



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

VIDEÓK FELCIMKÉZÉSE KÉPI TARTALOM ALAPJÁN

Szerzők: **Szántó Balázs**
mérnök informatikus MSc. szak,
I. évf.

Konzulensek: **Dr. Vámosy Zoltán**
egyetemi docens

Dr. Sergyán Szabolcs
egyetemi docens

Budapest, 2012.

1 TARTALOMJEGYZÉK

<u>1 TARTALOMJEGYZÉK</u>	<u>2</u>
<u>2 ABSZTRAKT</u>	<u>5</u>
<u>3 BEVEZETÉS: KÉP- ÉS VIDEÓ KERESÉS</u>	<u>6</u>
3.1 VIDEÓ KERESÉS A SZAKIRODALOMBAN	6
3.1.1 KERESSÜNK VIDEÓKAT	6
3.1.2 AZ IPARI ÉS TUDOMÁNYOS KÖZÖSSÉG	7
3.2 KITŰZÖTT CÉLOK ÉS FELHASZNÁLÁSI LEHETŐSÉGEK	7
3.2.1 KITŰZÖTT CÉLOK, AZ ALAP MODELL	7
3.2.2 A VIDEÓ KERESÉS LEHETSÉGES GYAKORLATI ALKALMAZÁSAI	8
3.2.3 KIHÍVÁSOK, KÉRDÉSEK	10
<u>4 IRODALOMKUTATÁS</u>	<u>12</u>
4.1 A MULTIMÉDIÁS TARTALMAK LEÍRÁSA, INDEXELÉSE, CSOPORTOSÍTÁSA	12
4.1.1 VIDEÓ STRUKTÚRA ANALÍZIS	12
4.1.2 KULCS KÉPKOCKA ELŐÁLLÍTÁS	12
4.1.3 JELLEMZŐ VEKTOROK	13
4.1.4 VIDEÓ ANNOTÁCIÓ ÉS KERESÉS	14
4.1.5 FELHASZNÁLÓI VISSZACSATOLÁS	14
4.2 RGB-D SENZOROK	14
4.2.1 3D MODELL VAGY 2D JELLEMZŐK?	14
4.3 A SZEGMENTÁLÁS ALAPÚ TECHNIKÁK	15
4.4 SZEMANTIKUS SZEGMENTÁLÁS	16
4.4.1 SZEMANTIKUS SZEGMENTÁLÓ ALGORITMUS ÁLTALÁNOS MODELLE	17
4.4.2 LOKÁLIS JELLEMZŐK KISZÁMÍTÁSA	17

4.4.3	JELLEMZŐ VEKTOROK TRANSZFORMÁLÁSA	18
4.4.4	OSZTÁLYOZÁS	20
4.4.5	PROBLÉMÁK	21
4.4.6	ESETTANULMÁNY – BELSŐ TEREK SZEMANTIKUS SZEGMENTÁLÁSA	22
4.5	ÁLTALÁNOS OBJEKTUMDETEKTÁLÁS	23
4.5.1	FELHASZNÁLT JELLEMZŐK	23
4.5.2	MODELLÉPÍTÉS	25
5	SAJÁT RENDSZEREM	27
5.1	A RENDSZER GLOBÁLIS FELÉPÍTÉSE	27
5.2	ELŐFELDOLGOZÁS	28
5.2.1	MÉLYSÉGI TÉRKÉP ELŐFELDOLGOZÁSA	28
5.2.2	RGB KÉP ELŐFELDOLGOZÁSA	28
5.3	VIZUÁLIS JELLEMZŐK	29
5.3.1	ÉL ÉS TEXTÚRA JELLEMZŐK	29
5.3.1	SZÍNJELLEMZŐK	31
5.3.2	FELÜLETJELLEMZŐK	31
5.4	OSZTÁLYOZÁS	32
5.4.1	GAUSS-KEVERÉKES MODELL (GAUSSIAN MIXTURE MODELL)	32
5.4.2	K- KÖZÉP (K-MEANS)	33
5.4.3	TÁMASZTÓ VEKTOR GÉP (SUPPORT VECTOR MACHINE)	33
5.4.4	DIMENZIÓK SZÁMÁNAK JELENTŐSÉGE	35
6	TESZTEK ÉS EREDMÉNYEK	37
6.1	FELHASZNÁLT SZOFTVEREK	37
6.1.1	POINT CLOUD LIBRARY (PCL)	37

6.1.2	ACCORD.NET	37
6.1.3	MATLAB IMAGE PROCESSING TOOLBOX	37
6.1.4	LIBSVM	37
6.2	TESZT ADATBÁZISOK	38
6.3	AZ ELKÉSZÜLT RENDSZER	38
6.3.1	PONTFELHŐ SZEGMENTÁLÁSA NORMÁLVEKTOROK ALAPJÁN	38
6.3.2	SZEGMENTÁLÓ/CÍMKÉZŐ ALRENDSEZER	41
7	ÖSSZEGZÉS	44
8	IRODALOMJEGYZÉK	45

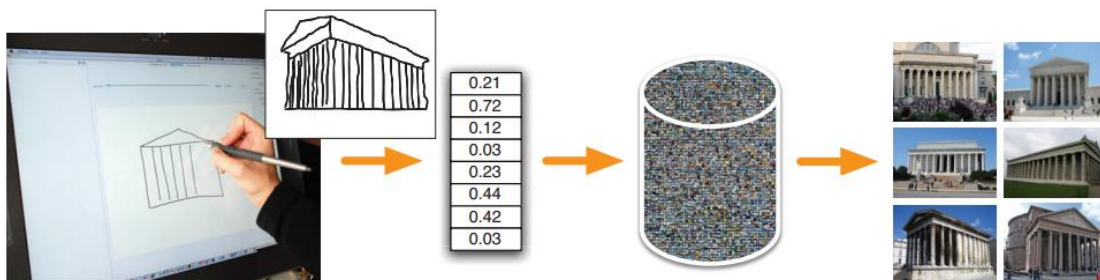
2 **ABSZTRAKT**

A kulcsszavas keresés mellett teret hódított a képi tartalomalapú keresés is. Ezzel együtt számos kutatás célozza már meg a videók kategorizálását és keresését, szimplán a vizuális tartalmat vizsgálva. A projekt célja a videókeresés lehetséges módszereinek és alkalmazási lehetőségeinek a bemutatása és értékelése. A metódusokat tekintve egy új szemléletet vizsgál a dolgozat, hogyan lehet szemantikus szegmentálás segítségével egy videót felcímkézni. Az implementált rendszerben elkészül egy olyan szegmentálási módszer, mely az adott képkocka minden pixelét kategorizálja a szemantikus tartalom alapján általános objektumdetektálás során felhasznált műveletekkel segítve. A videók Microsoft Kinect szenzorral készültek, így bemutatásra kerül, hogyan lehet szegmentálási metódusokat kiterjeszteni RGB-D adatokra. A megtervezett rendszer több nyelv segítségével, együttműködésével lett implementálva.

Kulcsszavak: videó keresés és indexelés, szemantikus szegmentálás, általános objektumdetektálás, RGB-D szegmentálás

3 BEVEZETÉS: KÉP- ÉS VIDEÓ KERESÉS

A képfeldolgozás egyik kurrens kutatási területének számít a képi tartalomalapú keresés („Content Based Image Retrieval” – továbbiakban CBIR). A hagyományos, kulcsszavak alapján történő keresés számos hátrányát próbálja kiküszöbölni ez az új módszer. Ilyen hátrálynak tekinthető a kulcsszavak objektív meghatározása, vagy az általános kulcsszavakra adott alacsony találati arány. Azonban, ha egy adatbázist a képi tartalom alapján címkézzük fel, vizuális kulcsszavakkal, az előbb említett problémák kiküszöbölhetők, ráadásul teljesen automatizált módon. Az ilyen jellegű tartalom sokféle lehet, választhatunk a kép színei és elrendezése alapján, vizsgálhatunk éleket, vagy akár jellemző pontokat. Nagyon sok lehetséges szempont létezik, mindig a feladatnak megfelelő képi jellemzőket kell felhasználni. A keresés, mint folyamat úgy történik, hogy egy példakép (keresendő kép) vizuális információi alapján keressük az adatbázis képeiről kinyert információk összevetésével. Egy másik projektem a CBIR egyik részterületén folytatott. Itt az elvárt bemenet nem egy színes kép, hanem egy, a felhasználó által rajzolt egyszerű, fekete-fehér skicc [1]. Pozsegovics Péterrel karöltve, témánkkal a XXX. Jubileumi OTDK, Informatikai szekciójában I. helyezést és Innovációs különdíjat nyertünk. Az 1. ábrán látható, hogyan is lehet egy ilyen rendszert használni.



1. ábra – Első lépésben a felhasználó gondol egy képre és ezt lerajzolja egyszerű skicc formájában. Majd a rajz alapján történik a keresés a képeket tartalmazó adatbázisban. A végeredmény a rajzhoz vizuálisan közel álló képek listája.
forrás: http://cybertron.cg.tu-berlin.de/eitz/pdf/2010_cag.pdf

A képek számának exponenciális növekedésével megegyező ütemben növekszik a videók száma is. Ezek a tartalmak rengeteg információt rejtnek önmagukban és kezelésére szintén modern osztályozó-kereső eszközöket kell biztosítani. Vizsgáljuk meg jobban ezt a területet.

3.1 VIDEÓ KERESÉS A SZAKIRODALOMBAN

A fejezet során bemutatom a vizuális tartalomalapú videóindexelés és -keresés kutatási területét. Összehasonlítom a videók és a képek keresésénél megismert metodikákat, továbbá prezentálom milyen fő előnyökkel és hátrányokkal szembesülünk. Röviden megvizsgálom, hogy milyen mozgalmak segítik elő a kutatási téma aktív fejlődését.

3.1.1 KERESSÜNK VIDEÓKAT

A vizuális tartalomalapú videóindexelés és -keresés („Visual Content-Based Video Indexing and Retrieval”) leírja, eltárolja és rendszerezi a videótartalmakat, hogy a felhasználók számára egyszerűen és gyorsan kereshető legyen különböző szemléletek alapján [2]. Az egyik lehetőség, amikor a videótartalom alapján kulcsszavakat készítünk. A másik lehetőség, hogy veszünk egy példa videót, és ahhoz hasonló után kutatunk. A két módszert persze érdemes vegyíteni, így az

egyes technikák gyengeségeit ki lehet küszöbölni a másik erősségeivel. A téma fontosságát már az is jelzi, hogy a Google az automatikus objektumfelismerő funkcióját videókon szabadalomként bejegyeztette.

3.1.2 AZ IPARI ÉS TUDOMÁNYOS KÖZÖSSÉG

A multimédia-tartalmakra sokáig nem létezett általános leíró formátum. Ezt az űrt az 1996-ban létrehozott MPEG-7 szabvány [3] hivatott betölteni („Moving Picture Experts Group”). Az MPEG-et kifejezetten multimédiás tartalmak leírására fejlesztették ki. Az ISO/IEC 15938 szabvány teljes neve: Multimédia Tartalom Leíró Interfész. Az MPEG-7 által definiált leírókat nem osztálykönyvtár formájában adják meg, hanem mint egy interfész leírást, amit nekünk kell megvalósítani. Lényegében egy ajánlást, szabványt kapunk. A videó tartalmak leírására irányuló formátumok és szabványok esetén még rendelkezésre állnak további kezdeményezések is, mint például SMIL [4] és Dublin Core Metadata Initiative [5].

Az [2] alapján 2005-2011 között már 14, a témát összefoglaló cikk jelent meg, ahol a területről átfogó beszámolók keretein belül bemutatták a legmodernebb eredményeket, amit eddig elértek. További lehetőséget nyújtanak az új algoritmusok és törekvések bemutatására a különböző versenyek, mint a TRECVID [6] vagy a VideoOlympics [7]. A TRECVID egy 2003 óta működő kezdeményezés. A célja egyértelműen a kutatás elősegítése a videókeresés területén. Ezt elősegítve jól meghatározott metrikákat és számos minta adatbázist bocsátanak rendelkezésre. A 2012-es versenyen már öt kategóriába lehetett nevezni: szemantikus indexelés, ismert videó keresése kulcsszavak és vizuális tartalom alapján, objektumok keresése videókon és három egyéb kategóriában események felismerése videón és azok elemzése. Továbbá minden évben az elkészített megoldásokhoz kapcsolódó cikkeket és prezentációkat elérhetővé teszik. Hasonló célokat követő VideoOlympics versenyt 2009-ben rendezték meg az ACM CIVR konferencián.

3.2 KITŰZÖTT CÉLOK ÉS FELHASZNÁLÁSI LEHETŐSÉGEK

A fejezetben bemutatjam a kitűzött céljaimat, amely a rendszerem alapvető modellje lesz. Kitérek a lehetséges felhasználási területekre. Végezetül pedig megvizsgálom a megoldáshoz vezető módszerekkel kapcsolatban felmerülő kérdéseket és kihívásokat.

3.2.1 KITŰZÖTT CÉLOK, AZ ALAP MODELL

A képi tartalomalapú keresés témájában megismert algoritmusokat és módszereket ki lehet terjeszteni videók tartalomalapú felcímkézésre, szegmentálására is. Ezen okból a feladatot felosztottam két részre. Az első részben a videót kell objektumokra bontani, szegmentálni. Egy videó számos többlet információval rendelkezik, mint egy kép. Ha pedig egy átlagos kamera (RGB képsáv) által rögzített videó mellett rendelkezésre áll mélységi térkép is, akkor még pontosabb algoritmusokat lehet implementálni. Sok alkalmazás használ már olyan szenzorrendszereket, mely mind RGB és mind távolság térképet („Depth” - D) is szolgáltat. Az RGB-D adatfolyam segítségével a környezet 3D térképét is fel lehet építeni.

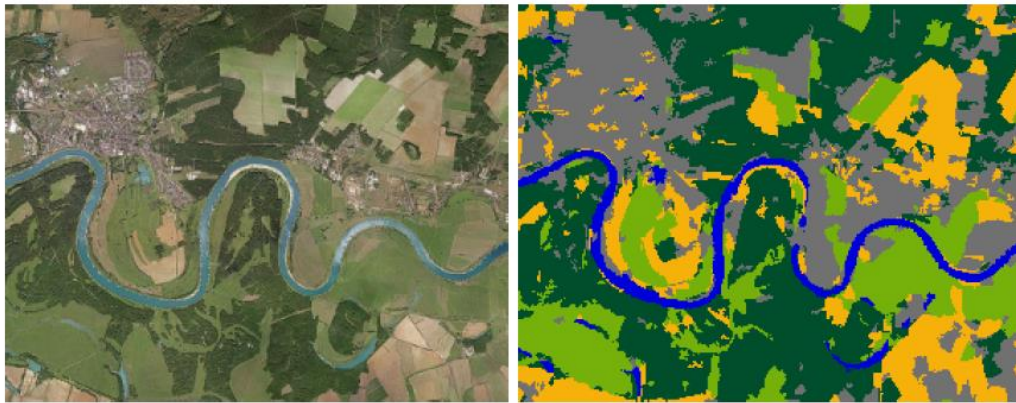
A dolgozatban a szegmentáló technikák megismerésére, implementálására és felhasználási lehetőségeire koncentrálok. A projekt jövőbeli célja, hogy az elkészített szegmentáló alrendszert egy másik, tartalomalapú keresőrendszerrel egészítsünk ki. Így a szegmentált részek szerint lehet háttéradatbázis alapján még több információt kinyerni az adott képszegmensről. Az információt

felhasználva nyilván fel lehetne egy adott videót címkézni, így később hatékonyan lehetne rákeresni.

3.2.2 A VIDEÓ KERESÉS LEHETSÉGES GYAKORLATI ALKALMAZÁSAI

Számos gyakorlati lehetőség kínálkozik, amit figyelembe vehetünk. Mint a „Kitűzött célok, az alap modell” című fejezetben, itt is vizsgálhatjuk a felhasználási lehetőségeket, mint a rendszer modulonkénti felhasználását.

Amennyiben csak magát a szegmentáló modul hasznosságát tekintjük, számos érvet találunk. Az itt elkészült algoritmusok számos környezetbe adaptálhatóak, mint például légi felvételek automatikus szegmentálása [8], amely egy nagyon fontos téma, mivel jól alkalmazható lehetne akár katasztrófavédelem során is. Például egy erdőtűz során egy autonóm repülőgép segítségével könnyen és automatikusan felmérhető lehetne a kár. Többek között az Óbudai Egyetem, Neumann János Informatikai Karon is folynak kutatások autonóm repülőgépek hardveres és szoftveres építésével kapcsolatban [9]. A 2. ábrán látható egy „szemantikusan” szegmentált légifelvétel.

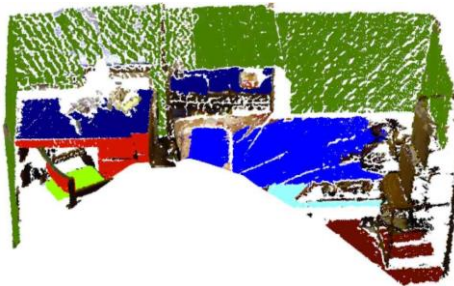


2. ábra – Bal oldalon látható egy légifelvétel, jobb oldalon pedig a szemantikusan szegmentálás eredménye [8].

További felhasználási lehetőség irodai vagy bármilyen más belső környezet (raktár például) manipulálása robotokkal [10, 11]. Ahhoz, hogy ez megvalósítható legyen, a robot/robotrepülő például Kinect kamera segítségével feltérképezi és felépíti az adott helyiség 3D-s modelljét, és ebben a modellben felcímkézi a különböző objektumokat és felületeket (lásd 3. és 4. ábra). Az ilyen rendszerek nagymértékben felhasználják a 3D-s pontfelhő által nyújtott lehetőségeket. Ellentétben a 2D-s információkkal, itt rendelkezésre állnak a geometriai tulajdonságok, mint alakjellemzők vagy az objektum pozíciója, elhelyezkedése (például: a monitor az asztalon helyezkedik el).



3. ábra – RGB-D szenzor adatai által rekonstruált környezet 3D modellje látható az ábrán [11]. A [11] szerzői ezt a felépített modellt címkézték fel.



4. ábra – A 3. ábrának az eredménye a címkézés után [11].

Mire jó még a videók felcímkézése? Például duplikátumok keresésére, mivel a videó-megosztó portálokon nagyon sok azonos tartalmat lehet találni [12] (a szakirodalomban „near duplicate video detection”), amelyek nagyban ronthatják a találati listát és a keresési élményt. A módszer segítségével ezek, illetve a jogszerűtlen tartalmak szűrése is megoldhatóvá válik.

További lehetőséget látok még a Google Project Glass-ban is [13]. A projekt célja egy olyan szemüveget elkészíteni, amely kiterjesztett valóságot nyújt a felhasználónak, úgy, hogy a szemüveg lencse kijelzőként funkcionál. Használata során megjeleníti, hogy levelünk érkezett-e, vagy megmutatja nekünk a környék térképét, hogy ne tévedjünk el. A szemüvegben kamera is található, amivel a saját szemszögünkből látottakat tudjuk megosztani másokkal. A kamerát külső vagy belső térben való navigációra is lehetne használni, így segítve látássérült embereket. A navigációhoz elengedhetetlen a környezet objektumainak a felismerése. A szemüveg várható prototípusa az 5. ábrán látható.



5. ábra – A Google Project Glass várható prototípusa [13].

forrás: <http://googlesystem.blogspot.hu/2012/04/googles-project-glass.html>

Ha kicsit elmozdulunk a reklámozás irányába, akkor elképzelhetünk egy forradalmian új modellt is. A felhasználó néz egy videót, és megtetszik neki egy autó vagy egy gitár. Amint rákattint, megjelennek a termékhez kapcsolódó reklámok és részletes információk is. Vegyük csak a YouTube példáját, milliók használják a szolgáltatást, így sok potenciális fogyasztóhoz tudnának ilyen módon célzott reklámokat eljuttatni, ami a Google számára jelentős profitként jelenhet meg. Úgy érzem, nagy hangsúlyt lehet arra fordítani, hogy a videókban megjelenő objektumokat extra tartalommal ruházzák fel. Egy lehetséges alkalmazást ábrázol a 6. ábra.



6. ábra – Tökéletes példa, hogy milyennek kell lennie az interaktív tartalomnak. Fontos alkalmazási területnek érzem. Mind a reklámozási oldalról innovatív, mind pedig a felhasználó élményét fokozná.

forrás: <http://lig.com/blog/wp-content/themes/modicus-remix/images/mobile%20recognition.png>

3.2.3 KIHÍVÁSOK, KÉRDÉSEK

Hogyan modellezzük a videó által szolgáltatott adatot? Az [14] cikk szerzői három különböző szintet definiálnak. Megközelíthető a probléma az úgynevezett nyers videó adat („raw data”), az

alacsony szintű képi jellemzők, és a szemantikus jellemzők oldaláról. Nyers adat, tulajdonság a videó felbontása, kamera típusa, képpont per másodperc arány, tehát egy fizikai tulajdonságról beszélünk. Alacsony szintű képi jellemzők a színek, elrendezésük, élek, textúrák csoportjai, jellemző pontok, vagy ezeknek különböző variációja. Természetesen számos kontextusban előfordulnak vizuális jellemzők. Ezt a szintet két csoportra osztották fel, mégpedig a statikus és dinamikus jellemzőkre. Statikus a jellemző, ha egy képkockáról származik, dinamikus pedig, ha több egymást követő képkockára van szükség. A szemantikus jellemzők képzik a harmadik szintet. Ez tekinthető a képfeldolgozás egyik legnehezebb feladatának, intenzitás értékekből következtetni szemantikus információra, mint objektumokra a képen és azok jelentésére. Míg vizuális jellemzőket teljesen automatizálva ki tudunk nyerni, addig szemantikus jellemzők előállításához gyakran felhasználói interakció szükséges, mint például tanító minta elkészítése. Egy negyedik szintet is modellezni kell egy videó esetében, mégpedig dinamikus tartalmat. Ha a statikus tartalomnak tekinthető az egyes képkockákról kinyert, szemantikus jelentéssel bíró objektumok vagy kép tartományok, akkor a dinamikus tartalom a statikus elemek közötti kapcsolat több képkockán keresztül. A szemantikus tartalom modellezése tehát az egyik legfontosabb lépés, amikor videókat szeretnénk vizuális tartalom alapján indexelni.

4 IRODALOMKUTATÁS

A fejezet első felében megvizsgálom milyen megközelítések és rendszerek léteznek a vizuális tartalomalapú videókeresés és -indexelés téma területén. A fejezetet a következő irodalmak alapján építettem fel: [2, 15, 16]. A hátralevő részben bemutatom az általam preferált, szegmentálás alapú módszereket és rendszereket, melyet összehasonlítok a hagyományos technikákkal.

4.1 A MULTIMÉDIÁS TARTALMAK LEÍRÁSA, INDEXELÉSE, CSOPORTOSÍTÁSA

Az [2, 15, 16] cikkek alapján bemutatom a multimédiás tartalmak (a videóra koncentrálok) leírásának, indexelésének és csoportosításának irodalmi hátterét.

4.1.1 VIDEÓ STRUKTÚRA ANALÍZIS

Egy videó tetszőleges sokaságú képkockából („frame”), snittből („shot”), jelenetből („scene”) és klipből („clip”) állhat. Ezek reprezentálják a képnek a struktúráját. A videó keresés érdekében szükségünk van rá, hogy meghatározzuk az egyes intervallumokat a jelenetek, snittek, és klipek között. Amennyiben tudjuk, hogy egy kifejezett jelenettel van dolgunk, akkor már élhetünk bizonyos feltételezésekkel, miközben kidolgozzuk a módszereinket.

A videók csoportosításának alapvető egysége a snitt, amelyen belül lévő képkockák erősen korrelálnak egymással tartalmi értelemben. Az egymást követő shotok között hirtelen változás figyelhető meg, amelyek mentén a videót különböző részekre tudjuk bontani. Általánosságban az algoritmusok először vizuális jellemzőket nyernek ki a képkockákról, és a kapott jellemzőket összehasonlítva kategorizálják az egyes képkockákat. A módszer hatékonysága a hasonlósági függvényről függ. Mind küszöbölés alapú, mind felügyelt és felügyelet nélküli tanításon alapuló módszereket is alkalmaznak. A dinamikus küszöbölést nehéz megoldani, a tanítással kapcsolatos technikáknál pedig megfelelő minőségű és számú tanítómintát kell biztosítani.

4.1.2 KULCS KÉPKOCKA ELŐÁLLÍTÁS

Egy snitten belül sok-sok redundáns képkocka van. Ha reprezentálni szeretnénk egy shotot, akkor ki kell számítani rá jellemző vektorokat, ezt megtehetjük úgy, hogy kitüntetünk egy, vagy akár több képkockát is, ami a kulcsképkocka lesz és annak a vizuális jellemzőit feleltetjük meg az egész snittre. A kulcsképkocka kinyerésére számtalan módszert alkalmaztak már. A szekvenciális összehasonlítás során az első képkockát kitüntetjük kulcsképkockának, majd megvizsgáljuk a rákövetkezőt. Ha a különbség jelentős, akkor az lesz az új kulcsképkocka. Az algoritmus addig megy, míg az utolsó elem is összehasonlításra került. Ez egyszerűen, gyorsan kiszámítható, azonban csak lokális különbségeket vesz figyelembe. A globálisan működő módszerek ezt küszöbölik ki, ahol az összehasonlítás során egy függvényt próbálnak minimalizálni (a kulcsképkocka tér el legkevésbé a többi képkockától). További lehetőség a képkockákat klaszterezni és a középpontokhoz legközelebb eső egyedet felhasználni. Másik érdekes módszer, hogy a képkockák jellemző vektorait a térben összekötik, és a problémát visszavezetik, arra, hogy meg kell keresni a görbén azokat a pontokat, amelyek legjobban reprezentálják a görbét. A módszer előnye, hogy az időrendi sorrendet fenntartja, viszont a számítási igénye magas. További meghatározási lehetőség az objektum- és eseménydetektálás. Ilyen esetekben, először meg kell határozni a videón objektumokat, majd ezeket a változás során nyilván kell

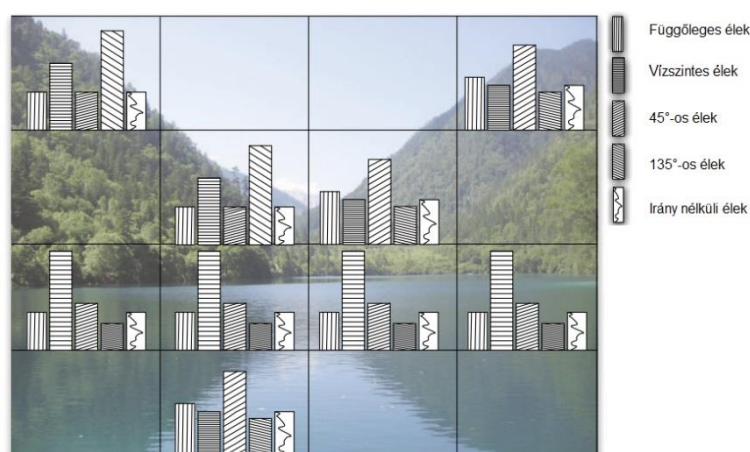
tartani. Az objektumok mozgása, eltűnése és megjelenése alapján már kisebb részekre lehet szegmentálni a videót, amennyiben ezt szeretnénk. A megközelítést viszont csak jól felépített környezetben lehet használni, bizonyos alkalmazási területeken. Az objektumdetektálás nagyban függ a tanítási mintáktól és a felállított szabályrendszerektől.

4.1.3 JELLEMZŐ VEKTOROK

Annak érdekében, hogy a videót részekre tudjuk bontani, illetve, hogy az egyes részek elemeit címkézhessük, a vizuális tartalmat több dimenziós jellemző vektorok formájában le kell írni. Ez lehet hisztogram, de egy képnek, kép résznek kitűntethetjük bizonyos tulajdonsággal rendelkező pontjait is a környezete leírásával. A szakirodalom három alacsony szintű jellemző csoportot határozott meg: szín, textúra és alak alapú jellemzők [15].

Az emberi érzékelés egyik legnagyobb részét a színérzékelés teszi ki. Maguk a színek, mint jellemzők a képfeldolgozásban az egyik legtöbbet felhasznált és legeredményesebben alkalmazott leírók. Lehet ilyen jellemző egyszerű RGB hisztogram is, de számos szofisztikáltabb algoritmusok is felhasználhatóak, mint a szín momentumok vagy az elrendezés leírók. Fontos kiemelni, hogy mielőtt bármilyen módszert felhasználnánk, mérlegelni kell, milyen szintérben dolgozzunk. Mivel az RGB szintér esetén kis megvilágítási változások nagy különbségeket mutathatnak a leírás során, így érdemes áttérni olyan szintérbe, ahol a megvilágítás komponensét valamilyen szinten szét tudjuk választani az objektum tényleges színétől. A színleíró módszerek nem rendelkeznek magas számítási igényekkel, azonban nem reprezentálják konkrétan az elemek alakját vagy textúrázottságát.

Az alak és textúra jellemzők ugyanúgy a képeken megjelenő élektől függenek. Amennyiben a videón nem csak homogén felületű objektumok találhatók meg, akkor a textúra leírást mindenképpen érdemes használni. Számos algoritmus áll már rendelkezésre, többek között az EHD (szemléltetesként szolgál a 7. ábra) és HOG leíró, amit sikeresen alkalmaztam a rajz alapú képkereső projektem során [1] és más rendszerek is gyakran használják [17]. Ezek az algoritmusok jól leírnak textúrákat és alakokat is egyben. Fontos felhasználni az élek adta információkat, azonban a különböző megvilágítású, részletezésű képekről gyakran sok fals pozitív és fals negatív élt állítunk elő, amely befolyásolja a leírás minőségét.



7. ábra – Az EHD leíró szemléltetése.

forrás: http://cybertron.cg.tu-berlin.de/eitz/pdf/2010_cag.pdf

Az alfejezetben csak egy áttekintést mutattam be a különböző jellemző kinyerési lehetőségekről. A projekt során felhasznált jellemzőket, a megvalósítással kapcsolatos algoritmusokat és problémákat, saját megoldásokat a későbbiekben, a saját rendszer fejezet keretein belül mutatom be.

4.1.4 VIDEÓ ANNOTÁCIÓ ÉS KERESÉS

Az előző alfejezetekben bemutatásra került, hogy a videókat, hogyan bonthatjuk fel strukturális egységekre, és ezeknél a metódusoknál mi alapján határoztuk meg a vizuális tartalmat. Az annotációs lépés során az egyes részeket előre meghatározott kategóriákba, koncepciókba soroljuk (pl.: autók, állatok, fák, emberek sétálnak, iroda). Az egyszerű koncepcióktól a teljesen összetett, már eseménynek nevezhető annotációs problémákra létezik már megoldás. Az előző fejezetben megismert csoportosítás során a videó egészére nézve kerültek megállapításra jellemzők, most azonban a videó egyes részeit figyelembe véve kellene bizonyos annotációkat hozzárendelni a videóhoz.

4.1.5 FELHASZNÁLÓI VISSZACSATOLÁS

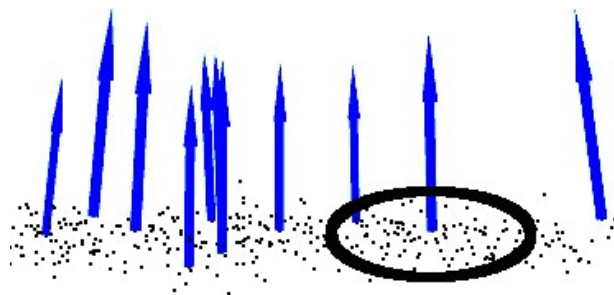
A keresés során a felhasználó a megjelenített eredményeken, explicit módon is javíthat, úgy hogy megadja a találati listából, melyek a számára releváns találatok. Ezt nevezik explicit felhasználói visszacsatolásnak. Ilyen esetekben két fő módszert alkalmaznak. Az egyik lehetőség, hogy a kereső jellemző vektort a felhasználó preferenciái szerint megváltoztatják a vektortérben, úgy, hogy a preferált találatok által reprezentált jellemző vektorokhoz essen közelebb. A másik megoldás, hogy a vektortérre egy maszkot alkalmazunk, amely súlyozza adott vektorok bizonyos dimenzióit. Ennek előnye a jobb találatok számának növekedése, azonban felhasználói oldalról kooperációt és türelmet igényel. Implicit módszerek is léteznek, mégpedig az úgynevezett kattintás analízis („Click-through analysis”) [2]. Itt a felhasználó által kiválasztott videók alapján történik a visszacsatolás.

4.2 RGB-D SENZOROK

A hagyományos 2D képalkotás miatt fontos információkat elveszítünk az objektumok alakjáról és térbeli kapcsolatairól. A pixelek térbeli elhelyezkedésének tudatában számos új, hatékony leírót fejleszthetünk ki objektumdetektálásra. Ilyen szenzorok építésére több lehetőségünk is van, példának okáért [18] 2D CCD kamerát kombinál egy ToF kamerával, hogy a távolságtérképet kinyerje. Sokan használják a hirtelen népszerűsége miatt tett Microsoft által kiadott Kinect RGB-D szenzort is, amely remek ár-érték aránnyal rendelkezik.

4.2.1 3D MODELL VAGY 2D JELLEMZŐK?

3D jellemzők bizonyos 3D beli pontok, pozíciók a térben, amelyek geometriai információkat írnak le az adott pont és környezetének tartalma alapján [19]. A 8. ábrán látható egy példa, ahol a tér egy pontjához hozzárendelünk egy vektort, ami a pont adott sugarú környezetére illeszthető síknak a normálvektora.



8. ábra – A mintavételezett pontok adott környezetére egy síkot illesztünk. A síknak a normálvektora lesz az adott pontra (és környezetére) jellemző vektor.

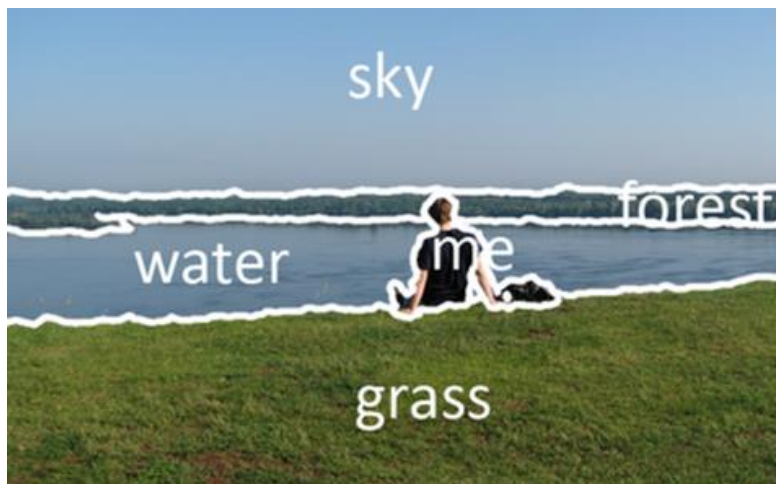
forrás: http://docs.pointclouds.org/trunk/group_features.html

A 3D jellemzőket gyakorlatilag 6 dimenziós adatokkal állítjuk elő, mégpedig a térbeli x, y, z koordinátákból és az r, g, b színösszetevőkből. Ezeket az adatokat pontfelhőnek nevezzük. Mivel az RGB-D szenzorok nagy mennyiségű ponthalmazt vagy pontfelhőt generálnak, így gyakran mintavételezni kell a bejövő adatokat, hogy a hatalmas számítási igényt csökkentsük. Számos algoritmus alapul a szomszédos pontok keresésén. Az algoritmusok minőségromlásának elkerülése érdekében, a mintavételezett ponthoz közeli elemeket az eredeti adathalmazban is lehet keresni. Ilyenkor az eredeti ponthalmazt, mint keresőfelületet használjuk fel. A projekt során a 3D és 2D jellemzőket is felhasználtam.

4.3 A SZEGMENTÁLÁS ALAPÚ TECHNIKÁK

A képfeldolgozó alkalmazások során sok esetben a végső cél a képek szegmentálása, az objektumok felismerése, és a kinyert tudás alapján az adott környezet automatikus manipulálása. A Google tudós kereső segítségével számtalan találat áll rendelkezésre tudományos cikkek, weboldalak formájában. A terület fejlődését elősegítik az objektumdetektáló versenyek, mint például a „The PASCAL Visual Object Classes (VOC) Challenge” [20]. A verseny fő célja objektumok detektálása képeken és ezek osztályozása (személy, állat, jármű, beltéri). Alapvetően ez egy felügyelt tanítással kapcsolatos probléma. Ha kategorizálnák a módszereket nehézség szerint, akkor három csoportot határozhatnánk meg: objektumok detektálása képen (megtalálható-e bizonyos objektum a képen vagy nem), objektumok lokalizálása (hol található meg) és objektumok szemantikus szegmentálása (mely pixelek melyik objektumhoz tartoznak). A következő bekezdésben definiálom a szakirodalomban található fogalmakat.

A szakirodalom szerint két megközelítés alapján „szegmentálható” egy kép (kép sorozat, „3D pont felhő – 3D point cloud”). Itt a szegmentálást értsük olyan értelemben, hogy az adott képekről milyen módon nyerünk ki információt. Az egyik oldalról „semantic segmentation - szemantikus szegmentálás”, mint fogalom áll a középpontban [11, 21]. Ilyenkor minden pixelt a képen a szemantikus hovatartozása alapján kell osztályoznunk (lásd 9. ábra). A másik megközelítés az úgynevezett interaktív vagy általános objektumdetektálás [20]. Számos detektor ablakot pontozva kell eldönteni, hogy melyik detektor ablak tartalmazhat objektumot (lásd 10. ábra). A felhasznált jellemzőkinyerési módszereket úgy kell megválasztani, hogy azok a lehető legnagyobb mértékben meghatározzák, hogy egy ablak objektumot tartalmaz-e vagy sem. A kérdés középpontjában az áll, hogy miért objektum egy objektum, melyek a megfelelő jellemzők, hogyan történik az emberi észékelés.



9. ábra – A kép a szemantikus jelentése alapján lett szegmentálva. Minden pixel kategorizálva lett.
forrás: <http://computerblindness.blogspot.com/2010/06/object-detection-vs-semantic.html>



10. ábra – Az objektumdetektálás eredményét láthatjuk. A zöld négyzetek jelölik a beazonosított elemeket. A környezet többi részéről nem kapunk többletinformációt.
forrás: <http://computerblindness.blogspot.com/2010/06/object-detection-vs-semantic.html>

Az irodalomkutatás során megismert módszereknél a videókat struktúrájuk alapján kategorizálják. A saját megoldásomban a kategorizált leírás helyett az egyes objektumokat szegmentálom. Miért szegmentálás alapú technikákat használok fel? Ha a videó kategorizálást választanám, akkor a kapott kulcs képkockák, melyek a videót reprezentálnák, azok nem biztosítanának mozgás adatokat. Továbbá a rendszer modularitása is szempont, így dinamikusabban lehet fejleszteni. Az elkészült eredményeket több másik projektben is újrafelhasználhatóvá lehet tenni. Azt persze meg kell jegyezni, hogy az irodalomban megismert és a projekt során később felhasznált módszerek nagyon hasonló lépésekből épülnek fel, csak más a cél és így a kimenet is.

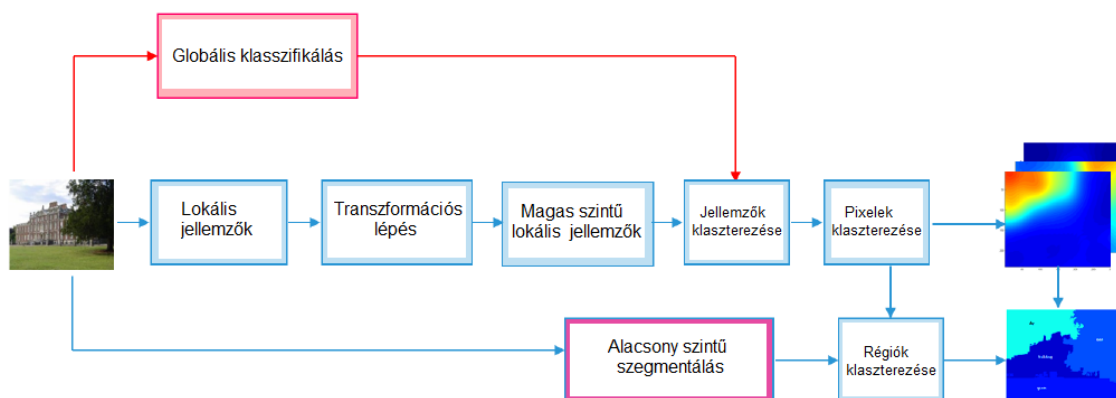
4.4 SZEMANTIKUS SZEGMENTÁLÁS

Mint már ismertettem a szemantikus szegmentálás (további elnevezések: „multi-class image annotation, shape guided object segmentation”) során, a képen a pixeleket különböző osztályokba soroljuk a szemantikus jelentésük alapján. A szemantikus szegmentálás koncepciója a

szemantikus webre, mint ötletre vezethető vissza [22], ahol a szerző kifejtette már 1990-ben, hogy idővel már minden adat megtalálható lesz az interneten. Az adatokat pedig számítógépek fogják analizálni és értelmezni. A szemantikus szegmentálásnak éppen ez a célja, a képi tartalmat modellezi és interpretálja jelentésre. Ez egy felügyelt probléma, mivel az egyes régiókat felcímkézi, nem úgy, mint a felügyelet nélküli tanításos szegmentálós rendszereknél (MeanShift, K-Means, Watershed szegmentálás). A szemantikus szegmentálás kimenete sokkal egyszerűbben értelmezhető és könnyebben használható. A [21] irodalom alapján a következő alfejezetben megvizsgálom, hogy milyen főbb sajátosságai vannak ezeknek az algoritmusoknak.

4.4.1 SZEMANTIKUS SZEGMENTÁLÓ ALGORITMUS ÁLTALÁNOS MODELLJE

A 11. ábrán látható a szemantikus szegmentáló algoritmusok általános modellje. Természetesen létezhetnek különböző más elképzelések is, de az irodalom [23] alapján ez tekinthető széleskörűen ismertnek és elfogadottnak. Ellentétben a felügyelet nélküli tanításon alapuló szegmentáló módszerekkel („MeanShift, K-Means, Watershed” szegmentálás) az algoritmus során felügyelt tanítást alkalmaznak. Az algoritmust négy-öt fázisra lehet felosztani, az első fázis a modellépítési fázis, amely során a felcímkézett tanító képekből (kép=objektum) alacsony szintű lokális jellemzőket nyerünk ki és ezek alapján építünk egy modellt. A modell alapján a tanító képekről (képek több objektummal) kinyert alacsony szintű jellemzőket áttanszformáljuk egy magasabb dimenzióba. Ez történik a második lépésben. A magas szintű leírók és felcímkézett tanító minták alapján betanítjuk az osztályozót a negyedik lépésben. A végső feladat az új képek pixel alapú osztályozása és esetleges globális optimalizálása.



11. ábra – A szemantikus szegmentálás általános folyamatábrája [23].

4.4.2 LOKÁLIS JELLEMZŐK KISZÁMÍTÁSA

Az első lépés során lokális vizuális jellemzőket kell kinyerni. Igaz, hogy minden pixelt kategorizálunk, de attól függetlenül még nem szükséges kiszámolni a jellemzőket minden pixelre. Sőt, ha belegondolunk, egy pixel sokkal kevesebb információval rendelkezhet, mint mondjuk a környezete. Ezért elég kiválasztani úgynevezett jellemző pontokat, amelyek környezete alapján kinyerjük a vizuális tartalmat („patch feature extraction” vagy „patch detection”) [23]. A legegyszerűbb, ha a képet felosztjuk egy grid segítségével, és az egyes csomópontokat vesszük jellemző pontoknak („dense sampling”) [24]. A pontok környezetét akkorára válasszuk meg, hogy egymással bizonyos mértékben átfedésben legyenek. Használhatunk továbbá sarokdetektáló

módszereket, mint a Harris detektor, vagy akár SIFT. Az utóbbiakat viszont kevésbé használják, mivel a kapott pontok segítségével nem tudjuk lefedni az egész képet, így bizonyos területen elhelyezkedő vizuális tartalmak leírás nélkül maradnának.

4.4.3 JELLEMZŐ VEKTOROK TRANSZFORMÁLÁSA

A következő lépés során, ezeket az alacsony szintű leírókat (szín, textúra,...), magas szintű leírókká transzformálják. A mögöttes matematikai megfontolás pedig a következő: egy magasabb dimenziós térben a vektorok szeparálása könnyebb. Azonban a dimenziók számának növekedésével exponenciálisan nő a számítási komplexitás, amit PCA segítségével eredményesen csökkenteni lehet [24].

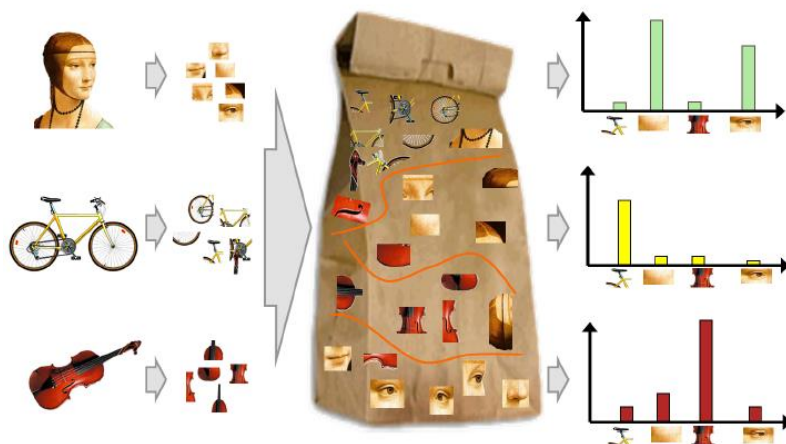
Az egyik széles-körben elterjedt módszer a („Visual Bag of Keywords vagy Bag of Words” – továbbiakban BOW) [25, 26, 24]. Az ötlet a természetes nyelvfeldolgozás analógiájából született meg, ahol dokumentumok csoportosítására használták fel (lásd 12. ábra). Egy előre elkészített szótár szavainak előfordulásából a dokumentumban előállítottak egy hisztogramot, ami jellemezte azt. Ezt a metódust, leírási módot a képfeldolgozás számos területén aktívan alkalmazzák, többek között objektumdetektálás és lokalizálásnál [20], vagy képi tartalomalapú keresők esetén [26].



12. ábra – A szavak mérete jelenti, hogy adott szövegben melyik szó, milyen gyakran fordult elő. Ezáltal könnyen csoportosítani lehet szövegeket csupán a kulcsszavak meghatározásával [23].

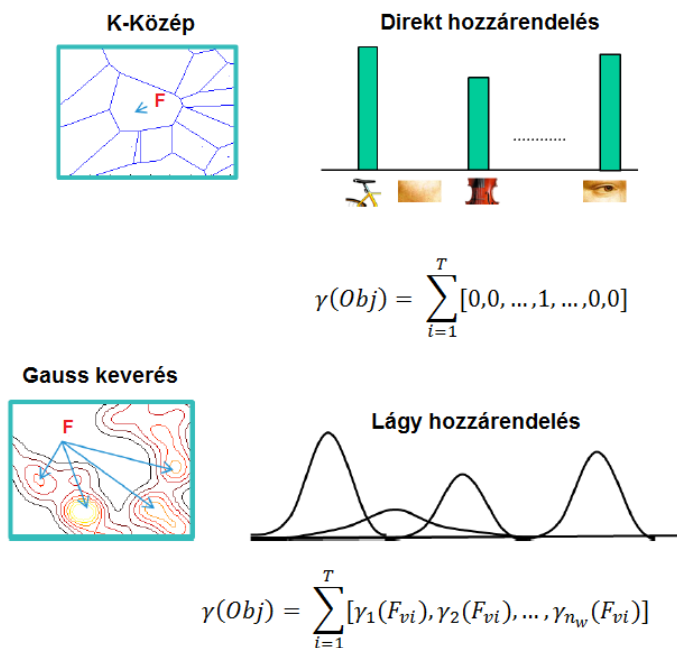
A BOW módszer segítségével alakítjuk át az előbb már definiált alacsony szintű jellemzőket magas szintű jellemzőkké. Az első lépésben fel kell építeni egy vizuális szótárt a következő paraméterekkel: objektumok száma (n_o), az egy objektumról kinyert vizuális jellemzők száma (n_v) és a vizuális szavak száma (n_w). Az alacsony szintű leírók tárgyalása során bemutattuk milyen módon lehet mintavételezni egy képet. Az n_v és n_w száma a problémától függ és az adatbázis sokszínűségétől. [26] esetén 500 jellemző/objektum, azonban [25] esetén ez mérték 5000. A vizuális szótár (D) egy n_w darab $\{M_i / i=0,1,2,...,n_w\}$ paraméter összessége. Az M_i paraméter az F_{vi} lokális jellemzők (tanító mintákról kinyert) egy klaszterét írja le. Ez lehet a klasztert reprezentáló vektor vagy paraméterek összessége (eloszlás, átlag, ...). A vizuális szótárt úgy építjük fel, hogy veszünk tanítómintákat, amelyről kinyert vizuális jellemzőket (F_{vi}) osztályozzuk és az egyes klasztereket feleltetjük meg egy a vizuális szavaknak. A legalapvetőbb módszernek a K-közép (K-Means) algoritmus tekinthető [26, 25, 24, 27] amit széles körben felhasználtak már. Ezen felül

még sokat használják a Gauss-keveréket [26, 28, 23] és a véletlen döntési fákat is [21, 8]. A klaszterek száma tipikusan 500-1000 körüli [26, 25]. A 13. ábra szemlélteti a BOW módszert.



13. ábra – A BOW módszer szemléltetése. Baloldalon jelennek meg a tanító objektumok, melyekről lokális jellemzőket nyerünk ki. A lokális jellemző osztályozása után el lehet készíteni a magas szintű jellemző vektort [23].

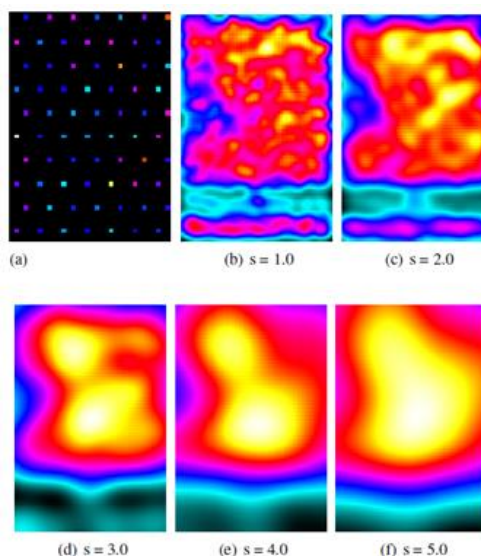
A következő kérdés, hogy a szótár alapján hogyan transzformáljuk át a szemantikus szegmentálandó képnek az alacsony szintű jellemzőit magas szintű leírókká. A BOW megközelítés alapvetően egy egész képet tud hatékonyan jellemezni, ahol megadja, hogy a képen az adott szótárból származó szavak milyen valószínűséggel találhatók meg a képen (így épül fel a hisztogram). Ezzel szemben a szemantikus szegmentálásnál nem egy egész képre kell egy hisztogramot előállítani, hanem egy jellemző vektorhoz kell egy magasabb dimenzionalitással rendelkező leíró létrehozni. Ha a BOW módszert használnánk, akkor az eredmény tipikusan a következő formájú lenne: $X_t = [0, 0, \dots, 1, \dots, 0, 0, 0]$, tehát a transzformáció megadja, hogy az F_{vi} lokális jellemző vektor melyik D szótárbeli szóhoz van legközelebb. Ez egy direkt hozzárendelés, az irodalomban „hard assignmentnek” nevezik [23]. Ez a leírás nem lenne hatékony, így valószínűség alapú leírást („lágý hozzárendelés – soft assignment”) kell alkalmaznunk, ami megadja, hogy az adott alacsony szintű jellemző, milyen valószínűséggel lehet egy adott szó a szótárból/ milyen messze van egy adott szótári szótól $X_t = [\gamma_1(F_{vi}), \gamma_2(F_{vi}), \gamma_3(F_{vi}), \dots, \gamma_{n_w}(F_{vi})]$. $\gamma_i(F_{vi})$ függvény megadja, hogy a F_{vi} lokális jellemző vektor milyen valószínűséggel lehet az M_i kódszó. A legkorszerűbb megoldás az irodalmak alapján a Gauss-keverék módszer, melyet a Fisher kernel használatával egészítenek ki [21, 8, 28, 23].



14. ábra – A direkt hozzárendeléshez képest a lágy hozzárendelés pontosabb jellemző transzformálást tesz lehetővé [23].

4.4.4 OSZTÁLYOZÁS

A felcímkézett tanító képek alapján kinyert magas szintű jellemzők segítségével be kell tanítani az osztályozót. Erre a célra gyakran használják az SVM-et vagy a Naiv Bayes osztályozót [25]. [21] továbbá tesztelte, hogyan lehet felhasználni a véletlenszerű döntési fákat. Mivel a képet mintavételeztük, így csak bizonyos pixelekre tudjuk elsőnek megállapítani, hogy melyik osztályba milyen valószínűséggel tartozhat. A végeredményből előállíthatunk egy valószínűségi térképet az egyes osztályokra (lásd 15. ábra). Az osztály valószínűségi térkép sokkal flexibilisebb leírást ad és könnyebben kiterjeszthető más modellekhez, mint a Markov Véletlen Mezőkhöz („Markov Random Fields”) vagy Feltételes véletlen mezőkhöz („Conditional Random Fields”) [21, 8]. Továbbá, ha adott osztályból származó pixel(ek) a képen egy küszöbnél kisebb valószínűséggel fordul(nak) elő, akkor azokat tekinthetjük úgy is, hogy egyáltalán nem jelennek meg címkeként [25], ezzel is javítva az eredményen.



15. ábra – (a): Az osztályozás megállapította, hogy a mintavételezési helyek milyen osztályokba tartoznak.
(b)-(f): A valószínűségi térképet Gauss szűrővel minden pontra kiterjesztünk.
[21].

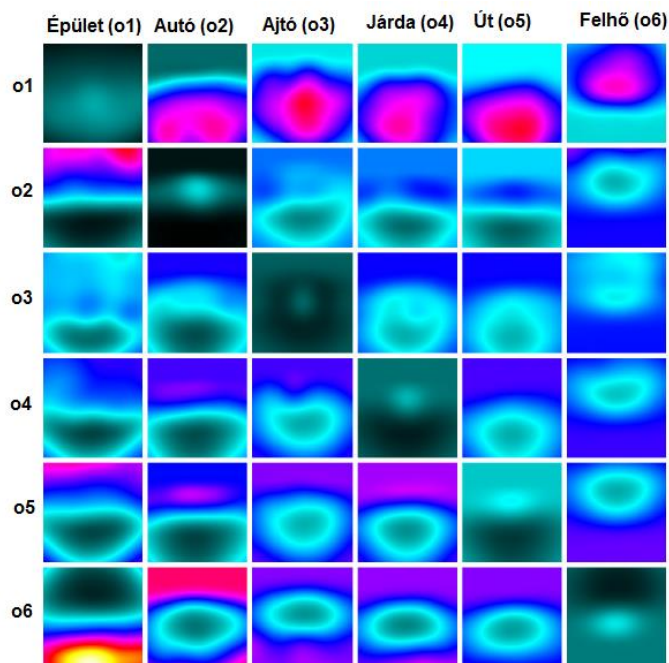
Ez a térkép akár a végeredmény is lehet, de [21, 8, 25] szerint sok esetben a végeredmény sok esetben nem hoz kielégítő eredményt. Viszont a térkép remekül felhasználható a túl-szegmentált képek (felügyelet nélküli tanításos algoritmusok) globális optimalizálására. Ilyen képről beszélünk, abban az esetben, ha a szegmensek száma nagyobb, mint az elvárt. A túl-szegmentálás oka egyszerű: két régiót egyszerűbb összevonni, mint szétválasztani. A gráf vágás (Graph Cut) egy gyakran használt algoritmus [21]. Legyen $G=\{V,E\}$ egy gráf, ahol V az egyes képszegmensek (a túlszegmentálás eredményei) jelölik a gráf csúcsait. E pedig az élek pedig a csúcsok közötti hasonlósági függvény eredménye. A cél a következő energia függvény minimalizálása:

$$E(\beta) = E_d(\beta) + \rho E_s(\beta)$$

Ahol $\beta = \{1, \dots, c\}^L$ címkék az L régiót tartalmazó képen. $E_d(\beta)$ adja meg, hogy milyen költséggel jár egy szegmensét címkézni, $\rho E_s(\beta)$ pedig a szegmenst jellemzi a vizuális tulajdonságai alapján.

4.4.5 PROBLÉMÁK

A szemantikus szegmentálás során számos problémába ütközhetünk. Az egyik probléma a komplexitás nagysága a számítás során. A kódkönyv generálása és a globális optimalizálás számít plusz költségnek. Nehézséget jelenthet az is, hogy többször is kell modellt építeni és az alapján osztályozni. Ezért megfelelő minőségű és mennyiségű tanító mintát kell rendelkezésre bocsátani. Ha megvizsgáljuk közelebbről például a BOW módszert, akkor látjuk, hogy csak lokális jellemzőket veszünk figyelembe, így elveszítjük az egyes objektumok globális kontextus tulajdonságait. [21] és [11] modellezte, hogy az egyes osztályok egymáshoz viszonyítva a képek adott pontjain milyen valószínűséggel fordulnak elő. Továbbá belátható, hogy bizonyos osztályokról tudhatunk előre információkat (az ég a kép felső részén helyezkedik el, a monitor az asztalon van, ...). A megoldást relatív valószínűségi távolságnak nevezik és kifejezi, hogy ha van egy p' pixel c' címkével, akkor egy p pixel $(\hat{u}, \hat{v})$ offszetre a p' pixeltől milyen valószínűséggel vesz fel c címkét [21]. A 16. ábrán látható a valószínűségi távolságok vizualizálása.



16. ábra – A relatív valószínűségi távolságok ábrázolása [21].

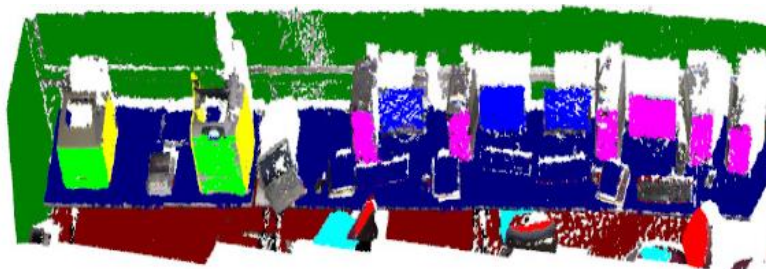
4.4.6 ESETTANULMÁNY – BELSŐ TEREK SZEMANTIKUS SZEGMENTÁLÁSA

Mivel [11] munkássága nagy motivációt jelentett a dolgozat elkészítése során, bemutatom az alfejezet keretein belül. Céljaik között szerepel belső terek szemantikus szegmentálása, felhasználva az objektumok egymáshoz viszonyított helyzetét. A szerzők szeretnék kihasználni az RGB-D szenzorok által nyújtott többletinformációt, így felülmúlni a 2D objektumcímkézés eredményeit.

Első lépésben az adott környezetről több irányból felvételeket készítenek, és ezek alapján az RGBDSLAM segítségével felépítik a 3D modellt (3D pontfelhő) (lásd 17. ábra). A következő lépésben túlszegmentálják a képet a 3D pontfelhőre kiszámított normálvektorok simasága és folytonossága alapján. Az egyes szegmensek felcímkézése a cél (végeredmény a 18. ábrán). Az 2D vizuális leírók mellett számos egyéb megközelítést alkalmaztak. Modellezték, hogy ha a szomszédos szegmensek hasonlóak, akkor nagy valószínűséggel egy objektumot határoznak meg. Eredményeinek sikerességét a szegmensek relatív geometriai jellemzők modellezésének tudják be. Például a monitor mindig az asztalon van, vagy a székek az asztalok közelében helyezkednek el.



17. ábra – Kinect felvételek alapján felépített 3D modell [11].



18. ábra – A szemantikus szegmentálás végeredménye [11].

4.5 ÁLTALÁNOS OBJEKTUMDETEKTÁLÁS

Az előző alfejezetben látott szemantikus szegmentálástól eltérő algoritmust fejlesztettek ki a [20] irodalomban. A hagyományos objektumlokalizáció során különböző objektumokat és vizuális jellemzőit tanítják be. A felismerés során pedig bizonyos ablakokról a felépített modell alapján eldöntik, hogy melyik objektumot tartalmazhatja. A [20] szerzői ezzel szemben egy általánosabb módszert találtak ki, ahol a fő cél az objektumoknak a lokalizálása, azonban nem adják meg, hogy melyik objektumról van szó, csak egy értékkel jelzik, hogy adott ablak milyen valószínűséggel tartalmaz objektumot. A hangsúlyt a jellemző kinyerésre fektették, hogy megtalálják azokat a vizuális leírókat, amelyek legnagyobb mértékben meghatároznak egy objektumot. Az objektum jól definiált szélekkel rendelkezik és jól elkülöníthető a környezetétől, továbbá a színei és textúrái egyedi megjelenést biztosítanak.

4.5.1 FELHASZNÁLT JELLEMZŐK

A szerzők 4 különféle leíró fejlesztettek ki („Multi-scale saliency map, Color Contrast, Edge Density, Superpixels Straddling”), melyeket egymás mellett használtak, így javítva a felismerés hatékonyságát [20]. Az ember a kutatások alapján két lépésben detektál objektumokat [29]. Az első lépésben („pre-attentive process”) olyan alacsony szintű jellemzők alapján, mint orientáció, élek, vagy intenzitás tekintünk bizonyos dolgokat objektumnak. A második lépésben („complex attention”) pedig eldöntjük, hogy miről is van szó pontosan. Az első lépésre alapozva fejlesztettek ki egy olyan jellemzőt, amely megadja egy adott környezetre vagy ablakra, hogy mennyire egyedi („Saliency detection”). Az információ elméletnél a kódolás érdekében a képet két részre bontották fel:

$$H(\text{image}) = H(\text{Innovation}) + H(\text{Prior Knowledge})$$

Ahol $H(\text{Innovation})$ jelzi a kép lényegi tulajdonságait és $H(\text{Prior Knowledge})$ pedig a redundáns információt, amit tömöríteni kell. A képen a hasonlóságok redundanciát jelentenek, amit statisztikai eszközökkel lehet vizsgálni. [29] a Fourier transzformáció segítségével vizsgálja a következőképpen:

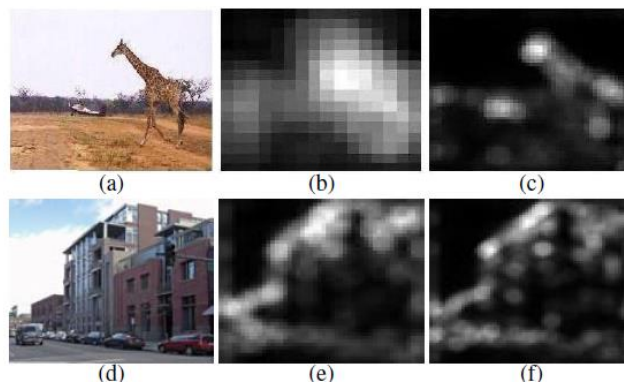
$$\mathcal{L}(f) = \log(\mathfrak{F}[I(x)]), \text{ ahol } \mathfrak{F} \text{ a Fourier transzformációt jelöli}$$

$$\mathcal{A}(f) - \text{A spektrum logaritmusának alakja,}$$

$$\text{megközelíthető a következőképpen: } h_n(f) * \mathcal{L}(f), \text{ ahol } h_n(f) = \frac{1}{n^2} \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}$$

$$\mathcal{R}(f) = \mathcal{L}(f) - \mathcal{A}(f),$$

$\mathcal{R}(f)$ inverz Fourier transzformáltja pedig megadja az egyediség mértékét (lásd 19. ábra). [20] szerzői ezt fejlesztették tovább. Mivel a Fourier transzformáció során a kép mérete is számít, így ezt felhasználva több szinten is kiszámolják az adott térképet.



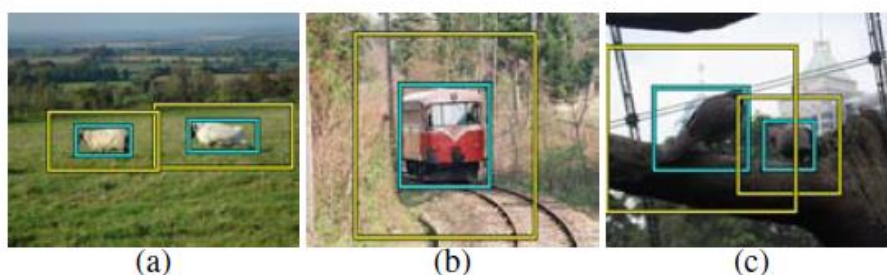
19. ábra – (b, c) és (e, f) az egyediség mértékét adják meg („saliency map”) [20].

A szín kontraszt („Color Contrast” – továbbiakban CC) leíró megmutatja, hogy egy adott ablakon belüli képnek a színei, mennyire térnek el a lokális környezettől. A lokális környezetet a következőképpen választották meg egy w ablakhoz képest (lásd 20. ábra), hogy a következő egyenlőség fennálljon:

$$\frac{|loc(w, \Theta_{CC})|}{|w|} = \Theta_{CC}^2 - 1$$

Θ_{CC} faktorról növelték meg w ablakot. A CC leíró a $L \times a \times b$ -ban kiszámolt h hisztogram alapján számolják:

$$CC(w, \Theta_{CC}) = \chi^2(h(w), h(loc(w, \Theta_{CC})))$$

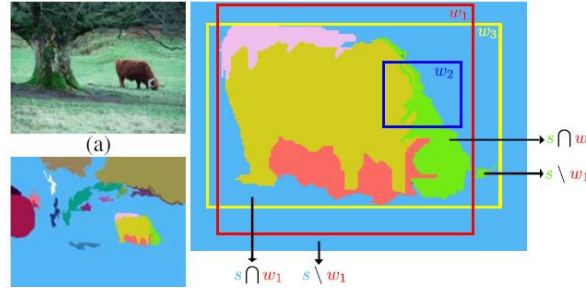


20. ábra – Az ábrán látható az ablakok mintavételezésének módja [20].

Az él sűrűség („Edge Density” – továbbiakban ED) leíró hasonlóan áll elő, csak a szín helyett az él pixelek számának a hányadosát kell venni. Fontos, hogy az él detektálás során biztosítsuk, hogy 1-pixel vastag élek álljanak elő az egész képen. Ezt a nem maximumok elnyomása („non-maximum-suppression”) metódussal meg lehet oldani. A következő nagyítást alkalmazták:

$$\frac{|loc(w, \Theta_{ED})|}{|w|} = \frac{1}{\Theta_{ED}^2}$$

Szuper pixel közrefogás („Superpixels Straddling” – továbbiakban SS) esetén a képet túl kell szegmentálni, és az egyes szegmensek lesznek a szuper-pixelek. A leíró megmondja egy detektor ablakra, hogy az adott ablakon belüli szuperpixelek milyen mértékben tartoznak az adott ablakhoz (lásd 21. ábra). Minél több rész helyezkedik el a szuperpixelből a detektor ablakon kívül, annál nagyobb a valószínűsége, hogy a detektor ablak, csak részben fedi le az adott objektumot.



21. ábra – A túlszegmentált képre adott helyzetekre kiszámítják az SS leírót [20].

A fent leírt jellemzőkkel egy probléma van. Ha a képen sok egymáshoz közeli objektum található, akkor a vektorok nem fognak pontos reprezentációt nyújtani, mivel ilyenkor nem különülnek el akkora mértékben a környezetüktől. Ezt a mintavételezés pontosságával lehet javítani, de ez növeli a számítási igényt.

4.5.2 MODELLÉPÍTÉS

A tanítás során 50 képet választottak ki, melyek összesen 291 objektumot tartalmaznak, amit 48 különböző kategóriába lehet sorolni. A tanító mintán megadták az egyes példányokat körbefoglaló ablakot, mint bináris maszkot. Mivel a cél nem azonosítás, hanem csak lokalizálás, így az egyes ablakok nincsenek annotálva. Az egyes jellemző vektorok paramétereit (Θ) a Bayes-osztályozó segítségével tanították. Minden tanító mintán generáltak 100000 véletlenszerű ablakot a képen. Egy w ablak tartalmaz egy o objektumot, ha:

$$\frac{|w \cap o|}{|w \cup o|} = 0.5$$

Ha tartalmaz egy objektumot, akkor pozitív mintának ($\mathcal{W}^{obj}$) tekintjük, különben negatívnak ($\mathcal{W}^{bg}$). Θ minden értékre megállapíthatók a következő valószínűségek:

$$p_{\Theta}(obj|CC(w, \Theta))$$

$$p_{\Theta}(CC(w, \Theta)|bg)$$

Az optimális paraméter a következő kifejezés maximalizálásával kapható meg:

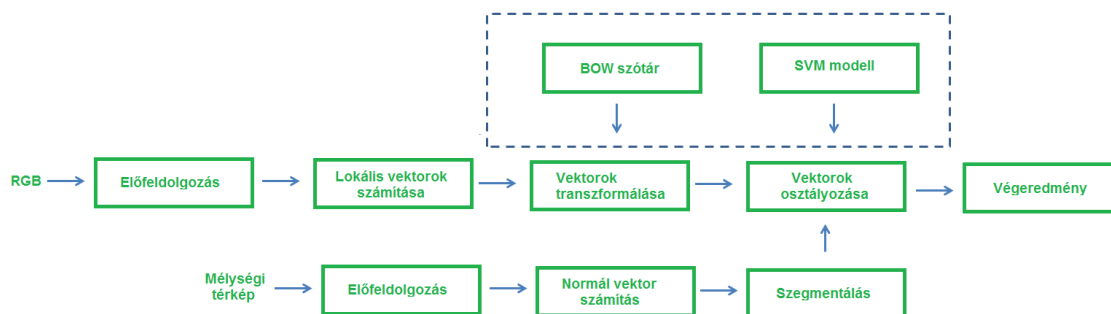
$$\arg \max_{\Theta} \prod_{w \in \mathcal{W}^{obj}} p_{\Theta}(obj|CC(w, \Theta))$$

A fentiekben a CC paraméterek tanulását mutattam meg. Az egyik lehetőség, hogy mindegyik paramétert külön-külön tanuljuk, vagy együtt, mint összefüggő egész. Az elkészített tesztek alapján azt a következtetést vonták le, hogy a jellemzők kombinált modellezése a Naiv-Bayes módszerrel eredményesebb volt.

5 SAJÁT RENDSZEREM

A fejezetben belül bemutatom a saját rendszerem globális felépítését. Majd megvizsgálom közelebbről az egyes építőelemeket. Az előfeldolgozástól a szegmentálásig bezárólag részletezem a felhasznált algoritmusokat és technikákat.

5.1 A RENDSZER GLOBÁLIS FELÉPÍTÉSE



22. ábra – A rendszerem globális folyamatábrája.

A fenti ábrán látható a rendszerem globális felépítése. A bemenet az RGB és a D adatfolyam, amit Microsoft Kinect kamerával készített adatbázisból származik. Ezen az adatokon még előfeldolgozást kell elvégezni a képpalkotás bizonyos problémái miatt, továbbá egy használhatóbb bemenetet generálása is cél. A mélységi térképet átalakítom pontfelhővé és kinyerem a pontokból mutató normálvektorokat. Ezek alapján már elő- szegmentálni tudom a pontfelhőt. A kérdés, ami felmerül, hogy az adott szegmensek milyen objektumnak vagy objektum résznek feleltethetők meg. Itt használom fel az RGB adatfolyamot. A kinyert lokális vizuális jellemzőket magas szintű jellemzőkké transzformálok és egy háttérmodell és BOW szótár alapján, amit tanítás során készítek el. A szegmentált pontfelhőt visszavetítem egy 2D képre és az eredeti RGB kép objektum maszkjaként alkalmazom. Az egy szegmensbe foglalt lokális jellemzőket összegzem, és ezt osztályozom, hogy melyik címkehez tartozik. Ezt a lépést követően már felcímkezett részekkel dolgozhatok tovább. Ha két szomszédos tartomány együtt jobban meghatároz egy címkét, akkor egyesítem. Röviden így tudnám összefoglalni a rendszerem alapjául szolgáló fő ötletet, melynek segítségével hatékonyan ki tudom használni a szenzor nyújtotta adatokat, ellentétben [11] cikk által bemutatott rendszerénél, ahol a jellemzőket is csak a pontfelhő alapján nyerik ki.

Mielőtt a rendszert használhatnám, el kell végezni egy tanítási folyamatot. Az Irodalomkutatás során megismert szemantikus szegmentálás során bemutatott módszereket vegyítettem az általános objektumdetektálásból adoptált technikákkal. A tanító minták alapján kinyert alacsony szintű vizuális jellemzőket Gauss-keverékes módszerrel osztályozom. A klasztereket jellemző eloszlások paraméterei határozzák meg a vizuális szavakat a BOW szótárban. A tanító minták minden objektumához meghatározok egy magas szintű jellemző vektort az objektumról kinyert lokális jellemzők és BOW szótár alapján (lásd 14. ábra). Az objektumhoz tartozó vektort és címkét egy többosztályos SVM osztályozóval klasszifikálok. A keletkezett modell alapján egy címke nélküli, magas szintű jellemzőről el tudom dönteni hova tartozik.

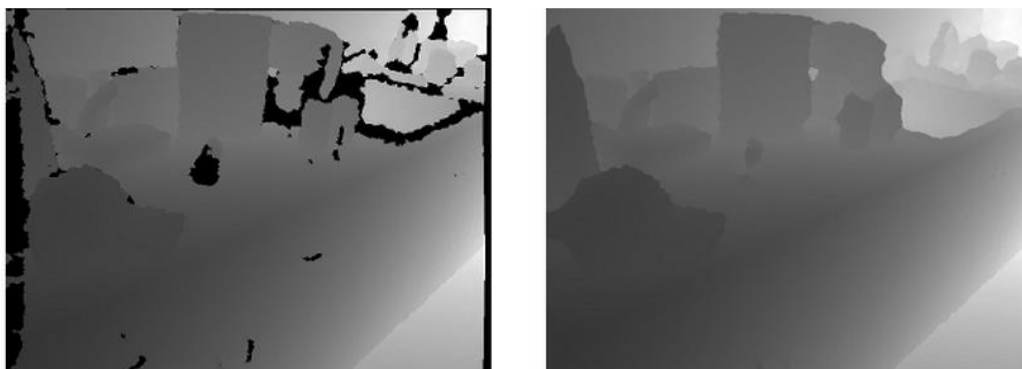
A következő fejezetek során bemutatom részletesen az egyes lépéseket.

5.2 ELŐFELDOLGOZÁS

Az előfeldolgozás az első lépés az algoritmus életciklusa során. Ahogy az RGB, úgy a mélységi adat is tartalmazhat zajokat, ezért a minőségén az első lépésben javítani kell. Sok esetben a probléma visszavezethető a szenzor tulajdonságaira. Számos minta adatbázis során a távolság értékek 2D kép formájában vannak eltárolva, és mivel a normál vektorok kiszámítása bemenetként egy pontfelhőt vár el, bizonyos esetekben ilyen átalakításokat is meg kell ejteni. Elsősorban megvizsgálom a mélységi térképpel kapcsolatos problémákat, majd kitérek az RGB adat korrigálására.

5.2.1 MÉLYSÉGI TÉRKÉP ELŐFELDOLGOZÁSA

Ahogy már korábbi fejezetben is említettük a projekt során (XBOX 360 és Windowsos Kinecthez) Microsoft Kinect szenzorral rögzített adatokon teszteltem az algoritmusaimat. A képkötés során bizonyos részek definiálatlanul maradnak a szenzor fizikai tulajdonságai miatt. Főként a fekete, átlátszó vagy tükröződő objektumoknál a térbeli távolság kiszámítása nem mindig lehetséges. Ezek a pixelek kimaradnának a jellemző számítási folyamatból. Mivel az üres részek összetartoznak, és nem véletlenszerűen maradnak ki, így rekurzívan alkalmazhatok medián szűrőt addig, amíg az összes hiányzó részt nem pótolom [30]. [31] munkájában egy teljesen másik megközelítést alkalmaznak. Feltételezésük szerint a hiányzó értékeknek nagy valószínűséggel a szomszédjainak értékeivel megegyező intenzitásuk van. Legközelebbi szomszédok interpolációval töltik ki a „lyukakat” és, hogy biztosítsák a sima felszínt 4x4-es medián szűrőt alkalmaznak.



23. ábra – A bal oldali mélységi térképen értékek hiányoznak a szenzor fizikai tulajdonságai miatt. A jobb oldali ábrán látható a rekurzív medián szűrővel feljavított verzió [30].

forrás: <http://www.cs.washington.edu/rgbd-dataset/>

5.2.2 RGB KÉP ELŐFELDOLGOZÁSA

Az egyes képek lehetnek zajosabbak, különbözhetnek a megvilágítás mértékében és irányában is. Ilyen körülmények között a jellemző vektorokat nem lehet hatékonyan összehasonlítani. Továbbá az RGB képen ránézés belüli kis különbségek a jellemző vektorok számítása, és osztályozás során nagy eltéréseket implikálhatnak. Színjellelmezők kalkulálása előtt érdemes a képen lévő színek számát csökkenteni azaz kvantálni, vagy a képeket másik színtérbe transzformálni, erre a célra a CIE L*a*b* vagy a HSV színterek megfelelőek [20].

5.3 VIZUÁLIS JELLEMZŐK

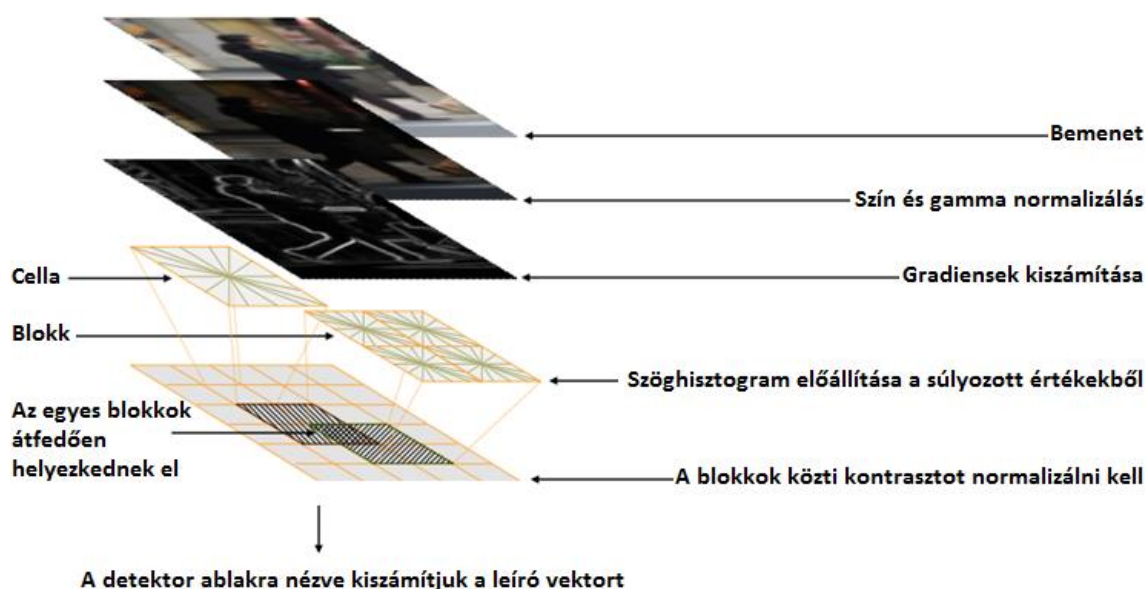
Elérkeztünk a rendszerem egyik alappillérét képező részegységéhez, mely a képek és objektumok tartalmát reprezentáló jellemző vektorok elkészítéséért felelős. Egy objektum egyedi színekkel rendelkezik, melyek különféle elrendezésben jelenhetnek meg. Egy teljes képhez viszonyítva azonban jóval kevesebb színt tartalmaz, így könnyebb megállapítanom azt a pár domináns színt, ami jellemző az adott objektumra. A textúra, és az alak szintén diszkriminatív leírást ad. A mélységi térkép pedig lehetőséget nyújt az egyes objektumok felületeinek elemzésében és térbeli elhelyezkedésében. Nagyon fontos feladat a diszkriminatív jellemzők összességének megtalálása és széleskörű tesztelése.

Az Irodalomkutatás című fejezetben láthattunk már példákat más rendszerektől bizonyos jellemző kinyerési módszerekre. A fejezetben bemutatom, milyen jellemzőket használtam fel a saját rendszerem elkészítése során. A módszereket négy csoportba lehet sorolni: szín, él, felület, mozgás. Végül kitérek a 3D-s objektumok leírásának lehetőségeire.

5.3.1 ÉL ÉS TEXTÚRA JELLEMZŐK

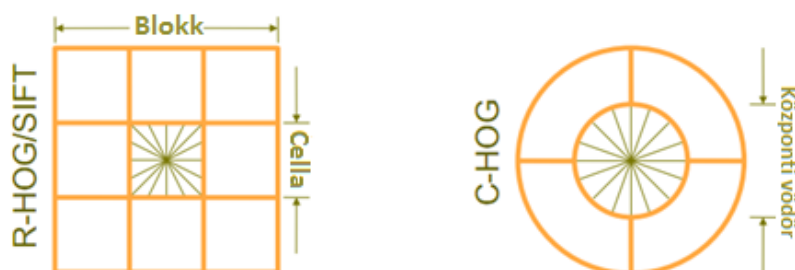
Korábbi projektem [1] során sikeresen implementáltam az MPEG-7 szabvány által ajánlott élhistrogram leíró ("Edge Histogram Descriptor" – továbbiakban EHD), mely a képen található lokális élek irányultsága és helyzete alapján épít histrogramot. A histrogram a leginkább alkalmazott struktúra, amit globális jellemzők ábrázolására használnak, innen származik az ötlet, hogy az élek irányultságát is így ábrázolják. A következőkben az EHD problémáit kiküszöbölő gradiens irány histrogram leíró ("Histogram of Oriented Gradients" – továbbiakban HOG [32]) és alak kontextus ("Shape Context" [34]) módszereket fogom bemutatni.

A HOG esetén régiókra bontom a képet, majd ezekhez olyan irányhistrogramokat rendelek, melynek vödreit az oda tartozó gradiensek nagyságaival vannak súlyozva. Vizsgáljuk meg közelebbről is a részletes mechanizmust, melyet a 24. ábra szemléltet.



24. ábra – A HOG jellemző előállításának szintjei [32].

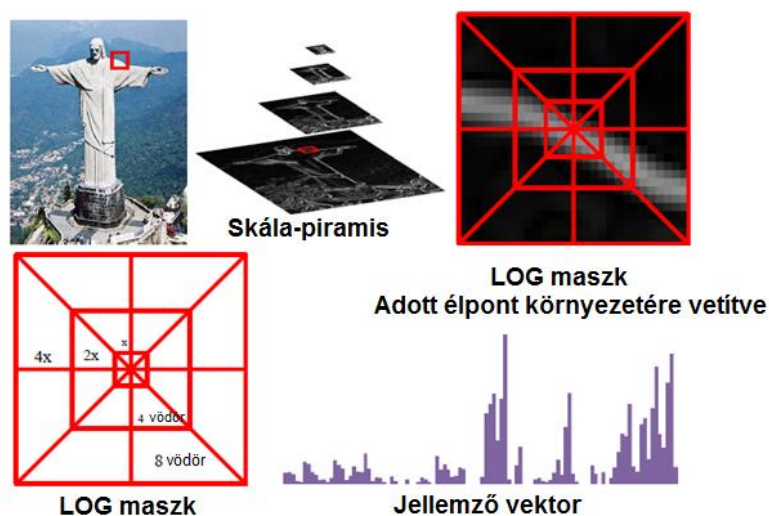
Az első lépésben végre kell hajtani egy gamma vagy színnormalizálást. A saját tapasztalatok és a szakirodalomban megjelenő tesztek során kiderült [33], hogy ez csak kis mértékben befolyásolja pozitív irányban a teljesítményt. A második lépésként éleket kell detektálnom. Több publikáció szerint is a leghatékonyabb az egydimenziós $[-1 \ 0 \ 1]$ szűrőmaszk $\sigma=0$ paraméterezéssel [17, 33]. A harmadik lépésben felosztom a képet – az EHD-nál használt módszertől eltérően – átfedő blokkokra, azokat pedig cellákra. A cellákhoz egy hisztogramot kell rendelni, melyet úgy állítok elő, hogy a pixelekhez kiszámítjuk a gradiens értékeket és az adott irányú vödröket nem egységnyiivel növelem, hanem a gradiens nagyságával súlyozottan. A vödrök a $0^\circ\text{--}180^\circ$ tartományban 20 fokként vannak előjel nélkül. Egy másik megoldás lehet, ha a teljes $0^\circ\text{--}360^\circ$ tartományt használom fel, ekkor azonban az előjeleket figyelembe kell venni. Az említett cellák lehetnek négyszögletesek, illetve kör alakúak (lásd 25. ábra). A negyedik lépésben normalizálást kell végrehajtani. A gradiens nagyságok igen széles skálán mozoghatnak, melyben szerepet játszik a megvilágítás, illetve az előtér és háttér kontrasztja. Ahhoz, hogy azonos léptékkal kezelhessük a különböző nagyságrendű értékeket, elengedhetetlen a normalizálás. A jó teljesítmény elérésének ez egy kulcs momentuma. A legjobb eredményeket az $L2$ norma használata által érhetem el [17].



25. ábra – A HOG esetében használt blokk típusok [32].

Ezt követően elő kell állítani a jellemző vektort az úgynevezett detektorablakra nézve. Ez egy, a kép méreténél általában kisebb, a kép egy részét lefedő képrészlet. A végső vektor mérete függ a paraméterezéstől. Megadhatjuk a cella méretét, ami tulajdonképpen az egy cellába eső pixelek száma. A blokkok tulajdonságait is lehet befolyásolni, úgymint méret, átfedés mértéke, normalizálás módja. Végül a hisztogram jellemzőit is beállíthatjuk, pl. vödrök száma, előjeles értékek használata, súlyozás módja. A különféle paraméterek értékei függenek a konkrét megoldandó feladat jellegétől. A HOG módszert általában egy adott alakú objektumdetektálására szokás használni valamilyen lineáris osztályozóval – általában SVM – kiegészítve.

Egy szintén sokat felhasznált módszer a lokális alak kontextus kontúr leíró [34]. Adott élpont relatív környezetét írja le. Az első lépésben egy Gauss-piramist kell felépíteni a pont adott környezetéről és minden szinten Canny éldetektálást kell végrehajtani. Ennek okán a leírás a továbbiakban skála független lesz. Minden szintre meg kell határozni a jellemző vektort. Egy kép során több lokális környezetre is kiszámolunk majd ilyen vektorokat, így érdemes az egész kép alapján Gauss-piramisokat építeni. Az adott pontra egy maszkot illeszt, mely meghatározza a végleges hisztogram formáját. Egy élpixel az erősségével számít bele a leírásba, továbbá az él kvantált irányával. A külső körön elhelyezkedő vödrök esetén az éleket 8, egyel beljebb 4, a legbelső esetén pedig 1 alvödörbe osztja. Eredményül egy $8 \times 8 + 4 \times 8 + 1 \times 8 = 104$ elemű leíró keletkezik. A leírot röviden összefoglalja a 26. ábra.



26. ábra – A kontextus kontúr leíró vázlatos működése [34].

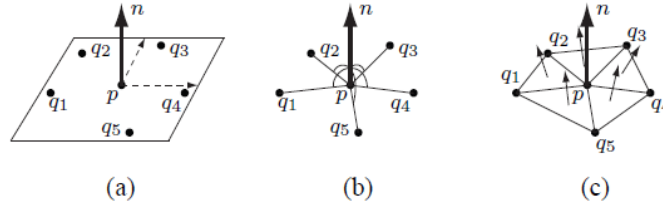
5.3.1 SZÍNJELLEMZŐK

Amennyiben hatékony szegmentáló rendszert szeretnénk elkészíteni, kihagyhatatlanok a szín alapú jellemzők. Annak ellenére, hogy számos esetben hatékonyan tudtak ilyen leírások alapján alkalmazásokat építeni, számos probléma merül fel, amikor egy objektum színéről beszélünk. Sokszor a képalkotó szenzor, a külső környezet fényhatásai beszűkítik a lehetséges módszerek számát. Az RGB kép előfeldolgozása című fejezetben leírtak alapján előfeldolgozható a kép. A másik lehetőség, hogy az alkalmazással kapcsolatban alapvető feltételezéssel élünk, vagy kialakítunk egy megfelelő környezetet (képek elkészítési módjának pontos meghatározásával és a külső környezeti változások minimálisra csökkentésével).

[34] ajánlása szerint transzformáljuk át a képet CIE $L^*a^*b^*$ színtérbe és használunk fel egyszerű lokális leírókat, mint a csatornánként vett középértéket és szórást (megmutatja továbbá a textúrázottság mértékét is). Véleményem szerint az előző (Él és textúra jellemzők) fejezetben említett alak kontextus leíró is lehet adoptálni, úgy, hogy az élek helyett színelfordulásokat tárolunk. Ilyenkor a tanítóminták alapján kell építeni egy szótárat, hogy a képeken előforduló színeket n darab osztályba csoportosítsuk.

5.3.2 FELÜLETJELLEMZŐK

Egy objektumra tekinthetünk úgy is, mint bizonyos tulajdonságú felületek összességére. A pontfelhő segítségével ki lehet számolni az egyes felületekre ható normál vektorokat, így megkülönböztethetővé tesszük a felületeket. [35] alapján három módszer létezik a normálvektorok számítására (lásd 27. ábra).



27. ábra – A főbb felületjellemző kalkulálási módszerek mögött meghúzódó ötletek [35]:

a, Egy síkot próbálnak p és környezetére illeszteni.

b, A normálvektor és a tangenciális vektor közötti szöveget maximalizálják.

c, A pontokból háromszögeket formálnak és az egyes háromszögekre kiszámolt normálvektorok átlagát veszik.

A legegyszerűbb módszer hogy egy pontnak vesszük a szomszédos pontjait, háromszögeket formálunk, (mint a 27. ábra c részén látható) és minden háromszögre meghatározott normálvektort átlagolunk. Az implementálása gyors, azonban az eredmény pontatlan lesz. A klasszikus módszer a pontokra egy síkot próbál illeszteni, és ennek a síknak a normálvektora lesz az adott pontból mutató vektor. Ez egy tipikus minimalizálási problémára vezethető vissza. A saját rendszerem során is hasonló megoldással élek.

5.4 OSZTÁLYOZÁS

A rendszerem számos pontján szükségem van vektorok osztályozására. Ez fontos a BOW szótár építésénél, vagy a magas szintű jellemzők csoportosításánál is. Az alfejezetben belül bemutatom a felhasznált módszereket.

5.4.1 GAUSS-KEVERÉKES MODELL (GAUSSIAN MIXTURE MODEL)

Vizsgáljuk meg először a Gauss-keverékes modell definícióját [36] alapján. A modell egy paraméteres sűrűségfüggvény, mely súlyozott Gauss eloszlások sűrűségfüggvényeinek összege. Magát a paramétereket iteratív módon a tanító adatok alapján számolják. A képfeldolgozáson kívül gyakran használják biometriai rendszerek (hangfelismerés) megvalósítása során [36]. Vezessük be a következő matematikai jelöléseket:

$$p(x|\lambda) = \sum_{i=1}^M w_i g(x|\mu_i, \Sigma_i)$$

Ahol x egy D dimenziós vektor (például jellemző vektor), $w_i, i = 1, \dots, M$ a keverés súlyok és kielégítik a $\sum_{i=1}^M w_i = 1$ feltételt. A paraméterek összességét jelöli az $\lambda = \{w_i, \mu_i, \Sigma_i\}$, ahol $i = 1, \dots, M$. Mindegyik komponens a következő alakú Gauss-eloszlás:

$$g(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2} |\Sigma_i|^{1/2}} e^{\left(-\frac{1}{2}(x-\mu_i)' \Sigma_i^{-1} (x-\mu_i)\right)}$$

Amikor is μ_i a középpérték vektor, a kovariancia mátrix pedig Σ_i . A fenn leírt paraméterek meghatározzák a Gauss-keverékes modellt. Ha ezt egy bemenő tapasztalati adatsorra meghatározzuk, akkor tudjuk használni osztályozó algoritmusként. Adott T darab megfigyelés

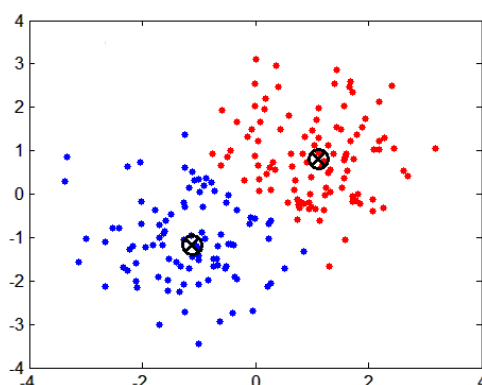
(esetünkben lokális jellemző vektor) $X = \{x_1, \dots, x_T\}$. A cél λ meghatározása, hogy az a legjobban közelítse X vektorok eloszlását, tehát a következő kifejezést kell maximalizálni:

$$p(X|\lambda) = \prod_{t=1}^T p(x_t|\lambda)$$

Mivel λ paraméterre nézve ez egy nem lineáris függvény, így direkt maximalizálás nem hajtható végre. Azonban avárható érték maximalizálásának elvével („Expectation Maximization” - EM) iteratív módon meghatározható a modell [37]. Az első lépésben inicializálni kell a kezdő modellt. A következőekben egy i . iterációban megtalált paraméterek alapján finomítják tovább a modellt, amíg egy adott küszöbnél nem kisebb a $p(X|\lambda)$ érték változása. Az algoritmus bemente az eloszlások száma a keveréshez és a küszöbszám.

5.4.2 K-KÖZÉP (K-MEANS)

A K-közép klaszterezés (lásd 28. ábra) egy iteratív algoritmus, egyes szakirodalmak Lloyd algoritmusnak nevezik [38]. A működést röviden a következő pár mondatban foglaljuk össze. Az első lépésben véletlenszerűen kiválasztjuk a klaszterek központi elhelyezkedését. Minden adatpontot elhelyezünk a megfelelő klaszterbe, az egyes klaszter-középpontoktól való minimális távolság alapján. Ezek után a középpontok elhelyezkedését úgy módosítjuk, hogy pont az egyes klaszterekhez tartozó vektorok súlypontja legyen. Az előző két lépést addig iteráljuk, amíg a súlypontok elmozdulnak. A módszer problémája, hogy lassú (a soros verzió lassú, jól párhuzamosítható), illetve előre meg kell adni az osztályok számát.



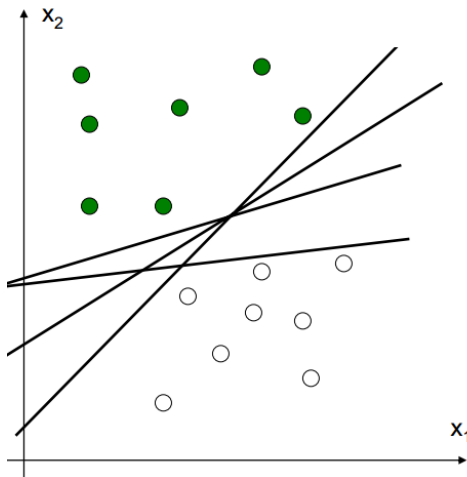
28. ábra – A K-közép algoritmus végeredményének szemléltetése (2 dimenziós esetben). Azonos színű pontok egy osztályba tartoznak. Az osztályokat a feketével jelzett körök reprezentálják [38].

[21, 24, 25, 26] is gyakran használt módszerként említi meg a K-Közép algoritmust BOW szótár generálására, azonban én [21] által preferált Gauss-keverékes algoritmussal állítok elő BOW szótárt. Sajnos a módszer jósága függ a kezdő Gauss-keverék eloszlások paramétereitől. [37] ajánlása szerint a lokális jellemzőkön végrehajtott K-Közép eredménye alapján inicializálom a kezdő paramétereket.

5.4.3 TÁMASZTÓ VEKTOR GÉP (SUPPORT VECTOR MACHINE)

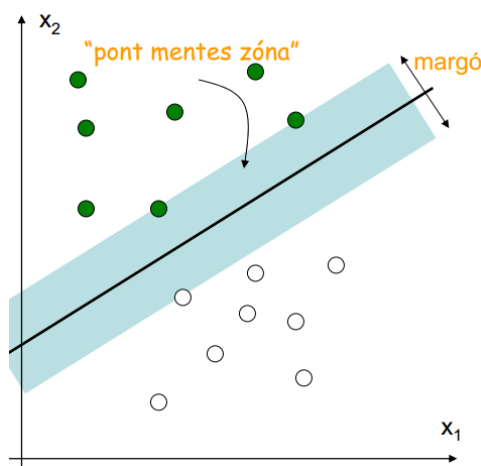
Az objektumokat jellemző vizuális vektorokat a támasztó vektor gép segítségével („Support Vector Machine” – továbbiakban SVM) tanítom be és osztályozom. Az SVM egy lineáris osztályozó, mely

a vektor teret úgynevezett hipersíkkal két részre osztja [39]. A pont hovatarozása attól függ, hogy a hipersík melyik felén helyezkedik el. Mivel sokféleképpen meg lehet ezt a síkot határozni, felmerül a kérdés: Hogyan választom ki ezt a sok lehetőség közül, mit vizsgáljak meg? A legjobb sík az, amelyiknek a margója a legnagyobb. Az így paraméterezett osztályozó általánosít a legjobban. A 29. ábra mutat pár lehetséges margót, a 30. ábra pedig az optimális megoldást mutatja meg.



29. ábra – Számos margót választhatunk ki.

forrás: <http://www.inf.u-szeged.hu/~ormandi/ai2/10-SVM.pdf>



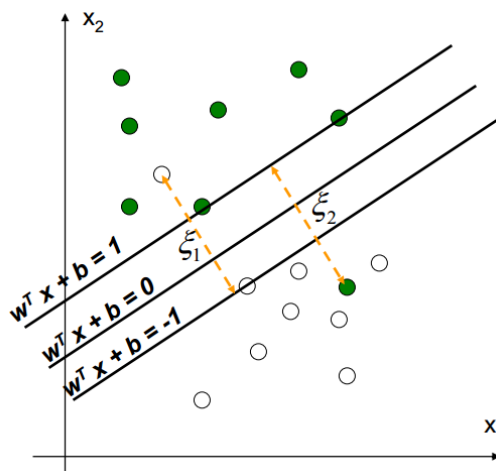
30. ábra – Az SVM azzal a hipersíkkal választja szét az adatokat, amelyeknek a margója maximális.

forrás: <http://www.inf.u-szeged.hu/~ormandi/ai2/10-SVM.pdf>

Mi történjen, ha a tanító adatbázis nem szeparálható (lásd 31. ábra)? Ebben az esetben a következő kifejezést kell minimalizálni a megoldás érdekében [39]:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^l \xi_i$$

$$y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i \text{ és } \xi_i \geq 0$$



31. ábra – Az adatok nem szeparálhatók, csak bizonyos hibaküszöb mellett.

forrás: <http://www.inf.u-szeged.hu/~ormandi/ai2/10-SVM.pdf>

C paraméterrel lehet szabályozni az általánosítás mértékét. Ha C kicsi, akkor a margó maximalizálása a cél, de ilyenkor nagyobb a hiba. A nem lineáris összefüggések kezelésére pedig felhasználjuk a kernel trükköt (ϕ függvény segítségével). Kernel-függvénynek nevezik a következőt:

$$K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$$

Az SVM egy lineáris osztályozó, azonban nem csak bináris problémák elkülönítésére lehet felhasználni, hanem kiegészíthető több osztályos klasszifikálásra is. Az egyik lehetőség az egy az összes ellen, tehát mindegyik SVM próbálja szétválasztani az adott klasztert a többitől lineárisan [40]. Az inputot az alapján csoportosítják, hogy melyik SVM tudja legnagyobb biztonsággal az adott helyre sorolni. Másik lehetséges megvalósítás az egy az egy ellen SVM [41]. Ilyenkor k osztályos tanítás során, $\frac{k(k-1)}{2}$ SVM osztályozót hoznak létre és mindegyik két osztályt szeparál, hogy minden osztályt minden osztállyal szemben klasszifikálhassanak. Egy új vektor (X_n) csoportosítása esetén vesszük az összes bináris osztályzás eredményét. Azt az osztályt fogjuk eredményül kiválasztani, amelyik legtöbbször szerepelt eredményként. Az általunk felhasznált LibSVM osztálykönyvtár megvalósítása is az előbb leírtak alapján (egy az egy ellen SVM – „one against one SVM”) működik [41].

5.4.4 DIMENZIÓK SZÁMÁNAK JELENTŐSÉGE

Az osztályozás végeredményének minőségét nagyban meghatározza a rendelkezésre álló tanítóminták vagy megfigyelések számossága és az egyes vektorok dimenzionalitása. Hogyan befolyásolják ezek a mennyiségek az eredményt? Ha egy N dimenziós tér elemeit leképezzük egy magasabb dimenzionalitású térbe meghatározott transzformáció során, akkor az egyes osztályokat könnyebben szétválaszthatjuk egymástól. Ezt nevezik kernel trükknek és gyakran használják nem lineáris problémák klaszterezése során [21].

Első ránézésre azt gondolhatjuk, hogyha minél részletesebben írunk le valamit, annál könnyebben különbséget tudunk tenni két elem között. Sajnos figyelembe kell venni a részletesség hátrányát is. [42] szerint a dimenzió növelésével az elvárt tanító minták száma exponenciálisan növekedik,

hogy megfelelő eredményt kapjunk. A jelenségnek a dimenzionalitás átka elnevezést adta. A cél olyan módszer felhasználása, amely az adatok dimenzióját csökkenti, a leírt információtartalom legkisebb torzulásával. Az egyik leggyakrabban használt metódus a főkomponens analízis („Principal Component Analysis” – továbbiakban PCA). A módszer geometriai értelmezése az algoritmusnak, hogy az adatokat egy másik, kisebb dimenziós koordináta rendszerbe transzformálja. [24] szerint a módszer lecsökkenti a számítási kapacitásbeli szükségleteket és lecsökkenti az adatok közti zajokat, melyek a jellemzők mintavételezése során jelentek meg. A PCA módszert sikeresen felhasználtam a projektem során, tapasztalataimat a következő fejezetben mutatom be.

6 TESZTEK ÉS EREDMÉNYEK

A fejezetben belül bemutatom a tesztrendszer, amit a MATLAB, C# és C++ nyelvek segítségével valósítottam meg. A MATLAB segítségével könnyen lehet felhasználni más nyelven megírt függvényeket, vagy programokat. Bizonyos esetben azért tértem el a MATLAB-os beépített függvényektől, mivel nem az a standard a képfeldolgozási közösségben belül. Döntésemet az is indokolja, hogy a felhasznált C# alatt megírt kódok hatékonyabbak is.

6.1 *FELHASZNÁLT SZOFTVEREK*

Ahogy említettem több nyelven készült a tesztprogram. Az alfejezetben belül részletesen bemutatom a főbb felhasznált osztálykönyvtárakat.

6.1.1 *POINT CLOUD LIBRARY (PCL)*

A Point Cloud Library egy BSD liszensz alatt álló, C++ alapú, open source projekt 2D/3D képek és pontfelhők feldolgozására [19]. A könyvtár számos jelenleg élen járó algoritmust implementál a következő témákban: szűrők, jellemzők, felületrekonstrukció, pontok regisztrálása 3D modellben, modellillesztés és szegmentálás. Az RGB-D szenzorok számának, típusainak és felhasználásának növekedésével egyre fontosabb, hogy elérhetőek legyenek a megfelelő feldolgozó könyvtárak, mint a PCL. A következő platformok alatt felhasználható: Linux, MacOS, Windows és Android/iOS. A PCL felhasználásával valósítottam meg a pontfelhő szegmentálását.

6.1.2 *ACCORD.NET*

A .NET keretrendszer alatt C# nyelven megírt osztálykönyvtár tudományos kutatások számára lett fejlesztve, mégpedig statisztikai számításokra, gépi tanulásra, jel és képfeldolgozásra [37]. A NuGet (osztály könyvtár menedzser, <http://docs.nuget.org/>) integrációnak köszönhetően könnyen elérhető bármilyen C# projekt alól. Jól dokumentált, egyszerűen használható és teljesítménye is megfelelő. A Gauss-keverék és dimenzió csökkentés során használtam fel a BOW szótár építésére.

6.1.3 *MATLAB IMAGE PROCESSING TOOLBOX*

A képfeldolgozás terület standardnak tekinthető MATLAB programcsomagot is felhasználtam [43]. Gyakorlatilag az itt megírt program a gerince az egész projektnek. Mivel lehetőség van mind C/C++ kódot MATLAB-ban hívni, így a jellemző kiszámítást ezzel lehetett gyorsítani. A C# komponenst is kényelmesen tudtam használni. Meglátásom szerint érdemes a MATLAB alatt elkészíteni mindig a demóalkalmazást, mivel számos nyelv együttműködését lehetővé teszi, ezzel növelve a programozó produktivitását.

6.1.4 *LIBSVM*

Az SVM osztályozás során a LibSVM-t használtam fel [41]. Jól dokumentált, számos példaalkalmazással demonstrálja az egyes típusú SVM osztályozók használatát. Rendelkezik Python, C++, MATLAB interfésszel, továbbá futtatható állományként a Windows felhasználókat is közvetlenül támogatja.

6.2 TESZT ADATBÁZISOK

Az alfejezetben belül röviden bemutatom a különböző RGB-D tesztadatbázisokat, amiket fel lehet használni szemantikus szegmentálás vagy objektumdetektálás kiértékelésére. Ezek az NYU Depth [44], a Washington RGD-D [30] és a B3DO: Berkeley 3-D Object Dataset [45]. Mindhárom adatbázis RGB-D szenzorral készítették el. Mindegyik során a nyers adatokat kalibrálták és a hiányzó mélységi értékeket feltöltötték. A Washington RGB-D tartalmazza a rekonstruált 3D pontfelhőket is. Az adatbázisokat nem egyszerű felhasználni. Ennek egyik oka, hogy mindegyik nagyon különböző szerkezetű és a felhasználáshoz biztosított szoftverek nagyon eltérőek. A másik probléma az adatbázisok nagy méretével van (NYU Depth > 400 GByte, Washington RGB-D > 100 GByte). Tesztelési céljaimra csak az NYU Depth adatbázist használtam fel.

6.3 AZ ELKÉSZÜLT RENDSZER

Az alfejezetben ismertetem a rendszeremmel kapcsolatos implementációs részleteket és a rendszer jelenlegi állapotát. Az implementált keretrendszert két részre osztanám fel. Az egyik a pontfelhő szegmentálása normálvektorok alapján, a másik pedig az RGB szegmentáló alrendszer. Az egyes részeket a MATLAB fogja össze egészszé különböző megoldásokkal. Az egyik lehetőség, hogy C/C++ Mex fájlokat készítünk, melynek segítségével C/C++ metódusokat hívhatunk, mint beépített MATLAB funkciókat. A funkció akkor is hasznos, ha a megvalósítás során feltétel a futási idő minimalizálása. További lehetőség a platformfüggetlenségre, hogy DOS parancsokat is lehetőség van futtatni, így a lefordított .NET vagy C/C++ alapú exe fájlokat probléma nélkül elindíthatjuk. A program beállításai és az algoritmusok paraméterei konfigurációs fájlokban találhatóak meg, amit mind kívülről, mind pedig kódból lehet módosítani.

6.3.1 PONTFELHŐ SZEGMENTÁLÁSA NORMÁLVEKTOROK ALAPJÁN

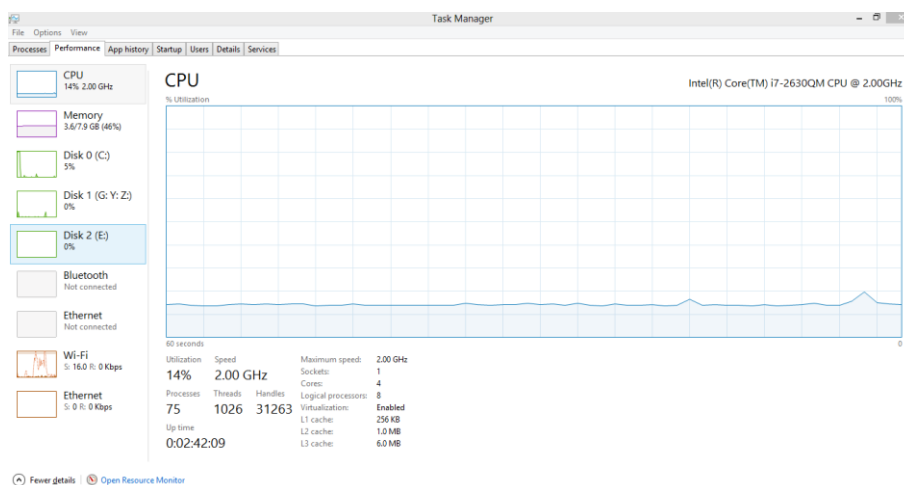
A pontfelhő szegmentálást teljes egészében C++ nyelven a PCL osztálykönyvtár segítségével készítettem el. Öt fő lépésből áll az algoritmus: pontfelhő megnyitása, előfeldolgozás optimalizálási célokkal, normálvektorok számítása és szegmentálás.

Az első lépésben egy pontfelhőt kell betölteni fájlnev alapján. Felmerülhet, hogy miért pontfelhőt használok, ami egy bonyolultabb adatszerkezet, mint az intenzitás képként ábrázolható mélységi térkép. Ha később videó adatokkal dolgozunk, akkor teljes 3D modelleket építhetünk például az RGD-SLAM segítségével [11], melyet elmenthetünk egy pontfelhő fájlba, illetve ilyenkor a jelenet egészére végre tudunk hajtani normálvektor számítást és szegmentálást nagyobb módosítás vagy bonyolult algoritmus nélkül. A nagyobb előny, hogy az eredményt egy kompakt egészben meg lehetne jeleníteni. Hogy ez működjön, el kell tárolnunk, hogy melyik RGB képkockához, melyik indexű pontok tartoznak a 3D felhőből. Ezt már csak azért is kell megtenni, mivel a szegmentálás alapvetően RGB képekről kinyert jellemzők alapján történik. Mivel egy pont a felhőből több RGB képhez is tartozhat, így szegmentálás eredménye miatt különböző képekre egy pont több címkét is kaphat. Itt egyszerűen csak ki kell választani a legnagyobb előfordulású eredményt. Mivel bizonyos adatbázisok csak mélységi térképeket (intenzitás kép) szolgáltatnak, így konverziós metódusokat kell készíteni, mely pontfelhővé alakítja ezeket.

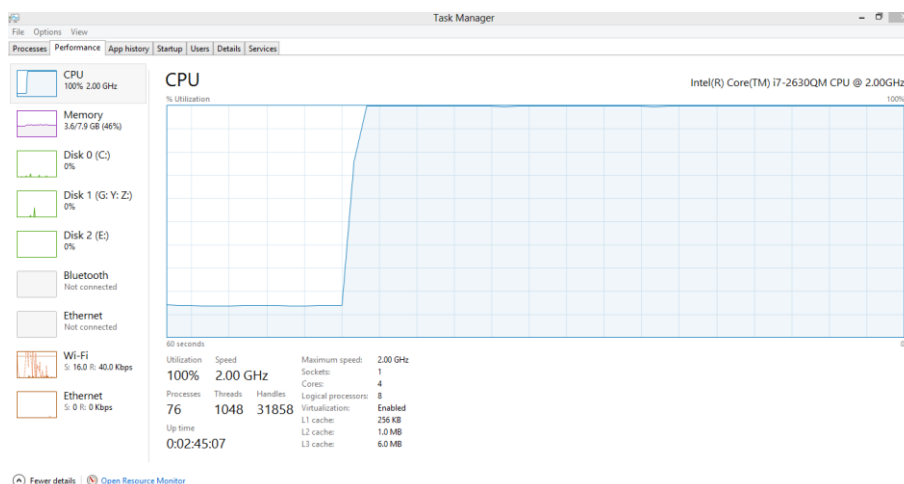
Az előfeldolgozás célja tisztán teljesítményoptimalizálási eredetű. Mikor először leprogramoztam, ezt a lépést hanyagoltam, és így az eredmény nagyon kiábrándító volt, mivel fél óra után sem kaptam értékelhető kimenetet. Először alkalmaztam a Voxelrács szűrőt („VoxelGrid filter”, ahol a voxel egy a tér minden irányában kiterjedéssel rendelkező pixel) [19]. Az algoritmus a 3D

modellben kis dobozokat helyez el rácsszerűen. Az egy dobozba eső pontok súlypontjával közelíti az ott lévő elemeket, így csökkentve a pontfelhő méretét. A közelítés alapja lehetne a dobozok középpontja is, így gyorsítva az algoritmust, azonban a mintavételezés minősége rosszabb lenne. Az eredeti pontfelhőt is meghagyjuk keresőfelületnek. A dobozok méretének megválasztásával lehet állítani a mintavételezés mértékét. Az esetleges megjelenő zajokat statisztikai megfontolású szűrőkkel el lehet távolítani, amennyiben ez iránt igény mutatkozik.

A következő lépésben számolom ki a normálvektorokat a csökkentett elemszámú pontfelhőre, ahol az eredeti adat lesz a keresőfelület. Az algoritmus lassú, mivel sok pontra sorosan meg kell határozni a legközelebbi szomszédjait, és kiszámolni azok alapján a normálvektort. A PCL szerencsére rendelkezik OpenMP API-t használó párhuzamosított verzióval is, aminek segítségével ki lehet használni a többmagos rendszerekben rejlő számítási erőt. A soros és párhuzamos implementáció, processzor kihasználtság különbségeit szemlélteti a 32. és 33. ábra.



32. ábra – A normálvektorok kiszámítása a soros algoritmussal. A processzor kihasználtsága 14%. (4 magos Intel(R) Core(TM) i7-2630QM CPU – 8 logikai mag, 2.00 GHz).

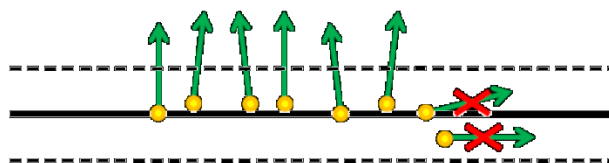


33. ábra – Az OpenMP-vel támogatott normálvektor számítás az inicializálás után 100%-os processzorkihasználtság mellett fut. (4 magos Intel(R) Core(TM) i7-2630QM CPU – 8 logikai mag, 2.00 GHz).

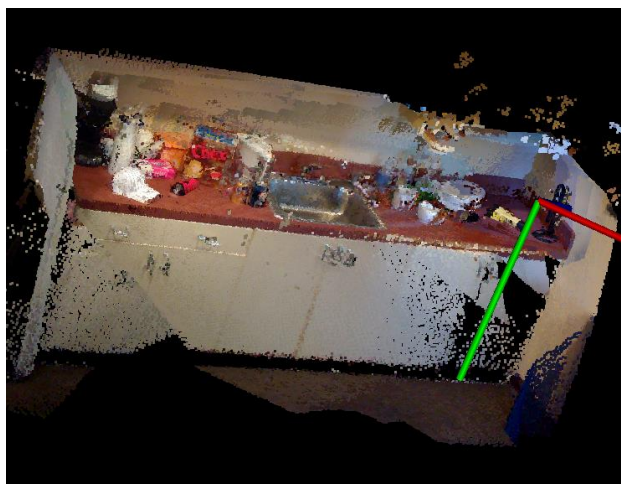
A pontfelhő szegmentálása során PCL osztálykönyvtár a RANSAC módszerét használtam fel. Ez egy iteratív algoritmus, melynek segítségével N db megfigyelés részhalmazára illeszthető matematikai modell paramétereit becsülhetjük meg [46]. Első körben a bemeneti adathalmazból kiválaszt egy részhalmazt (belső halmaz), amelyre a modellt illesztjük (angolul „inliers”), a többit kívül eső pontnak tekinti („outliers”). Kiválasztja a kezdeti paramétereket és a külső adatokra is teszteli a hipotézist. Amennyiben kielégítik a modellt, őket is a belső halmaz elemeinek lehet tekinteni. Ezek után a modellt újra kiértékelik, és a paramétereken finomítanak. Ez egészen addig fut, amíg hiba nem csökken egy adott küszöb alá (vagyis a becslés bizonyossága elég magas lesz). A pontfelhő szegmentálás során a következő modellt alkalmazom.

$$ax + by + cz + d = 0$$

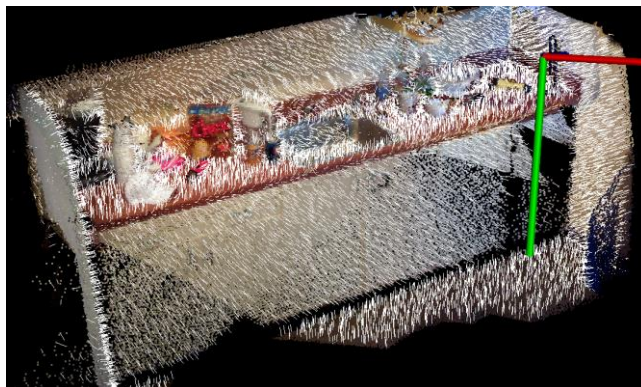
Mikor pontfelhő részhalmazára megfelelően illeszkedik egy matematikai modell, akkor ezeknek a pontoknak adok egy címkét és hozzáadom őket egy másik (kezdetben üres) pontfelhőhöz. A megmaradt adatokra újra próbálom modellt illeszteni, és ezt addig folytatom, amíg az eredeti pontfelhő adott százalékánál még több pont áll a rendelkezésre. Tipikusan ez az érték 5-10% a programom során. Mivel korábban kiszámoltam az egyes pontokból mutató normálvektorokat is, ezért ezeket hatékonyan fel lehet használni. Ha modell által meghatározott sík normálvektora és az adott pont normál vektora közötti elhajlási szög („angular deviation”) nagyobb egy küszöbnél, akkor el kell vetni a pontot, annak ellenére is, hogy egy síkban helyezkednek el (lásd 34. ábra). A 35, 36. és 37. ábra megmutatja a normálvektor alapján működő szegmentálás egyes lépéseit.



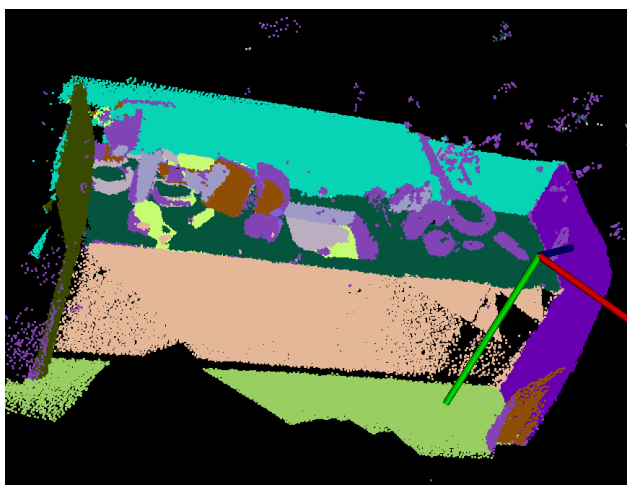
34. ábra – A modell normálvektorával nem eléggé párhuzamos vektorokat hanyagoljuk.
forrás: <http://www.pointclouds.org/assets/files/presentations/ICCV2011-segmentation.pdf>



35. ábra – A 3D pontfelhő betöltése és megjelenítése PCL segítségével.
forrás: saját program



36. ábra – A pontfelhőre ható normálvektorok kiszámítása és megjelenítése PCL segítségével.
forrás: saját program



37. ábra – Az egyes szegmensek elkülönítése és megjelenítése PCL segítségével.
forrás: saját program

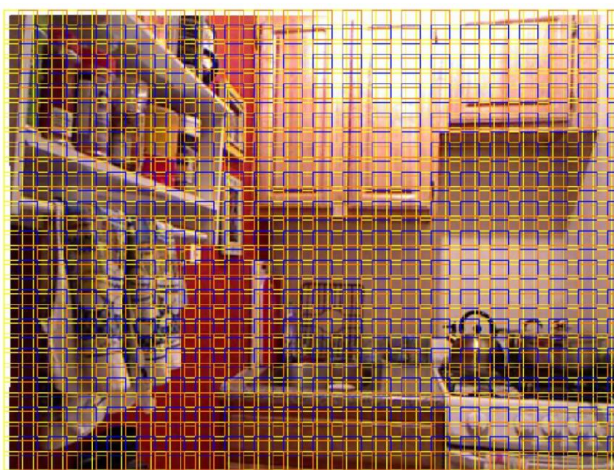
6.3.2 SZEGMENTÁLÓ/CÍMKÉZŐ ALRENDSZER

Ahogy a „A rendszer globális felépítése” kifejtettem a szegmentált pontfelhő egy maszkként fog megfelelni, és az egyes szegmenseket kell felcímkézni és később egyesíteni, ha együtt nagyobb bizonyossággal meghatároznak egy objektumot. Az alfejezetben bemutatom, hogy az egyes szegmenseket mi alapján lehet azonosítani.

Az alrendszert egy tanuló és egy osztályozó fázisra bontottam szét. A magas szintű jellemzőket az Accord .NET osztálykönyvtár segítségével állítottam elő a MATLAB-ban kinyert jellemzők alapján. Az első lépésben a lokális jellemzőket számolom ki. Jelen rendszerben csak a HOG jellemzőket használtam fel. A 38. és 39. ábrán látható a két mintavételezési stratégia. Ahhoz, hogy a kép összes információját leírják a jellemzők, átfedő ablakokra kell kiszámítani a HOGokat (lásd 39. ábra). Ezeket a jellemzőket Gauss-keverékes módszerrel osztályoztam, de előtte a 32 dimenziós HOG jellemzők PCA-val csökkentem. Az elkészült szótár alapján előállítom a magasabb dimenzionalitású vektorokat. Azt, hogy egy vektor melyik szótárbeli szó legyen, az egyes osztályokat képviselő Gauss eloszlások sűrűség függvénye alapján állapítom meg. Mindig a legnagyobb eredmény határozza meg, hogy melyik szóról beszélünk.



38. ábra – Mintavételezés vizualizációja nem átfedő blokkokkal.
forrás: saját program



39. ábra - Mintavételezés vizualizációja átfedő blokkokkal.
forrás: saját program

Az SVM tanítás során nagyon fontos, hogy megfelelő paraméterekkel és megfelelő kernellel inicializáljuk a módszert. [39] ajánlása szerint a radiál bázis függvényt választottam („radial basis function” – továbbiakban RBF kernel). Az adatok és hozzátartozó annotációk betöltése után, a vektorokat $[0,1]$ intervallumra lineárisan skáláztam, mivel a nagy értékekből a magasabb dimenzionalitásba transzformálása és az osztályozása esetén számítási problémák adódhatnak. Mivel RBF kernelnek van két paramétere is, így egy modellválasztásnak kell megtörténnie. [39]-ben írtak szerint a C és γ paraméterek kiválasztására egy grid alapú keresést valósítottam meg. Egyszerűen megkeresem azt a (C, γ) párt, amelyikre a legjobb keresztellenőrzési értéket kapom. (C, γ) értékeket exponenciális növekedéssel vizsgálom, tehát a $\{2^{-5}, 2^{-4}, \dots, 2^4, 2^5\}$ halmazból kerülnek ki az értékek. Megjegyzésként a keresztellenőrzés során a tanító adatokat felosztom v részre. Minden egyes részre megvizsgálom, hogy a $v-1$ elemre történő tanítás milyen hatékonyságú. Ilyenkor az egyetlen rész annotációját nem használom fel.

A tanítás során közel 50%-os keresztellenőrzési rátát sikerült elérni, melynek több oka is van. Először is fontos beszélni arról, hogy csak HOG jellemzőket használtam eddig fel, ami a

környezetben előforduló éleket ábrázolja hisztogram segítségével. Véleményem szerint csak az élinformációk alapján nem lehet megfelelően eldönteni, hogy adott objektumot ábrázoló jellemző vektor melyik osztályba tartozik. A másik oka a rossz teljesítmények az NYU mintaadatbázis. A tesztelés során konyhai képeket használtam fel, és egy-egy objektumról kevés felvétel készült, ezért egy-egy osztályt kevés vektor jellemzett.

7 ÖSSZEGZÉS

A videók kereséséhez, interaktív felhasználásához és intelligens tárolásához nagy szükség van a megfelelő eszközök biztosítására. Az első ilyen lépés a témában az automatikus szegmentáló algoritmusok fejlesztése és tesztelése. Ehhez szorosan kapcsolódnia kell az annotációs eljárások fejlődésének is. Számos alkalmazási terület és piaci ré áll nyitva a kutatók előtt. A dolgozat „Kitűzött célok és felhasználási lehetőségek” alfejezetében számos hasznos alkalmazást megemlítettem, mint légi felvételek szemantikus szegmentálását, videók annotálását reklámokkal vagy a duplikátum és jogtalan tartalmak szűrését videó megosztó portálokon. Az elsődleges célom a videók címkézéssel kapcsolatban a szegmentálási lehetőségek megvizsgálása volt.

A célkitűzésem egy szegmentáló rendszer építése volt, amely mind szemantikus szegmentálásnál felhasznált, mind pedig az általános objektumdetektálásnál alkalmazott technikákat ötvözte. A projekt során RGB-D adatokkal dolgoztam, azért, hogy megmutassam, mennyi potenciál rejlik az ilyen szenzorokban, szenzorrendszerekben.

A projektem során összeállítottam egy keretrendszert, amit a jövőben bővíteni fogok további jellemző leírási módokkal is. Három különböző nyelvet is felhasználtam, aminek nagyon pozitív előnyeit élveztem. Így egy projekten belül tudtam alkalmazni különböző nyelven megírt korszerű osztálykönyvtárakat. Ezáltal produktívabban tudtam dolgozni. Implementáltam a Gauss-keveréken alapuló BOW módszert, ahol rács alapú lokális jellemző mintavételezést alkalmaztam. A vektorok dimenzionalitásából adódó problémákat pedig PCA segítségével mérsékeltem. A BOW módszerem a későbbiek során a Fischer kernel használatával szeretnék kiegészíteni [21]. A rendszer két fő részből áll, ezeket külön-külön megvalósítottam, de még nem illesztettem őket össze.

Egyre fontosab, hogy több és több példaadatbázis legyen elérhető, mely RGB-D jeleneteket tartalmaz. Mivel mindegyik adatbázis különböző struktúrába van szervezve, így sokat kell azzal foglalkozni, hogy egységes formára hozzuk az adatokat. Célszerű lenne a témában standardizálni.

8 IRODALOMJEGYZÉK

- [1] **B. Szántó, P. Pozsegovics, Z. Vámosy, S. Sergyán**, "Sketch4match — Content-based image retrieval system using sketches," *Applied Machine Intelligence and Informatics (SAMII), 2011 IEEE 9th International Symposium on*, Herlány, Slovakia pp. 183-188, 27-29 January 2011.
- [2] **H. Weiming, X. Nianhua, L. Li, Z. Xianglin, S. Maybank**, "A Survey on Visual Content-Based Video Indexing and Retrieval," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, Vol.41, No.6, pp.797-819, November 2011.
- [3] **S. W. Chee, K. P. Dong, P. Soo-Jun**, "Efficient Use of MPEG-7 Edge Histogram Descriptor" *ETRI Journal on*, Vol. 24, No. 1, pp. 23-30, 2002.
- [4] **Synchronized Multimedia Integration Language** – Description of the SMIL 3.0. [Online]. <http://www.w3.org/TR/SMIL3/>, utolsó látogatás: 2012.09.05.
- [5] **Dublin Core Metadata Initiative** – Making it easier to find information [Online]. <http://dublincore.org/metadata-basics/>, utolsó látogatás: 2012.09.05.
- [6] **P. Over, G. Awad, J. Fiscus, A. F. Smeaton, W. Kraaij, G. Quénot**, "TRECVID 2011 -- An Overview of the Goals, Tasks, Data, Evaluation Mechanisms and Metrics," *Proceedings of TRECVID 2011 (to appear)*, December 2011.
- [7] **C. G. M. Snoek, M. Worring, O. de Rooij, K. E. A. van de Sande, R. Yan, A. G. Hauptmann**, "VideOlympics: Real-Time Evaluation of Multimedia Retrieval Systems," *IEEE MultiMedia on*, Vol. 15 No. 1, pp. 86-91, January 2008.
- [8] **B. Fröhlich, J. Denzler**, "Semantic Segmentation Algorithms for Aerial Image Analysis and Classification," *One Day BMVA Symposium at the British Computer Society on Aerial Image Analysis & Classification*, pp. 1-2, 2010.
- [9] **D. Stojcsics, A. Molnár**, "AirGuardian UAV hardware and software system for small size UAVs," *International Journal of Advanced Robotic Systems*, Croatia, 2012, ISSN 1729-880 6 (megjelenés alatt).
- [10] **C. Cigla, A. A. Alatan**, "Segmentation in multi-view video via color, depth and motion cues," *Image Processing, ICIP 2008. 15th IEEE International Conference on*, pp. 2724-2727, 12-15 October 2008.
- [11] **H. Koppula, A. Anand, T. Joachims, A. Saxena**, "Semantic Labeling of 3D Point Clouds for Indoor Scenes," *Conference on Neural Information Processing Systems (NIPS)*, 2011.
- [12] **L. Shang, L. Yang, F. Wang, K. Chan, X. Hua**, "Real-time large scale near-duplicate web video retrieval," *In Proceedings of the international conference on Multimedia (MM '10) ACM*, New York, NY, USA, pp. 531-540. 2010.

- [13] Google Operating System – Google Project Glass [Online]. <http://googlesystem.blogspot.hu/2012/04/googles-project-glass.html>, utolsó látogatás: 2012.11.10.
- [14] **M. Petkovic**, "Content-based Video Retrieval", *VII. Conference on Extending Database Technology (EDBT), Ph.D. Workshop*, Konstanz, Germany, March 2000.
- [15] **P. Geetha, V. Narayanan**, "A Survey of Content-Based Video Retrieval," *Journal of Computer Science*, Vol. 4, No. 6, pp. 474-486, 2008.
- [16] **A. G. Hauptmann, M. G. Christel, Y. Rong**, "Video Retrieval Based on Semantic Concepts," *Proceedings of the IEEE*, Vol. 96, No. 4, pp. 602-622, April 2008.
- [17] **F. Suard, A. Rakotomamonjy, A. Bensrhair, A. Broggi**, "Pedestrian Detection using Infrared images and Histograms of Oriented Gradients," *Intelligent Vehicles Symposium*, pp. 206-212, 2006.
- [18] **E. Rodner**, "Learning from Few Examples for Visual Recognition Problems," *PhD Dissertation*, Friedrich-Schiller-Universität Jena, pp. 1-209, December, 2011.
- [19] **R.B Rusu, S. Cousins**, "3D is here: Point Cloud Library (PCL)," *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp.1-4, 9-13 May 2011.
- [20] **B. Alexe, T. Deselaers and V. Ferrari**, "What is an object?," *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10)*, 2010.
- [21] **B. Fröhlich**, "Semantic Segmentation," *Diploma Thesis*, Friedrich-Schiller-Universität Jena, pp. 1-79, October 2009.
- [22] **T. Berners-Lee, M. Fischetti**, "Weaving the Web : The Original Design and Ultimate Destiny of the World Wide Web by its Inventor," Harper San Francisco, September 1999.
- [23] **G. Csurka**, "BOV and Fisher Vectors in Large Scale Image Classification and Retrieval" *19th Summer School on Image Processing*, Szeged, Hungar, July, 2011.
- [24] **H. Nakayama, T. Harada, Y. Kuniyoshi**, "Dense sampling low-level statistics of local features," *In Proceedings of the ACM International Conference on Image and Video Retrieval (CVIR '09)*, No. 17, pp. 1-8, New York, 2009.
- [25] **C. Dance, J. Willamowski, L. Fan, C. Bray, G. Csurka**, "Visual categorization with bags of keypoints," *ECCV International Workshop on Statistical Learning in Computer Vision*, Prague, 2004.
- [26] **M. Eitz, R. Richter, K. Hildebrand, T. Boubekur, M. Alexa**, "Photosketcher: Interactive Sketch-Based Image Synthesis," *Compute Graphics and Applications, IEEE*, Vol. 31, No. 6, pp. 56-66, November-December 2011.
- [27] **A. Lakdashti, H. Ajourloo**, "IRFUM: Image Retrieval via Fuzzy Modeling" *Computing and Informatics*, Vol. 22, pp. 1001-1029, 2003.

- [28] **F. Perronnin, L. Yan, J. Sánchez, H. Poirier**, "Large-scale image retrieval with compressed Fisher vectors," *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3384-3391, 13-18 June 2010.
- [29] **H. Xiaodi, Z. Liqing**, "Saliency Detection: A Spectral Residual Approach," *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp.1-8, 17-22 June 2007.
- [30] **K. Lai, B. Liefeng, R. Xiaofeng, D. Fox**, "A large-scale hierarchical multi-view RGB-D object dataset," *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1817-1824, 9-13 May 2011.
- [31] **X. Lu, C. Chia-Chin, J.K. Aggarwal**, "Human detection using depth information by Kinect," *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pp.15-22, 20-25 June 2011.
- [32] **D. Navneet, B. Triggs, C. Schmid**, "Histogram of Oriented Gradients (HOG) for Object Detection," 2009.[Online]
http://vision.stanford.edu/teaching/cs223b_winter0910/lecture/HoG.pdf , utolsó látogatás: 2012.11.10
- [33] **D. Navneet, B. Triggs**. "Histograms of Oriented Gradients for Human Detection" *In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1, pp. 886-893, 2005.
- [34] **M. A. Johnson**, "Semantic Segmentation and Image Search," *PhD Dissertation, University of Cambridge*, pp. 1-171, April 2008.
- [35] **K. Klasing, D. Althoff, D. Wollherr, M. Buss**, "Comparison of surface normal estimation methods for range sensing applications," *Robotics and Automation, 2009. ICRA '09, IEEE International Conference on*, pp. 3206-3211, 12-17 May 2009.
- [36] **D. Reynolds**, "Gaussian Mixture Models," *MIT Lincoln Laboratory*, pp. 1-5
- [37] **C. R. Souza**, "The Accord.NET Framework," April 2012. [Online].
<http://accord.googlecode.com> , utolsó látogatás: 2012.11.10.
- [38] **A. Kaehler, G. Bradski**, "Learning OpenCV," *O'Reilly Media*, 2008.
- [39] **C. W. Hsu, C. C. Chang, C. J. Lin**, "A practical guide to support vector classification," *Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan*, 2003.
- [40] **A. Ben-Hur, J. Weston**, "A User's Guide to Support Vector Machines Data Mining Techniques for the Life Sciences," *Springer* (2009).
- [41] **C. Chih-Chung, L. Chih-Jen**, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol*, Vol. 2, No. 3, pp. 27 May 2011.
- [42] **R. Bellman**, "Adaptive Control Process: A Guided Tour," *Princeton University Press*, 1961.

[43] The MathWorks Inc. Image Processing Toolbox.[Online]. <http://www.mathworks.com/products/image/> , utolsó látogatás: 2010.11.10.

[44] **N. Silberman, R. Fergus, D. Hoiem, P. Kohli**, "Indoor Segmentation and Support Inference from RGBD Images," *ECCV*, 2012.

[45] **A. Janoch, S. Karayev, Y. Jia, J.T. Barron, M. Fritz, K. Saenko, T. Darrell**, "A Category-Level 3-D Object Dataset: Putting the Kinect to Work," *ICCV Workshop on Consumer Depth Cameras in Computer Vision*, 2011.

[46] **M. A. Fischler, R. C. Bolles**, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Comm. of the ACM*, Vol. 24, pp. 381-395, 1981.