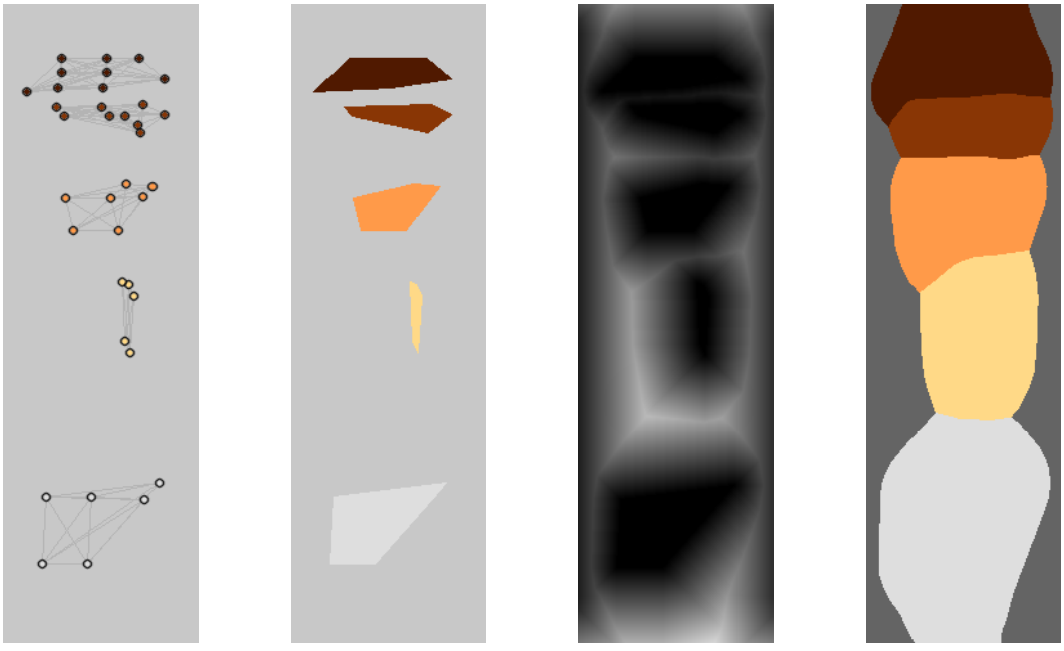
- Vegyük a klaszterek alappontjait, melyeket a megfelelő koordináta elhagyásával ábrázoljunk a fentebb említett síkokon (lásd 4.3a. ábra).
- Második lépésben elkészítjük a síkokon felvett alappontok konvex burkait (lásd 4.3b. ábra) és a burkokat alkotó pontokat hozzávesszük az eredeti klaszterekhez. Ennek reprezentálására egy bináris képet használtunk, ahol a 0-ás érték jelentette azt, hogy az adott pixel valamelyik klaszterhez tartozik és a 1-es érték jelentette azt, hogy a pixel egyik klaszterbe sem esik bele (tehát nem alappont és nem is pontja egyik konvex buroknak sem).
- A pszichológiai kísérletek eredményét befolyásoló emberi tényező miatt lehetnek olyan konvex burkaink, melyek átfedik egymást. Ezen átfedéseket meg kell szüntetni, mivel azt az érzetet keltik, hogy az átfedésben lévő klaszterekhez tartozó pontok egy klaszterhez tartoznak, nem pedig több különbözőhöz. Így az egymást átfedő klaszterekből elhagyjuk a közös részeket.

- Következő lépésben az RGB tér, HSI térbe transzformált pontjait, az előbb ismertetett konvex sokszögek pontjaihoz kell mérni. A vizsgált pontot abba a klaszterbe kell sorolni, amelyhez a legközelebb esik. Ennek eldöntésére egy Chamfer távolság-transzformációt (lásd 2.4.5. fejezet) hajtunk végre az előbbi bináris képen a lentebb látható bináris maszkkal (lásd 4.2. ábra).

	11		11	
11	7	5	7	11
	5	p	5	
11	7	5	7	11
	11		11	

4.2. ábra. A távolság-transzformáció során használt maszk.

- A távolság-transzformáció eredményeképpen egy szürkeskálás képet kapunk (lásd 4.3c. ábra), melynek a Voronoi felosztását (lásd 2.5. fejezet) elkészítve megkapjuk az egyes klaszterek között húzódó határoló egyeneseket. Az eljárás eredménye a 4.3d. ábrán látható.



(a) (b) (c) (d)

4.3. ábra. A HI sík szegmentálásának lépései a hajszín esetében: (a) a klaszterek alappontjai, (b) a konvex burkok, (c) a távolság-transzformált, (d) a távolság-transzformált Voronoi felosztása.

4.3 Szín meghatározása

Az egyes jellemzők színét a szegmentált terület alatti domináns szín (a legnagyobb gyakorisággal előforduló szín) határozza meg. Ezt a színt a legegyszerűbben úgy kaphatjuk meg, ha megkeressük a kép hisztogramjának a maximumát. Mivel az eredeti kép a háromdimenziós RGB színtérben van definiálva, így az előbb ismertetett maximumot egy háromdimenziós hisztogramban (16 millió elem közül) kellene megkeresni, ami elég nehéz és számításigényes feladat.

Ahelyett hogy a teljes színtérben keresnénk meg a domináns színt, majd a CLUT-ban megkeresnénk a hozzátartozó címkét, sokkal jobban járunk, ha a következő eljárást választjuk: a szegmentált jellemzők képpontjainak színét helyettesítsük a CLUT-ban hozzájuk tartozó színkategóriák középszíneivel. Az előbbi eljárás eredményeként egy olyan színhisztogramot fogunk kapni, amely csupán öt elemből fog állni (az öt színkategória középszínéből). Végezetül tehát az adott jellemző színét a legnagyobb számban előforduló színkategória határozza meg (lásd 4.4. ábra).

4.4. ábra. A színek számának csökkentése az arcbőr esetén.

Tehát a színinformáció alapján történő keresés teljesen analóg módon történik a szegmentálással. Első lépésben a kép sorfolytonos bejárásával az egyes RGB pontok, HSI térbe történő áttranszformálása történik. Az így kapott HSI színtérbeli pontoknak először kromatikus–akromatikus besorolását végezzük el, az SI síkra vetítésükkel. Ezután az alábbi két eset állhat fent:

- A vizsgált RGB pont akromatikus: ekkor az RGB pontnak megfelelő szín, az akromatikus értékeket tartalmazó CLUT–ból kiolvasott szín lesz.

- A vizsgált RGB pont kromatikus: ez esetben szükség van a kromatikus osztályok közötti besorolásra (mely az RGB pontból nyert térbeli HSI pont) HI síkra történő vetítésével kapható meg. Az RGB pont tényleges színe, a kromatikus értékeket tartalmazó CLUT–ból kiolvasott szín lesz.

5 Összefoglalás és kitekintés

5.1 Eredmények

A fentebb ismertetett színreprezentáció segítségével egy megbízható színkinyerő módszert lehet megvalósítani. A kinyert színek segítségével az arcok osztályozhatóvá válnak. A vizuális információn alapuló keresőrendszerünk validálására egy újabb pszichológiai kísérletet végeztünk (melynek menete ugyanaz volt, mint amit a 4.1. fejezetben ismertettem). A kísérlet során egy 20 képből álló adatbázist használtunk, melyekről a kísérletben résztvevőknek el kellett dönteniük, hogy az egyes arci jellemzőket melyik előre definiált színkategória (lásd 4.1. táblázat) írja le a legjobban. Ugyanezekre a képekre lefuttattuk a keresőrendszerünket is, mellyel minden egyes jellemzőre meghatároztunk egy százalékos értéket. Ez az érték azt adja meg, hogy a szegmentált terület hány százalékát alkotják a domináns színkategória pixelei.

Következő lépésben egy összehasonlítást végeztünk a kísérletben résztvevők véleményére és a program kimenetére vonatkozólag. A két eredmény közötti kapcsolatot a függelék 9.3. fejezetében összesítettem. Látható hogy a keresőrendszerünk által jósolt színkategóriák több mint 75%-ban megegyeznek a megkérdezettek véleményével. Sőt sok esetben csak egy színárnyalatnyi különbség mutatkozik a két eredmény összehasonlítása során (pl. a program sötétbarna helyett, barnának detektálja a bőrszínt). Ha az egy színárnyalatnyi eltéréseket is pozitív találatként fogadjuk el az értékek összehasonlítása során, akkor a keresőrendszerünk 95% feletti helyes színdetektálási elérésére képes. A két halmaz közötti korreláció átlagosan 0.73, továbbá jellemzőkre lebontva a következő értékek adódnak: 0.79 az arcbőrre, 0.61 a szemre és 0.78 a hajra. A viszonylag rossz eredmény a szem esetében a jellemző alacsony felbontásának köszönhető. A szemet tartalmazó kép felbontásának növelésével nő a korreláció, de egyben nő a keresőrendszer időigénye is (mert a szem méretének növekedésével egyenes arányban nő az arc mérete is).

A fentebb ismertetett vizuális információn alapuló keresőrendszerünket használhatjuk egy tartalom-alapú képkereső rendszer automatikus indexelésére is. Ennek érdekében a színkinyerő alkalmazásunkat integráltuk egy web-alapú képkereső rendszerbe. Az elkészült webalkalmazás két weboldalból és egy képadatbázisból áll. A felhasználóknak lehetősége van ebbe a képadatbázisba új képeket feltölteni. A feltöltött képeket paraméterként adjuk át a színkinyerő alkalmazásunknak, mellyel meghatározzuk az egyes arci jellemzők színét és a

kép nevével együtt eltároljuk azokat egy meta leírófájlban. A webalkalmazás főoldalán az adatbázisbeli képeket listázhatjuk ki az előbbi meta leírófájl alapján. Tehát az egyes arci jellemzőkhöz színkategóriákat választhatunk ki, mely alapján lekérdezéseket hajthatunk végre az arcbőr-, a szem-, és a haj színére vonatkozólag. Természetesen a rendszer elérhető az interneten[1] keresztül is.

[1] http://www.inf.unideb.hu/ipgd/FaceColorSegmentation

5.2 Összefoglalás

A 2.3 fejezetben megmutattam az objektumdetektálás egy megközelítését, amely alacsony számítási idővel magas detektálási arányt képes elérni. Mindezt három fontos tulajdonságán keresztül láthattuk.

Az első fontos tulajdonság a jellemzők számításához használt integrál kép reprezentáció, mely jelentősen csökkenti a kezdeti képfeldolgozást. A második fontos tulajdonság az AdaBoost-on alapuló jellemző kiválasztás, ami egy agresszív és hatékony technika a jellemző kiválasztásra. A harmadik fontos tulajdonság az osztályozók kaszkádja, mely radikálisan csökkenti a számítási időt, míg növeli a detektálási pontosságot.

A 3. fejezetben az egyes arci jellemzők leírására alkalmas szegmentációs módszereket ismertettem, melyek segítségével alacsony számítási idő alatt érhetünk el jó eredményeket. A módszer azon a tényen alapszik, hogy a Viola-Jones detektorok segítségével viszonylag alacsony számítási idővel tudjuk elérni a vizsgált jellemzők magas detektálási arányát.

A 4. fejezetben ismertettem, hogy az adott jellemző színét a neki megfelelő szegmentált terület alatti domináns szín írja le. A jobb használhatóság érdekében drasztikusan csökkentettük a keresőrendszerben felhasználható színek számát. A teljes színtér és az arci jellemzőknek megfelelő színkategóriák közötti kapcsolat leírására egy szín modellt definiáltunk, melyet humán megfigyelések alapján készítettünk el.

A színeknek e reprezentációja könnyebbé és gyorsabbá tette az arci jellemzők színének meghatározását. A fentebb ismertetett színkinyerő alkalmazás működési elvét demonstrációs célból beépítettük egy tartalomalapú keresőrendszerbe, amely egy képadatbázist felhasználva képes a bőr-, a szem-, és a hajszín online lekérdezésére.

A színkinyerő eljárásunkat a keresőrendszeren túl megpróbáltuk rasszok osztályozásra is felhasználni. Azonban azt tapasztaltuk, hogy a színinformáció egyedüliként nem robosztus a rasszok felismerésére, de eredményét egyéb más módszerekkel vegyítve (pl. a fej formájának detektálása, az arci jellemzők közötti távolságok mérése) a probléma szempontjából jó végeredményt kaphatunk.

6 Köszönetnyilvánítás

Szeretnék köszönetet mondani Dr. Fazekas Attilának a témakörben nyújtott gondos útmutatásért és tanácsaiért, és hogy lehetőséget kaptam az IPGD csoport tagjaként megismerni a csapatmunka előnyeit rendkívüli emberek között. Köszönöm Sajó Leventének a türelmét, gyakorlati ötleteit és tanácsait, melyek segítettek mind az elméleti témakör gyorsabb megértésében, mind a hatékonyabb szoftverfejlesztésben.

7 Ábrák és táblázatok jegyzéke

7.1 Ábrák jegyzéke

7.2 Táblázatok jegyzéke

8 Irodalomjegyzék

[1] I. Szakadát, "Keresőrendszerek a weben," *PKI Tudományos Napok 2003*, pp. 1-3, 2003.

[2] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR , 1998.

[3] M. Tkalcic and J. F. Tasic, "Colour spaces - perceptual, historical and applicational background," *The IEEE Region 8 EUROCON 2003 proceedins*, pp. 304-308, 2003.

[4] A. Ford and A. Roberts. (2010, Apr.) Colour Space Conversions. [Online]. http://www.poynton.com/PDFs/coloureq.pdf

[5] A. Albiol, L. Torres, and E. J. Delp, "Optimum color spaces for skin detection," *Proceedings of the International Conference on Image Processing (ICIP)*, p. 122–124, 2001.

[6] T. Cheng, H. W. Park, and Y. Kim. (2010, Apr.) Image filtering in HSI color space. [Online]. http://www.freepatentsonline.com/6631206.html

[7] Intel® IPP. (2010) Open Source Computer Vision Library. [Online]. http://sourceforge.net/projects/opencvlibrary/

[8] R. Lienhart and J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection," *IEEE ICIP*, pp. 900-903, 2002.

[9] K.-K. Sung and T. Poggio, "Example-based learning for view-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 39-51, 1998.

[10] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 22-38, 1998.

[11] T. Kanade and H. Schneiderman, "A statistical method for 3d object detection applied to faces and cars," International Conference on Computer Vision, 2000.

[12] D. Roth, M. Yang, and N. Ahuja, "A snowbased face detector," *Neural Information Processing*, vol. 12, 2000.

[13] Y. Amit, D. Geman, and K. Wilder, "Joint induction of shape features and tree classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1300-1305, 1997.

[14] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," *Sixth International Conference on Computer Vision (ICCV'98)*, p. 555, 1998.

[15] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," *Computational Learning Theory: Eurocolt '95*, vol. 37, pp. 23-37, 1995.

[16] K. Tieu and P. Viola, "Boosting Image Retrieval," *International Journal of Computer Vision*, vol. 56, pp. 17-36, 2004.

[17] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods," *The Annals of Statistics*, vol. 26, no. 5, pp. 1651-1686, 1998.

[18] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[19] J. K. Tsotsos, et al., "Modeling visual attention via selective tuning," vol. 78, no. 1-2, pp. 507-545, 1995.

[20] L. Itti, C. Koch, and E. Niebur, "A Model of Saliency-based Visual Attention for Rapid Scene Analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 11, pp. 1254-1259, 1998.

[21] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, pp. 81-106, 1986.

[22] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno, "2D Euclidean distance transform algorithms: A comparative survey," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, pp. 1-14, 2008.

[23] F. Porikli and T. Kocak, "Fast Distance Transform Computation using Dual Scan Line Propagation," in *Real-time image processing*, 2007.

[24] F. Sárközy. (2010, Apr.) Delaunay háromszögelés - Voronoi sokszögek. [Online]. http://gisfigyelo.geocentrum.hu/sarkozy_terinfo/tbev.htm

[25] Z. Savas, "Real-Time Detection and Tracking Of Human Eyes in Video Sequences," *thesis*, pp. 59-61, 2005.

[26] P. Norvig and S. J. Russell, *Mesterséges Intelligencia - Modern megközelítésben*, 2nd ed. Panem Kiadó Kft., 2005.

[27] E. L. van den Broek, T. E. Schouten, and P. M. F. Kisters, "Modeling human color categorization," *Pattern Recognition Letters 29.*, vol. 29, no. 8, p. 1136–1144, 2008.

[28] P. Frost, "European hair and eye color: A case of frequency-dependent sexual selection?," *Evolution and Human Behavior*, vol. 27, pp. 85-103, 2006.

[29] L. Bickford. (2010, Apr.) The EyeCare Reports. [Online]. http://www.eyecarecontacts.com/eyecolor.html

[30] Caltech Face Datebase. (2010, Apr.) [Online]. http://www.vision.caltech.edu/html-files/archive.html

[31] CBCL Face Database. (2010, Apr.) MIT Center For Biological and Computation Learning. [Online]. http://www.ai.mit.edu/projects/cbcl

[32] E. L. van den Broek, "Human-Centered Content-Based Image Retrieval," *thesis*, pp. 53-64, 2005.

[33] A. Hajdu and G. Fazekas, "Képfeldolgozási módszerek," *egyetemi jegyzet*, pp. 12-19, 2004.

9 Függelék

9.1 Az AdaBoost algoritmus

A 2.3.5. fejezetben szó volt róla, hogy az egyes gyenge tanulókat az osztályozó függvényeknek arra a halmazára szorítottuk meg, melyek csak egy egyszerű jellemzőtől függnek. A gyenge osztályozókból a tanuló algoritmus, az AdaBoost algoritmus egy változatával állítja elő az erős osztályozót, melyek a lépcsők szerepét töltik majd be.

Az alábbiakban, az AdaBoost tanuló algoritmus leírása következik, mely egy kérdés online megtanulásának menet ét írja le. Minden T hipotézis egy egyszerű jellemzőt használ. A végső hipotézis a T hipotézisek egy súlyozott lineáris kombinációja, ahol a súlyok fordítottan arányosak a tanulási hibával.

- Adottak az $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$ tanító példák, ahol x_i egy kép és $y_i = 0{,}1$ attól függően, hogy a példa pozitív, vagy negatív.
- Inicializáljuk a súlyokat a következőképpen (m és l a negatív és pozitív példák száma):

$$w_{1,i} = \begin{cases} \frac{1}{2m}, & \text{ha } y_i = 0, \\ \frac{1}{2l}, & \text{ha } y_i = 1. \end{cases}$$

- $for\ t = 1, \dots, T$
  1. Normalizáljuk a súlyokat, hogy w_t valószínűségi eloszlás legyen:
  $$w_{t,i} = \frac{w_{t,i}}{\sum_{j=0}^{n} w_{t,j}}.$$
  2. Minden j jellemzőhöz tanítsunk be egy h_j osztályozót, ami csak egy jellemzőtől függ. A w_t-re vonatkozó ε_t súlyozott hiba:
  $$\varepsilon_t = \sum_i w_i |h_j(x_i) - y_i|.$$
  3. Válasszuk ki a legkisebb ε_t hibájú h_t osztályozót.
  4. Módosítsuk a súlyokat a következőképpen:
  $$w_{t+1,i} = w_{t,i} \beta_t^{e_i},$$
  ahol $e_i = 0$, ha az x_i osztályozása helyes, különben $e_i = 1$, és $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.

- A végső erős osztályozó:

$$h(x) = \begin{cases} 1, ha\ \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0, különben \end{cases} .$$

9.1. táblázat. Az AdaBoost algoritmus.

9.2 A vízesés struktúra tanítása

A vízesés struktúra tervezésénél a motivációt a minél jobb detektálási arány és a minél jobb detektálási arány és a minél alacsonyabb számításigény elérése jelenti. A vízesés szerkezetű osztályozó hamis pozitív aránya:

$$F = \prod_{i=1}^{K} f_i,$$

ahol K az osztályozók száma f_i az i-edik osztályozó hamis pozitív aránya azokon a példákon, amelyek eljutnak hozzá. A detektálási arány:

$$D = \prod_{i=1}^{K} d_i,$$

ahol d_i az i-edik osztályozó találati aránya a hozzá eljutott példákon.

A kép pásztázása során kiértékelt jellemzők száma szükségszerűen egy valószínűségi folyamat függvénye. Bármely alablak addig halad a kaszkádon (egyszerre csak egy osztályozón át), míg el nem döntjük, hogy az ablak negatív, vagy ritka körülmények között az alablak minden teszten végigmegy, így pozitív eredményt ad. Minden osztályozó kulcsértéke a "pozitív aránya", azaz azoknak az ablakoknak az aránya, melyekre az osztályozó pozitív választ ad. A kiértékelt jellemzők számának várható értéke:

$$N = n_0 + \sum_{i=1}^{K}\left(n_i \prod_{j<i} p_j\right),$$

ahol N a kiértékelt jellemzők várható száma, K az osztályozók száma, p_i az $i.$ osztályozó pozitív aránya, és n_i a jellemzők száma az $i.$ osztályban. Érdekes módon, attól fogva, hogy az objektumok nagyon ritkák, a „pozitív arány” egyenlő a hamis pozitív aránnyal.

A kaszkád elemeit tanító folyamat némi odafigyelést igényel. Az AdaBoost a hibák minimalizálására törekszik, és nem kifejezetten magas detektálási arányok elérésére tervezték magas hamis pozitív arányok költsége mellett. Magasabb küszöbbel kevesebb hamis pozitívat detektáló osztályozót eredményez, alacsonyabb detektálási aránnyal. Alacsony küszöbbel pedig több hamis pozitívat és magasabb detektálási arányt elérő osztályozót kapunk.

A teljes tanítási folyamat kétféle, egymásnak ellentmondó kritérium optimalizálását igényli. Többnyire egy osztályozó több jellemzővel magasabb detektálási arányt és alacsonyabb hamis pozitív arányt fog elérni. Ugyanakkor egy több jellemzővel rendelkező

osztályozó több számítási időt igényel. Alapjában véve definiálhatunk egy optimalizációs keretet, amiben

- az osztályozó szintek száma,
- a jellemzők száma, n_i minden szinthez, és
- minden szint küszöbe,

úgy kivitelezett, hogy minimalizáljuk N jellemzők számát, adott F és D célértékek mellett. Sajnos ennek a minimumnak a megtalálása egy igen nehéz feladat. Gyakorlatilag egy nagyon egyszerű alkalmazást használnak hatékony osztályozók előállításához. A felhasználó kiválasztja a minimum elfogadható f_i és d_i arányokat.

A kaszkád minden szintjét AdaBoosttal tanítják, úgy hogy a jellemzők számát addig növelik, míg a célul kitűzött detektálási és hamis pozitív arányokat el nem érik. Az arányokat a detektor tesztelésével kapják meg egy érvényes halmazon. Ha még nem érték el a célul kitűzött hamis pozitív arányt, további szinteket adnak a detektorhoz. A negatív halmazt a következő szintek tanításához az aktuális detektor futtatása során kapott hamis pozitív eredményekből kapják.

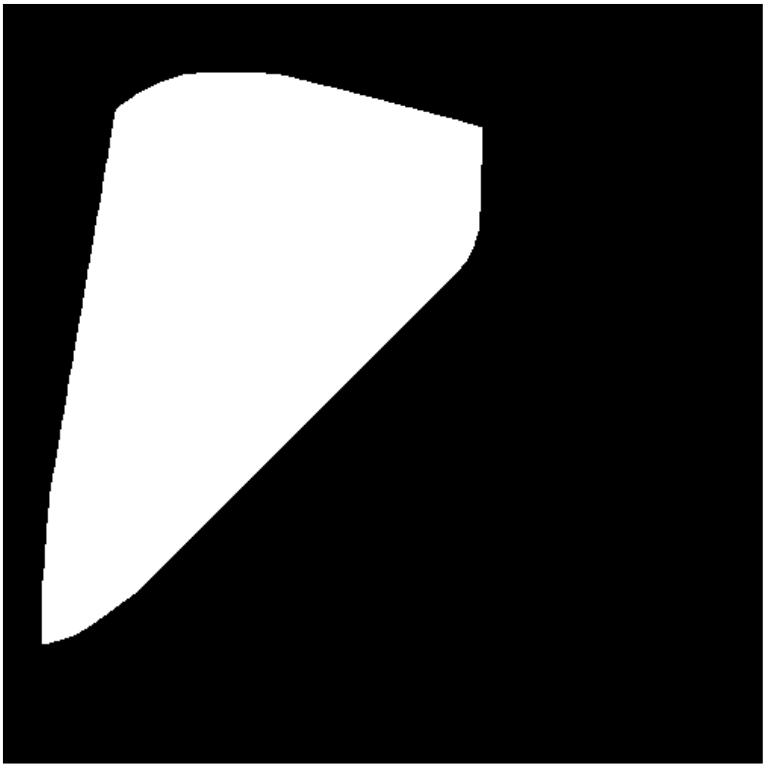
9.3 A keresőrendszer teszteredményei

Az elkészült keresőrendszerünk tesztelésére 20 kép felhasználásával egy újabb pszichológiai kísérletet végeztünk. A teszt során tulajdonképpen egy összehasonlítást végeztünk a kísérletben résztvevők véleményére és a program kimenetére vonatkozólag. A két eredmény közötti kapcsolat a 9.2. táblázatban látható. A táblázat minden egyes cellájában a program által kimenetként szolgáltatott az egyes színkategóriákra vonatkozó százalékos értékek láthatók.

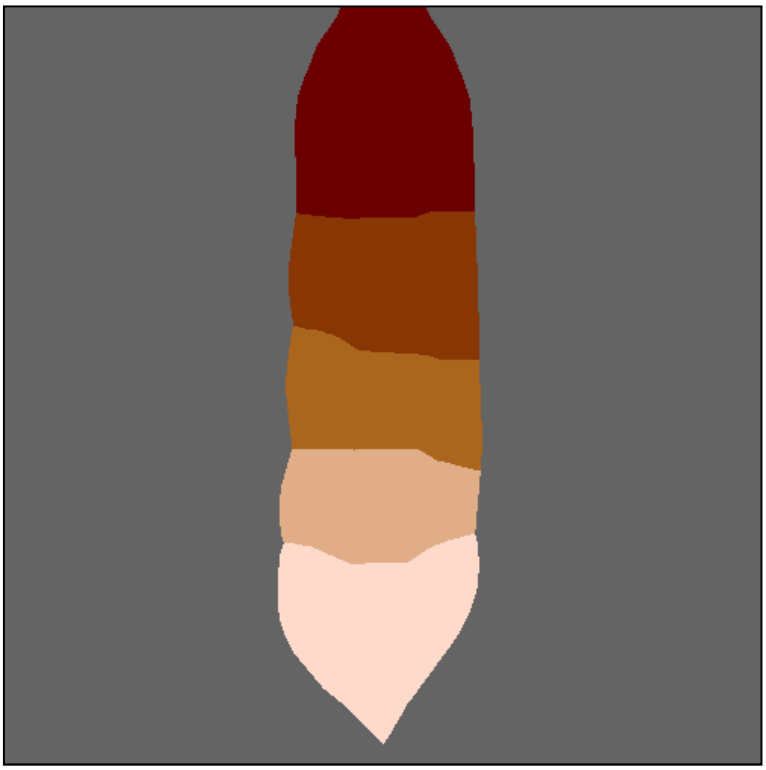
	Képek																			
nagyon világos	.04	.08	**.43**	.34	.23	.11	**.35**	0	.20	0	.04	**.23**	0	0	0	.14	.03	0	0	0
világos	.47	**.56**	.28	**.52**	**.26**	.34	.19	.33	**.61**	.05	**.54**	.22	0	0	0	**.36**	**.37**	0	.37	0
átmeneti	.45	.34	0	.02	.08	.27	0	**.50**	.12	.25	.20	0	.01	0	.02	.24	.32	0	.60	.01
sötét vagy barna	.01	0	0	.02	.02	.10	0	.15	0	**.53**	0	0	**.27**	.24	.29	.02	.23	**.51**	.01	.30
nagyon sötét	0	0	0	.02	0	0	0	0	0	0	0	0	.25	**.67**	.41	0	0	.46	0	**.52**
kék	0	0	**.30**	.02	.01	.01	**.53**	.01	.01	0	0	0	0	0	0	0	.02	0	.05	0
szürke	0	0	.10	.01	.04	.01	.06	0	.01	0	0	0	0	.01	0	.01	.20	0	.14	.08
zöld	.21	.06	0	.10	.04	.06	0	.24	.51	.09	.22	**.66**	.02	.23	.07	**.58**	.18	.06	.50	.09
barna	.21	.77	.08	**.59**	**.53**	**.38**	0	.34	.39	.40	**.30**	.11	.12	.18	.22	.06	.46	.28	.29	.13
sötétbarna	**.44**	.09	0	.09	.19	.18	0	.10	.02	.33	.08	0	.18	**.40**	**.37**	0	.12	**.58**	0	**.53**
szőke	0	0	.10	0	0	0	.51	.23	.08	0	.05	.76	0	**.40**	0	.07	0	0	.01	0
aranybarna	0	.03	**.76**	.19	.05	.02	.02	0	**.58**	0	0	0	0	.03	.01	0	0	0	**.52**	0
barna	0	**.48**	.10	**.53**	.31	.17	0	0	.16	.02	0	0	.16	0	.04	0	0	.01	.18	0
fekete vagy sötétbarna	**.39**	.46	0	.22	.17	**.58**	0	0	0	**.58**	0	0	**.33**	0	**.87**	0	0	**.79**	0	0
szürke vagy fehér	0	0	0	0	0	0	**.43**	**.65**	0	0	**.70**	.23	0	0	0	.31	**100**	0	0	0

9.2. táblázat. A kísérletben résztvevők véleményének és a program kimenetének összehasonlítása. Zöld színnel a kísérletben résztvevők által kiválasztott kategóriát, narancssárga színnel a program által visszaadott színkategóriát jelöltük, kékszínű félkövér kiemeléssel jelöltük, ha az előbbi két kategória megegyezik.

9.4 Az arcbőrhöz tartozó SI és HI síkok

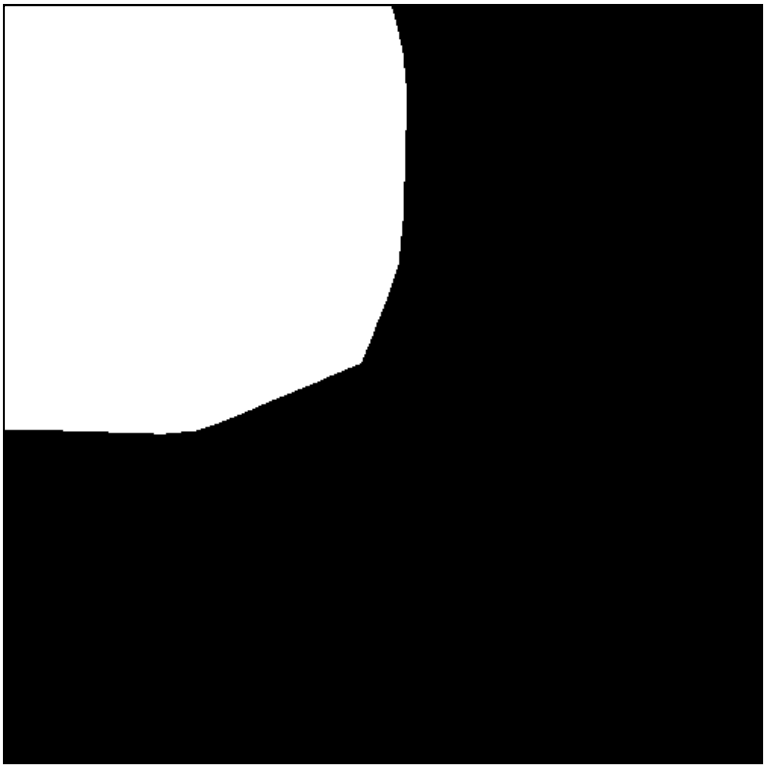


9.1. ábra. Az arcbőrhöz tartozó SI sík: fehér színnel jelöltük a színtér kromatikus tartományába eső pixeleit, valamint fekete színnel jelöltük az akromatikus pixeleket.

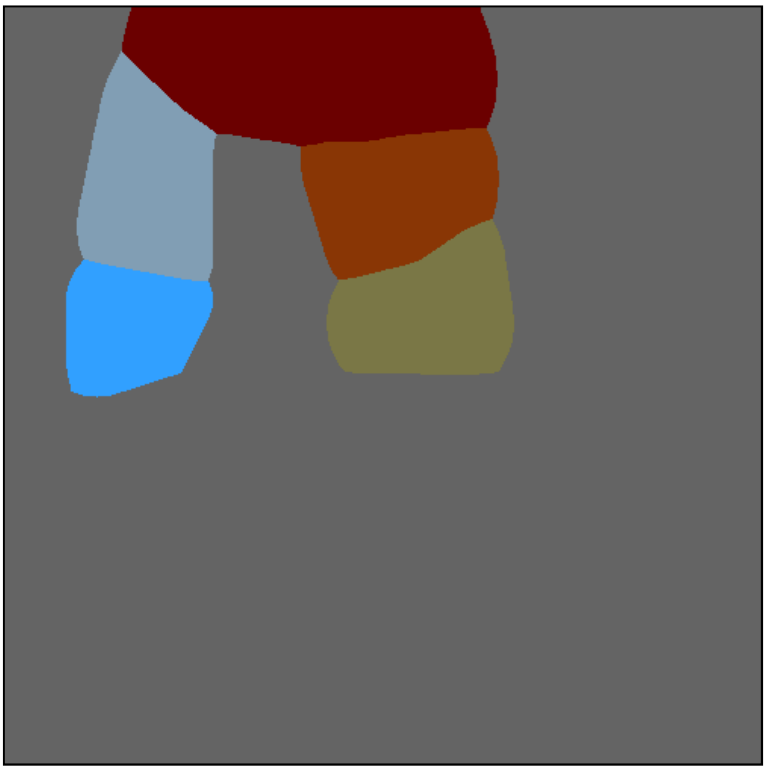


9.2. ábra. Az arcbőrhöz tartozó HI sík: A képen az egyes színkategóriák középszínei láthatók.

9.5 A szemhez tartozó SI és HI síkok

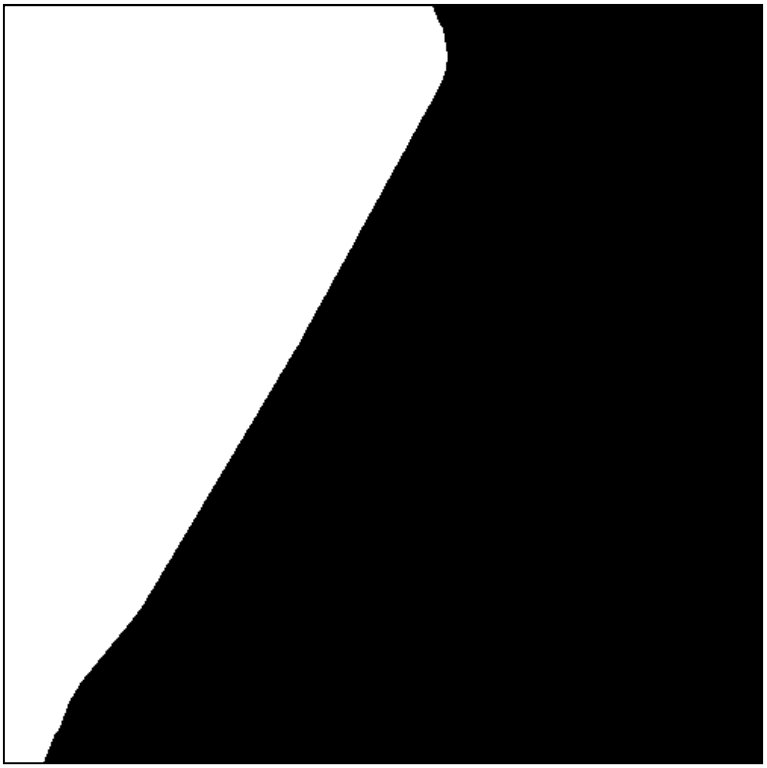


9.3. ábra. A szemhez tartozó SI sík: fehér színnel jelöltük a színtér kromatikus tartományába eső pixeleit, valamint fekete színnel jelöltük az akromatikus pixeleket.

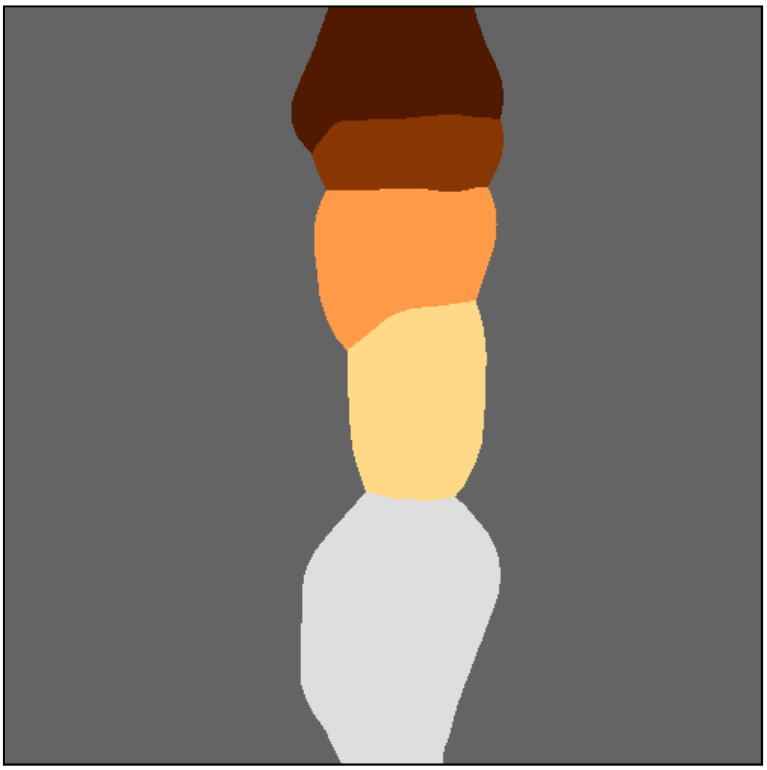


9.4. ábra. A szemhez tartozó HI sík: A képen az egyes színkategóriák középszínei láthatók.

9.6 A hajhoz tartozó SI és HI síkok



9.5. ábra. A hajhoz tartozó SI sík: fehér színnel jelöltük a színtér kromatikus tartományába eső pixeleit, valamint fekete színnel jelöltük az akromatikus pixeleket.



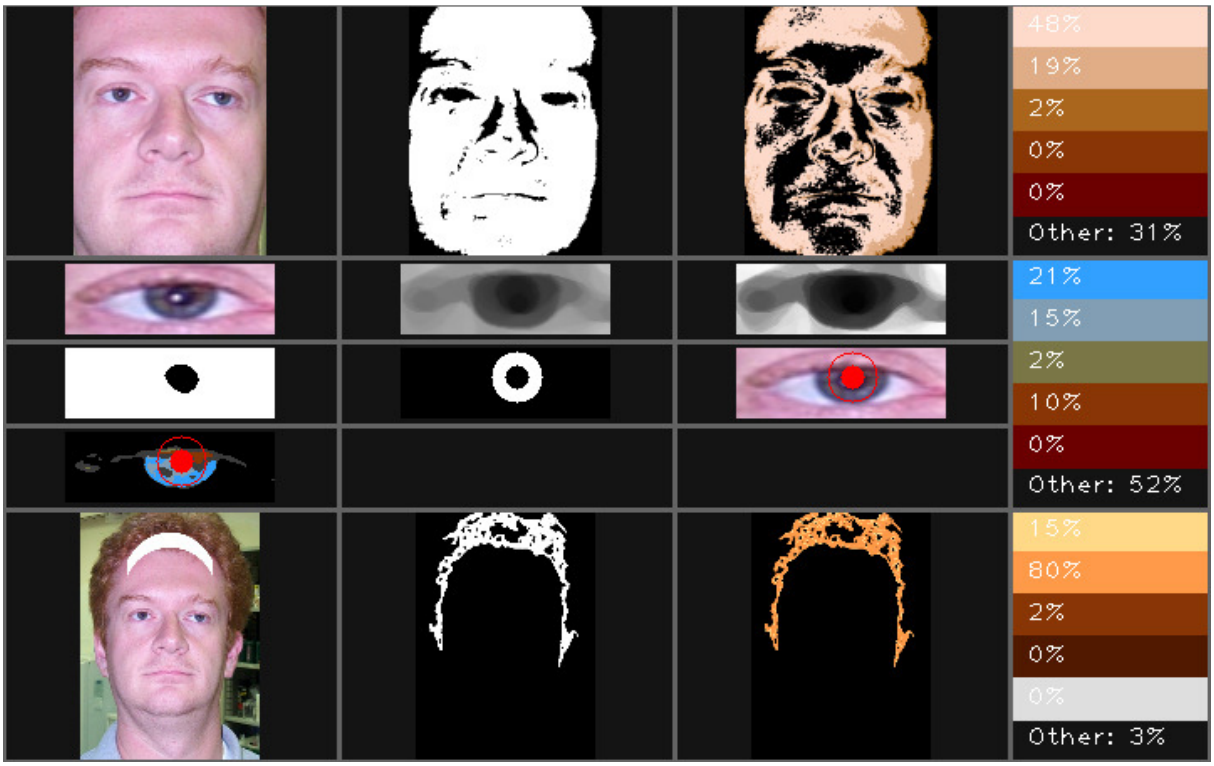
9.6. ábra. A hajhoz tartozó HI sík: A képen az egyes színkategóriák középszínei láthatók.

9.7 A program kimenete

A lentebb látható ábrán a program által szolgáltatott kimenet látható. Az ábra első három oszlopában az egyes jellemzők színének meghatározása során alkalmazott lépések grafikus szemléltetései láthatók, az utolsó oszlopában pedig az egyes színkategóriákra jósolt százalékos értékek helyezkednek el. Az utolsó oszlopában a sorok színei, a 4.1. táblázatban szereplő színkategóriák elemeinek középszínei.

Az „other” címszóval jelölt színkategória a szegmentált régió alatti akromatikus pixelek százalékos arányát jelenti. A keresőrendszer a legnagyobb százalékban előforduló kromatikus színt választja az egyes jellemzők színének. Tehát arci jellemzőkként a 4.1. táblázatból választja ki az adott jellemző színének azt a kategóriát, melynek színei a legnagyobb arányban fordulnak elő a szegmentált régió alatti területen.

Jelen esetben az arcbőr színe, a nagyon világos kategóriába, a szemszín a kék kategóriába, a hajszín pedig az aranybarna kategóriába esik.



9.7. ábra. A keresőrendszer kimenete.

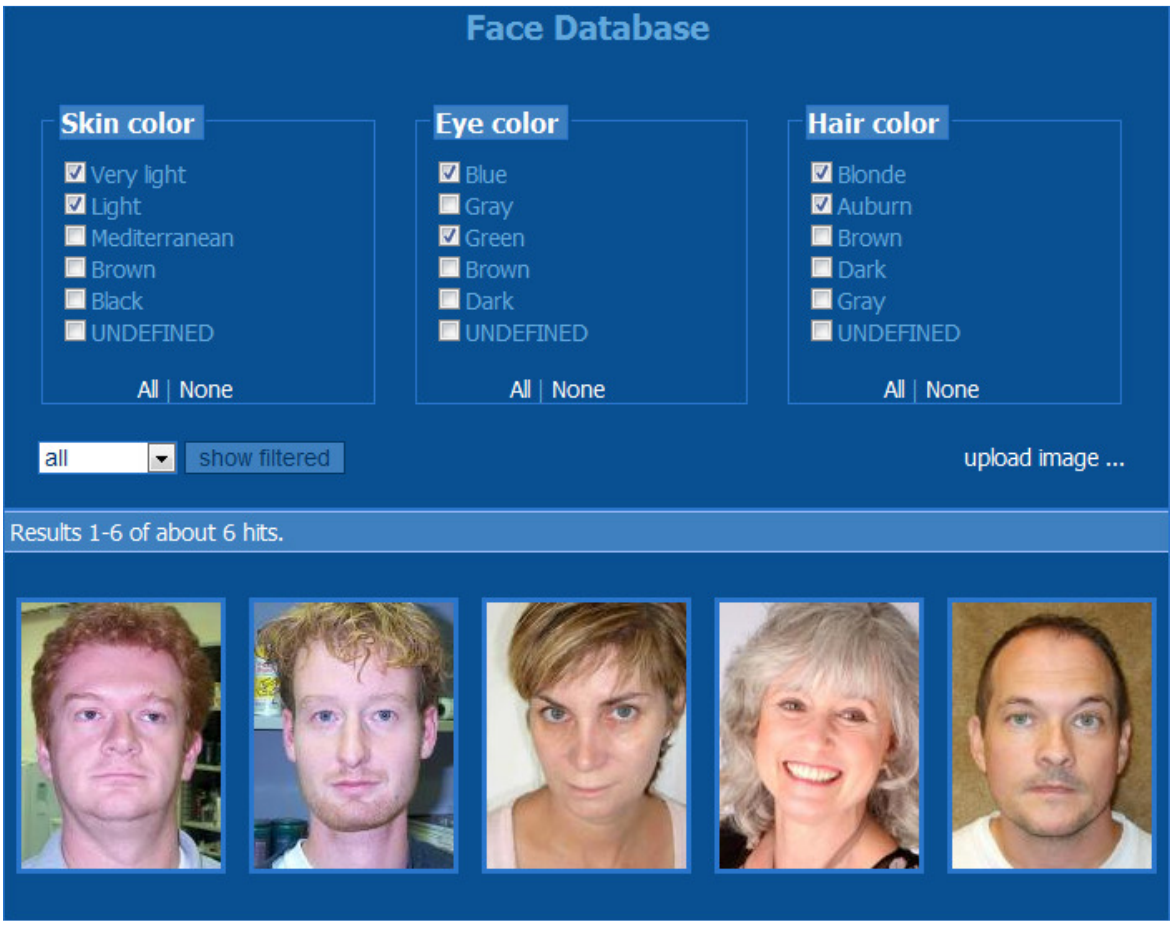
9.8 A keresőrendszerhez tartozó webalkalmazás

A lentebb látható ábrán a keresőrendszerünkhöz készített webalkalmazásunk felhasználói felülete látható. A webalkalmazás segítségével jellemzőkként kérdezhetjük le egy metafájlból a felöltött-, vagy a demó képek adatbázisát.

Mint ahogy azt már az 5.1. fejezetben ismertettem, a felhasználóknak lehetősége van a képadatbázisba új képeket feltölteni. A feltöltött képeket paraméterként adjuk át a színkinyerő alkalmazásunknak, mellyel meghatározzuk az egyes arci jellemzők színét, és azokat a kép nevével együtt eltároljuk egy meta leírófájlban.

Az alábbi ábrán az egyes arci jellemzőknél látható UNDEFINED kategóriába azok a képek tartoznak, melyeknél a program nem tudta meghatározni a színét az adott jellemzőnek (pl. az arcdetektor nem talált egyetlen pozitív találatot sem a bemeneti képen, így a program nem tudta meghatározni a bőrt tartalmazó régiót).

Jelen esetben a nagyon világos-, vagy világos bőrű, kék-, vagy zöld szemű és szőke-, vagy aranybarna hajú embereket listázzuk ki a teljes adatbázisból.



9.8. ábra. A webalkalmazás felhasználói felülete.

Debreceni Egyetem
Informatikai Kar

VIZUÁLIS INFORMÁCIÓN ALAPULÓ KERESŐRENDSZER ARCI ADATBÁZISOKBAN

Témavezető:
Dr. Fazekas Attila
Egyetemi docens

Készítette:
Bertók Kornél
V. PTM

Debrecen
2010

Tartalomjegyzék

1 Bevezetés

Az emberiség történelme folyamán kevés új technológia fejlődött olyan hatalmas sebességgel és tört be mindennapjainkba olyan hirtelen, mint az informatika. Ma már nehéz olyan tevékenységet találni, amely műveléséhez ne lenne elengedhetetlenül szükséges, vagy legalábbis rendkívül hasznos a számítógépek használata. A kezdetben szűk szakmai közösségek által kifejlesztett és felhasznált eszközök egyre szűkösebbé váltak azáltal, hogy a felhasználók köre kiszélesedett. A különböző felhasználások a számítógéppel való kommunikáció újabb és újabb változatos formáit igénylik.

A HCI (ember-számítógép interakció) kutatási feladatai közé tartozik, hogy olyan új, alternatív kommunikációs (adat ki- és beviteli) eszközöket és módszereket fejlesszen, amelyek segítik az ember és gép közötti kapcsolatot az ember számára minél természetesebbé, magától értetődővé tenni.

Diplomamunkámban az ember-számítógép kommunikáció egy újszerű felhasználói eszközével, egy vizuális információn alapuló keresőrendszer kifejlesztésével foglalkozom. A rendszert az IPGD csoportban Dr. Fazekas Attila témavezetése mellett Sajó Levente doktorandusz hallgatóval fejlesztettük ki és implementáltuk C++ nyelven, operációs rendszertől független környezetben.

A dolgozat célja, egy olyan keresőrendszer ismertetése, mely segítségével lehetőségünk nyílik egy arcokat tartalmazó képi adatbázison összetett lekérdezéseket végrehajtani. A jelenlegi rendszer segítségével az adatbázisban szereplő emberek arc-, szem-, és haj színének lekérdezését tehetjük meg.

Fontos leszögezni, hogy a keresés kizárólag vizuális információkon alapul. Tehát nem állnak rendelkezésre úgynevezett metainformációk, melyek segítségével az érzékszervi információ, nyelvi információvá transzformálható. A keresés során a képeket nem foglaljuk gyűjteményekbe, hanem csak mint vizuális információt tároljuk metainformáció nélkül. Fogalmazhatunk úgy is, hogy a rendszernek nincs kiinduló információja arról, hogy valójában mit ábrázol a kép.

1.1 Keresőrendszerek

Napjaink információval túlterhelt társadalmában különösen fontos szerepet töltenek be a keresőrendszerek. E rendszerek segítségével vagyunk képesek megtalálni és elkülöníteni a számunkra hasznos információkat a lényegtelenektől.

Keresőrendszer alatt az informatikában olyan szolgáltatást értünk, ami adatbázisok rendszeres vagy egyéni kérésre történő rendezését, nyomon követését és kivonatolását biztosítja. Mindezek mellett lehetővé teszi a tartalomnak a felhasználó, és általában a szélesebb nyilvánosság részére történő rendelkezésre bocsátását. A keresést általában egy olyan kommunikációs helyzetnek minősítjük, melyben valaki, valahol, valamilyen módon rendelkezésre álló információt szeretne megtalálni. Jelen dolgozat kontextusában úgy érdemes szűkíteni a fogalom jelentéstartományát, hogy a kommunikáció általános modelljében a két kommunikáló fél közül az egyik egy természetes személy, a másik egy számítógép. Ez utóbbi fél digitális adatbázisokban rögzített információkat próbál meg visszakeresni és visszaadni kommunikációs partnere, az ember számára [1].

A megjelenítendő tartalom neve – amely szavakat, logikai operátorokat és egyéb attribútumokat tartalmazó összetett nyelvi kifejezés lehet – a keresőszó, melyet az ember ad meg a keresőrendszernek. Következő lépésben a rendszer megkeresi és kijelzi mindazt, amit ezzel kapcsolatban tud. Keresés alatt általában címszavas keresést értünk, vagyis a rendszer bekéri a keresendő címszót (pl. egy beviteli űrlapmezőben), majd megjeleníti azokat a tartalmi egységeket, amelyek a rendszer szerint a címszóhoz kapcsolódnak.

A keresés modelljének alaposabb kifejtése érdekében célszerű megválaszolni azt a kérdést, hogy „Mit lehet egyáltalán keresni?”. A rövid válasz erre nyilván az, hogy információt, de ebből azonnal következik az újabb kérdés, hogy „milyen információtípusok vannak?”. A válasz előtt rögzítenünk kell, hogy két szempont alapján, érzékszervi, illetve nyelvi szinten különíthetjük el egymástól a különböző információtípusokat. A digitális kommunikáció világán belül érzékszervi szinten a hallás és a látás érzékszerveire támaszkodó információtípusokkal érdemes foglalkozni. E dolgozat a keresés modelljének keresztmetszetét, a multimédiás tartalmakra azon belül is a látás érzékszerveire támaszkodó információtípusokra szűkíti le.

1.2 Tézisek

1. A Viola-Jones objektum detektorok robosztus valósidejű objektumdetektálásra képesek. Érzéketlenek, azaz robosztusak a beérkező kameraképre, gyors feldolgozásra képesek magas detektálási arány mellett. Egy 30 kép/másodperc teljesítményű 640x480 képpont felbontású webkamera képén, a Viola-Jones arcdetektor 30 kép/másodperc teljesítményt ér el egy 1.6 GHz-es Intel Pentium M processzorral rendelkező notebook-on, tehát képes az összes beérkező kép feldolgozására.
2. A távolság transzformáció egy hatékony eljárás lehet a színtér szegmentálására. Segítségével egész aritmetikát használva hatékonyan tudjuk közelíteni a színek és az egyes színklaszterek közötti euklideszi távolságot.
3. A színtér távolság transzformáltjának Voronoi felosztásával ki tudjuk alakítani azt a rácsszerkezetet, melynek minden belső pontja közelebb van az adott klaszter alappontjaihoz, mint az összes többi ponthoz.
4. Az arci jellemzők szegmentálása elvégezhető az azok szerkezeti információja alapján.
5. Az arci jellemzők színének kinyerése során sokkal gyorsabb eljárást és valósághűbb eredményt kapunk, ha humán megfigyelések alapján a színtér minden egyes elemét színkategóriákba soroljuk be.

1.3 A dolgozat felépítése

Dolgozatom 2. fejezetében keresőrendszerünk működéséhez szükséges elméleti hátteret fektetem le, különös figyelmet fordítva az arci jellemzők színének meghatározásának alapját jelentő objektumdetektálást (lásd 2.3. fejezet). A 2.4. és a 2.5. fejezetben, egy a színterünk hatékony és gyors szegmentálását megvalósító algoritmus alapköveit (távolság transzformáció és Voronoi diagram) ismertetem.

A 3. és 4. fejezetben a keresőrendszer tényleges működésével, az arci jellemzők színének meghatározásával foglalkozom. A jellemzők színét két lépésben határozzuk meg: először szerkezeti információ alapján szegmentáljuk az arcbőrt, a szemeket és hajat (lásd 3. fejezet). Majd második lépésben az eredeti képen a szegmentált terület alatti pixelek világosságkódjainak nagy elemszámú halmazát cseréljük le egy kisebb halmazra – melyet az adott jellemzők színének meghatározásához használunk fel. A felhasznált színmodellünket humán megfigyelések alapján készítettük el. Ezen a modellen alapszik a színmeghatározó metódusunk, melyet a 4. fejezetben részletesen ismertetek.

2 Elméleti háttér

Az elkészült keresőrendszer az IPGD csoportban eltöltött több mint kétéves kutató és fejlesztő munkám eredménye. A fejlesztés során a digitális képfeldolgozás számos szerteágazó területét tanulmányoztam és sajátítottam el.

Ebben a fejezetben a keresőrendszer felépítéséhez szükséges algoritmusokat, módszereket és elméleti alapokat, vagyis a rendszer alapköveit tekintjük át.

2.1 Színek használata

A szín voltaképpen a látható tartományba eső elektromágneses hullámok által kiváltott érzet [2], amely a hullámok spektrális eloszlásán (fizikai tulajdonságain) kívül döntő mértékben függ a szem és az agy működésétől, továbbá pszichológiai jelenségektől. A színek érzékelése tehát személyes élmény (nem mérhető objektivitás), vizsgálata emiatt a fizikától a biológián és a pszichológián át egészen a képzőművészetekig vezet. Ezzel a megállapítással eljutottunk a digitális képfeldolgozás legalapvetőbb és legnehezebben megválaszolható kérdéséhez: Mikor mondjuk egy színre, hogy az piros, vagy barna, stb.? Nyilvánvaló, hogy egy keresőrendszernek, mely például az arc színét hivatott meghatározni, annak az előző kérdésre kell helyest választ szolgáltatnia.

2.2 Színterek

Az előző kérdés megválaszolásához mindenekelőtt szükségünk lesz egy modellre, amely alkalmas a színek és a közöttük fennálló relációk matematikai leírására. Erre a modellre a továbbiakban színtérként fogok hivatkozni. A színterek a színek ábrázolására használható virtuális terek (koordinátarendszerek), melyben az egyes színek rögzített összefüggések alapján reprezentálhatók [3,4]. A színtér határozza meg, hogy milyen módon tárolunk egy színt. Általában egy–, kettő–, három–, vagy négydimenziós teret definiálnak, ahol egy–egy dimenzió rendre egy–egy jellemző számszerű kifejezésének felel meg, így az egyes színeket azok koordinátái fejezik ki. A színtérben az ábrázolható színek valamilyen rend szerint kerülnek elhelyezésre (pl. az alapján, hogy a színtér alapszíneinek milyen arányú keverésével állíthatók elő), és a pozíciójukat meghatározó koordinátákkal kerülnek azonosításra. A képfeldolgozásban leggyakrabban az RGB, HSV, HSI, CMYK, színtereket alkalmazzák, de ezeken kívül rengeteg színtér létezik még.

A színeknek rendkívüli jelentősége van az ember-számítógép interakciókban. De problémákat is jelentenek a feldolgozásban, hiszen az érzékelő számára a színek

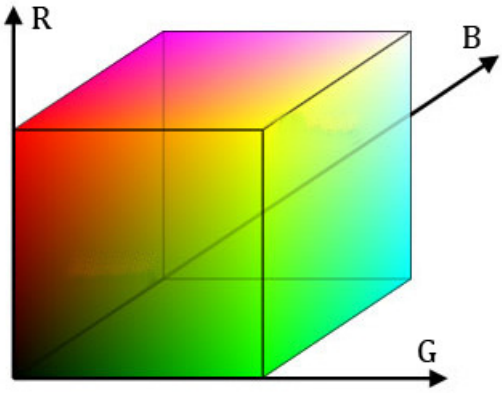
megváltozhatnak a felület orientációjával, a kamera nézőpontjának, a megvilágítás pozíciójának vagy spektrumának megváltozásával, valamint annak függvényében, hogy a fény hogyan hat az egyes objektumokra.

A számítástechnikában legismertebb RGB színtér nem az emberi szem érzékeléséhez igazodik, így használata csak akkor javasolt, ha a kameraképen minimális változások vannak. Az RGB térrel szemben léteznek invariáns tulajdonságokkal rendelkező színterek is. Olyanok, ahol a színárnyalat nem függ az objektum elhelyezkedésétől, ezért jobban használható a lényeges információk kinyerésére. E tulajdonságokkal rendelkező terekben a színek csoportosítása – magyarán a teljes színtér szegmentálása – egyszerűbben és átláthatóbban elvégezhető, mint más terek esetén.

2.2.1 RGB színtér

A bemeneti képünk RGB színtérben van definiálva, így az egyes pixelpozícióiban lévő intenzitás értékek, RGB koordinátáknak felelnek meg. Az RGB színrendszerben a színek a három alapszín, a vörös (R – Red), zöld (G – Green), kék (B – Blue) egymásra vetítésével állíthatók elő, tulajdonképpen ezt nevezik additív színkeverésnek (lásd 2.1. ábra). Ez a fajta színkeverési rendszer a kisugárzott, illetve az érzékelt fényen alapul, ezért csak fényt kibocsátó berendezésekkel hozható létre, illetve azokban alkalmazzák (pl. monitorokban).

Az RGB tér legnagyobb problémája, hogy a színcsoportok pozíciója és a csoport elemeit befoglaló alakzat formája a színtéren belül, egyáltalán nem kézenfekvő. Ezért a színtér szegmentálására érdemes, a színek elhelyezkedését illetően egy strukturáltabb színteret használni, mint pl. a HSI tér [5,6].



2.1. ábra. Az RGB színtér 3D-s reprezentációja. Az egyes tengelyek az alapszíneknek megfelelő címkékkel vannak ellátva.

2.2.2 HSI színtér

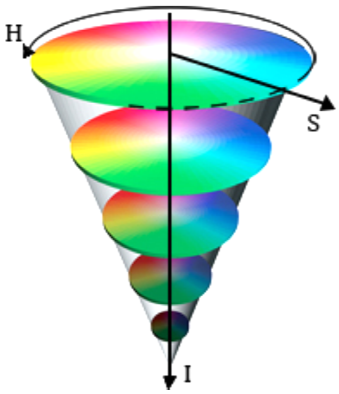
HSI színtér esetén, az RGB koordinátarendszer testátlója felől nézzük a színeket jelentő pontokat. Az R tengely irányát tekintjük *nulla* foknak, és ehhez képest határozzuk meg a pontok irányát, ami a HSI tér H (Hue – Színezeti szög) komponense lesz. A 2.1. táblázatban az RGB tér alap–, és a kiegészítő színeinek oldalszöge látható, mindazonáltal fontos megjegyezni, hogy a színezeti szög nem azonos a színnel, hiszen ezen túl van még két érték, amely befolyásolja a színérzetet. Úgy is fogalmazhatunk, hogy a színezeti szög, csak a fény hullámhosszúságának függvénye.

H (fok)	0°	60°	120°	180°	240°	300°
Szín	Piros	Sárga	Zöld	Kékeszöld	Kék	Bíbor

2.1. táblázat. Az alapszínek és kiegészítő színek jellegzetes színezete

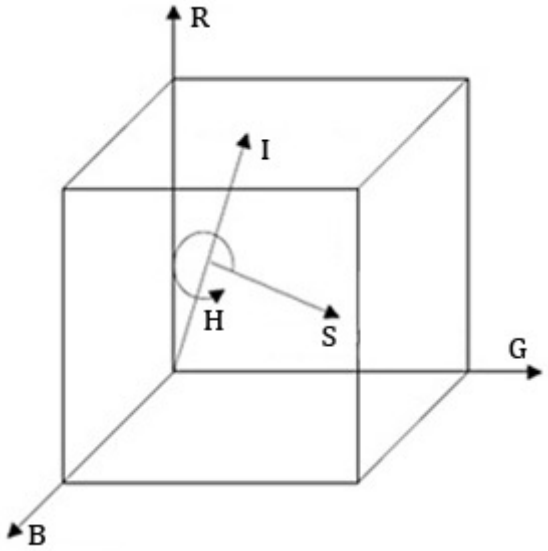
A HSI színtér második eleme az S koordináta (Saturation – Telítettség). Ez a komponens a szín élénkségét, vagyis a fehér összetevő mennyiségét fejezi ki. A spektrum-színek 100%-os telítettségűek, nincs fehér összetevőjük. Ugyanakkor például a rózsaszín néhány százalékban fehér összetevőt tartalmazó vörös. Egy átlagos szem húsz különböző telítettségi fokozatot tud elkülöníteni Az alapszínek és kiegészítő színek telítettsége maximális. A szürke szín árnyalatainak a telítettsége pedig nulla. Az 2.2. ábrán a kúp keresztmetszetét alkotó kör középpontjában a telítettség nulla, a kör kerületén pedig egy.

A színrendszer harmadik eleme az I koordináta (Intensity – Intenzitás), mely az adott szín sötétségét fejezi ki. Ez a fényforrás által kibocsátott fotonok mennyisége, illetve az egységnyi felületre beérkező fotonok száma. Például a barna szín spektrális eloszlása a sárgáéval azonos, de más a világosság értéke. Átlagosan mintegy ötszáz intenzitásfokozatot tudunk a szemünkkel megkülönböztetni.



2.2. ábra. A HSI színtér 3D-s reprezentációja.

A fenti fejezetből látható, hogy a HSI színtér használata a képfeldolgozásban rendkívül előnyös, mivel a feladatok egy része, egyetlen komponens módosításával elvégezhető. Ennek nem csak azért van jelentősége, mert kevesebb számítást igényel, hanem azért is, mert bizonyos jellemzők vizsgálatára (telítettség, színezeti szög, relatív világosság, világosság) az RGB térben nincs lehetőségünk – márpedig a vizuális információn alapuló keresőrendszerünk működéséhez e mennyiségek nélkülözhetetlenek.



2.3. ábra. A kapcsolat az RGB és a HSI színterek között. Az ábrán az RGB tér bonyolult struktúrája látható a színek elhelyezkedését illetően.

2.3 Objektumdetektálás

A keresőrendszerünk sikeres működéséhez elengedhetetlen egy nagy megbízhatóságú, robosztus objektumdetektáló rendszer. A robosztusságon azt értjük, hogy a rendszernek érzéketlennek kell lennie a beérkező kép minőségére.

Az IPGD csoportban 2008. első felében az Intel OpenCV [7] szabadon felhasználható képfeldolgozó könyvtár segítségével elkészítettünk egy olyan szoftver eszközt, amely a gyakorlati kísérletek során bebizonyította, hogy megfelel az előbbi bekezdésben megfogalmazott elvárásainknak.

Az alábbiakban ennek a rendszernek az elméleti hátterét tekintjük át a hozzá kapcsolódó algoritmusokkal együtt. Ezzel a komponenssel határozzuk meg a bőr-, szem-, és hajszín megállapításához szükséges arc és szem objektumok pozícióját a bemeneti képeken.

2.3.1 Viola-Jones detektorok

Paul Viola és Michael J. Jones egy olyan frontális arcdetektor rendszert alakítottak ki [8], mely eléri az előtte publikált legjobb eredmények [9,10,11,12,13] találati és hamis pozitív arányát (3. és 4. definíció). Az arcdetektor mintájára szem-, és szájdetektorokat készítettünk az Intel OpenCV szabadon felhasználható képfeldolgozó-könyvtár segítségével, melyben Viola és Jones közzétették arcdetektorukat és az implementált tanítási algoritmust.

Arcdetektáló rendszerük tisztán kiemelkedik az eddigi megközelítések közül gyorsaságában. Valós időben 30 kép/másodperc teljesítményt érhetünk el, 640x480 pixel felbontású kameraképen egy 1.6 GHz-es Intel Pentium M processzorral rendelkező notebookon. A Viola-Jones detektor szürkeskálás képekből kapott információk alapján képes magas képfeldolgozási arányt elérni. A következő definíciók a további részek érthetősége miatt kerültek ide.

1. Definíció. Hamis találat, vagy hamis pozitív

Ha a detektor a kép egy részletén igaz eredménnyel tér vissza, miközben a képrészleten nem szerepel a keresett objektum, akkor azt hamis találatnak vagy hamis pozitívnak nevezzük.

2. Definíció. Hamis negatív

A nem detektált objektumot hamis negatívnak, vagy hamis elutasításnak nevezzük.

3. Definíció. Találat

Ha a detektor a kép egy részletre igaz eredményt ad, és a képrészlet valóban a keresett objektumot ábrázolja, akkor ezt az eseményt találatnak nevezzük.

4. Definíció. Hamis pozitív arány, vagy hamis találati arány

Detektor tesztelése során a hamis találatok számát osztva a vizsgált képrészletek számával kapjuk a hamis pozitív-, vagy hamis találati arányt. Ez egy racionális szám a [0..1] intervallumból.

5. Definíció. Találati vagy detektálási arány

Detektor tesztelése során a találatok számát osztva a keresett objektumok számával kapjuk a találati vagy detektálási arányt. Ez egy racionális szám a [0..1] intervallumból.

Gondolatmenetünket három fő témakör köré csoportosíthatjuk:

1. Az integrált képreprezentáció bevezetése. A rendszer a kép intenzitásértékei helyett képrégiók jellemzőivel dolgozik [14]. Ezek a jellemzők a Haar-féle bázisfüggvényekre emlékeztetnek, bár néhány esetben ezeknél valamivel bonyolultabbak. A jellemzők többféle méret melletti gyors kiszámítása érdekében vezetjük be az integrált képet, amely az eredeti képből képpontonként néhány elemi művelettel – tehát rendkívül gyorsan és hatékonyan – előállítható. Az integrált képreprezentáció ismeretében a jellemzők értéke bármely pozícióban vagy skálázás mellett konstans időben meghatározható.

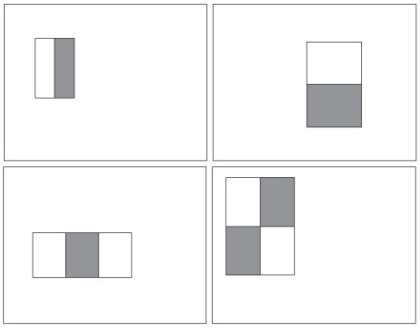
2. A második fontos tulajdonság egy osztályozó létrehozásának folyamata, a fontos jellemzők egy kis halmazának kiválasztásával, az AdaBoost-on alapuló tanuló algoritmust használva [15]. Egy kép részablakán belül az összes Haar jellemzők száma nagyon nagy, sokkal nagyobb, mint a pixelek száma. Azért, hogy a gyors osztályozást biztosítsuk, a tanuló folyamatnak ki kell zárnia az elérhető jellemzők nagy többségét, és a kritikus jellemzők kis halmazára kell koncentrálnia. Tieu és Viola munkája által motiváltan, a jellemző kiválasztás az AdaBoost egy egyszerű módosítása lett: minden gyenge osztályozót kényszerítenek, hogy annak eredménye csak egy jellemzőtől függjön [16]. Ennek eredményeként a Boost folyamat minden köre, amely új gyenge osztályozót választ, megfelel egy jellemző kiválasztó folyamatnak [17,18,14].

3. Az utolsó lépés, hogy az egyre bonyolultabb osztályozókat láncba, egy vízesésszerű sémába rendezzük. Az ötlet azon az észrevételen alapul, hogy gyakran egyszerű azt eldönteni, hogy egy objektum hol fordulhat elő, így a bonyolultabb feldolgozást csak ezeken a „bíztató” területeken kell elvégezni [19,20,13]. Az osztályozók teljesítményének fontos mértéke az úgynevezett „hamis negatív” találati arány, vagyis azon részablakok aránya, amelyeket elutasítunk, annak ellenére, hogy tartalmazzák a keresett objektumot. Ideális esetben a képen ténylegesen szereplő összes célobjektum régióját meg kell találni.

Egy gyors arcdetektornak széles alkalmazási köre lehet. A detektálás sebességének növekedése valós idejű arcdetekálást tesz lehetővé olyan rendszereken, melyeken ez korábban kivitelezhetetlen volt. Implementálható alacsony teljesítményű eszközök széles skáláján, beleértve például a kézikamerákat és integrált processzorokat is. A detektor készítői például egy Compaq iPaq kézikamerán implementálták rendszerüket, és azon 2 kép/másodperc mellett detektáltak arcot. (Az eszköznek egy alacsony teljesítményű 200 MIPS sebességű StrongArm processzora volt, mely nem volt képes lebegőpontos számításra.)

2.3.2 Jellemzők

A Viola-Jones objektumdetektáló rendszer egyszerű jellemzők értékei alapján osztályoz képeket. Sok érv szól a jellemzők használata mellett, a képpontok közvetlen használata helyett. A legáltalánosabb ok, hogy a jellemzők ad-hoc tudás területeket képesek kódolni, melyeket nehéz megtanulni véges mennyiségű tanító adathalmaz használatával. A rendszer egy másik fontos motivációja a jellemzők használatára: a jellemző-alapú rendszer sokkal gyorsabban dolgozik, mint a képpont-alapú.



2.4. ábra. Téglalap jellemzők a detektor ablakon belül. A fehér téglalapon belül fekvő pixelek összegét vonjuk ki a fekete téglalapon belül fekvő pixelek összegéből.

A használt egyszerű jellemzők emlékeztetnek a Haar-féle bázisfüggvényekre, melyeket Papageorgiou és társai használtak [14]. A Viola-Jones rendszer ezeknek a jellemzőknek három fajtáját használja. A *két-téglalap jellemző* értéke a pixelek összegének különbsége két téglalap terület között. A területeknek ugyanakkora a mérete és alakja, és függőlegesen vagy vízszintesen szomszédosak (lásd 2.4. ábra). A *három-téglalap jellemző* két szélső téglalap pixeleinek összegét vonja le a középső téglalap pixeleinek összegéből. Végül a *négy-téglalap jellemző* diagonális téglalap párok közötti különbséget számít.

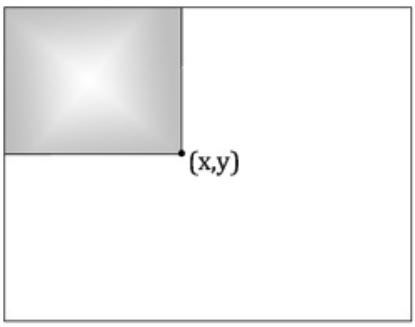
2.3.3 Integrál kép

Egy adott képet többfajta kódolással lehet tárolni. A Viola-Jones detektorok csak szürkeskálás képeket használnak. Ha színes képeket használunk, akkor az algoritmus először átalakítja a képeket szürkeárnyalatossá, majd ezeken dolgozik.

A szürkeárnyalatú képek egy lehetséges tárolási formája egy olyan mátrix, amelynek minden eleme a kép egy adott pixelének szürkeárnyalatát mutatja. Általában ez az elem egy [0,255]-ig terjedő egész szám. A képhez tartozó integrál kép egy ugyanolyan méretű mátrix, amelynek elemei az eredeti kép adott pixelétől balra és felfelé elhelyezkedő pixelértékek összegét tartalmazza. Tehát az integrál kép az x, y pontban a tőle balra fent levő téglalapban elhelyezkedő pixelek intenzitásértékeinek összege:

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

ahol *ii*(*x; y*) az integrál kép, és *i*(*x; y*) az eredeti kép (lásd 2.5. ábra).



2.5. ábra. Az integrál kép értéke az (x,y) pontban a balra fent levő téglalapban elhelyezkedő pixelek intenzitásértékeinek összege.

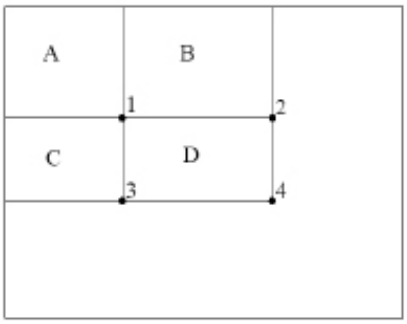
Jelölje $s(x,y)$ az intenzitásértékek területmérő függvényét, és definíció szerint legyen $s(x,-1) = 0$, továbbá $\mathrm{ii}(-1,\mathrm{y}) = 0$. Ekkor igazak az alábbi rekurzív összefüggések:

$$\mathrm{s}(\mathrm{x},\mathrm{y}) = \mathrm{s}(\mathrm{s},\mathrm{y}-1) + \mathrm{i}(\mathrm{x},\mathrm{y}), \tag{2.2}$$

$$\mathrm{ii}(\mathrm{x},\mathrm{y}) = \mathrm{ii}(\mathrm{x}-1,\mathrm{y}) + \mathrm{s}(\mathrm{x},\mathrm{y}). \tag{2.3}$$

A fentiekből látható, hogy az integrál kép az eredeti kép egyszeri végigolvasásával előállítható, majd a későbbiekben a jellemző kiértékelésben felhasználható.

Az integrál képet használva egy téglalap összeg négy tömbhivatkozással számítható (lásd 2.6. ábra). Két téglalap közötti különbség nyolc hivatkozással számítható. A két-téglalap jellemző mivel szomszédos téglalapok különbségét veszi, hat tömbhivatkozással számítható, a három-téglalap jellemző nyolc hivatkozással, a négy-téglalap jellemző kilenc hivatkozással.



2.6. ábra. A D-be eső képpontok összege az integrál kép négyszeri elérésével meghatározható. Az integrál kép értéke az 1-es pozícióban az A-ba eső képpontok intenzitásösszege. 2-ben az érték $\boldsymbol{A}+\boldsymbol{B}$, 3-ban $\boldsymbol{A}+\boldsymbol{C}$, 4-ben $\boldsymbol{A}+\boldsymbol{B}+\boldsymbol{C}+\boldsymbol{D}$. Így a D-be eső összeg a $\mathbf{4}+\mathbf{1}-(\mathbf{2}+\mathbf{3})$ összefüggés alapján számítható ki.

2.3.4 A jellemzők előnye a detektálás során

Annak érdekében, hogy – egyszerűségük és látszólagos rugalmatlanságuk ellenére – értékelni tudjuk a téglalap alakú jellemzők használhatóságát, vizsgáljunk meg hatékonysági szempontból egy hagyományosabb megközelítést. Több objektum detektorhoz hasonlóan a mi rendszerünk is többféle skálázás mellett olvassa végig a digitális képet. Kezdetben az objektumokat 24×24-es négyzetekben keressük, és a képet összesen 11 skálázás mellett olvassuk végig. Az ablak minden lépésben 1.25-ször nagyobb, mint az előző mérete volt.

A hagyományos megközelítés ezzel szemben az, hogy a képről egy 11 skálázás melletti piramist építünk fel, amelynél minden szint 1.25-ször kisebb, mint az előző. Majd egy rögzített skálájú detektort futtatunk le minden egyes szinten. Bár a piramis felépítése egyszerű, mégis igen költséges.

Hagyományos személyi számítógépeken még a 15 kép/másodperc sebesség elérése is nehézséget jelent. Ezzel szemben mi jellemzők egy halmazát definiáltuk, amely azzal a tulajdonsággal bír, hogy bármely eleme tetszőleges skálázás mellett néhány művelettel, rendkívül gyorsan meghatározható az értéke. Így akár a 15 kép/másodperc sebességű objektumdetektálás is könnyűszerrel elérhető, ami kevesebb időbe kerül, mint önmagában a

11 szintű piramis felépítése. Következésképpen minden olyan detektor, amely a hagyományos piramis módszert alkalmazza, várhatóan lassabb lesz az általunk tárgyalt megközelítésnél.

2.3.5 Az osztályozó függvények tanítása

Ha adott a jellemzőknek, továbbá pozitív és negatív tanítópéldáknak egy-egy halmaza, akkor a gépi tanulás bármely megközelítése használható egy osztályozó függvény tanulásához. Feltételezve, hogy a detektor alap felbontása 24×24-es, azt kapjuk, hogy a téglalap jellemzők halmazának számossága – a Haar-féle bázisfüggvényektől eltérően – igen nagy: 45.396. Ez a szám sokkal nagyobb, mint a képpontok száma. Habár minden jellemző hatékonyan számítható, a jellemzők teljes halmazának számítása megengedhetetlen. Ezeknek a jellemzőknek már igen kicsi halmazával is létrehozható hatékony osztályozó. Az igazi kihívás ezeknek az osztályozóknak a megtalálása.

A Viola-Jones rendszerben az AdaBoost egy változatát használják mind a jellemző kiválasztásra, mind pedig az osztályozó tanítására. Eredeti formájában az AdaBoost gyenge osztályozó függvények kombinálásával képes egy erős osztályozót előállítani, ezáltal tetszőleges tanuló algoritmus javítására használható.

Az alapgondolat a következő: a gyenge osztályozót lefuttatjuk egy tanítóhalmazon. A teljes tanítóhalmaz osztályozása után az előző osztályozó által hibásan osztályozott példákat újra súlyozzuk, és újra elvégezzük a tanítást. Így a végső erős osztályozó egy perceptron lesz, a gyenge osztályozók lineáris kombinációja, kiegészítve egy küszöbértékkel. Freund és Shapire bebizonyította, hogy az erős osztályozó tanulási hibája a körök számával párhuzamosan exponenciálisan tart nullához. Még fontosabbak az általános teljesítmény számeredményei, melyeket később bizonyítottak [17].

A hagyományos AdaBoost folyamatot könnyen értelmezhetjük egy mohó jellemző kiválasztó folyamatként. A fokozás általános problémáját figyelembe véve, amiben a jellemző osztályozó függvények egy nagy halmazát kombináljuk össze súlyozott többségi szavazás alapján, az igazi kihívás, hogy a jó osztályozókhoz nagy súlyt adjunk, míg a kevésbé jó függvényekhez kicsit. Az AdaBoost egy agresszív eljárás jó osztályozók kis halmazának kiválasztására, amik jelentősen eltérőek lehetnek. Egy analógiát rajzolva a gyenge osztályozók és a jellemzők közé, az AdaBoost egy hatékony eljárás jó jellemzők egy kis halmazának keresésére, melyek jelentősen eltérőek lehetnek.

Egy gyakorlati eljárás ennek az analógiának a kiegészítésére, hogy megszorítjuk a gyenge tanulót az osztályozó függvényeknek arra a halmazára, melyek csak egyszerű jellemzőktől függnek. Ennek eléréséhez a gyenge tanuló algoritmust úgy tervezték meg, hogy azt az egyszerű téglalap jellemzőt válassza ki, amely a legjobban választja szét a pozitív és negatív példákat Minden jellemzőhöz a gyenge tanuló meghatározza az optimális küszöbosztályozó függvényt, ami a példák minimális számú halmazát osztályozza rosszul.

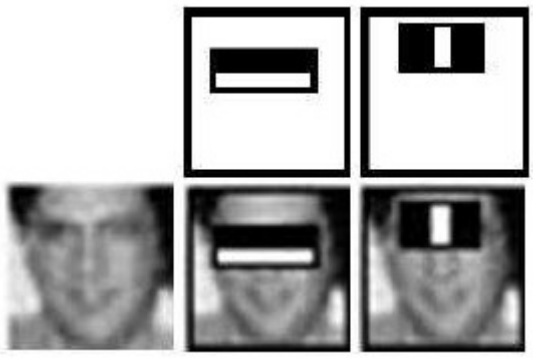
Így egy gyenge osztályozó ($h_j(x)$) egy jellemzőből (f_j), egy küszöbből (θ_j) és egy paritásból (p_j) áll, ahol a paritás az egyenlőtlenség jel irányát jelöli:

$$\mathrm{h_j(x)} = \begin{cases} 1, \text{ha } \mathrm{p_j f_j(x)} < \mathrm{p_j \theta_j}, \\ 0, \text{különben.} \end{cases} \tag{2.4}$$

Ahol x egy 24×24 pixel felbontású kép-alablak. Az AdaBoost algoritmus pszeudókódja, a függelék 9.1. fejezetében található.

2.3.6 A tanítás eredménye

Az implementált rendszerrel végzett kezdeti kísérletek azt mutatták, hogy már mindössze 200 jellemzőből konstruált osztályozó is elfogadható pontossággal működik. Számszerűsítve, 95%-os találati arány mellett a rendszer mindössze 1 hamis pozitív objektumot adott a 14.084-es teszthalmazból. Az arcdetektálás esetében az AdaBoost által kiválasztott jellemzők szemléletesek és könnyen értelmezhetők. Az első jellemző az arc azon tulajdonságára koncentrál, hogy a szemek környéke gyakran sötétebb, mint az orr és a szemek alatti rész.



2.7. ábra. Az AdaBoost által választott első és második jellemző. A két jellemző látható a felső sorban, az alsó sor pedig a jellemzőket egy tanító arcra helyezve mutatja meg.

Ez a jellemző az ablak méretéhez mérten meglehetősen nagy, következésképpen érzéketlen az arc méretére vagy elhelyezkedésére. A második kiválasztott jellemző azt írja le, hogy a szemek sötétebbek, mint az orrnyereg (lásd 2.7. ábra). Az eredmények bíztatóak, de

nem elegendőek a valós idejű alkalmazások készítéséhez. Sajnos a detektálási arány javításának legegyszerűbb módja – újabb jellemzők bevonása – közvetlenül növeli a számítási költségeket, így csökkenti a detektor sebességét.

2.3.7 A sebesség növelése

Ebben a részben röviden áttekintünk egy algoritmust, amellyel osztályozóinkat olyan „vízesés" struktúrába tudjuk rendezni, amely a detektálási arány növelése mellett a sebességet is rendkívüli mértékben megnöveli. Az alapötlet az, hogy könnyű olyan gyenge osztályozókat készíteni, amelyek a negatív példák nagy részét visszautasítják, viszont a pozitívokat elfogadják. Egyszerű osztályozókat használunk tehát a negatív példák elutasítására, és az összetettebb osztályozókat csak a biztató területeken futtatjuk le, hogy minél alacsonyabb hamis pozitív arányt érhessünk el.

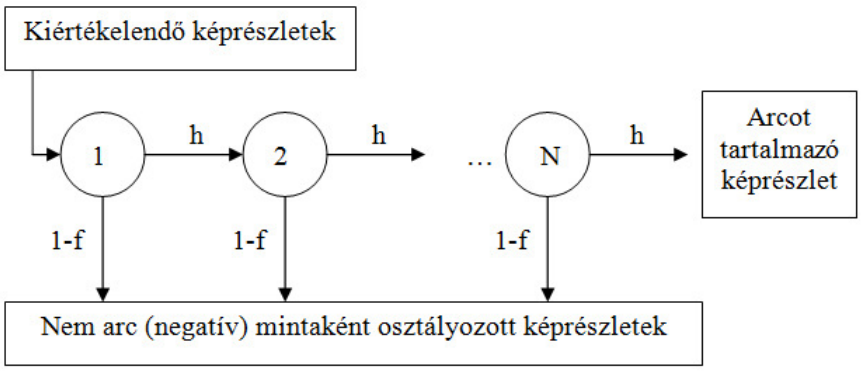
A vízesés egyes lépcsőiben szereplő osztályozókat az AdaBoost algoritmussal tanítjuk be. Az egy, két jellemzős gyenge osztályozó hatékony arcdetektor lehet, ha a küszöbértéket úgy hangoljuk, hogy a hamis pozitív arányt minimalizálja. Az AdaBoost kezdeti küszöbértéke, $\frac{1}{2}\sum_{t=1}^{T}\alpha_t$. Ha ezt csökkentjük, nő a detektálási, de sajnos a hamis pozitív arány is. A kísérletek azt mutatták, hogy a két jellemzős detektor képes az arcok 100%-át detektálni 40%-os hamis pozitív arány mellett. Ez természetesen elfogadhatatlan a gyakorlati alkalmazásokban. Mindamellett nagyon kevés művelettel képes lecsökkenteni azon ablakok számát, amelyek további feldolgozást igényelnek:

1. A téglalap jellemzők kiértékelése (jellemzőként 6-9 olvasás).
2. A gyenge osztályozó kiszámítása minden jellemzőre (jellemzőként 1 művelet).
3. A gyenge osztályozók kombinálása (jellemzőként egy szorzás, egy összeadás, és a végén egy küszöbérték ellenőrzés).

A végső osztályozó egy elfajuló döntési fa, amelyet mi „vízesésnek" nevezünk [21]. Az első osztályozó pozitív válasza elindítja a második osztályozót, amely szintén nagyon magas detektálási aránnyal rendelkezik. Ha ez pozitív választ ad, elindul a harmadik osztályozó, és így tovább. Ha bármely pontnál a válasz nemleges, az ablakot azonnal visszautasítjuk, ahogy ezt a 2.8. ábra is mutatja.

Az osztályozó szerkezete is azt tükrözi, hogy egy képen belül a részablakok túlnyomó többsége negatív példa. Ennek érdekében a vízesés a lehető legkorábbi szakaszban igyekszik elutasítani minél több negatív példát. Mivel a pozitív példára mindegyik osztályozó pozitív

választ ad, így egy ilyen megtalálása az összes ellenőrzéshez képest nagyon ritka esemény. A döntési fához hasonlóan egy adott osztályozót csak az előző osztályozók által pozitívnak ítélt példákkal tanítunk. Következésképpen, a második osztályozónak nehezebb dolga van, mint az elsőnek. Adott detektálási arány mellett a mélyebben elhelyezkedő osztályozók magasabb hamis pozitív aránnyal rendelkeznek. A vízesés struktúra tanítása a függelékben 9.2. fejezetében található.



2.8. ábra. Egy $\boldsymbol{N}$ fokozatú döntési fa szerkezeti felépítése. A fokozatok találati aránya $\boldsymbol{h}$, $\boldsymbol{f}$ pedig a hibásan osztályozott minták aránya. A teljes rendszerre vonatkozó találati arány $\boldsymbol{h^N}$, a hibás osztályozás arány pedig. $\boldsymbol{f^N}$.

2.4 Távolságmérés

2.4.1 Metrika

Metrikus tér alatt egy olyan (X, d) rendezett párt értünk, ahol X egy tetszőleges halmaz, $d: X^2 \to R^+$pedig olyan nemnegatív valós szám értékű függvény, melyre tetszőleges $x, y, z \in X$ esetén:

1. $d(x, y) = 0 \Leftrightarrow \mathrm{x} = \mathrm{y}$ (egyenlőségi tulajdonság) (2.5)
2. $d(x, y) = d(y, x)$ (szimmetria) (2.6)
3. $d(x, z) \leq d(x, y) + d(y, z)$ (háromszög–egyenlőtlenség) (2.7)

Ha (X, d) metrikus tér, X elemeit pontoknak, a $d(x, y)$ függvényt az X feletti metrikának vagy távolságfüggvénynek szokás nevezni.

2.4.2 Euklideszi távolság

A hagyományos euklideszi sík–, és térgeometria pontjai modellezhetőek valós számok rendezett n–eseivel, azaz n–dimenziós vektorokkal. Pl. a sík egy pontja megadható egy $P = (x, y)$ számpárral, a tér egy pontja egy $P = (x, y, z)$ számhármassal. Általában akárhány n-dimenziós térben is a pontok megadhatóak a $P = (p_1, p_2, \ldots, p_n)$ szám–n–essel, ahol a p_i számot a pont i–edik koordinátájának nevezzük. A valós szám–n–esek halmazát R^n–nel jelölve, értelmezhető a következő d_n–nel jelölt metrika:

$$d_n(P, Q) \coloneqq \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{2.8}$$

Az euklideszi távolságot definiálhatjuk pont és ponthalmaz között is:

$$\mathrm{d}(\mathrm{x}, \mathrm{A}) \coloneqq \min_{\mathrm{y} \in \mathrm{A}} \mathrm{d}(\mathrm{x},\mathrm{y}) \tag{2.9}$$

2.4.3 Probléma a színtér szegmentálása során

Az egyes objektumok színének meghatározásához célszerű a színterünk egyes színeit klaszterekbe foglalni. A művelet elvégzéséhez kiindulásként az egyes klaszterekbe alappontokat kell felvennünk, majd az RGB tér szegmentálását ezen alappontok mentén végezzük el. A szegmentálás során a tér minden egyes pontjára meg kell határozni az adott pont és az egyes klaszterek, mint ponthalmazok távolságát. Mivel az RGB tér nagyjából 16 millió pontot tartalmaz, így látható, hogy a 2.9-es képlet a színek száma miatt nem alkalmazható a gyakorlatban. Ez a fejezet ennek a problémának a megoldására kíván megoldást nyújtani.

2.4.4 Távolság transzformáció

Ebben a fejezetben egy az euklideszi távolság approximálására alkalmas technikát fogunk bemutatni [22,23]. A módszer neve távolság transzformáció, mely széles körben használható módszer, ugyanis a távolságmérés központi szerepet játszik a bináris képek összehasonlításában, az él-, illetve sarokdetektálásban, osztályozási feladatokban és térképészeti alkalmazásokban.

Alkalmazása azért előnyös számunkra, mert egész aritmetikát használva tudjuk a bináris képpontoknak a színklaszterektől mért távolságát közelíteni. Úgy is fogalmazhatunk, hogy a távolság transzformáció egy útkereső eljárás, amely meghatározza, hogy a színtér egyes pontjaiból kiindulva, melyik színklaszter érhető el a legrövidebb úton.

Legyen adva egy kétdimenziós bináris képünk $I(x,y)$, osszuk a képpontjait két halmazba (az objektumpontok-, illetve a háttérpontok halmazába).

$$\mathrm{I(x,y)} \in \{\mathrm{Ob, Bg}\} \tag{2.10}$$

Ennek a bináris képnek az $I_d(x,y)$ távolság távolságtranszformáltját úgy kapjuk, hogy minden egyes pixelét felcímkézzük a pixel és a hozzá legközelebb eső háttérpont távolságával. Matematikailag:

$$\mathrm{I_d(x,y)} = \begin{cases} 0, \text{ha } \mathrm{I(x,y)} \in \{\mathrm{Bg}\}, \\ \min(\|\mathrm{x}-\mathrm{x_0}, \mathrm{y}-\mathrm{y_0}\|, \forall \mathrm{I(x_0,y_0)} \in \mathrm{Bg}) \text{, ha } \mathrm{I(x,y)} \in \{\mathrm{Ob}\}, \end{cases} \tag{2.11}$$

ahol $\|x,y\|$ egy kétdimenziós metrika. Értelemszerűen különböző metrikák, különböző távolság transzformáltat eredményeznek. Az esetek többségében az 2.8-as képletben ismertetett euklideszi távolságot használjuk.

Ez egy izotróp metrika, melyben a mért távolságok függetlenek a tárgy helyzetétől, természetesen azzal a megkötéssel, hogy a tárgyak határai digitálisak így pontjaink csak diszkrét helyeken vannak értelmezve. A legnagyobb korlátozása az euklideszi metrikának az, hogy a digitális képfeldolgozás világában, komplex alakzatok esetében nehéz hatékonyan kiszámolni. Ennek érdekében több olyan közelítést is definiáltak, melyek a kétdimenziós digitális képek rácsszerkezetének köszönhetően könnyebben használhatók. Ezek közül az első a „city block”, vagy Manhattan távolság (lásd 2.9b. ábra), mely az L_1 normát használja:

$$\|\mathrm{x,y}\|_{\mathrm{L_1}} = |\mathrm{x}| + |\mathrm{y}|, \tag{2.12}$$

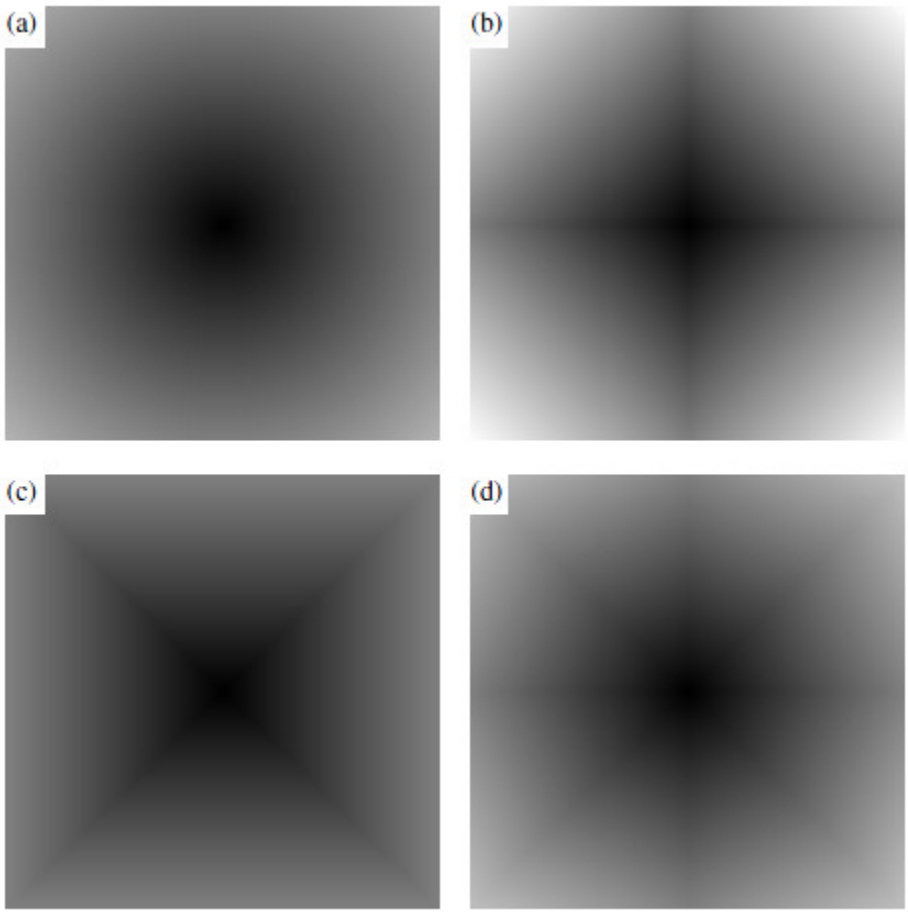
ahol a távolság az (x,y) pontba jutáshoz szükséges vízszintes és függőleges lépések számát jelenti. A technika egyik nagy hátránya, hogy csak a 4-szomszédok irányába tudunk lépni,

tehát egy átlós lépést, csak egy függőleges és vízszintes lépés segítségével lehet realizálni. Egy másik általánosan használt módszer, a sakktábla metrika (lásd 2.9c. ábra), mely a L_∞ normát használja:

$$\|\mathrm{x},\mathrm{y}\|_{\mathrm{L}_\infty} = \max(|\mathrm{x}|,|\mathrm{y}|) \qquad 2.13$$

Mely egy sakktáblán a királynak az (x, y) pontba jutáshoz szükséges lépéseinek darabszámát méri. Ezzel a módszerrel a pontok 8-szomszédait lehet elérni egy lépés során. Az euklideszi távolság közelítése érdekében különböző metrikák széles halmazát definiálták. Ezek közül több a Manhattan és a sakktábla távolság egyszerű számítási szabályain alapszik. Pl. az előző kettő átlagaként definiált hibrid metrika (lásd 2.9d. ábra):

$$\|\mathrm{x},\mathrm{y}\|_{\mathrm{Hibrid}} = \frac{1}{2}\left(|\mathrm{x}| + |\mathrm{y}| + \max(|\mathrm{x}|,|\mathrm{y}|)\right) \qquad 2.14$$



2.9. ábra. Grafikus összehasonlítása a különböző metrikáknak. Minden képen a képpontoknak a kép középpontjától mért távolságát szemléltetjük. (a) euklideszi távolság; (b) city block távolság; (c) sakktábla távolság; (d) hibrid metrika, az (b) és (c) keveréke.

2.4.5 Chamfer távolság transzformáció – kétmenetes algoritmus

A módszer segítségével a bináris képet kétszer bejárva – egyszer a bal felső sarokból indulva a jobb alsó felé, másodjára pedig a jobb alsótól indulva a bal felső felé - határozhatjuk meg annak a távolság transzformáltját. Ezzel a két bejárással határozzuk meg az egyes pixeleknek a tárgy felső-, és bal élétől mért, valamint a tárgy alsó-, és jobb élétől mért távolságát.

A bejárás során távolság maszkokat használunk. A maszk egy pixelnek és a hozzá legközelebbi szomszédjának távolságot adja vissza. Ezt két irányba futtatva egy olyan képet ad eredményül, ahol a tárgyak két széle között lévő legtávolabbi, vagyis középső pontok lokális maximumként jelentkeznek. Ha egy 3×3-as maszkot használunk, akkor az első bejárás az aktuális pixelnek a három felette lévő pixel és a tőle közvetlenül balra lévő pixel távolsága közül a legkisebbet adja vissza, képlettel:

$$\mathrm{I_d}(\mathrm{x},\mathrm{y}) = \min\begin{pmatrix}\mathrm{I_d}(\mathrm{x}-1,\mathrm{y}-1)+\mathrm{b}, & \mathrm{I_d}(\mathrm{x},\mathrm{y}-1)+\mathrm{a}, & \mathrm{I_d}(\mathrm{x}+1,\mathrm{y}-1)+\mathrm{b}, \\ \mathrm{I_d}(\mathrm{x}-1,\mathrm{y})+\mathrm{a} & & \end{pmatrix}, \quad 2.15$$

ahol az a és b növekmények értéke attól függ, hogy a képpontok 4-, vagy 8-szomszédossági relációban vannak-e egymással. A háttérpontoknak előzetesen 0 értéket adtunk.

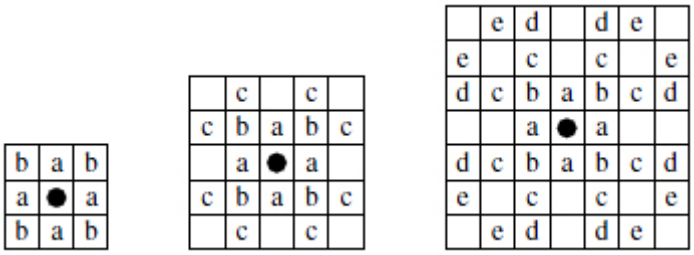
A második bejárás az aktuális pixelnek a három alatta lévő pixel és a tőle közvetlenül jobbra lévő pixel távolsága közül a legkisebbet adja vissza. A második bejárás csak akkor írja felül a távolság értéket, ha az kisebb az első körben számított értéknél. Ugyanis ebben az esetben a vizsgált pixel közelebb helyezkedik el a tárgy alsó-, vagy jobb széléhez:

$$\mathrm{I_d}(\mathrm{x},\mathrm{y}) = \min\begin{pmatrix}\mathrm{I_d}(\mathrm{x},\mathrm{y}), & \mathrm{I_d}(\mathrm{x}+1,\mathrm{y})+\mathrm{a}, & \mathrm{I_d}(\mathrm{x}-1,\mathrm{y}+1)+\mathrm{b}, \\ \mathrm{I_d}(\mathrm{x},\mathrm{y}+1)+\mathrm{a}, & \mathrm{I_d}(\mathrm{x}+1,\mathrm{y}+1)+\mathrm{b} & \end{pmatrix}. \quad 2.16$$

Az a és b növekmények különböző értékei, különböző metrikákat eredményeznek. A city block távolságot, az $a = 1$ és $b = 2$ értékekkel realizálják, a sakktábla távolság esetében $a = b = 1$. Az euklideszi távolságnak egy jobb közelítését kapjuk, ha az $a = 3$ és $b = 4$ növekményeket használjuk, és az eredményt elosztjuk 4-el. Ez egy a 2.9d. ábrán látható nyolcszög alakú mintát eredményez.

Általánosságban elmondható, hogy a növekmények optimalizálásával egyre pontosabb távolságokat kapunk, de a közelítés pontossága nagyban függ a maszk méretétől is. Ugyanis egy nagyobb maszk segítségével több utat tudunk összehasonlítani, 3×3-as esetben 4-et, 5×5-ös esetben 8-at, 7×7-es esetben pedig 16-ot (lásd 2.10. ábra). Egy 5×5-ös maszk

elfogadható kompromisszumot szolgáltat a számítás komplexitása és a közelítés pontossága között. A Chamfer távolság transzformáció műveleteinek száma minden egyes pixel esetében fix, továbbá elmondható még, hogy az időbonyolultsága a kép pixeleinek számával arányos.



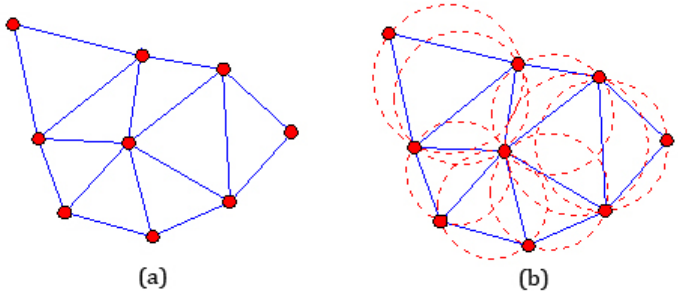
2.10. ábra. A növekmények helyei a 3×3-as, 5×5-ös, 7×7-es méretű maszkokon belül. Az üres helyek nem vesznek részt a számításban.

2.5 Voronoi felosztás

Színterünk szegmentálása során a klaszterek alappontjai köré egy olyan rácsszerkezetet szeretnénk kialakítani, melynek minden belső pontja közelebb van az adott klaszter alappontjaihoz, mint az összes többi ponthoz [24]. Ezáltal minden egyes klaszterhez egy konvex sokszöget rendelünk hozzá, melyek az azonos színűnek látott pixeleket foglalják magukban. Ez a fejezet az előbb felvázolt probléma megoldására ismertet egy Voronoi felosztásnak nevezett módszert.

2.5.1 Delaunay háromszögelés

Egy adott *P* ponthalmaz Delaunay-háromszögelése egy olyan egyenes szakaszokból álló vonalhálózat, aminek sokszögtartományai köré írt gömbjei csak határukon tartalmazzák a *P* ponthalmaz pontjait. A korlátos sokszögtartományok tehát húrsokszögek. Ha ezek a tartományok nem mind háromszögek, akkor a Delaunay-háromszögelés elfajuló, egyébként valódi (lásd 2.11. ábra). A Delaunay-háromszögelés akkor és csak akkor valódi, ha a *P* halmaz pontjai között semelyik három nincs egy egyenesen, és semelyik négy nincs egy körön. Az élek száma lineárisan függ a pontok számától. A *P* ponthalmaz valódi Delaunay-háromszögelésének fontos tulajdonsága, hogy a *P* halmaz összes háromszögelés között maximalizálja a háromszögek legkisebb szögét.



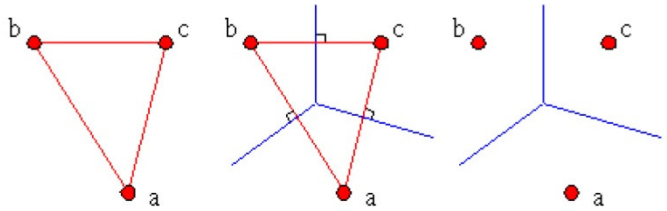
2.11. ábra. (a) Delaunay háromszögelés; a (b) ábrán látható, hogy a háromszögelés valódi Delaunay-háromszögelés.

2.5.2 Voronoi diagram

Legyenek a síkon (térben) szabálytalan elrendezésű pontjaink. Minden pont köré szerkeszthető egy olyan sokszög (poliéder), melynek belső pontjai (összes pontja a határát alkotó pontok kivételével) közelebb vannak a kérdéses ponthoz, mint az összes többi ponthoz. Az ilyen tulajdonsággal rendelkező sokszögek (poliéderek) konvexek és folytonosan töltik ki a síkot (teret). A meghatározásból következik, hogy a sokszög oldalai (a poliéder

oldallapjai) merőlegesek a körülvett pontot a többi ponttal összekötő egyenesekre és felezik azokat. Tulajdonképpen a Voronoi diagram egy pont halmaz közelségi diagramjaként fogható fel, mely segítségével a teret úgy oszthatjuk fel, hogy a tartomány minden pontja közelebb van a tartomány pontjához, mint bármely más ponthoz.

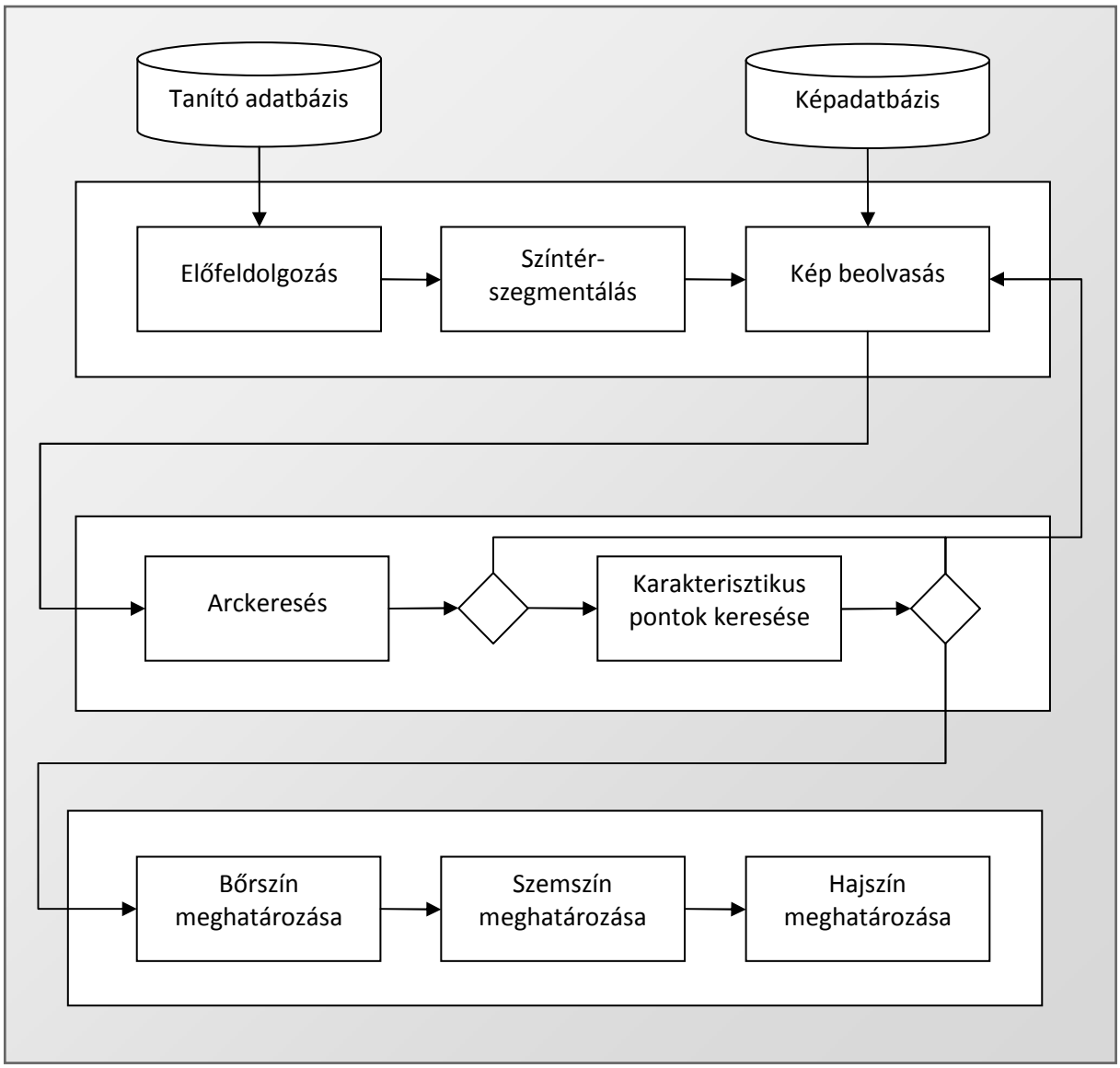
A Voronoi diagram valójában a Delaunay háromszögelés geometriai duálisa. Duális lévén egyik a másikból megkapható. A módszer az, hogy a Delaunay háromszögek éleinek felezőmerőlegeseit behúzzuk (lásd 2.12. ábra).



2.12. ábra. Delaunay háromszög Voronoi diagrammá alakítása.

3 Arci jellemzők szegmentálása

A rendszer tervezése és fejlesztése során az eddigiekben ismertetett módszerek közül több is megvalósításra került, ezek közül a végleges alkalmazás a valamely szempontból jobbnak bizonyult eljárásokat használja, amelyek e fejezetben kerülnek felhasználásra. Mivel a cél egy valós idejű rendszer tervezése, így az algoritmusok kiválasztásánál, illetve fejlesztésénél a fő szempont azok sebessége volt, de természetesen a hatékonyság is hasonló jelentőséggel bír. Ezen megfontolások alapján a rendszer vázát három fő modul képzi, amelyek az adatbázisból kapott kép alapján képesek arc–, illetve annak a karakterisztikus pontjainak a detektálására, valamint ezek alapján bizonyos adottságok meghatározására (lásd 3.1. ábra).



3.1. ábra. A rendszer szerkezeti felépítése.

3.1 Előkészületek

A dolgozat elsődleges célja az volt, hogy egy arcokat tartalmazó adatbázisból – meta információk használata nélkül – lekérdezhetők legyenek bizonyos arci paraméterek, mint pl. a bőrszín, a szemszín, és a hajszín. Az előbbi kérdések megválaszolásához, rendelkeznünk kell azokkal az ismeretekkel, hogy a bemeneti kép melyik területén kell vizsgálatokat folytatnunk.

Az arci jellemzők színének meghatározásához első lépésben arcot kell detektálni az adatbázisbeli képeken. Az arcdetektálásnak széles irodalma van, számos megoldást publikáltak ebben az irányban. Talán az egyik legsikeresebb eljárás, a Viola és Jones által kidolgozott módszer, mely hatékonyságát tekintve nem marad alul a többivel szemben, de a gyorsaságát tekintve felülmúlja az összes eljárást.

A 2.3-as fejezetben ismertetett Viola-Jones detektorok segítségével elég nagy találati arány mellett végezhető el az arc detektálása, de még így is születhetnek hamis találatok. Ezek azonban könnyen kiszűrhetők a következő algoritmussal: egy Viola-Jones szemdetektorral detektáljunk szemet az arcdetektor által visszaadott képrégió felső felében. Ha a szemdetektor nem ad vissza egyetlen találatot sem, akkor vessük el a detektált arcot hamis pozitívként, ellenkező esetben fogadjuk el pozitív találatként. Ezzel a módszerrel egyrészt a szivárványhártya színének meghatározásához szükséges régiót is megkapjuk, másrészt az arcdetektor egyfajta validálásának is felfogható a szemdetektálás.

A fentebb vázolt módszerrel tehát képesek vagyunk a keresett objektumok színének meghatározására. Az egyes jellemzők szegmentálásához további előkészületek szükségesek, melyeket az alábbi fejezetekben külön-külön ismertetek. A szegmentálás eredményét a színmeghatározó modulban fogjuk felhasználni (lásd 4. fejezet).

Habár a szegmentálás nagyrészt szerkezeti információkon alapszik, mégis néhány esetben rosszul szegmentált régiókat eredményezhet. Azonban jó végeredmény elérésére vagyunk képesek, ha ezt az eredményt a színtér szegmentálása során szerzett tapasztalatokkal vegyítjük.

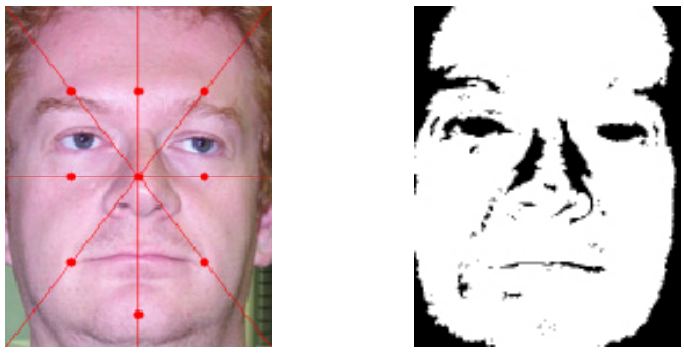
3.2 Az arcbőr szegmentálása

Az arcbőr szegmentálása során az a célunk, hogy a bőrt elkülönítsük a detektált arc többi részétől (pl. szem, haj, arcszőrzet), majd a színmeghatározó modul ezek közül a szegmentált pixelek közül fogja a leggyakrabban előfordulót bőrszínnek választani.

Abból a feltevésből indulunk ki, hogy az arcon az arcbőr alkotja a legnagyobb hasonló világosságkóddal rendelkező összefüggő régiót – olyan régió, melyben a színek közötti távolság kisebb, mint egy tapasztalati úton számított küszöbérték.

Az arcbőrrégió meghatározásához egy régiónövelő eljárást fogunk használni. Az algoritmus szétválogatja az arcbőrhöz tartozó pixeleket, azoktól a részektől ahol az arcbőr nem látható, vagy azoktól ahol annak a színe megváltozik (pl. rossz megvilágítás miatt). Az eljárás menete nagyon hasonló a számítógépes grafikában alkalmazott üregkitöltő algoritmusokéhoz, amikor egy homogén színű zárt tartományt töltünk ki egy adott színnel.

- A különböző megvilágítási hatások és az arc helyi gödrei miatt a régiónövelést több különböző pontból indítjuk (lásd 3.2a. ábra).
- Minden egyes lépésben a külső határpontok világosságkódját hasonlítjuk össze az addig elkészült régió átlagolt színével. Azokat a pontokat adjuk hozzá a régióhoz, melyek színének távolsága a régió átlag színétől, tíz százalék alá esik.
- A régiónövelések által kapott régiókat egyesítjük egy közös maszkba, amit a továbbiakban az arcbőr helyének tekintünk (lásd 3.2b. ábra).



(a) (b)

3.2. ábra. A bőrrégió meghatározása: (a) a régiónövelés kezdőpontjai, (b) a régiónövelés által eredményezett bináris maszk.

3.3 A szem szegmentálása

A szemszín meghatározása komplikáltabb, mint a bőrszíné. Ugyanis ebben az esetben nem járható út az, hogy a szemrégió színét számítjuk ki. Ebben az esetben előbb a szemrégión belül elhelyezkedő szivárványhártyát kell megkeresnünk, hiszen ténylegesen ez határozza meg a szem színét.

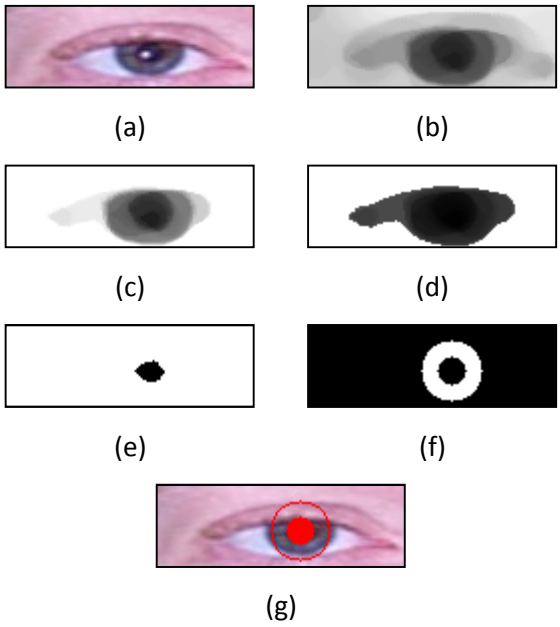
Az algoritmus első lépésében egy Viola-Jones detektor segítségével szemet detektálunk az arcon (lásd 3.3a. ábra). A hamis találatok kiszűrése érdekében a szemdetektálást az arc bal felső negyedébe korlátozzuk, ily módon csak a bal szemet keressük. Emlékeztetőül: ha a detektor nem talált szemet az arcon, akkor az arcot hamis találatként értelmezzük és egyetlen vizsgálatokat sem végzünk rajta. Következő lépésben a pupilla és a szivárványhártya régió pontos helyét keressük meg a szemdetektor által visszaadott szemen:

- A zajok (pl. szempilla, szemöldök) kiküszöbölése érdekében egy mediánszűrést (11×11-es maszkkal) végzünk a képen.
- Átkonvertáljuk a szemet tartalmazó képet CIE XYZ színtérbe, majd kimentjük a Z csatornát egy egycsatornás – magyarán szürkeskálás – képbe (lásd 3.3b. ábra). A „Z” csatorna használatával egy sokkal kontrasztosabb képet kapunk, mintha egyszerűen csak szürkeskálás képpé konvertálnánk a szemet tartalmazó képet [25,4]. Ez a lépés azért hasznos számunkra, mert a szemszín meghatározó algoritmusunk a szemrégiót alkotó viszonylag kontrasztos szivárványhártya–pupilla részek és a kevésbé kontrasztos szemhéj és környezete részek szétválasztásán alapul. Így azt szeretnénk elérni, hogy e részek között, a kontrasztosságot tekintve a lehető legnagyobb legyen a különbség.
- További kontrasztnövelést hajtunk végre a szemen (lásd 3.3c. ábra).
- A szürkeskálás képen a pupilla és szivárványhártya részek általában sötétebbek, mint a szem összes többi része. Egy hisztogram-kiegyenlítést végzünk ezen a képen, abból a célból, hogy pupilla és szivárványhártya részek az összes képen a legsötétebb intenzitásértékkel legyenek kigyújtva (lásd 3.3d. ábra).
- Majd az így kapott eredményképen egy küszöbölés segítségével szétválasztjuk a pupilla és szivárványhártya részeket a szem többi részétől (20-as szürkeségi küszöbértéket használtunk).

- A kisebb zajokat egy erózió morfológiai operátor segítségével eltávolítjuk a képről (11×11-es téglalapalakú maszkot használtunk) (lásd 3.3e. ábra).
- A szivárványhártya és a pupilla elkülönítésére egy kör alakú maszkot használunk (lásd 3.3f. ábra), melynek sugarát a Viola-Jones detektor által visszaadott ablak dimenziója alapján számítottuk ki. Abban az esetben, ha a visszaadott ablak $W \times H$ méretű, akkor a kör alakú maszk sugara:

$$R_{mask} = H * 0.8$$

A maszkot az előző lépés által visszaadott képen a fekete színű pixelek súlypontjára helyezzük rá (lásd 3.3g. ábra).

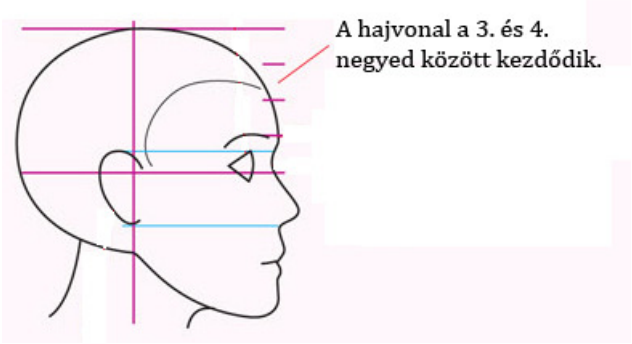


3.3. ábra. A szivárványhártya detektálása: (a) a detektált szem, (b) XYZ transzformáció, (c) kontrasztnövelt kép, (d) hisztogramkiegyenlített kép, (e) küszöbölés utáni eredmény, (f) a szivárványhártyához tartozó maszk, (g) maszkolt kép.

3.4 A haj szegmentálása

A számos hajviseletnek, stílusnak, hajszínnek és egyéb tényezőknek (pl. kopaszodás) köszönhetően, a hajszín meghatározása során koránt sincs annyira egyszerű dolgunk, mint az előző fejezetekben. Ebből kifolyólag lesznek olyan esetek, amikor a hajszín nem állapítható meg egyértelműen. A hajszín meghatározásához először a bőrrégiót kell megtalálni, mivel feltételezhető, hogy a bőrrégió feletti félhold alakú területben helyezkedik el a haj. A haj szegmentálásának lépései, a következők:

- Első lépésben átméretezzük a képeket úgy, hogy az arc mérete 400×400-as legyen. Majd az átméretezett képeken egy mediánszűrést végzünk (7×7-es maszkkal) az apró zajok eltávolításának érdekében.



3.4. ábra. A hajvonal kezdete a fejen. A hajvonal a szem alatt kezdődő, a fej tetejéig húzódó régió 3. és 4. negyede között helyezkedik el.

- Következő lépésben egy félhold alakú maszkot illesztünk az arcbőrt tartalmazó régió tetejére (lásd 3.5a. ábra), melynek a kiterjedését és centrumát az emberi fej biológiai arányai alapján számítjuk ki. Ha a Viola-Jones arcdetektor által visszaadott régió W hosszú és H magas, akkor a félhold alakú tartomány dimenzióit a következő formula alapján számíthatjuk ki:

$$W_{mask} = W * 0.8, H_{mask} = H * 0.4$$

- A félhold alakú régió ívének szintén az arc biológiai arányai miatt a fej felső felének 3. és 4. negyede között kell mennie (lásd 3.4. ábra).

- A hajat egy az arcszín szegmentálása során ismertetett régiónövelő eljárás segítségével szegmentáljuk. Első lépésben a félhold alakú régióban keressük meg a legnagyobb összefüggő komponenst, öt véletlenszerűen kiválasztott pont esetében. Második

lépésben elvégzünk egy újabb régiónövelést abból a pontból kiindulva, amelyikből a legnagyobb összefüggő komponenst kaptuk, de ezúttal nem korlátozzuk le az eljárást a félhold alakú maszkra (lásd 3.5b. ábra).

(a) (b)

3.5. ábra. A hajrégió megkeresése: (a) a félhold alakú maszk, (b) a régiónövelő eljárás eredménye.

4 Színinformáció kinyerése

A színterünk elméletileg végtelen, de a gyakorlatban is óriási mennyiségű szín reprezentálását teszi lehetővé. Azonban a legtöbb alkalmazás szempontjából elegendő lenne csak véges sok szín használata. Gondoljunk bele, hogy hány különböző hajszínt szoktak az emberek megnevezni? Ebből kifolyólag a gyakorlatban használható közel 16 millió színt tartalmazó színterünknek elegendő csak egy kis részhalmazát használni a lekérdezések során [26]. Ennek érdekében a színtéren olyan osztályozást kell definiálnunk, amelyeknek eredményeként létrejött osztályokat a lekérdezések során, az emberi érzékelés szempontjából egy közös címkével látunk el. Magyarán a színtér minden egyes elemét színkategóriákba soroljuk be. Ennek a leképezésnek a menetét a színmodellünkben definiáljuk.

A színmodellünkben a Broek által kifejlesztett szabályrendszert használjuk fel [27]. Az alapötlet az, hogy a folytonos színterünket humán megfigyelések alapján képezzük le diszkrét részhalmazokra. Vagyis az osztályozást pszichológiai kísérletek alapján végezzük el Emberek véleményét kértük ki, hogy milyennek látják a tanító adatbázisbeli képeken szereplő emberek bőr–, szem–, és hajszínét. Az eredmények összegzése és feldolgozása után az adatbázisbeli képek mindegyikéhez három címkét rendeltünk, melyek mindegyike az általunk kérdezett objektum színének nevét takarja. Azonban a színtér szegmentálásához nem szöveges információkra, hanem konkrét világosságkódokra van szükségünk, így a tanító adatbázisbeli képekhez rendelt szöveges címkéket le kell képeznünk a színterünk színeire.

A színtér szegmentálásának első lépésében tehát a tanító adatbázisbeli képek és a pszichológiai kísérletek alapján csoportosítjuk a színeket, majd a tényleges lekérdezések során ennek a csoportosításnak az eredményét használjuk fel a színinformáció kinyerésére.

4.1 Színek csoportosítása

A színtér szegmentálásának első lépésében színkategóriákat kell definiálnunk, a pszichológiai kísérletek elvégzéséhez. A kísérletek két fázisból állnak. Az első fázisban ki kell jelölnünk a lehetséges színkategóriákat. Minden egyes jellemzőhöz öt kategóriát rendeltünk [28,29], melyek az 4.1. táblázatban láthatók. A kategorizálás speciális jelleméből fakadóan a szóba jöhető színek nem fedik le teljesen a színterünket, így minden egyes jellemzőhöz definiáltunk még egy hatodik kategóriát is, melybe az összes többi színt soroltuk be.

Bőr	Szem	Haj
nagyon világos	kék	szőke
világos	szürke	aranybarna
átmeneti	zöld	barna
sötét vagy barna	barna	fekete vagy sötétbarna
nagyon sötét	sötétbarna	szürke vagy fehér
nem bőrszín	nem szemszín	nem hajszín

4.1. táblázat. A színkategóriák.

A második fázisban 40 emberrel töltettünk ki egy internetes kérdőívet (lásd 4.1. ábra). A kérdőív 40 oldalból állt és minden egyes oldal egy szemből fényképezett arcot (800×600-as felbontással) tartalmazott, melyről a kísérletben résztvevőknek el kellett dönteniük, hogy az arc egyes jellemzőinek színe az 4.1. táblázatban megadott kategóriák közül melyikbe tartozik.

A fényképeket több arcadatbázisból [30,31] és az internet segítségével a Google képkereső szolgáltatásával gyűjtöttük össze. A megkérdezettek véleményének összesítésével megkaptuk az egyes jellemzők színkategóriáinak alapelemeit. Magyarán minden adatbázisbeli képhez három címkét rendeltünk, melyek az adott jellemzőknek az emberek többsége által látott színét jelenti.

Bizonyos esetekben, a kísérletben résztvevő emberek egymásnak ellentmondó módon szavaztak a jellemzők színére. A válaszokat két csoportba osztottuk: külön csoportba kerültek azok a képek, melyek a szavazás során minden egyes jellemzőjük esetén a nyertes színkategória 50% feletti eredményt ért el. Itt ugyanis a többi színkategóriával vett különbség erősen szignifikáns, így a felcímkézett jellemzőt eredményesen használhatjuk fel a színtér szegmentálása során. A másik csoportba azok a képek kerültek, melyek jellemzőinek a színe

a megkérdezettek szavazatai alapján egyértelműen nem dönthető el. Ezeket a képeket kivettük az adatbázisból, így azok nem vesznek részt a további feldolgozások során.

Mint már említettem a színtér szegmentálásához nem szöveges információkra, hanem konkrét világosságkódokra van szükségünk, így a tanító adatbázisbeli képekhez rendelt szöveges címkéket le kell képeznünk a színterünk színeire. Magyarán a minden egyes jellemző, minden egyes címkéjéhez egy RGB értéket kell rendelnünk.

A leképezés első lépésében szegmentáljuk az adott jellemzőt, tehát meghatározzuk azt a régiót, amely a képen az jellemzőt tartalmazza. Majd következő lépésben az elkészült bináris maszkot felhasználva átlagoljuk a maszk alatti RGB értékeket. A címkékhez az átlagolt RGB értékeket rendeljük hozzá. Az azonos címkével rendelkező átlagolt RGB értékeket – jellemzőkként csoportosítva – a címke alapján klaszterekbe foglaljuk, majd a továbbiakban e klaszterek segítségével végezzük el a színtér szegmentálását.

4.1. ábra. Az internetes kérdőív.

4.2 Színtér szegmentálása

Az előző fejezet során előállltak azok a klaszterek, melyek alapján az RGB tér szegmentálása elvégezhető. Az RGB tér minden pontjára, egyesével fogjuk meghatározni, hogy az melyik klaszterhez tartozik. A kapott eredményeket egy CLUT (Color LookUp Table – Szín Visszakeresési Táblázat) nevezetű táblázatban tároljuk, hogy az időigényes szegmentálást csak egyszer, az inicializáló fázisban kelljen elvégezni. A keresés során ennek a CLUT–nak a közvetlen címzésével határozható meg az, hogy az adott intenzitásértéket az emberek többsége milyen színnek látja.

Az RGB tér szegmentálását végezhetnénk úgy, hogy kiszámoljuk minden egyes pontjának, a minden egyes klasztertől mért euklideszi távolságát (lásd 2.4.2 fejezet), majd ahhoz a klaszterhez soroljuk be a pontot, amelyik klaszterhez a legkisebb távolság adódott. Azonban ez az RGB tér elemszáma miatt meglehetősen lassú módszer, továbbá a másik probléma az, hogy az egyes csoportok között nem lesznek letisztultak a határok.

A szín–klaszter összerendelési problémára az előző bekezdésben ismertetett eljárás helyett, a szegmentálást a HSI sík két 2D-s vetületével végezzük el. A szegmentálásnak ez a típusa azért hasznos számunkra, mert a színeink két nagy csoportba oszthatók: vannak akromatikus színek (pl. fehér, szürke, fekete), illetve kromatikus színek (pl. a kék, zöld, piros). A HSI térben az akromatikus színek leírásában nem játszik jelentős szerepet a H koordináta, így az elhagyható az akromatikus–, és a kromatikus színcsoportok szeparálása során. A kromatikus színek valódi színét pedig elsősorban a H és az I koordináták határozzák meg (pl. az RGB tér alap–, és kiegészítő színeinek a telítettsége egyenlő eggyel), így az S koordináta minden további jelentős következmény nélkül elhagyható a kromatikus színek szegmentálása során [27]. Ennek megfelelően a három dimenzióban történő szegmentálás helyettesíthető, két kétdimenziós sík szegmentálásával. Magyarán a 3D-ban történő szeparáló hipersíkok keresése helyett, csak kétdimenziós szeparáló síkok egyenletét kell meghatároznunk [32].

Tehát a színtér szegmentálása az alábbi két lépés segítségével oldható meg:

- Először elvégezzük az akromatikus és kromatikus színek szeparálását azzal, hogy a 3D-s színterünket rávetítjük az SI síkra, majd az alappontok alapján elvégezzük az SI sík szegmentálását.

- Másodszor elvégezzük a kromatikus színek szegmentálását azzal, hogy rávetítjük a 3D-s színterünket a HI síkre, és az alappontok mentén elvégezzük az HI sík szegmentálását.

Mivel a szegmentálás a HSI színtérben történik, viszont a bemeneti képeink és a klasztereink alappontjai RGB színtérben vannak értelmezve, így első lépésben le kell képeznünk az RGB színterünk elemeit a HSI színtérre. A leképezést az alábbi képlet segítségével végeztük el [4]:

$$H = \begin{cases} Nincs\ értelmezve, ha\ S = 0 \\ 360° - cos^{-1}\left(\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)*(G-B)]^{1/2}}\right), ha\ \frac{B}{I} > \frac{G}{I}, \\ cos^{-1}\left(\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)*(G-B)]^{1/2}}\right), egyébként. \end{cases}$$

$$S = \begin{cases} Nincs\ definiálva, ha\ I = 0, \\ 1 - \frac{3}{R+G+B} * \min(R,G,B)\ , különben, ahol \end{cases}$$

$$I = \frac{R+G+B}{3}.$$

4.2. táblázat. Az RGB-HSI konverzió

A 2D-s HSI síkok szegmentálása az alább ismertetett algoritmus segítségével történik:

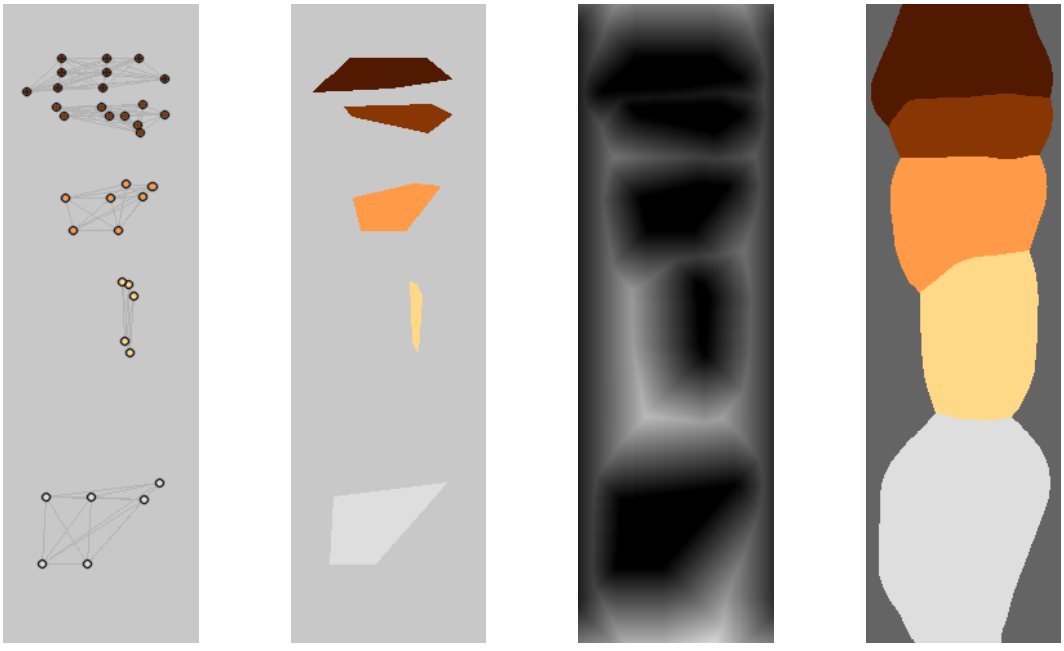
- Vegyük a klaszterek alappontjait, melyeket a megfelelő koordináta elhagyásával ábrázoljunk a fentebb említett síkokon (lásd 4.3a. ábra).
- Második lépésben elkészítjük a síkokon felvett alappontok konvex burkait (lásd 4.3b. ábra) és a burkokat alkotó pontokat hozzávesszük az eredeti klaszterekhez. Ennek reprezentálására egy bináris képet használtunk, ahol a 0-ás érték jelentette azt, hogy az adott pixel valamelyik klaszterhez tartozik és a 1-es érték jelentette azt, hogy a pixel egyik klaszterbe sem esik bele (tehát nem alappont és nem is pontja egyik konvex buroknak sem).
- A pszichológiai kísérletek eredményét befolyásoló emberi tényező miatt lehetnek olyan konvex burkaink, melyek átfedik egymást. Ezen átfedéseket meg kell szüntetni, mivel azt az érzetet keltik, hogy az átfedésben lévő klaszterekhez tartozó pontok egy klaszterhez tartoznak, nem pedig több különbözőhöz. Így az egymást átfedő klaszterekből elhagyjuk a közös részeket.

- Következő lépésben az RGB tér, HSI térbe transzformált pontjait, az előbb ismertetett konvex sokszögek pontjaihoz kell mérni. A vizsgált pontot abba a klaszterbe kell sorolni, amelyhez a legközelebb esik. Ennek eldöntésére egy Chamfer távolság-transzformációt (lásd 2.4.5. fejezet) hajtunk végre az előbbi bináris képen a lentebb látható bináris maszkkal (lásd 4.2. ábra).

	11		11	
11	7	5	7	11
	5	p	5	
11	7	5	7	11
	11		11	

4.2. ábra. A távolság-transzformáció során használt maszk.

- A távolság-transzformáció eredményeképpen egy szürkeskálás képet kapunk (lásd 4.3c. ábra), melynek a Voronoi felosztását (lásd 2.5. fejezet) elkészítve megkapjuk az egyes klaszterek között húzódó határoló egyeneseket. Az eljárás eredménye a 4.3d. ábrán látható.



(a) (b) (c) (d)

4.3. ábra. A HI sík szegmentálásának lépései a hajszín esetében: (a) a klaszterek alappontjai, (b) a konvex burkok, (c) a távolság-transzformált, (d) a távolság-transzformált Voronoi felosztása.

4.3 Szín meghatározása

Az egyes jellemzők színét a szegmentált terület alatti domináns szín (a legnagyobb gyakorisággal előforduló szín) határozza meg. Ezt a színt a legegyszerűbben úgy kaphatjuk meg, ha megkeressük a kép hisztogramjának a maximumát. Mivel az eredeti kép a háromdimenziós RGB színtérben van definiálva, így az előbb ismertetett maximumot egy háromdimenziós hisztogramban (16 millió elem közül) kellene megkeresni, ami elég nehéz és számításigényes feladat.

Ahelyett hogy a teljes színtérben keresnénk meg a domináns színt, majd a CLUT-ban megkeresnénk a hozzátartozó címkét, sokkal jobban járunk, ha a következő eljárást választjuk: a szegmentált jellemzők képpontjainak színét helyettesítsük a CLUT-ban hozzájuk tartozó színkategóriák középszíneivel. Az előbbi eljárás eredményeként egy olyan színhisztogramot fogunk kapni, amely csupán öt elemből fog állni (az öt színkategória középszínéből). Végezetül tehát az adott jellemző színét a legnagyobb számban előforduló színkategória határozza meg (lásd 4.4. ábra).

4.4. ábra. A színek számának csökkentése az arcbőr esetén.

Tehát a színinformáció alapján történő keresés teljesen analóg módon történik a szegmentálással. Első lépésben a kép sorfolytonos bejárásával az egyes RGB pontok, HSI térbe történő áttranszformálása történik. Az így kapott HSI színtérbeli pontoknak először kromatikus–akromatikus besorolását végezzük el, az SI síkra vetítésükkel. Ezután az alábbi két eset állhat fent:

- A vizsgált RGB pont akromatikus: ekkor az RGB pontnak megfelelő szín, az akromatikus értékeket tartalmazó CLUT–ból kiolvasott szín lesz.

- A vizsgált RGB pont kromatikus: ez esetben szükség van a kromatikus osztályok közötti besorolásra (mely az RGB pontból nyert térbeli HSI pont) HI síkra történő vetítésével kapható meg. Az RGB pont tényleges színe, a kromatikus értékeket tartalmazó CLUT–ból kiolvasott szín lesz.

5 Összefoglalás és kitekintés

5.1 Eredmények

A fentebb ismertetett színreprezentáció segítségével egy megbízható színkinyerő módszert lehet megvalósítani. A kinyert színek segítségével az arcok osztályozhatóvá válnak. A vizuális információn alapuló keresőrendszerünk validálására egy újabb pszichológiai kísérletet végeztünk (melynek menete ugyanaz volt, mint amit a 4.1. fejezetben ismertettem). A kísérlet során egy 20 képből álló adatbázist használtunk, melyekről a kísérletben résztvevőknek el kellett dönteniük, hogy az egyes arci jellemzőket melyik előre definiált színkategória (lásd 4.1. táblázat) írja le a legjobban. Ugyanezekre a képekre lefuttattuk a keresőrendszerünket is, mellyel minden egyes jellemzőre meghatároztunk egy százalékos értéket. Ez az érték azt adja meg, hogy a szegmentált terület hány százalékát alkotják a domináns színkategória pixelei.

Következő lépésben egy összehasonlítást végeztünk a kísérletben résztvevők véleményére és a program kimenetére vonatkozólag. A két eredmény közötti kapcsolatot a függelék 9.3. fejezetében összesítettem. Látható hogy a keresőrendszerünk által jósolt színkategóriák több mint 75%-ban megegyeznek a megkérdezettek véleményével. Sőt sok esetben csak egy színárnyalatnyi különbség mutatkozik a két eredmény összehasonlítása során (pl. a program sötétbarna helyett, barnának detektálja a bőrszínt). Ha az egy színárnyalatnyi eltéréseket is pozitív találatként fogadjuk el az értékek összehasonlítása során, akkor a keresőrendszerünk 95% feletti helyes színdetektálási elérésére képes. A két halmaz közötti korreláció átlagosan 0.73, továbbá jellemzőkre lebontva a következő értékek adódnak: 0.79 az arcbőrre, 0.61 a szemre és 0.78 a hajra. A viszonylag rossz eredmény a szem esetében a jellemző alacsony felbontásának köszönhető. A szemet tartalmazó kép felbontásának növelésével nő a korreláció, de egyben nő a keresőrendszer időigénye is (mert a szem méretének növekedésével egyenes arányban nő az arc mérete is).

A fentebb ismertetett vizuális információn alapuló keresőrendszerünket használhatjuk egy tartalom-alapú képkereső rendszer automatikus indexelésére is. Ennek érdekében a színkinyerő alkalmazásunkat integráltuk egy web-alapú képkereső rendszerbe. Az elkészült webalkalmazás két weboldalból és egy képadatbázisból áll. A felhasználóknak lehetősége van ebbe a képadatbázisba új képeket feltölteni. A feltöltött képeket paraméterként adjuk át a színkinyerő alkalmazásunknak, mellyel meghatározzuk az egyes arci jellemzők színét és a

kép nevével együtt eltároljuk azokat egy meta leírófájlban. A webalkalmazás főoldalán az adatbázisbeli képeket listázhatjuk ki az előbbi meta leírófájl alapján. Tehát az egyes arci jellemzőkhöz színkategóriákat választhatunk ki, mely alapján lekérdezéseket hajthatunk végre az arcbőr-, a szem-, és a haj színére vonatkozólag. Természetesen a rendszer elérhető az interneten[1] keresztül is.

[1] http://www.inf.unideb.hu/ipgd/FaceColorSegmentation

5.2 Összefoglalás

A 2.3 fejezetben megmutattam az objektumdetektálás egy megközelítését, amely alacsony számítási idővel magas detektálási arányt képes elérni. Mindezt három fontos tulajdonságán keresztül láthattuk.

Az első fontos tulajdonság a jellemzők számításához használt integrál kép reprezentáció, mely jelentősen csökkenti a kezdeti képfeldolgozást. A második fontos tulajdonság az AdaBoost-on alapuló jellemző kiválasztás, ami egy agresszív és hatékony technika a jellemző kiválasztásra. A harmadik fontos tulajdonság az osztályozók kaszkádja, mely radikálisan csökkenti a számítási időt, míg növeli a detektálási pontosságot.

A 3. fejezetben az egyes arci jellemzők leírására alkalmas szegmentációs módszereket ismertettem, melyek segítségével alacsony számítási idő alatt érhetünk el jó eredményeket. A módszer azon a tényen alapszik, hogy a Viola-Jones detektorok segítségével viszonylag alacsony számítási idővel tudjuk elérni a vizsgált jellemzők magas detektálási arányát.

A 4. fejezetben ismertettem, hogy az adott jellemző színét a neki megfelelő szegmentált terület alatti domináns szín írja le. A jobb használhatóság érdekében drasztikusan csökkentettük a keresőrendszerben felhasználható színek számát. A teljes színtér és az arci jellemzőknek megfelelő színkategóriák közötti kapcsolat leírására egy szín modellt definiáltunk, melyet humán megfigyelések alapján készítettünk el.

A színeknek e reprezentációja könnyebbé és gyorsabbá tette az arci jellemzők színének meghatározását. A fentebb ismertetett színkinyerő alkalmazás működési elvét demonstrációs célból beépítettük egy tartalomalapú keresőrendszerbe, amely egy képadatbázist felhasználva képes a bőr-, a szem-, és a hajszín online lekérdezésére.

A színkinyerő eljárásunkat a keresőrendszeren túl megpróbáltuk rasszok osztályozásra is felhasználni. Azonban azt tapasztaltuk, hogy a színinformáció egyedüliként nem robosztus a rasszok felismerésére, de eredményét egyéb más módszerekkel vegyítve (pl. a fej formájának detektálása, az arci jellemzők közötti távolságok mérése) a probléma szempontjából jó végeredményt kaphatunk.

6 Köszönetnyilvánítás

Szeretnék köszönetet mondani Dr. Fazekas Attilának a témakörben nyújtott gondos útmutatásért és tanácsaiért, és hogy lehetőséget kaptam az IPGD csoport tagjaként megismerni a csapatmunka előnyeit rendkívüli emberek között. Köszönöm Sajó Leventének a türelmét, gyakorlati ötleteit és tanácsait, melyek segítettek mind az elméleti témakör gyorsabb megértésében, mind a hatékonyabb szoftverfejlesztésben.

7 Ábrák és táblázatok jegyzéke

7.1 Ábrák jegyzéke

7.2 Táblázatok jegyzéke

8 Irodalomjegyzék

[1] I. Szakadát, "Keresőrendszerek a weben," *PKI Tudományos Napok 2003*, pp. 1-3, 2003.

[2] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR , 1998.

[3] M. Tkalcic and J. F. Tasic, "Colour spaces - perceptual, historical and applicational background," *The IEEE Region 8 EUROCON 2003 proceedins*, pp. 304-308, 2003.

[4] A. Ford and A. Roberts. (2010, Apr.) Colour Space Conversions. [Online]. http://www.poynton.com/PDFs/coloureq.pdf

[5] A. Albiol, L. Torres, and E. J. Delp, "Optimum color spaces for skin detection," *Proceedings of the International Conference on Image Processing (ICIP)*, p. 122–124, 2001.

[6] T. Cheng, H. W. Park, and Y. Kim. (2010, Apr.) Image filtering in HSI color space. [Online]. http://www.freepatentsonline.com/6631206.html

[7] Intel® IPP. (2010) Open Source Computer Vision Library. [Online]. http://sourceforge.net/projects/opencvlibrary/

[8] R. Lienhart and J. Maydt, "An Extended Set of Haar-like Features for Rapid Object Detection," *IEEE ICIP*, pp. 900-903, 2002.

[9] K.-K. Sung and T. Poggio, "Example-based learning for view-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 39-51, 1998.

[10] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 22-38, 1998.

[11] T. Kanade and H. Schneiderman, "A statistical method for 3d object detection applied to faces and cars," International Conference on Computer Vision, 2000.

[12] D. Roth, M. Yang, and N. Ahuja, "A snowbased face detector," *Neural Information Processing*, vol. 12, 2000.

[13] Y. Amit, D. Geman, and K. Wilder, "Joint induction of shape features and tree classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1300-1305, 1997.

[14] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," *Sixth International Conference on Computer Vision (ICCV'98)*, p. 555, 1998.

[15] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," *Computational Learning Theory: Eurocolt '95*, vol. 37, pp. 23-37, 1995.

[16] K. Tieu and P. Viola, "Boosting Image Retrieval," *International Journal of Computer Vision*, vol. 56, pp. 17-36, 2004.

[17] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods," *The Annals of Statistics*, vol. 26, no. 5, pp. 1651-1686, 1998.

[18] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[19] J. K. Tsotsos, et al., "Modeling visual attention via selective tuning," vol. 78, no. 1-2, pp. 507-545, 1995.

[20] L. Itti, C. Koch, and E. Niebur, "A Model of Saliency-based Visual Attention for Rapid Scene Analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 11, pp. 1254-1259, 1998.

[21] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, pp. 81-106, 1986.

[22] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno, "2D Euclidean distance transform algorithms: A comparative survey," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, pp. 1-14, 2008.

[23] F. Porikli and T. Kocak, "Fast Distance Transform Computation using Dual Scan Line Propagation," in *Real-time image processing*, 2007.

[24] F. Sárközy. (2010, Apr.) Delaunay háromszögelés - Voronoi sokszögek. [Online]. http://gisfigyelo.geocentrum.hu/sarkozy_terinfo/tbev.htm

[25] Z. Savas, "Real-Time Detection and Tracking Of Human Eyes in Video Sequences," *thesis*, pp. 59-61, 2005.

[26] P. Norvig and S. J. Russell, *Mesterséges Intelligencia - Modern megközelítésben*, 2nd ed. Panem Kiadó Kft., 2005.

[27] E. L. van den Broek, T. E. Schouten, and P. M. F. Kisters, "Modeling human color categorization," *Pattern Recognition Letters 29.*, vol. 29, no. 8, p. 1136–1144, 2008.

[28] P. Frost, "European hair and eye color: A case of frequency-dependent sexual selection?," *Evolution and Human Behavior*, vol. 27, pp. 85-103, 2006.

[29] L. Bickford. (2010, Apr.) The EyeCare Reports. [Online]. http://www.eyecarecontacts.com/eyecolor.html

[30] Caltech Face Datebase. (2010, Apr.) [Online]. http://www.vision.caltech.edu/html-files/archive.html

[31] CBCL Face Database. (2010, Apr.) MIT Center For Biological and Computation Learning. [Online]. http://www.ai.mit.edu/projects/cbcl

[32] E. L. van den Broek, "Human-Centered Content-Based Image Retrieval," *thesis*, pp. 53-64, 2005.

[33] A. Hajdu and G. Fazekas, "Képfeldolgozási módszerek," *egyetemi jegyzet*, pp. 12-19, 2004.

9 Függelék

9.1 Az AdaBoost algoritmus

A 2.3.5. fejezetben szó volt róla, hogy az egyes gyenge tanulókat az osztályozó függvényeknek arra a halmazára szorítottuk meg, melyek csak egy egyszerű jellemzőtől függnek. A gyenge osztályozókból a tanuló algoritmus, az AdaBoost algoritmus egy változatával állítja elő az erős osztályozót, melyek a lépcsők szerepét töltik majd be.

Az alábbiakban, az AdaBoost tanuló algoritmus leírása következik, mely egy kérdés online megtanulásának menet ét írja le. Minden T hipotézis egy egyszerű jellemzőt használ. A végső hipotézis a T hipotézisek egy súlyozott lineáris kombinációja, ahol a súlyok fordítottan arányosak a tanulási hibával.

- Adottak az $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$ tanító példák, ahol x_i egy kép és $y_i = 0{,}1$ attól függően, hogy a példa pozitív, vagy negatív.
- Inicializáljuk a súlyokat a következőképpen (m és l a negatív és pozitív példák száma):

$$w_{1,i} = \begin{cases} \frac{1}{2m}, & \text{ha } y_i = 0, \\ \frac{1}{2l}, & \text{ha } y_i = 1. \end{cases}$$

- $for\ t = 1, \dots, T$
  1. Normalizáljuk a súlyokat, hogy w_t valószínűségi eloszlás legyen:
  $$w_{t,i} = \frac{w_{t,i}}{\sum_{j=0}^{n} w_{t,j}}.$$
  2. Minden j jellemzőhöz tanítsunk be egy h_j osztályozót, ami csak egy jellemzőtől függ. A w_t-re vonatkozó ε_t súlyozott hiba:
  $$\varepsilon_t = \sum_i w_i |h_j(x_i) - y_i|.$$
  3. Válasszuk ki a legkisebb ε_t hibájú h_t osztályozót.
  4. Módosítsuk a súlyokat a következőképpen:
  $$w_{t+1,i} = w_{t,i} \beta_t^{e_i},$$
  ahol $e_i = 0$, ha az x_i osztályozása helyes, különben $e_i = 1$, és $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$.

- A végső erős osztályozó:

$$h(x) = \begin{cases} 1, ha\ \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0, különben \end{cases} .$$

9.1. táblázat. Az AdaBoost algoritmus.

9.2 A vízesés struktúra tanítása

A vízesés struktúra tervezésénél a motivációt a minél jobb detektálási arány és a minél jobb detektálási arány és a minél alacsonyabb számításigény elérése jelenti. A vízesés szerkezetű osztályozó hamis pozitív aránya:

$$F = \prod_{i=1}^{K} f_i,$$

ahol K az osztályozók száma f_i az i-edik osztályozó hamis pozitív aránya azokon a példákon, amelyek eljutnak hozzá. A detektálási arány:

$$D = \prod_{i=1}^{K} d_i,$$

ahol d_i az i-edik osztályozó találati aránya a hozzá eljutott példákon.

A kép pásztázása során kiértékelt jellemzők száma szükségszerűen egy valószínűségi folyamat függvénye. Bármely alablak addig halad a kaszkádon (egyszerre csak egy osztályozón át), míg el nem döntjük, hogy az ablak negatív, vagy ritka körülmények között az alablak minden teszten végigmegy, így pozitív eredményt ad. Minden osztályozó kulcsértéke a "pozitív aránya", azaz azoknak az ablakoknak az aránya, melyekre az osztályozó pozitív választ ad. A kiértékelt jellemzők számának várható értéke:

$$N = n_0 + \sum_{i=1}^{K}\left(n_i \prod_{j<i} p_j\right),$$

ahol N a kiértékelt jellemzők várható száma, K az osztályozók száma, p_i az $i.$ osztályozó pozitív aránya, és n_i a jellemzők száma az $i.$ osztályban. Érdekes módon, attól fogva, hogy az objektumok nagyon ritkák, a „pozitív arány” egyenlő a hamis pozitív aránnyal.

A kaszkád elemeit tanító folyamat némi odafigyelést igényel. Az AdaBoost a hibák minimalizálására törekszik, és nem kifejezetten magas detektálási arányok elérésére tervezték magas hamis pozitív arányok költsége mellett. Magasabb küszöbbel kevesebb hamis pozitívat detektáló osztályozót eredményez, alacsonyabb detektálási aránnyal. Alacsony küszöbbel pedig több hamis pozitívat és magasabb detektálási arányt elérő osztályozót kapunk.

A teljes tanítási folyamat kétféle, egymásnak ellentmondó kritérium optimalizálását igényli. Többnyire egy osztályozó több jellemzővel magasabb detektálási arányt és alacsonyabb hamis pozitív arányt fog elérni. Ugyanakkor egy több jellemzővel rendelkező

osztályozó több számítási időt igényel. Alapjában véve definiálhatunk egy optimalizációs keretet, amiben

- az osztályozó szintek száma,
- a jellemzők száma, n_i minden szinthez, és
- minden szint küszöbe,

úgy kivitelezett, hogy minimalizáljuk N jellemzők számát, adott F és D célértékek mellett. Sajnos ennek a minimumnak a megtalálása egy igen nehéz feladat. Gyakorlatilag egy nagyon egyszerű alkalmazást használnak hatékony osztályozók előállításához. A felhasználó kiválasztja a minimum elfogadható f_i és d_i arányokat.

A kaszkád minden szintjét AdaBoosttal tanítják, úgy hogy a jellemzők számát addig növelik, míg a célul kitűzött detektálási és hamis pozitív arányokat el nem érik. Az arányokat a detektor tesztelésével kapják meg egy érvényes halmazon. Ha még nem érték el a célul kitűzött hamis pozitív arányt, további szinteket adnak a detektorhoz. A negatív halmazt a következő szintek tanításához az aktuális detektor futtatása során kapott hamis pozitív eredményekből kapják.

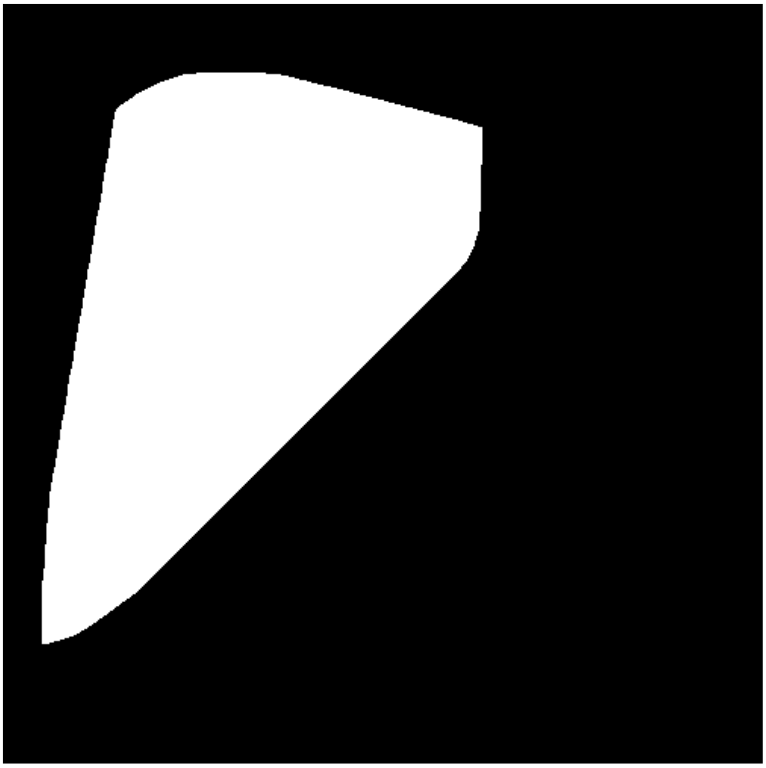
9.3 A keresőrendszer teszteredményei

Az elkészült keresőrendszerünk tesztelésére 20 kép felhasználásával egy újabb pszichológiai kísérletet végeztünk. A teszt során tulajdonképpen egy összehasonlítást végeztünk a kísérletben résztvevők véleményére és a program kimenetére vonatkozólag. A két eredmény közötti kapcsolat a 9.2. táblázatban látható. A táblázat minden egyes cellájában a program által kimenetként szolgáltatott az egyes színkategóriákra vonatkozó százalékos értékek láthatók.

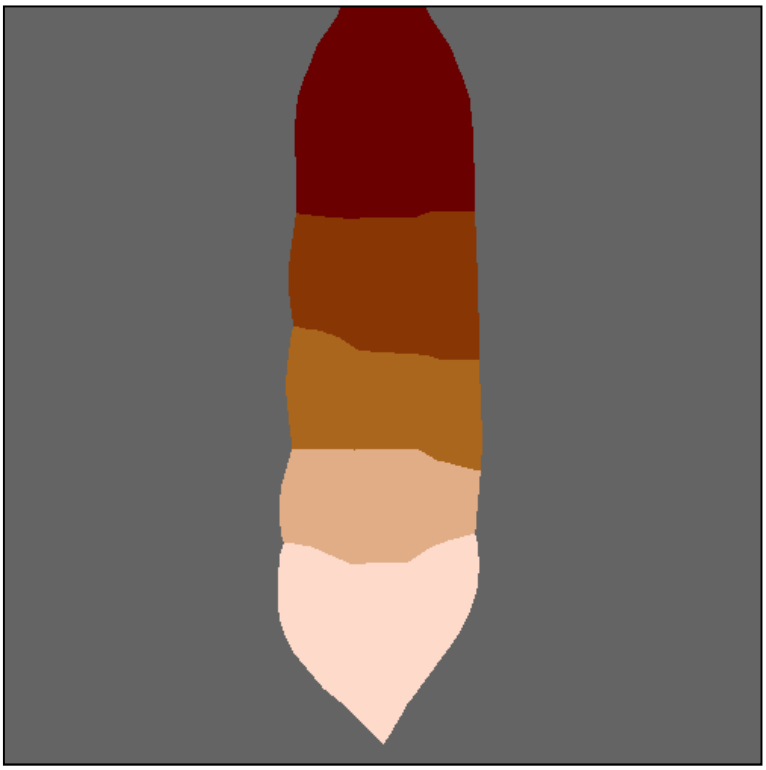
	Képek																			
nagyon világos	.04	.08	**.43**	.34	.23	.11	**.35**	0	.20	0	.04	**.23**	0	0	0	.14	.03	0	0	0
világos	.47	**.56**	.28	**.52**	**.26**	.34	.19	.33	**.61**	.05	**.54**	.22	0	0	0	**.36**	**.37**	0	.37	0
átmeneti	.45	.34	0	.02	.08	.27	0	**.50**	.12	.25	.20	0	.01	0	.02	.24	.32	0	.60	.01
sötét vagy barna	.01	0	0	.02	.02	.10	0	.15	0	**.53**	0	0	**.27**	.24	.29	.02	.23	**.51**	.01	.30
nagyon sötét	0	0	0	.02	0	0	0	0	0	0	0	0	.25	**.67**	.41	0	0	.46	0	**.52**
kék	0	0	**.30**	.02	.01	.01	**.53**	.01	.01	0	0	0	0	0	0	0	.02	0	.05	0
szürke	0	0	.10	.01	.04	.01	.06	0	.01	0	0	0	0	.01	0	.01	.20	0	.14	.08
zöld	.21	.06	0	.10	.04	.06	0	.24	.51	.09	.22	**.66**	.02	.23	.07	**.58**	.18	.06	.50	.09
barna	.21	.77	.08	**.59**	**.53**	**.38**	0	.34	.39	.40	**.30**	.11	.12	.18	.22	.06	.46	.28	.29	.13
sötétbarna	**.44**	.09	0	.09	.19	.18	0	.10	.02	.33	.08	0	.18	**.40**	**.37**	0	.12	**.58**	0	**.53**
szőke	0	0	.10	0	0	0	.51	.23	.08	0	.05	.76	0	**.40**	0	.07	0	0	.01	0
aranybarna	0	.03	**.76**	.19	.05	.02	.02	0	**.58**	0	0	0	0	.03	.01	0	0	0	**.52**	0
barna	0	**.48**	.10	**.53**	.31	.17	0	0	.16	.02	0	0	.16	0	.04	0	0	.01	.18	0
fekete vagy sötétbarna	**.39**	.46	0	.22	.17	**.58**	0	0	0	**.58**	0	0	**.33**	0	**.87**	0	0	**.79**	0	0
szürke vagy fehér	0	0	0	0	0	0	**.43**	**.65**	0	0	**.70**	.23	0	0	0	.31	**100**	0	0	0

9.2. táblázat. A kísérletben résztvevők véleményének és a program kimenetének összehasonlítása. Zöld színnel a kísérletben résztvevők által kiválasztott kategóriát, narancssárga színnel a program által visszaadott színkategóriát jelöltük, kékszínű félkövér kiemeléssel jelöltük, ha az előbbi két kategória megegyezik.

9.4 Az arcbőrhöz tartozó SI és HI síkok

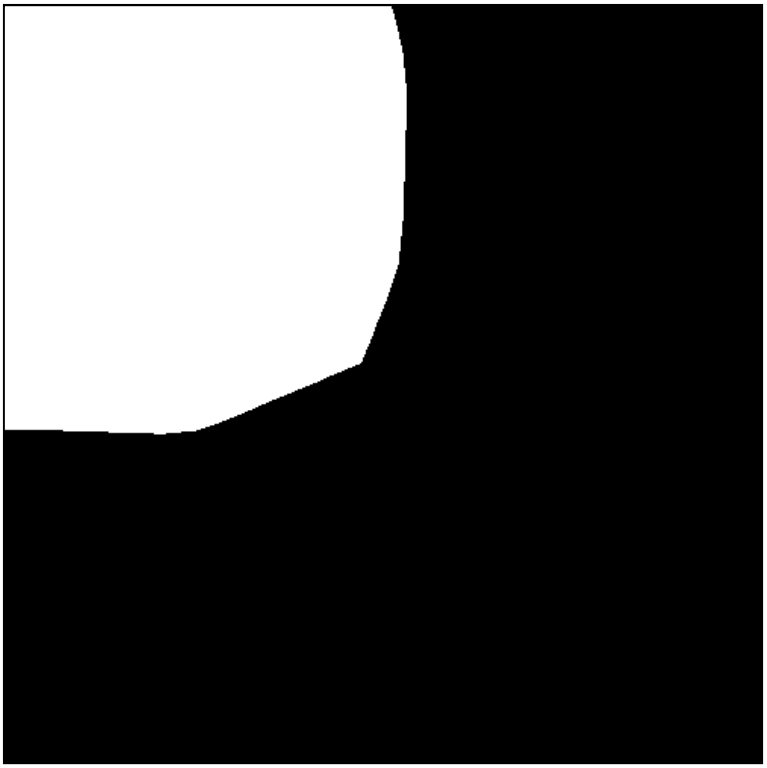


9.1. ábra. Az arcbőrhöz tartozó SI sík: fehér színnel jelöltük a színtér kromatikus tartományába eső pixeleit, valamint fekete színnel jelöltük az akromatikus pixeleket.

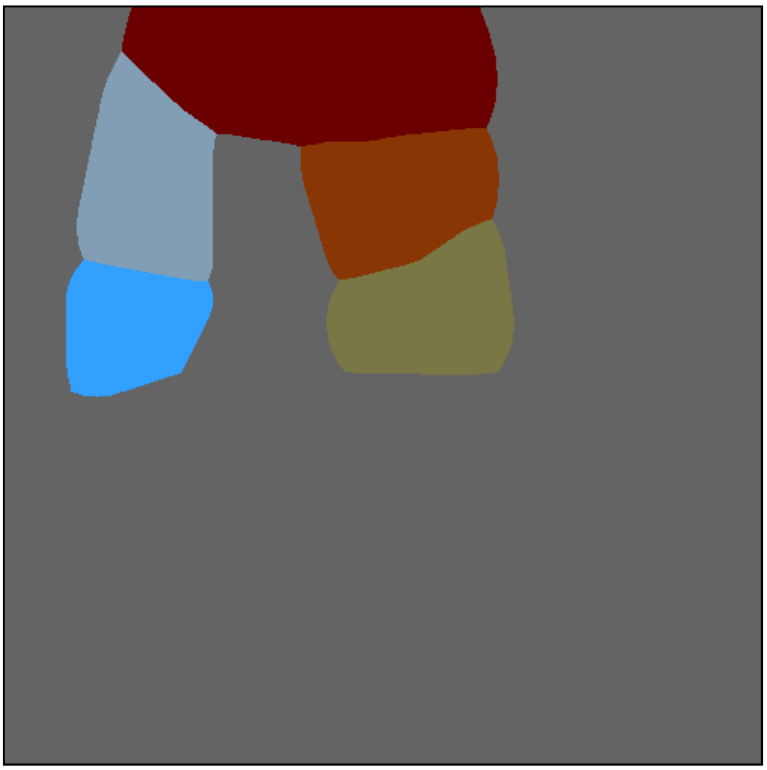


9.2. ábra. Az arcbőrhöz tartozó HI sík: A képen az egyes színkategóriák középszínei láthatók.

9.5 A szemhez tartozó SI és HI síkok

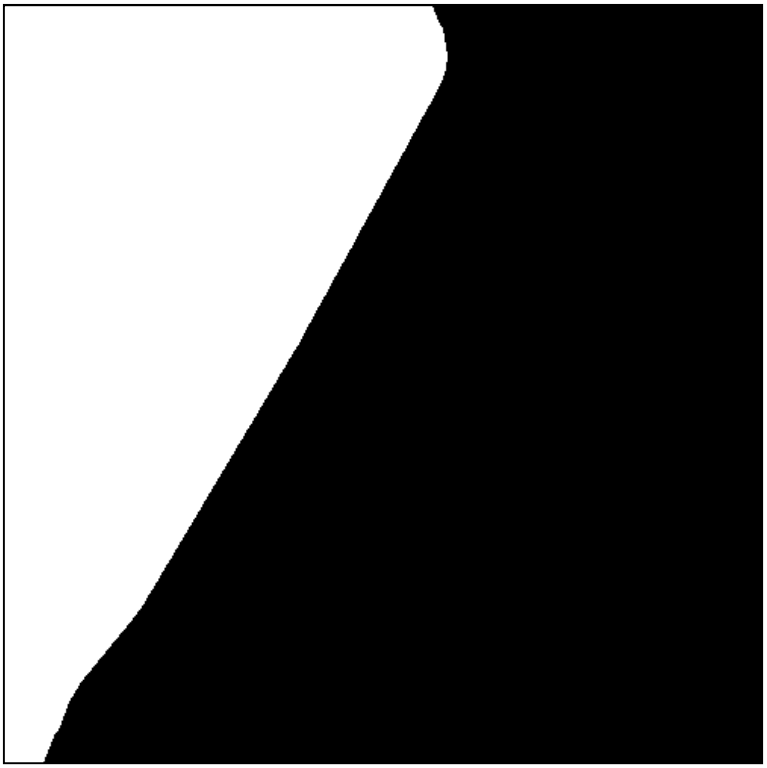


9.3. ábra. A szemhez tartozó SI sík: fehér színnel jelöltük a színtér kromatikus tartományába eső pixeleit, valamint fekete színnel jelöltük az akromatikus pixeleket.

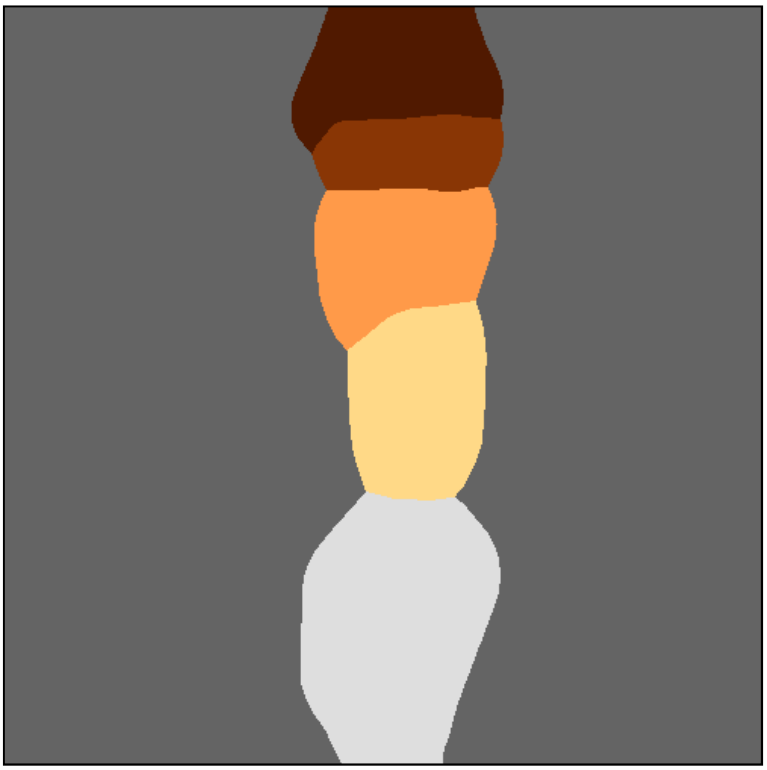


9.4. ábra. A szemhez tartozó HI sík: A képen az egyes színkategóriák középszínei láthatók.

9.6 A hajhoz tartozó SI és HI síkok



9.5. ábra. A hajhoz tartozó SI sík: fehér színnel jelöltük a színtér kromatikus tartományába eső pixeleit, valamint fekete színnel jelöltük az akromatikus pixeleket.



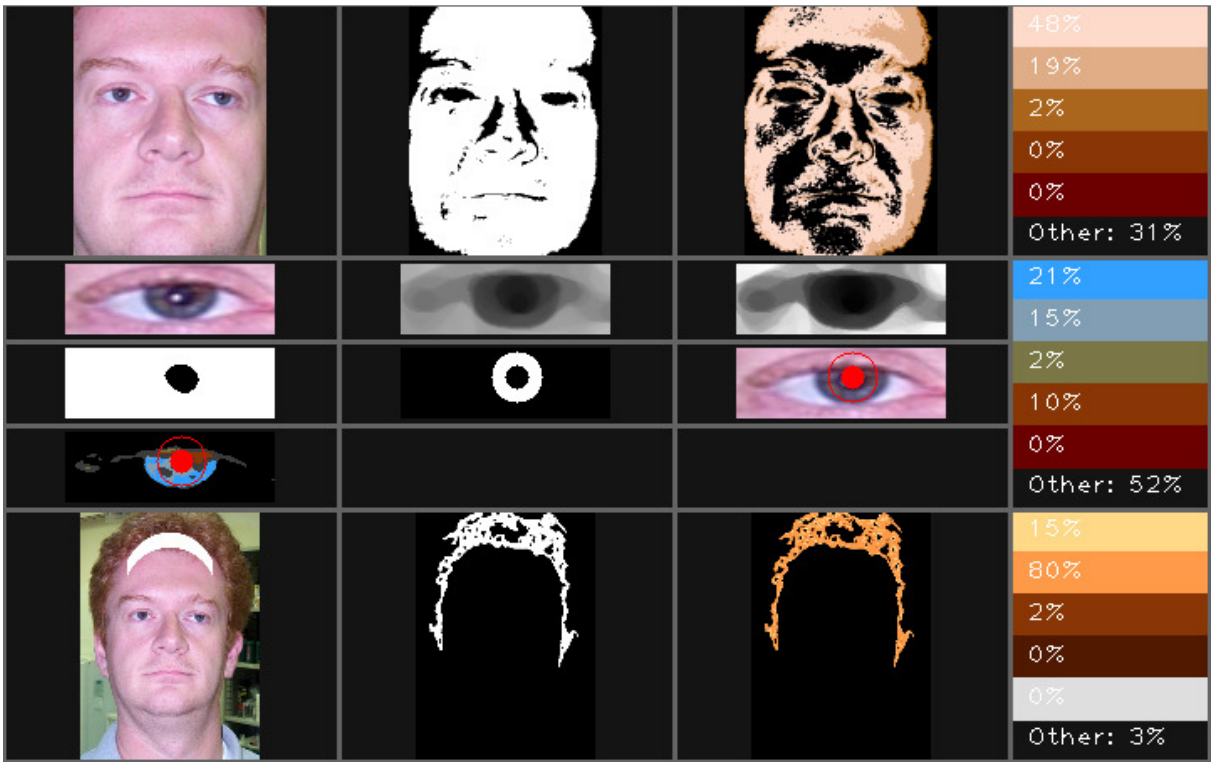
9.6. ábra. A hajhoz tartozó HI sík: A képen az egyes színkategóriák középszínei láthatók.

9.7 A program kimenete

A lentebb látható ábrán a program által szolgáltatott kimenet látható. Az ábra első három oszlopában az egyes jellemzők színének meghatározása során alkalmazott lépések grafikus szemléltetései láthatók, az utolsó oszlopában pedig az egyes színkategóriákra jósolt százalékos értékek helyezkednek el. Az utolsó oszlopában a sorok színei, a 4.1. táblázatban szereplő színkategóriák elemeinek középszínei.

Az „other” címszóval jelölt színkategória a szegmentált régió alatti akromatikus pixelek százalékos arányát jelenti. A keresőrendszer a legnagyobb százalékban előforduló kromatikus színt választja az egyes jellemzők színének. Tehát arci jellemzőkként a 4.1. táblázatból választja ki az adott jellemző színének azt a kategóriát, melynek színei a legnagyobb arányban fordulnak elő a szegmentált régió alatti területen.

Jelen esetben az arcbőr színe, a nagyon világos kategóriába, a szemszín a kék kategóriába, a hajszín pedig az aranybarna kategóriába esik.



9.7. ábra. A keresőrendszer kimenete.

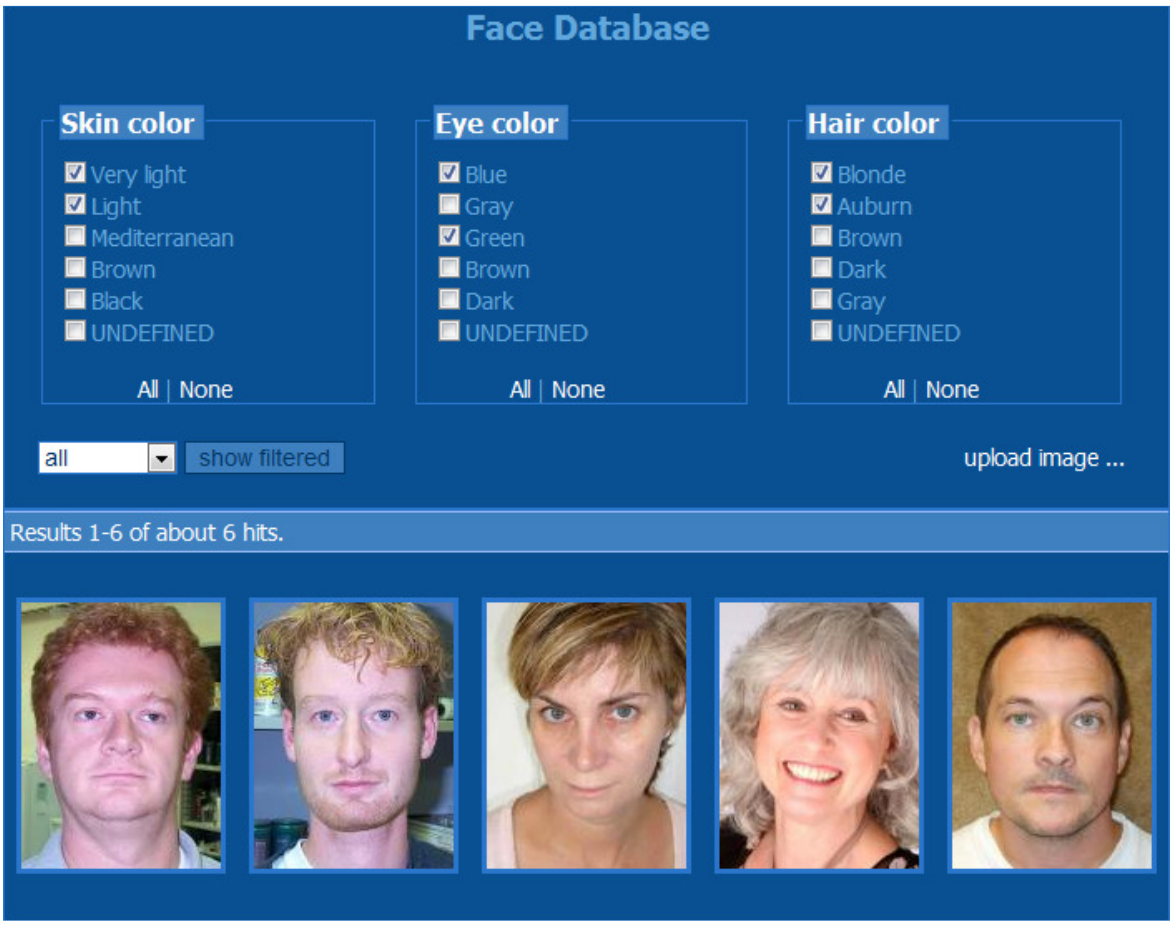
9.8 A keresőrendszerhez tartozó webalkalmazás

A lentebb látható ábrán a keresőrendszerünkhöz készített webalkalmazásunk felhasználói felülete látható. A webalkalmazás segítségével jellemzőkként kérdezhetjük le egy metafájlból a felöltött-, vagy a demó képek adatbázisát.

Mint ahogy azt már az 5.1. fejezetben ismertettem, a felhasználóknak lehetősége van a képadatbázisba új képeket feltölteni. A feltöltött képeket paraméterként adjuk át a színkinyerő alkalmazásunknak, mellyel meghatározzuk az egyes arci jellemzők színét, és azokat a kép nevével együtt eltároljuk egy meta leírófájlban.

Az alábbi ábrán az egyes arci jellemzőknél látható UNDEFINED kategóriába azok a képek tartoznak, melyeknél a program nem tudta meghatározni a színét az adott jellemzőnek (pl. az arcdetektor nem talált egyetlen pozitív találatot sem a bemeneti képen, így a program nem tudta meghatározni a bőrt tartalmazó régiót).

Jelen esetben a nagyon világos-, vagy világos bőrű, kék-, vagy zöld szemű és szőke-, vagy aranybarna hajú embereket listázzuk ki a teljes adatbázisból.



9.8. ábra. A webalkalmazás felhasználói felülete.