Hallgató neve: **Kiss Árpád**

Törzskönyvi száma: T 1082/FI38878/N

A dolgozat címe:

MODERN KÖNYVKATALÓGUS

Modern book catalog

A feladat

Tervezzen meg és implementáljon egy olyan mobilalkalmazást, amely egy könyv borító-járól készített fotója alapján képes azt kikeresni az adatbázisban tárolt könyvek között, továbbá képes a könyvhöz tartozó további információkat letölteni az internetről. Az alkalmazáshoz szükséges egy webes háttérrendszer fejlesztése is, hogy a dokumentumok bővíthetősége biztosítva legyen. A dokumentumok tárolásához alkalmazzon séma nélküli adatbázist. A rendszerben alkalmazzon valamilyen adatbányászatban alkalmazott megoldást a döntési tér szűkítésére.

A dolgozatnak tartalmaznia kell:

- a képi adatbázisok általános megközelítéseit, problémáit,
- a könyvborító felismeréshez szükséges lépéseket,
- a felismeréshez felhasználható képi jellemzők ismertetését,
- optikai karakterfelismerés bemutatását és alkalmazását,
- a legmegfelelőbb módszer kiválasztását, és a rendszer tervezésének lépéseit,
- részletes működési tervet és az implementáció bemutatását,
- valamint a rendszert alkotó modulok tesztelési eredményeit, jóságuk értékelését.

TARTALOMJEGYZÉK

1. ABSZTRAKT

A projekt célja egy olyan rendszer megtervezése és kifejlesztése volt, amely a könyvtárak látogatóinak kínál egy újszerű, hatékony módot az ott tárolt dokumentumok háttér-információinak megszerzésében, egy könyv borítójáról készült fotó alapján történő keresés által. A rendszer a képfeldolgozás, a gépi tanulás és az adatbányászat eszközeit alkalmazza.

A dolgozat részletesen ismerteti a könyvborítók felismeréshez szükséges előfeldolgozási algoritmusokat, a jellemzők kinyerésének módjait és leírókat, továbbá a felismerés során alkalmazható felügyelt és felügyelet nélküli gépi tanulókat. Kifejti a képi jellemzők adatbázisban tárolásának és kereshetőségének problematikáját, a keresés gyorsításának több megközelítési módját. Bemutatja az optikai karakterfelismerés általános megközelítéseit, az utófeldolgozás során felmerülő megoldandó problémákat, illetve lehetséges megoldásait.

A rendszer magja C++ nyelven íródott, a hálózati kommunikációt egy Node JS alapokon készült REST API kezeli, amelyhez egy Objective-C nyelven íródott mobil kliens tartozik. A könyvekhez tartozó meta adatok séma nélküli JSON formátumban kerülnek tárolásra, a leírók a fájlrendszeren tárolódnak, XML formátumban. A rendszer OpenCV keretrendszert használ egyes képfeldolgozási algoritmusokhoz, az optikai karakterfelismerés Tesseract keretrendszer felhasználásával történik.

ABSTRACT

My main goal in this project was to design and develop a system, which can provide a new, efficient way for library's visitors to discover the stored documents through a photo based search method. The system is applying the tools of image processing, machine learning and data mining.

This paper is deeply describing the preprocessing algorithms, which are used for the book cover recognition, multiple approaches to gain feature vectors, and as well the supervised and unsupervised machine learners. Explains the problems of storing and searching pictorial features, shows more method for speeding up the process. Presents the general approaches of optical character recognition, problems of post-processing and possible solutions.

The core of the system was written in C++, the network communication managed by a Node JS based REST API, which have an Objective C written mobile client application. The metadata of the books are stored in schema-less JSON format, the features are stored on the file system in XML format. The system is using Open CV for certain image processing algorithms, the optical character recognition based on Tesseract framework.

2. BEVEZETÉS

2.1 Célkitűzés

A megvalósított alkalmazás célja az eddigi, hagyományosan beviteli mező alapú könyvtári katalógusban történő keresés kiterjesztése egy, a felhasználók szempontjából izgalmasabb, életszerűbb megközelítéssel. Az alkalmazás a hagyományos böngészésen kívül lehetőséget nyújt a felhasználóknak saját kívánságlisták, könyvjelzők összeállítására, az egyes dokumentumokról mélyebb információk megismerésére.

Célom, hogy a felhasználó képes legyen az okos telefonja felhasználásával, egy könyvborítóról készített fotóval keresést végezni a rendszerben. Fontos, hogy ez a folyamat gyorsan és egyszerűen történjen. Az új megközelítés képfeldolgozás, leíró kinyerés és gépi tanulás alapokon működik.

2.2 Digitális könyvtári katalógusok

A könyvtári katalógusokat [1] már az ókor óta használjuk. Napjainkra a hagyományos, cédula alapú katalógusok helyét átvették ezek infó-kommunikációs technológiákon alapuló változatai. Ezek a rendszerek minden esetben egy interaktív felületet kínálnak a felhasználók számára, hogy képesek legyenek könnyen és gyorsan keresni a könyvtár archívumában tárolt dokumentumok között. A dokumentumokat fő és alkategóriákhoz szokták rendelni, előre meghatározott szempontrendszer szerint, amelynek neve Egyetemes Tizedes Osztályozás. Ez a szempontrendszer nemzetközileg is elismert, egy számjegyekből álló kódrendszerként működik. Egy példa erre az 599.742.13-as azonosító, amely az Állatrendszertanon belül a kutyát jelenti, nyelvtől függetlenül. A kódrendszer karbantartását egy nemzetközi szervezet végzi, így biztosítva van a fogalmak közötti összefüggés és a hierarchikus felépítés. Az ETO általánosan alkalmazott osztályozását a melléklet első pontja szemlélteti.

A digitális könyvtári rendszerek terjedésével az ETO veszített jelentőségéből, viszont a könyvtárak túlnyomó többségében e rendszer alapján kerülnek a dokumentumok elhelyezésre illetve magában, a digitális rendszerben tárolásra.

A dokumentumok adatbázisba történő felvitele és kategorizálása során fontos tényező azok formai feltárása. A formai feltárás során történik az egyes dokumentumok alapvető adatainak összegyűjtése, így mint cím, alcím, eredeti nyelvű cím, sorozati cím, szerző, fordító, illusztrátor, rendező, konzulens és más fontos közreműködő neve, a kiadás éve, helye, a kiadó neve és a terjedelmi adatok, technikai adatok. Ezen kívül tárolni szokás az egyes dokumentumok ISBN számát (International Standard Book/Serial Number), amely egy nemzetközileg elfogadott egyedi azonosító szám, az országot és a kiadót is tartalmazza. A szerzők megnevezésekor nem szokás a tudományos és egyéb rangok

egzakt tárolása, mert problémát okozhat betűrendbe rendezéskor, de szokás ezt külön mezőben tárolni.

A formai feltárás mellett a tartalmi feltárás is fontos lépés. A dokumentumok tartalmi jegyeket direktben nem tartalmaznak, ezt a dokumentum felvitelekor kell elvégezni. Ezen jegyekre tárgyszavakat, más szóval címkéket szokás alkalmazni. Ennek mennyisége a dokumentum témájának gazdagsága és a feltáró munka mélységétől függ. Ez a lépés fontos tényező a későbbi, katalógusban történő keresés szempontjából, illetve a dokumentumok között fellelhető esetleges asszociációk feltárásának céljából.

2.3 Képi jellemzők adatbázisban tárolása

A megvalósított rendszer egyik kulcs funkciója, hogy a felhasználó képes legyen egy általa készített, a könyvborítót tartalmazó fotó alapján mélyebb információhoz jutni az adott dokumentumról. Az azonosítási folyamat a fényképen található könyvborítóról kinyert, úgy nevezett leírók, illetve az adatbázisban tárolt leírók közötti korreláció vizsgálatával történik. A leírók általában tetszőleges dimenziószámú vektorokként reprezentálhatóak.

A rendszerben fontos szempont, hogy az azonosítási folyamat a lehető legrövidebb idő alatt menjen végbe. Tegyük fel, hogy az adatbázisban 1000 darab könyvet, borítót és a hozzájuk tartozó leíró vektorokat tároljuk. Hagyományos megközelítésben ahhoz, hogy megállapíthassuk, mely tárolt könyv tartozik a keresendő mintánkhoz, szükséges, hogy minden tárolt könyv esetén elvégezzük az összehasonlítást, majd a legtöbb egyezést tartalmazó könyvet vegyük, mint lehetséges jó megoldást. Könnyen belátható, hogy ez a megoldás a tárolandó adatmennyiséggel exponenciálisan aránylik, hiszen az egyes minta összehasonlítás O (N*K) idővel növeli a komplexitást, ahol N a vizsgálandó vektor dimenziószáma, K pedig az aktuálisan tárolt minta dimenziószáma.

2.4 Lehetséges problémák

Mint minden, képfeldolgozással kapcsolatos rendszerben, ahol a valós térből kinyert információ feldolgozás van jelen, jelentős problémát tud okozni a különböző megvilágításból adódó információvesztés. A különböző megvilágítás azt is eredményezheti, hogy a vizsgálandó képen lévő objektumunk színei oly mértékben változnak meg, hogy ez a mintafelismerés lépésre is hatással van, a képről kinyert leírók egy jelentős része elveszik.

Második probléma az objektumunk orientációjából adódhat. Ahhoz, hogy képesek legyünk a valós térben elhelyezkedő könyvről jónak tekinthető leírókat kinyerni, előtte az objektumot a kép síkjára kell transzformálnunk. A transzformációhoz egy könyv alakú modellt kell alkalmaznunk, amely X és Y irányú komponense úgy aránylik egymáshoz, ahogy az a könyvek esetében jellemző, vagyis 1 az 1,456-hoz. A probléma

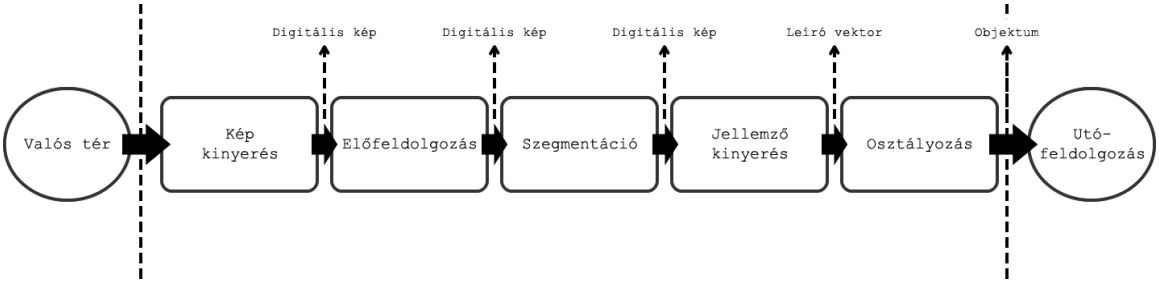
akkor jelentkezik, amikor a transzformált képünk hosszabb főtengelye a modellünk keskenyebb tengelyére képeződik le, a transzformáció során alkalmazott interpolációs lépés ekkor jelentős információvesztést okoz, hiszen adott esetben a kontúrok és ez által a jellemző pontok pozíciói is megváltozik. Emiatt szükséges, hogy a transzformáció előtt megállapítsuk az objektumunk orientációját, hogy adott esetben módosítani tudjuk a leképzés során használt modellünket. Ez a módosítás csupán egy affin transzformációt, 90 fokos forgatást jelent. Az eredményként kapott kép orientációtól függetlenül használható a későbbi lépésekben, csupán a projekció eredményeként kapott kép arányainak jósága befolyásolja az eredményt.

A harmadik, és talán legjelentősebb probléma a sebességből adódhat. Mivel a célkitűzésem része, hogy a lehető leggyorsabban megtörténjen az azonosítás lépése, szükséges, hogy a fennálló, akár több ezer könyvborítót tartalmazó adatbázisunknak csak azon elemeit vizsgáljuk az összehasonlítás lépésében, amelyekkel előzetesen szignifikáns korrelációt vélünk felfedezni a vizsgált, előfeldolgozott mintánk között. Adott esetben, ha az összehasonlítás során nincs lehetőségünk többszálú feldolgozásra, a lineáris keresés jelentős erőforrás és időigénnyel jelentkezhet. Emiatt szükséges, hogy a vizsgálandó borítók előválogatása és pontossága a lehető legjobb legyen, az adatbázis oldalán jelentkező szűrés lépése gyorsan történjen, illetve ügyelnünk kell a vizsgálat során alkalmazott szempont rendszerre.

3. IRODALOMKUTATÁS

3. 1 Könyvborító felismerés

Ahhoz, hogy képesek legyünk a valós térben elhelyezkedő könyvborítót további vizsgálatoknak alávetni, szükséges, hogy a borítót önmagában képesek legyünk kezelni. Ehhez a lépéshez szükséges, hogy a bejövő képet különböző előfeldolgozási algoritmusokkal a kívánalmaknak megfelelőre alakítsuk, eltávolítsuk róla a fotózáskor keletkezett zajokat, megtaláljuk magát a könyvborítót tartalmazó szegmenst a képen, majd az a kapott objektumot további feldolgozás céljából a kép síkjára transzformáljuk.



1. ábra: a hagyományos képfeldolgozó programok folyamatábrája

3.1.1 Előfeldolgozás

Az előfeldolgozás fázisa három részre osztható: zajszűrés, szegmentálás, majd a szegmens térbe transzformálása. A szűrés egy ablak alapú művelet, ahol az eredményképen az adott pixelekhez tartozó értékek azok valamely környezetéből származtatottak. A szűrés során az eredménykép mérete nem változik a kiindulás során használt képtől, a számítás pixelről pixelre történik, az ablakot egy mátrix reprezentálja.

A zajszűrés során a bejövő képünkön végzünk egy szűrést, amely képes a mintavételezés során keletkezett kiugró pixel intenzitás értékkel jelentkező, sérült pixeleket eltávolítani, adott esetben korrigálni egy, a környezetéből származtatott új értékkel. A zajszűrés történhet lineáris és nem lineáris módon. Lineáris esetben alkalmazhatunk ú. n. átlagoló szűrőt, ahol az új pixel intenzitás értéke a környezetében található pixelek intenzitásértékeinek átlaga lesz, illetve Gauss szűrést, ahol a maszk által lefedett pixelekhez egy súlyozó értéket rendelünk. Gauss szűrésnél általános esetben a középső pixel kapja a legnagyobb súlyt, majd a további súlyok ettől a ponttól távolodva csökkennek. Az utóbbi maszk közelítése a Gauss függvénynek. Nem lineáris esetben ez történhet mediánszűréssel.

A szegmentálás lépésben feladatunk a vizsgált objektumot az adott képen fellelhető kontextusából kiemelni. Az I (x,y) kép szegmentálása annak kapcsolódó részképekre bontását jelenti. Az egyes régiók akkor alkotnak egységet, ha valamilyen hasonlósági

feltétel teljesül rájuk. Ez a hasonlósági feltétel lehet intenzitás egyenlőség, vagy ha az adott régió intenzitásai között fellelhető különbség nem halad meg egy küszöbszámot, továbbá az adott régióban fellelhető intenzitás értékek szórása kicsi. Akkor beszélhetünk sikeres szegmentálásról, ha az adott régiók homogének, nem találunk lyukakat, illetve ha a szomszédos régiók között szignifikáns különbség áll fent.

A szegmentálás történhet binarizálás, él, régió és illesztés alapon. A binarizálás esetében beszélhetünk globális, lokális illetve dinamikus küszöbölésről, ahol a kiindulás minden esetben az adott kép pixel intenzitás értékeiből alkotott hisztogramja. Él-alapú esetében a kép deriválása során kapott, ú. n. grádiens képből indulunk ki. A grádiens alapú megközelítés esetében azok a területek képeznek összefüggő régiókat, amelyek jól kapcsolódó, zárt határvonalakkal rendelkeznek. Ebben az esetben a képeken jelentkező zajok képesek ú. n. ál-éleket képezni, ezért fontos, hogy minden esetben szükséges valamilyen zajszűrés lépés a grádiens kép előállítása előtt. Éldetektálás során nem csak az adott lokális grádiens értékről kaphatunk információt, hanem azok irányáról is, ez a későbbi lépésben elvégzett jellemző-pont detektálás során még jól jön.

3.1.2 Ortografikus projekció

A következő lépésben a könyvborítót reprezentálható szegmenst először a kép síkjára kell transzformálnunk. Mivel a könyvborító a kamera által elkészített fotón gyakorlatilag bármilyen orientációban elhelyezkedhet, ezért ügyelnünk kell, hogy a képet készítő kamera fókusztávolságát is figyelembe kell vennünk e művelet során, hiszen ez, mint torzító jelenség nagyban befolyásolhatja az eredményünket. Mivel jelen esetben 3D-2D-ós leképzést végzünk, egy jó megoldás lehet, ha ortografikus projekciót alkalmazunk a könyvborító transzformációs mátrixának meghatározására.

Az ortografikus projekció esetében az objektum felől a kamerába érkező sugarakat párhuzamosnak tekintjük, vagyis a fókusztávolságot végtelennel közelítjük. Az objektumunk egy jól körbehatárolható, téglalap alakú, négy sarokponttal rendelkező szegmensként reprezentálódik. A háromdimenziós térben elhelyezkedő $[a_x, a_y, a_z]$ pont leképzése $[b_x, b_y]$ pontba a következő módon számolható:

$$b_x = s_x a_x + c_x \qquad (1)$$

$$b_y = s_z a_z + c_z$$

ahol s az úgy nevezett skálázó tényező, c pedig az eltolás. A parametrikus egyenletek mátrixokkal felírva a következő:

$$\begin{bmatrix} b_x \\ b_y \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} + \begin{bmatrix} c_x \\ c_z \end{bmatrix} \quad (2)$$

3.1.2 Leíró kinyerés

A perspektív transzformáció után megkapjuk a könyvborítónk térbe transzformált leképzését. Mivel a célunk az adatbázisban található mintáinkkal való összevetés, szükséges, hogy valamilyen jellemzőket gyűjtsünk össze a képről. Olyan jellemzőket kell gyűjtenünk, amelyek invariánsak különböző geometriai és fotometriai transzformációkra, így mint eltolás, forgatás, skálázás, továbbá fényesség és expozíció. Általános megközelítés, hogy lokális jellemzőket gyűjtsünk, mivel ezek jól jellemzik az adott objektumunkat, robusztusak, képenként nagy mennyiség lelhető fel belőlük. A leírók lehetnek szín, alak, struktúra, textúra illetve jellemző pont alapú megközelítésből származó vektorok.

A jó jellemző pontok általában egy kis környezetben találhatóak. Azokat a pontokat nevezzük jellemző pontnak, ahol erőteljes válasz lelhető fel az adott környezetéhez képest. A jellemző pontdetektálás a korábban említett grádiens képből indul ki, azok a pontok rendelkeznek erőteljes válasszal, amelyek több él metszéspontjában találhatóak, tehát az adott pont egy pici környezetében számított sajátértékei lokális maximumként jelentkeznek. Jellemző pontok keresésénél fontos, hogy olyan detektort alkalmazzunk, amely invariáns eltolásra, skálázásra és affin transzformációkra. Általános esetben alkalmazhatunk Gauss kép-piramisokat az eredmény javítására. Ezen jellemző pontokból nyert leírók általában tartalmaznak az adott pont valamilyen kis környezetéből származó információkat, ezért arról is gondoskodnunk kell, hogy a leírók is invariánsak legyenek.

Egy ilyen leíró kinyerési módszer a Multiscale Oriented Pathches [3] leíró. Az algoritmus a jellemző pontok körüli 40x40 pixeles ablakból indul ki. A jellemző számításához az ablak méretét előszűrést alkalmazva először lecsökkentik 8x8 pixelesre, majd vízszintes irányba forgatják. A forgatáshoz a pont simított gradiens irányából számított szöget használják. Utolsó lépésben ezeknek a 8x8-as ablakok normalizált intenzitásait használják, mint leírót, ezt először az átlaggal csökkentik, majd elosztják a standard szórással.

A szín alapú megközelítésben a képek konkrét pixel intenzitás értékein végzett számításokból nyert értékeket használják leíróként. Ez lehet például az adott kép pixelintenzitásainak átlaga, minimális és maximális értéke valamint varianciája.

Az alakra, struktúrára és textúrára vonatkozó leírókról ebben a dolgozatban nem esik szó, hiszen itt egyértelműen definiálható téglalap alakú objektumokat kívánunk

összehasonlítani, és a transzformált borító további szegmentálásából nyert alakzatok felhasználása rendkívüli módon növelné a komplexitást.

3.2 Képek indexelése és keresés az adatbázisban

A képek visszakereshetőségének kérdése egy rendkívül összetett és sok kihívást tartalmazó feladat. Ez adódhat az adatbázis méretéből, a képek számítógép általi értelmezésének nehézségéből, a lekérdezések megfogalmazásának bonyolultságából, továbbá az eredmények kiértékelésének komplexitásából. Több megoldás létezik már a problémára, ilyen többek között az IBM QBIC[4] rendszere, az MIT Photobook [5], WebSEEK [6], CMU Informedia [7] illetve a Stanford WBIIS [8] melyek még a 90-es években születtek. A közös jellemzője ezeknek a rendszereknek, hogy mindegyik egy úgynevezett képekből kinyert szignatúra alapján kívánja megoldani a képek kereshetőségének problémáját.

A szignatúrák jellemzően pixel alapú leírókból származnak és előre definiált összehasonlítási szabályokat definiálnak a rendszer működéséhez. A szignatúrák alkotóelemei a korábban említett jellemzőkből származnak. Egyik előnye a szignatúrák használatának szemben a pixel intenzitás értékekkel, hogy a kép reprezentáció nagymértékben tömöríthetővé válik, habár alkalmazásának valódi oka inkább a kép reprezentáció és azok szemantikája között fellelhető korreláció felismeréséből adódó adatbázis particionálhatóság, továbbá ennek következtében fellépő jelentős rejlő sebesség növekedésben keresendő. Ennek okán a szignatúrák meghatározásakor a legfőbb feladat a képek pixeles reprezentációjának szemantikai elemzése. Az ilyen jellegű rendszereket összefoglaló névvel Content Based Image Retrieval (CBIR) rendszereknek nevezik, a szignatúrák kinyerésének megközelítései alapján három kategóriát különböztethetünk meg: szín-hisztogram alapú, szín-elrendezés alapú és régió alapú keresés. Léteznek ezen megközelítések ötvözését tartalmazó rendszerek is.

A szignatúrák kinyerése utáni következő lépés azoknak a szabályoknak a meghatározása, amelyek alapján össze tudjuk hasonlítani a kívánt képeket, vagyis azon hasonlóság mértékek meghatározása, amelyek képesek megadni az egyes képek közötti korreláció valószínűségét. A legtöbb rendszerben lekérdezéshez ugyancsak egy képet használnak, egyes rendszerekben[9] lehetőség van csupán ezen kép egy előre definiált régiója alapján való keresésre.

A hisztogram alapú keresés során a kép karakterisztikáját annak pixel intenzitás értékei eloszlásából származtatott görbe és az adatbázis képeinek ugyanezen eloszlásból származtatott görbe közötti távolságokat vizsgálják. Sok távolságforgalmat használnak e görbék összehasonításaihoz, így mint Euklideszi távolság, Histogram intersection távolság[10], illetve a Historgram quadratic távolságot, amely a hisztogram egyes vizsgált pontjainak kereszt-korrelációjából származtatott távolság fogalom. Ezen megközelítés hátránya, hogy nincs semmilyen információnk a vizsgált objektumunk

helyzetéről, alakjáról illetve textúrájáról, továbbá a szín-hisztogram alapú keresés érzékeny a megvilágításból, vágásból illetve az intenzitások varianciájából adódó eltérésekre.

A szín-elrendezés alapú megközelítés e hibákból adódó problémákat próbálja meg kiküszöbölni úgy, hogy a képeket blokkokra partícionálja[4], és az ezekben számított átlagos intenzitás értékeket tárolja, de létezik olyan rendszer[8], ahol Daubechies wavelet transzformáció koefficienseit tárolják, mint blokk leírót. Utóbbi megoldásban a blokkméret és a wavelet transzformáció szintjének növelésével az egyes színintenzitás értékek behangolhatóak az egyes blokkokra és ez jobb eredményt produkálhat kereséskor. Egy másik előnye ennek a megközelítésnek, hogy megfelelő felbontáskor képes megtartani az objektumaink alakját, elhelyezkedését és textúráját, ugyanakkor a keresés továbbra is érzékeny marad az eltolásra, skálázásra és forgatásra.

A WALRUS [11] nevű rendszerben az eltolásból és a skálázásból adódó problémákat úgy próbálták meg kiküszöbölni, hogy a teljes körűen particionálták fel a képeket blokkokra. Ezt a particionálást több méretű mozgó kernelmaszkkal oldották meg, ahol minden ablakra kiszámolták a szín-szignatúráját a blokknak. A kép összehasonlítást ezután a részképek összehasonlításával végezték. Hátulütője ennek a rendszerek a megnövekedett számítási teljesítményben és a keresési idő megnövekedésében jelentkezik, továbbá az objektumok alakja és textúrája továbbra is figyelmen kívül hagyott, mint jellemző, csupán a részképek blokkjainak átlagos színeit alkalmazzák, mint jellemzőt.

A harmadik csoportot a régió alapú keresést alkalmazó rendszerek alkotják. Ezen rendszerek célja, hogy a keresés objektum szinten valósítsák meg, ezzel küszöböljék ki a szín alapú megközelítések problémáit. Ezen rendszerekben a képeket először szegmentálják, majd a képek ezen dekompozícióit használják fel a keresés során. Ez a megközelítés arra hivatott, hogy közelítse a képek általi kép-kontextus értelmezést az emberi érzékeléshez, annak ellenére, hogy a valós képek csupán 2D-ós projekciói az ember által is érzékelt 3D-s világnak, és a számítógépek nincsenek megtanítva a 3D-s tér ember módjára történő érzékelésére.

Mivel e rendszerekben a képeken elhelyezkedő objektumokat önmagukban kezeljük, így könnyebb azok összehasonlítása más képeken szereplő objektumokkal, függetlenül az adott objektum orientációjától és méretétől.

Két ilyen rendszer a NeTra [9] és a Blobworld [12]. Ezen rendszerek közös tulajdonsága, hogy a képeket az egyes régiók összehasonlításán keresztül vizsgálják, az egyes képeken fellelhető régiók száma limitált, az egyes találatok az egyedülálló régiókra adott keresési találatok összevonásán alapszik. Vagyis adott esetben, ha egy képen három objektum található, az eredmények az egyes régiókhoz tartozó találatokból álló lista lesz, a végső döntést a felhasználóra bízzák. Belátható, hogy ez a megközelítés

a felhasználó számára nagyobb kontrollt biztosít, tekintettel, hogy a felhasználónak lehetősége van a régiók textúrájának, színének figyelembevételével keresni, ugyanakkor az egyes régiók reprezentációi eltörpülnek a felhasználó szemantikus megértése mellett.

E két rendszer további hátránya, hogy nem fektetnek nagy hangsúlyt a hasonlóság mértékek meghatározására, inkább egyes régiókból származó információkra alapoznak. Ugyanakkor létezik egy megközelítés [13], amely erre a hiányra próbál megoldást kínálni. Ebben a megközelítésben minden régióhoz azok karakterisztikája alapján címkéket rendelnek, és ezekből a címkékből készített leíró határozza meg az egyes régiókat a továbbiakban. A leíróból egy úgy nevezett CRT leírót készítenek, amely hivatott a címkék relatív sorrendje alapján meghatározni a régiókat. A vizsgálat során ezeket a CRT leírókat használják összehasonlításra. Ez a megoldás sajnos nem elég robusztus, mivel a címkék sorrendje változhat eltolás, skálázás és rotáció esetén, továbbá a rendszer teljesítménye nagyban függ a címkéket tartalmazó szótár méretétől. Az egyes címkéket szín, alak és textúra jellemzőkből származtatják, az értékkészlet növelésével a számítási komplexitás exponenciálisan nő, ezért nem biztos, hogy nagy méretnél a hatékonyság biztosítható.

Az egyik hibrid rendszer, amely az előző rendszerek problémáit hivatott kiküszöbölni a SIMPLYcity [14], amely abból az állításból indul ki, hogy a releváns képek szemantikai karakterisztikája nagyon hasonló. Ennek a rendszernek az alapja egy felügyelt tanulási megközelítés, célja, hogy a képek különböző osztályokba sorolásával jelentősen csökkentsék a képek keresésének idejét. A rendszer a tárolt és a keresett képeken elvégez egy osztályozást, majd az alábbi szemantikai alkategóriákba sorolja a ezeket: grafika, fénykép, textúrázott, nem texttúrázott, beltéri, kültéri, város, táj, embereket tartalmaz, illetve nem tartalmaz embereket. Minden egyes szemantikai osztályhoz egy-egy különböző adatbázist használ, és egyedi osztályozást végez, továbbá a tárolt jellemzők is egyediek. Textúrázott képeknél például nem használ alakra vonatkozó jellemzőt, hiszen itt például a textúra energia alapú jellemzők sokkal szignifikánsabbak. A régiók szegmentását csatornánkénti szín-átlagolással és magas frekvenciájú wavelet transzformáció során kinyert 6 jellemzővel határozza meg. A képeket először 4x4-es blokkokra bontja, és ezekben számít jellemzőket. A blokkok között számított K-means klaszterezés eredményeként határozza meg a régiókat, ahol az elvárt csoportok száma egy maximális értékhez közelít, továbbá alkalmaz egy empirikus alapon számított, jóságot meghatározó küszöb értéket, mint kilépési feltételt. Színek esetében LUV színteret használ, L a fényesség, U és V pedig színességet adja meg. Ez a színtér kiválóan alkalmas színtávolságok számítására. A wavelet transzformációt Haar transzformációval számítja. A rendszer fejlesztői módszerükben bevezettek egy integrated region matching (IRM) távolság mérést, amely gyenge szegmentálásból adódó hibákat hivatott javítani. Az előzetes osztályozás és szegmentálás után a rendszer szemantikai osztálytól függően használ textúra, szín és alak alapú jellemzőket. Végeredményben ez a rendszer egy rendkívül robusztus megoldást kínál az előzőekben

felsorolt problémákra, akkor is képes helyes eredményeket produkálni, ha a keresett minta orientációja, színintenzitásai, felületi jellemzői változnak.

3.3 Leírók összehasonlítása és osztályozás

A kinyert leírók és az adatbázisban tárolt leírók közötti korreláció vizsgálata egy komplex feladat a CBIR rendszerekben. Szín alapú leírók esetében feladatunk olyan jellemzők feltárása, amelyek szignifikánsan jellemezhetik az adott képünket, figyelembe véve a megvilágításból adódó szín-torzulások okozta hibát. Az előzőekben tárgyalt szín-elrendezés alapú megközelítés lehet a leghatékonyabb módszer e jellemzők vizsgálatára, viszont gondoskodnunk kell arról, hogy olyan színtérben végezzük összehasonlítást, ahol a távolságok uniformisak lehetnek az egyes csatornák eltéréseire. Könnyen belátható, hogy ez a tulajdonság az általánosan használt RGB színtérnél nem áll fent, az egyes csatornákon fennálló intenzitás érték változás ugyan adott esetben nem változtatja jelentősen a három csatorna intenzitás értékeiből képzett, az RGB színtérben elhelyezkedő új pont és az eredeti pont között fennálló távolságot, mégis emberi érzékelés számára egészen más színérzékelés tapasztalható. Ezen probléma kiküszöbölésére alkalmas többek között az XYZ és az LAB színtér, ahol az Z illetve az L komponens csupán a szín fényességét adja meg, míg az X-Y, és A-B komponens azok színárnyalatát kódolják. Az LAB színtér valójában az XYZ színtér egy nem lineárisan tömörített változata, célja az emberi szín érzékelés közelítése volt. Ezen színterekben a távolságok számítás általában az Euklideszi formulával történik.

A pontpárokon alapuló képillesztés a két képen található jellemző pontok párba állításán alapszik. Ezen módszerek hasonlósági mérték optimalizálásra törekednek közvetlenül egyenletrendszerek megoldásán keresztül, vagyis cél egy direkt stratégia követése. Fontos, hogy ugyanazon jellemző pontokat szeretnénk megtalálni mindkét képen, vagyis az adott pont-csoportok között fennálló transzformációk hasonlóak legyenek. Hogy ezt a transzformációt megtaláljuk, olyan módszert kell alkalmaznunk, amely toleráns eltolásra, forgatásra, skálázásra és affin transzformációkra.

A direkt stratégia célja a pontpárok megtalálása, általában egy adott pont és a másik képen lévő pontok közötti távolság minimalizálásából indulunk ki. Ez a módszer akkor is működik, ha a párosítás kétértelmű, viszont fontos, hogy valamilyen jóság/hasznosság függvényt vezessünk be a helytelen találatok kezelésére. A jóság vizsgálatához általában meghatároznak egy küszöb értéket, amely a pontok közötti távolságot szabályozza. Matematikailag X a magyarázó változókat, Y a magyarázandó változókat, $f(.) : X \rightarrow Y$ az osztályozás függvényét jelenti. Cél $E[U(Y, f(X))]$ várható érték maximalizálása, ahol $U(y, y')$ az előbb említett hasznossági függvény, és y' az y függvény közelítése.

A feladat megoldására gyakorta használnak K-Nearest Neighbour [35] algoritmust, vagy Support Vector Machine [29] -t. Mindkét módszer egy felügyelt tanulás módszer,

ahol a bemeneti adatunkról szeretnénk eldönteni, melyik osztályba tartozik, vagyis szeretnénk egy címkét rendelni hozzá. Az osztályozás egy Euklideszi vektortérben történik. A tanító adatokat általános esetben numerikus értékek reprezentálják, ezek dimenziószáma határozzák meg azon vektortér dimenzióinak számát, ahol az osztályozást végezzük. Az osztályozás hipotézise, hogy az egyes osztályokba tartozó dokumentumok összefüggő régiót alkotnak a vektortérben. Az osztályozások között megkülönböztetünk particionáló és kiválasztó osztályozást, utóbbi esetben az egyes osztályok kizárják egymást.

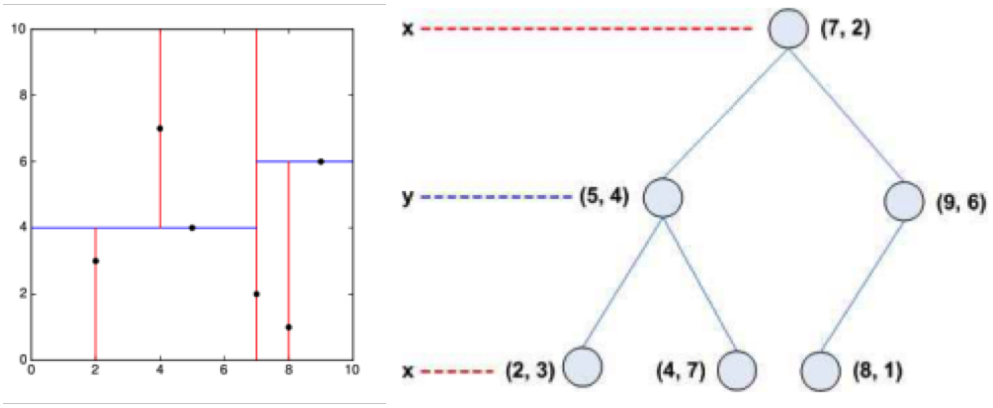
A tanítás során történik meg az úgy nevezett modellépítés, mikor az egyes tanítóadatokhoz címkéket rendelünk. A címkéhez rendelés felügyelettel történik, a tanítás során minden osztály esetében meg kell határozni a tartalmazott pontjai által alkotott halmaz és annak komplementer halmazai között található szeparáló hipersíkokat, ez lesz az a határ, amely az osztályozás során meghatározza majd a vizsgált objektumunk adott osztályba sorolhatóságát.

3.4 Keresés további gyorsítása

Nagy értékkészlet esetében az osztályozás rendkívül számítás igényes lehet, hiszen minden objektumunkra meg kell határoznunk azok elhelyezkedését az előzőleg becsült N dimenziós vektortérben. Az egyik gyorsítási lehetőség, ha már a predikciós lépésben elvégezzük ezt a modellépítést, így csupán az adott osztályba esés valószínűségét kell vizsgálnunk, a tanítási lépést nem szükséges minden vizsgálat során újra és újra elvégeznünk. Egy másik gyorsítási lehetőség lehet, ha az egyes osztályokat valamilyen szempontrendszer szerint csoportosítjuk, ezen technikák összefoglaló neve a Branch-and-bound. Ilyenkor gondoskodnunk kell róla, hogy az egyes osztályokat úgy rendezzük adott csoportokba, hogy azok eloszlása a lehető legjobban közelítse a osztályozni kívánt objektumaink várható értékeinek eloszlását, más szóval az adott osztályba sorolhatóságának valószínűségét. További gyorsítási lehetőség lehet, ha a vizsgálat során valamilyen szempont szerint sorba rendezzük a vizsgált mintáinkat, így az általános O(n) idő alatt végbemenő lineáris keresést visszavezethetjük bináris keresésre, azaz ideális esetben a keresés csupán O(log n) idő alatt végbemehet.

Továbbá alkalmazhatunk KD[15] fákat a keresés gyorsítására, ahol az adott döntési teret hipertéglatestekre bontjuk. Osztályozáskor azt a hipertéglatestet vesszük, amely tartalmazza az adott osztályt, a döntés a vizsgált pontunkhoz legközelebb álló osztályra esik. Viszont amennyiben a vizsgált pontunk közelebb esik egy ilyen hipertéglatest határához, a szomszédos hipertéglatestben található osztályt is felvesszük az értékkészletünkbe, ezzel finomítjuk a kiértékelést. KD fa építésénél ügyelnünk kell, hogy a fa kiegyensúlyozott legyen, minden elágazás és levél egy hipertéglatest dimenzióját tartalmazza. Építéskor egy egyszerű heurisztikát kell követnünk, a

tengelyek kiválasztásánál a legnagyobb szórású osztályokból indulunk ki, osztópontnak pedig az adott osztály pontjainak mediánját vesszük.



2. ábra, KD-fa hipertéglatestei és fa reprezentációja

3.5 Optikai karakterfelismerés

Kép alapú azonosítás során problémaként jelentkezhet, ha az adott könyv borítója sérült, vagy adott esetben a könyv egyes kiadásai különböző borítóval rendelkeznek. Továbbá a felismerés során előállhat olyan eset, amikor az egyes könyvborítók nem tartalmaznak elegendő jellemzőként felhasználható információt, csupán az adott könyv címére és szerzőjére utaló információt. Ahhoz, hogy az ebből származó hibás osztályozást elkerülhessük, szükséges, hogy a képi jellemzők mellett a borítón található szöveges információk segítségével javítsuk a keresést. Számos megközelítést alkalmaznak a karakterek felismerésére, a felismerés mindig egy előfeldolgozási fázissal kezdődik. Az egyes karakterek azonosításához fontos, hogy mint önálló objektum kezelni tudjuk őket. Erre jó megközelítés, ha a képek szürkeárnyalatos változatain végzünk el binarizálás vagy él alapú szegmentálást, majd az egy irányba eső, jól szeparálható régiókon végezzünk el további vizsgálatokat. A karakterek keresése e régiók illesztésével történik, általában alakra vonatkozó jellemzők, így mint skeleton, alakszám, nyomatéki vagy csúcspontokból származtatott jellemzők alapján történik.

A betűk szegmentálásához egy gyakran használt algoritmus, a Stroke Width Transform [32], ami abból a feltevésből indul ki, hogy a karakterek vastagsága jellemzően korrelál egy értékhez, csupán egy kis környezetben változik.

A jellemzők kinyerése utáni második fázisban a cél, az egyes objektumok azonosítása. Mivel a számok és az írásjelek kölcsönösen kizárják egymást, általában kiválasztó osztályozást alkalmaznak ezeknek a karaktereknek a felismerésére.

A harmadik fázis az utófeldolgozás, ahol több problémával is szembesülhetünk. Az első a hibásan osztályozott karakterekből adódik, általában lokálisan-érzékeny hasítást [16], szomszédsági és/vagy Levenshtein [17] távolságot alkalmaznak e hibák kiküszöbölésére. Ezen vizsgálatok elvégzéséhez általában szükséges valamilyen szótár megléte, mivel az adott karakterek nyelv-specifikus előfordulása eltér, az egyes karaktercsoportok együttes előfordulására alapozva jobb eredmény érhető el.

A Levenshtein távolság arra a problémára hivatott megoldást kínálni, hogy két hasonló szó esetében hány permutációból állna a keresett szó vizsgált szóba történő transzformációja. Matematikailag *a*, *b* szó közötti távolságot a következő rekurzív formulával definiáljuk:

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \quad (3)$$

Példaként vegyük az angol kitten és sitting szavakat, amelyeknek távolsága három, mivel a következő három permutáció szükséges, hogy átvigyük az egyik szót a másikba és nincs rövidebb út ennek megtételére:

- kitten → sitten ("s" helyettesítése "k"-val)
- sitten → sittin ("i" helyettesítése "e"-vel)
- sittin → sitting ("g" beszúrása a végére)

A Levenshtein távolság öt szabályt definiál az egyes távolság értékek alsó és felső határaira vonatkozóan:

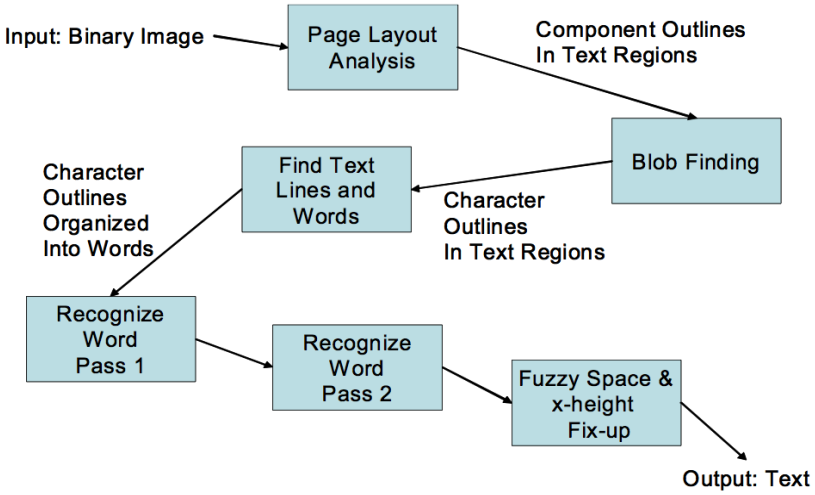
- a távolság mindig legalább a két szó hosszának különbsége
- a távolság maximum a hosszabbik szó hosszával egyezhet meg
- csak abban az esetben zérus a távolság, ha a két szó megegyezik
- ha a két szó hossza megegyezik, a távolság étéke egyenlő a két szó Hamming távolságával
- két szó közötti Levenshtein távolság maximum akkora lehet, mint a távolságuk összege egy harmadik szótól

Könnyen belátható, hogy e távolságfogalom használata nagyban segítheti az egyes szóegyezésre vonatkozó vizsgálatainkat.

3.5.1 Tesseract by Google

Ezt a keretrendszert a Google [18] fejlesztette és gondozza, nyílt forráskódú, így bárki módosíthatja, és adott feladatra szabhatja. A keretrendszer eredetileg latin nyelvek feldolgozására készült, de a Google berkein belül sok erőfeszítést tesznek az alkalmazhatóság kiterjesztésére vonatkozóan a kínai [19], japán, koreai, arab és hindi nyelvet illetően. Ezen nyelveknél külön kihívást jelent a meglévő karakterkészletek óriási nagyságrendje, a szövegek írásának iránya, arab esetében külön nem szeparálható betűk, ezért igyekeznek ezeket a nyelveket külön nyelv-specifikusan értelmezni. Nyilvánvaló, hogy könnyebb egy adott nyelvre specifikus értelmező készítése, mint egy globálisan használható megoldás, hiszen az nagy értékkészletre történő osztályozás még napjainkban is problémás kutatási terület.

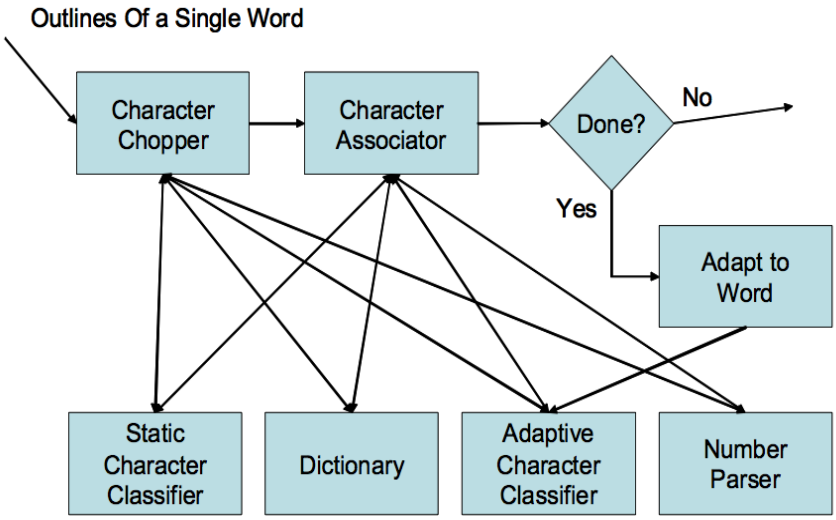
A rendszerük első lépésben valamilyen globális optimum megtalálása után elkészíti a bemeneti kép binarizált változatát. A továbbiakban ezen a bináris képen végeznek kapcsolódó komponensek analízisét, hogy megtalálják a karaktereket reprezentálható régiókat a képen. Mivel a keretrendszer fejlesztésekor célként fogalmazták meg, hogy a rendszernek képesnek kell lennie fehér alapon fekete illetve fekete alapon fehér szövegek felismerésére is, ezért a továbbiakban csupán e régiók határvonalait használják tovább. Mivel a latin betűk esetében nagyon ritkán áll fent, hogy egy karakter kettőnél több lyukat tartalmazzon, így ezt jól tudják használni, mint terminálási feltételt az előválogatás során.



3. ábra, a Tesseract keretrendszer blokksémája [18]

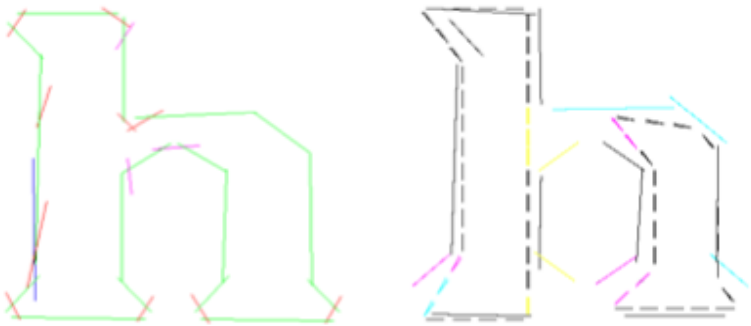
Miután eldöntötték, mely régiók körvonalai reprezentálhatnak karaktereket, meghatározzák a szöveg egyes sorainak orientációját, ehhez a régiók alapvonalait használják. Miután megtalálták ezen alapvonalait egy szóköz detektáló megvizsgál minden sort, hogy megtalálja az egyes szóközöket, majd ezek meghatározása után a sorokat szavakra particionálják, vagyis a szóközzel határolt régió csoportokat

összevonják. A következő lépésben végzett vizsgálat ezen összerendelt szavakon végez vizsgálatokat, az egyes szavakat külön-külön kezeli.



4. ábra, a szófelismerés blokkdiagramja [18]

Jellemzően minden régió egy adott karaktert reprezentál, ezért a szó felismerő először külön-külön osztályozza a lehetséges karaktereket. A karakterek osztályozásához azok alakját először poligon approximációval felbontják, majd minden szakasz esetén előállítanak egy négydimenziós leíró vektort a szakaszfelező x, y koordinátáiból, illetve a szakasz irányának és hosszának értékeiből. Ez után ezeket a négydimenziós vektorokat klaszterezik, ezzel képeznek egy prototípust a karakterből. Felismeréskor a különböző dimenziószámú poligonokat egységes dimenziószámra transzformálják, hogy függetlenítsék a leíró vektortól.



5. ábra, a) a h betű Times New Roman betűtípusból felépített prototípusa, b) pedig egy találatként vett prototípus reprezentációja az adott modellhez viszonyítva [18]

Az a) képnél a zöld vonalak azt jelölik, hogy azok klaszterezett jellemzők, vagyis szignifikánsnak tekinthetőek az összehasonlítás lépésében. A kék szín azt jelöli, hogy ez két szignifikáns klaszter összevonásából képződött klaszter. A magenta szín azt jelöli, hogy ezeket a szegmenseket nem vizsgáljuk, mert már beletartoznak egy meglévő

klaszterbe. a Piros pedig azt jelöli, hogy az adott klaszter nem szignifikáns, nem tartalmaz elég mintát, ezért nem sorolható egyik klaszterbe sem.

A b) képnél a rövid szaggatott vonalak a találat jellemzői, a hosszabb vonalak pedig a prototípus jellemzői. A színek a találat minőségét jelszik, a fekete jó, a magenta elfogadható, a cián a gyenge és a sárga a találat nélkülit jelöli. A függőleges találatok általában jók, annak ellenére, hogy a vizsgált karakter erősen hiányos.

A karakter osztályozás két lépésből áll: először az összevetendő objektumok számát lecsökkentik 1-10 karakterre lokálisan-érzékeny hasítás alkalmazásával, majd ezután a csökkent értékkészleten végeznek távolság számításokat. A távolságvizsgálat során az egyes klaszterek között euklideszi távolságot számítanak az x, y és hossz értékekből, majd korrigálják ezt a klaszter 4-ik dimenzióján tárolt szöggel, amelyhez egy súlyt is alkalmaznak:

$$d_f = d^2 + w\theta^2 \tag{4}$$

Ez az osztályozás az ideálistól vett távolságot számolja, ahhoz hogy ezt finomítsák, bevezettek egy úgy nevezett bizonyosság mértéket, amely:

$$E_f = \frac{1}{1 + kd_f^2} \tag{5}$$

ahol k egy kontrolláló faktor a távolságra. Jellemzők vizsgálatakor jelentkező találat esetén ezt az E_f faktort átmásolják a prototípusra E_p, mivel a párosítás során több jellemzőt is összevetnek. A jó találatok gyűjtése függetlenül történik, a jellemzők összege és a prototípus bizonyossága eltérhet. A végső távolságot végül a jellemzők számával és prototípusok számával L_p normalizálják, majd - mint távolságot- használják tovább:

$$d_{final} = 1 - \frac{\sum_f E_f + \sum_p E_p}{N_f + \sum_p L_p} \tag{6}$$

A kétlépéses osztályozás lehetőséget ad harmadik felek számára, hogy egy-egy karakterhez több betűtípus felhasználásával készített prototípusokkal bővítsék rendszerünket.

A numerikus karakterekre és írásjelekre a rendszer egy elkódolt szótárat alkalmaz, ennek bővítése nem lehetséges.

Miután az egyes karakterek azonosítása megtörtént, egy szótár segítségével, kontextus vizsgálatával bizonyosodik meg az adott szó helyességéről, vagyis hogy a detektálás sikeres. A rendszer lehetőséget biztosít ezen szótár bővítésére, illetve saját maga is alkalmaz adaptív szótárbővítést. A keresés egy direkt aciklikus szó gráfban történik, ahol az egyes csúcspontokat 8 bites karakterek reprezentálják. A rendszerben két ilyen gráf különböztethető meg: bizonyos és bizonytalan szavak szótára.

Hogy megbizonyosodjanak egy újonnan bekerült szó megbízhatóságáról, két szabályt fogalmaztak meg: az alakfelismerőnek egy tiszta győztest kell szavaznia az adott szó variánsai közül, vagyis szignifikánsan nagyobb bizonyossággal kell rendelkeznie, mint a második lehetséges variáns. A második szabály pedig, hogy nem létezhet az adott szó alakjára vonatkozó legjobb választás a bizonyos szavak szótárában.

A két gráf alkalmazása jelentős sebességnövekedést okoz, mivel a szótárbővítés után egy adott szó gyorsan kikereshető a bizonyos szavak szótárában és nincs szükség további vizsgálatokra.

3.5.2 Azonosított szövegek összevetése az adatbázisban tároltakkal

A kinyert szövegeink az adatbázisban tárolt mintákkal való összevetése esetén probléma lehet, hogy azonosított és egy csoportba sorolható karakter/szó csoportjainkat szerző, cím, vagy alcímként kezeljük. A csoportba sorolhatóság jellemzően egy adott sorba esésként definiálható. Mivel a szerzők nevei jellemzően 2 és 4 szót alkotnak (feltéve, hogy egy szerzőt veszünk), érdemes az ilyen szó csoportokkal indítani a keresést. Továbbá feltehetjük, hogy dokumentumok címe általában nagyobb betűmérettel rendelkeznek, így ezt is figyelembe vehetjük a keresés gyorsítása során. Amennyiben egyik tulajdonság sem teljesül, minden fentebb felsorolt, illetve a képen található összefüggő szövegrész párosítását el kell végeznünk.

A keresés tovább gyorsítható, ha az egyes attribútumokban tárolt szavakat lemmatizáljuk, vagy szótövezést alkalmazunk, majd az előálló új szavakat, mint független címkéket tároljuk. Kereséskor legtöbb egyezést tartalmazó könyveken kell további vizsgálatokat végezni. A lemmatizálás során a szavak végéről vesszük le az egyes toldalékokat, szótövezés esetén az egyes szavakat helyettesítjük azok szótöveivel.

Angol nyelvnél a legáltalánosabban alkalmazott szótövező a Porter [20] algoritmus. Az algoritmusban szekvenciálisan hajtunk végre fázisokat, a fázisok műveletek csoportjaiból állnak. Cél azon művelet kiválasztása fázisonként, amelyek a legnagyobb mértékben csökkentik a szó hosszát.

Fontos kiemelni, hogy az egyes szótövezők jellemzően nyelv specifikusak. Magyar nyelvet is ismerő szótövező például a Snowball [21], amely a Porter algoritmus egy modifikációja. A nyelvenként eltérő nyelvtani szabályrendszerek miatt szükséges, hogy csak olyan szótövezőt alkalmazzunk, amely bizonyosan jól illeszkedik a vizsgált szövegünk nyelvére. Ahhoz, hogy a megfelelő szótövezőt választhassuk, meg kell bizonyosodjunk a vizsgált szövegünk nyelvéről.

A nyelv meghatározására számos módszer létezik. Az egyik leggyakrabban alkalmazott az N-gramm analízis [22], amely során a szavakat n hosszúságú átfedő részekre bontják, majd ezeknek a szeleteknek előfordulási valószínűségét vizsgálják tovább. Egy k betűből álló szónak k + 1 bi, k + 2 tri és k + 3 4-grammja van. Az egyes N-grammok valószínűségét tokenizálással előkészített szövegen egy hasító tábla segítségével határozzák meg, inputként javasolt néhány 10000 szóból álló mintaszöveg felépítése. A leggyakoribb 300 N-gramm jó korrelációval rendelkezik az adott nyelvre vonatkozóan, így elegendő csupán azokat tárolni. Amennyiben egy N-gramm több nyelv hasító táblájában is szerepel, a nagyobb valószínűségűt veszik, mint jó eredményt.

Alkalmazható továbbá a rövid szó vizsgálat, ahol 4 és annál kevesebb karakterből álló szavakat használnak a felismerés során. A kinyert kulcsszavak/N-gramok osztályozásához alkalmazható például a Naiv-Bayes [23] osztályozás, amely a Bayes féle feltételes valószínűség tételéből indul ki. Utóbbi esetben minden egyes szóra meghatározzák annak előfordulási valószínűségét. Másik megközelítés a rejtett markovi modellek nyelvfelismerésre alkalmazása lehet.

Egy korábbi egyetemi kutatásban [24] statisztikai jellemzőket alkalmaztak nyelvfelismerésre, ezen jellemzők voltak a szó végi karakterek, átlagos szóhossz, az A/E és S/T karakterek aránya, továbbá 1-2 karakter változásig vizsgált betű-átmenet gyakoriság. A módszer 1500 karakter fölött már átlagosan 95%-os pontossággal képes egy adott szöveg nyelvének meghatározására, továbbá egy nagyságrenddel gyorsabban, mint az N-grammos osztályozás. A mérések során angol, német, francia, spanyol és magyar nyelvek statisztikája alapján osztályoztak, a szóvégi karakterek esetén az angol ábécé egy empirikus módon meghatározott részhalmazát vizsgálták.

3.6 Hasonló rendszerek bemutatása

3.6.1 Book Cover Recognition Project

Ez a rendszer [25] az University Of California San Diego egy PhD hallgatójának gondozásában készült. A rendszerben a leíró vektorok kinyeréséhez SIFT [26] algoritmust alkalmaz. Az algoritmus első lépésben egy Gauss-féle konvolúciós szűrést alkalmaz a kép simítására, majd az előálló képből több szintű piramist épít. A piramis egyes szintjeiből különbség szinteket képez (DoG) az aktuális Gauss piramis szint és az alatta található szintek között a következő módon:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, \sigma) - L(x, y, \sigma) \quad (6)$$

ahol σ az alkalmazott Gauss függvény szórása, k a vizsgált szint, $I(x, y)$ a bemeneti kép, $*$a konvolúciós operátor, $D(x, y, \sigma)$ pedig a különbség szinttek képzéséhez használt konvolúciós maszk.

A következő lépésben a D függvény által előállított különbség szintekből álló térben keres lokális szélsőértékeket, ezek a pontok kerülnek - mint jellemző pontok -, kiválasztásra. Minden szinten elvégzi ezen lokális szélsőértékek keresését, a szintek közötti átszámolásra interpolációt alkalmaz. A kulcspontok számának redukálására az így összegyűjtött pontokon egy Laplace operációt alkalmaz küszöbölve, a gyenge élpontok eltávolítására.

Utolsó lépésben történik a jellemző vektorok meghatározása. Az algoritmus minden jellemző ponthoz egy él-orientációs hisztogramot számol a lokális gradiens jellemzők alapján. A gradiens nagyságot és irányt az alábbi módon határozza meg:

$$m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2} \quad (7)$$

$$\theta(x, y) = \text{arctg} ((L(x, y + 1) - L(x, y - 1)) / (L(x + 1, y) - L(x - 1, y)) \quad (8)$$

ahol $m(x, y)$az adott x, ypontbeli gradiens nagysága, θ pedig az iránya. A számítást az adott pontok 16x16 pixeles környezetében végzik, amelyet további 4x4 pixeles ablakokra osztanak. Az egyes ablakokban minden pontra kiszámolják azok 8 irányban vizsgált gradiens értékeit és ebből képeznek egy globális hisztogramot minden ablakra. A leíró ebből a 16 * 8, vagyis 128 dimenziós vektorból áll. A módszer előnye, hogy rendkívül robusztus, forgatást 60-fokos szögig tolerálja, továbbá mivel az éltérképen végzett számításra alapul, így viszonylag jól tolerálja a megvilágításból adódó problémákat. Az algoritmus ezen kívül alkalmas valós idejű működésre is. Az algoritmusnak létezik egy főkomponens analízisen alapuló változata is, amely a leíróvektort 36 dimenziósra csökkenti le, ezzel jelentős sebesség gyorsulást eredményezve leíró vektorok párosításakor, viszont maga a leíró kinyerés költsége megnő.

A könyvek felismerésére a szerző K-Means [27] felügyelet nélküli tanulást, csoportosítást, más szóval klaszterezést alkalmaz. A csoportosítás általános esetben egy hasonlósági függvény használatával kíván adott objektumokat csoportokba szervezni, az egyes csoportokba tartozást egy tagsági függvényel határozza meg. Megkülönböztetünk

hierarchikus és particionáló klaszterezést, előbbi esetben az egyes csoportok tartalmazhatnak alcsoportokat is. Ezen kívül a csoportosítás lehet top down, ahol a teljes halmazból indulunk ki, és minden esetben kettéosztunk egyes csoportokat, vagy bottom-up, ahol az egyes klaszterekből indulunk ki, és lépésenként végzünk összevonást. A csoportok hasonlóságának vizsgálata egy súlyozott dendogramm felhasználásával történik, cél, hogy azok a csoportok kerüljenek összevonásra, amelyek a leghasonlóbbak. Az összevonás lépése egészen addig ismétlődik, amíg el nem ér egy úgy nevezett terminálási számot, vagyis egy előre meghatározott klaszterszámot. Minden összevonás után az egyes klaszterekhez tartozó súlyt az algoritmus újra számolja.

K-Means esetében abból a feltevésből indulunk ki, hogy az egyes klaszterek középpontja, más szóval centroidja jól reprezentálja az adott csoportot. A kiértékelés során egyes centroidoktól vett távolság alapján történik a csoportba sorolás. Algoritmus első lépésben véletlenszerűen meghatároz K darab kiinduló elemet, majd iterálva elkezdi ezen K elemek környezetében található legközelebbi pontok összerendelését. Lépésenként 1 pont kerül egy adott csoportba, majd az összerendelés után újonnan létrejövő csoportok centroidjának újraszámolásával módosítja a hasonlósági függvényt. Az algoritmus egészen addig végzi az iterációt, amíg egy terminálási feltétel nem teljesül, mint például ha minden pont tartozik egy csoporthoz, az algoritmus elért egy meghatározott iterációs számot, a partíciók már nem változnak, a centroidok helyzete nem változik. A végeredmény K darab, jól szeparálható klaszter lesz.

Ez az algoritmus kiválóan alkalmas minták illesztésére, továbbá felhasználható képek színintenzitás alapú szegmentálására is, amennyiben a vizsgált kép színei között jó hasonlósági függvény definiálható. Többek között a SIMPLYcity [14] rendszer is ezt az algoritmust alkalmazza a képek régiókra bontása során, a szegmentáláskor LUV színteret alkalmaz.

3.6.2 Book Cover Recognition project by Wang

Ez a rendszer [28] a Columbia Egyetem egy hallgatója által készült. Ebben a rendszerben az előfeldolgozás során a készítő bilaterális szűrést, Canny [39] éldetektálást, Hough transzformációt, majd küszöbölés alapú vágás alkalmaz. A leírókat jellemző pontokból számolt, korábban említett SIFT algoritmussal nyeri, és a felismeréshez lineáris Support Vector Machine [29]-t alkalmaz. Minden könyvhöz 15 mintát tárol az adatbázisban, paraméterként 5-szörös keresztvalidációt használ. Állítása szerint a rendszer 99% körüli hatékonysággal működik.

A SVM, magyarul szupport-vektor gép az egyes osztályok közötti szeparáló hipersíkok közötti eltérés maximalizálására törekszik. A döntési függvényt a tanuló adatok részhalmazai, a szupport vektorok határozzák meg. Ez az egyik leghatékonyabb osztályozási módszer, az osztályozás során csak azokat a tanuló adatokat veszi

figyelembe, amelyek az adott döntési határok közelében helyezkednek el. A módszer során amennyiben az algoritmus nem talál egyértelműen definiálható szeparáló hipersíkot, egy nemlineáris transzformációval bővíti a teret, bevezet egy új dimenziót, majd így állítja elő az új osztály határokat. Ugyanakkor fontos kiemelni, hogy az SVN instabil a kisméretű tanító adatokra, az optimális hipersíkok megtalálása szempontjából.

3.6.3 BookSpineOCR - Hybrid Approach to Mobile Book Spine Recognition

Ez a rendszer [30] a képi jellemzők és az optikai karakterfelismerést ötvözi annak érdekében, hogy a kiértékeléskor minél nagyobb pontosságot tudjon elérni. A rendszert alapvetően könyv gerinc felismerésre fejlesztették ki, a betűk lokalizálására Maximally Stable Extremal Regionst [31] (MSER) és Stroke Width Transformot [32] (SWT) alkalmaznak. A szegmentálás él alapon történik, Canny eldetektálót használnak az MSER által megtalált betűk vágásához, majd a Tesseract keretrendszer segítségével értékelik ki az eredményt. A Tesseract által megkövetelt szótárat az adatbázisban tárolt dokumentumok szövegeiből készítik elő.

A keresés gyorsításához a szótár szavaiból felépítenek egy adatbázist (W), amely minden szóhoz eltárolja azon dokumentumok azonosítóit, amelyek azok metaadataiban megtalálhatóak. Tehát minden $w_i \in$ Wszó esetén meghatározzák ezen $L(w_i)$ mutatót. Keresés során a felismert $q_j \in$ Qszóhoz megkeresik a szótárban található m_j szót, teljes vagy Nearest Neightbour alapon, a legközelebbi szót kiválasztva. Az egyes k-adik találat jóságának meghatározását a következő képpen végzik:

$$s_t(k) = \sum_j I(k \in L(m_j)), \tag{9}$$

ahol I( ) az indikátor függvény, értéke 1 ha $L(m_j)$tartalmazza k dokumentumot, egyébként nulla. Az egyes dokumentumok jóságát ezután terminus-frekvencia és az inverz dokumentumgyakorisággal súlyozzák. A terminus frekvenciát a szó előfordulásainak száma a vizsgált dokumentumban osztva a dokumentum szavainak száma adja, számítása a következő formulával történik:

$$tf(t,d) = 0.5 + \frac{0.5 \times f(t,d)}{\max\{f(w,d) : w \in d\}} \tag{10}$$

ahol $f(t,d)$ megadja t szó előfordulásainak számát d dokumentumban. Az idf-et a terminus egész korpuszon vett ritkasága alapján lehet számolni a következő módon:

$$\mathrm{idf_i} = 1 / \log(\mathrm{n} / \mathrm{df_i}) \tag{11}$$

ahol df_i azon dokumentumok száma, amelyek tartalmazzák i terminust. A súly ezen két érték skaláris szorzata adja meg.

A könyvgerincek szín alapú jellemzőit SURF [33] algoritmussal nyerik ki, amely nagyban épít a SIFT algoritmus megközelítésére, ugyanakkor pontosságban és robusztusságban jobb teljesítményt ígér. A keresés gyorsításához Vocabulary Tree [34]-t alkalmaznak.

3.7 Értékelés

Ebben a szekcióban ismertettem a könyvborító felismerés általános megközelítéseit, így mint az előkészítési fázis feladatait, a leírók kinyerésének több módját, illetve a kinyert jellemzők párosításainak problematikáját. Kifejtettem a képi adatbázisok építésének általános megközelítéseit, a keresés gyorsításának több féle megközelítését, a keresés során fellelhető problémákat és lehetséges megoldásait. Részletesen kifejtettem az optikai karakter felismerés feladatait, esetleges felhasználását illetve az ehhez kapcsolódó megoldandó problémákat. Bemutattam három hasonló rendszert és ott alkalmazott algoritmusokat.

Mivel a rendszer működését tekintve fontos szempont a sebesség, ezért az egyes könyvborító osztályozását a SIMPLYcity [14]-hez hasonlóan, két lépésben kívánom elvégezni. A rendszerben szín-elrendezést alapú előválogatást kívánok végezni, LAB színtérben. Mivel az LAB színtér L csatornája az adott A-B csatorna által meghatározott színnek csupán a fényességét szabályozza, adott esetben elhagyható a vizsgálat szempontjából, ezzel csökkentve az előszűrés során vizsgálandó vektorok dimenziószámát.

Mivel a rendszerben nem csupán hasonlóság alapon kívánok találatokat megjeleníteni, mint a szín-hisztogram alapú példában, hanem konkrét egyezést, ezért az a módszer már csak számítási komplexitása miatt sem alkalmazható. Helyette jellemző pont alapú leírót kívánok alkalmazni, azon belül az SIFT-hez nagyon hasonló, Haar wavelet válasz alapú SURF leírót.

A leíró vektorok osztályozásához jelen esetben a SVN nem nyújtana elég hatékony megoldást, egy részt a számításigénye, másrészt a tanítóadatok mennyisége miatt. Mivel egyes könyveknél csupán 1 adott könyvborítót kívánok tárolni, ezért kiválasztó K-Nearest Neighbour[35] osztályozás tűnik a legcélravezetőbbnek. Ez az algoritmus lehetővé teszi, hogy a tanítás fázisát futásidőben végezzem, így a memória használat jelentősen csökkenthető.

A hibás osztályozások javítására a Tesseract keretrendszert kívánom alkalmazni, melyhez az egyes könyvek metaadatainak dekompozíciójával kívánok szótárat felépíteni. A találatok jóságának vizsgálatára a BookSpineOCR-ben alkalmazott vizsgálathoz hasonló megoldást kívánok implementálni.

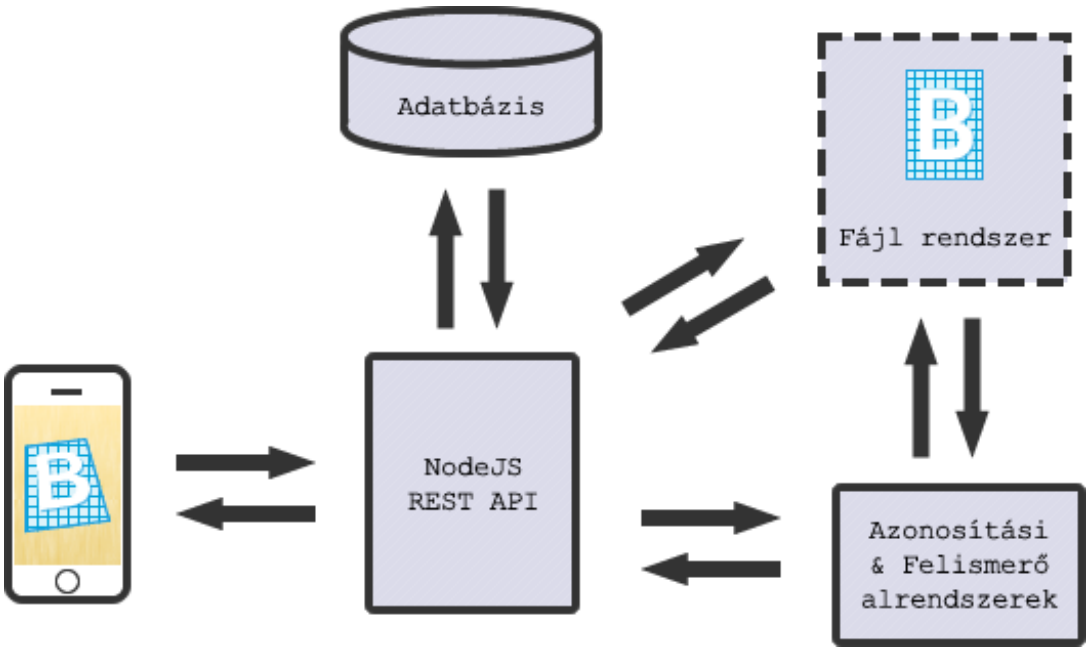
4. RENDSZERTERV

Mint az előbbiekben kifejtettem a rendszer egy alapvetően webes technológiákon alapuló katalógus-alkalmazás, amelyben valós térből kinyert információk alapján történő azonosítási folyamaton van a hangsúly. A rendszer egy iOS platformon készült kliens, egy NodeJS alapon készült, HTTP-n kommunikáló REST API-ból, illetve magát az információ kinyerést végző C++ nyelven íródott alrendszeren alapul. Az adatbázis saját megvalósításon alapszik, az adatok séma nélküli JSON formátumban kerülnek tárolásra.

4.1 Architektúra

Az architektúra 4, jól elszeparálható részre bomlik: iOS kliens, API, Azonosítási és Felismerő alrendszer továbbá az Adatbázis. A rendszer tervezésekor Three-tier tervezési mintát vettem alapul. A tárolt leírók a fájlrendszeren találhatóak, az adatbázisban csupán az ezekhez tartozó azonosítók és egyéb információk vannak tárolva. Az Azonosítási és Felismerő alrendszer magát a fájlrendszert használja a leírók betöltéséhez, illetve a feltöltött képekről kinyert leírók is a fájlrendszeren tárolódnak. Az API az alapvető lekérdezéseken kívül, mint köztes réteg vesz részt az azonosítási folyamatban.

Az architektúra felépítését az alábbi ábra szemlélteti, ahol a nyilak az egyes alrendszerek közötti kommunikációs csatornákat szemléltetik:



6. ábra, a rendszer felépítése

4.2 Azonosítási alrendszer

Az azonosítási alrendszer feladata a felhasználó által készített fotó feldolgozása, az objektum azonosításához szükséges leíró kinyerése. A bemenet egy RGB kép, a kimenet pedig egy N dimenziós, dimenziónként M darab leíró vektort tartalmazó vektor.

4.2.1 Előfeldolgozás

Az előfeldolgozás feladata, hogy a valós térben elhelyezkedő könyvborítót olyan feldolgozható adathalmazra transzformálja, hogy lehetővé váljon jó minőségű leírók kinyerése. Az előfeldolgozás több részből áll: zajok eltüntetése, éldetektálás, szegmentálás és síkba transzformálás.

4.2.1.1 Medián szűrés

A medián szűrés egy nem lineáris szűrés. Matematikában művelet szekvenciákat akkor nevezzük lineárisnak, ha teljesül rájuk a konkatenitás és kojuktivitás matematikai tulajdonság.

A szűrés során egy 3x3-as maszkot mozgatunk a gép felett. A maszk értékei az adott pixel és környezetének intenzitásait veszi fel, az új pixel intenzitásának ezen értékek sorba rendezése után kapott sorozat középső elemét rendeljük. A mediánszűrő kiválóan alkalmas a kiugró intenzitású zajok eltüntetésére, hiszen ezen értékek a sorba rendezett sorozat elején vagy végén helyezkednek el. Az algoritmus működését gyorsíthatjuk, hogyha a sorba rendezés során használt algoritmusunkat úgy módosítjuk, hogy csupán addig csak a lépésig történjen a rendezés, amíg az eredményhalmazunk 5. elemét meg nem találtuk.

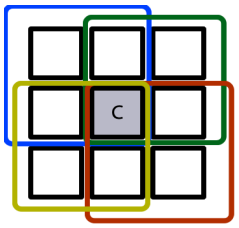
4.2.1.2 Morfológiai szűrő

A matematikai morfológia egy d-dimenziós euklideszi vektortér tetszőleges részhalmazain értelmezett művelet, ahol X és Y képpontok a V halmaz részhalmazai, kétdimenziós kép esetében $V = Z^2$. Két alapműveletet [36] különböztetünk meg: dilatáció és erózió. Ezen két alapművelet kombinációjából további műveletek írhatóak le, így mint nyitás, zárás, a morfológiai szűrés. A műveletvégzés általános esetben bináris képeken történik, egy szerkesztő elem felhasználásával. Az erózió során a kép szegmenseinek széleit csökkenjük, fogyasztjuk a képet. Az eredményként előálló képen a szegmensek mérete csökken, az eddig zajként jelentkezett pontok eltűnnek. A dilatáció során a szegmenseket hizlaljuk, ennek következtében a szegmensek belsejében lévő lyukak mérete csökken, esetleg megszűnik. A morfológiai nyitást egy erózió és dilatáció egymás utáni végrehajtása adja, zárás esetében a sorrend fordított. A nyitás és

zárás művelet idempontens, egymás utáni ismétlésük nem változtatnak a képen. A morfológiai szűrő a nyitás és zárás egymás utáni végrehajtását jelenti, a zajok és lyukak eltüntetése, illetve a szegmensek kontúrjainak simítását eredményezi.

4.2.1.3 Kuwahara-Nagao szürés

A Kuwahara-Nagao [37] szűrés ugyancsak egy nem lineáris szűrés. Az algoritmus egy 3x3-as maszkalt operál, amelyben további 4, 2x2-es átfedő ablakot definiál. Eredményként a minimális varianciájú, vagyis a maximálisan homogén ablak átlagos intenzitás értéke lesz az középső pixel új intenzitás értéke.



7. ábra, a Kuwahara-Nagao algoritmus szűrő maszkja az átfedő ablakokkal

Az algoritmus alapvetően színes képek feldolgozására ajánlott, az eredményként előálló kép hasonló jelleget vesz fel, mint bilaterális szűrés esetében, vagyis a homogénebb régiók összemosódnak, de az élek továbbra is jól kapcsolódnak egymáshoz. A bilaterális szűréshez képest szignifikáns különbséget a sebesség, illetve a zajtűrő képesség szemlélteti. [38]

8. ábra, a) az eredeti kép b) Kuwahara-Nagao c) bilaterális szűrés [38]

9. ábra, a) az eredeti kép b) Kuwahara-Nagao c) bilaterális szűrés [38]

Összességében ez a szűrés önmagában kiválthatja a medián és morfológiai szűrés lépéseit.

4.2.1.4 Symmetric Nearest Neighbour

Ahhoz hogy képesek legyünk a valós térben elhelyezkedő, könyv alakú objektumokat további feldolgozás céljából használni, szükséges, hogy képesek legyünk ezt, mint összefüggő szegmenst kivágni a képünkről. A szegmensek meghatározására egy lehetséges megközelítés az éldetektálás, ahol a képeket diszkrét tartományban deriváljuk, más szóval előállítjuk azok grádiens képét. Erre a feladatra szolgál megoldással a Symmetric Nearest Neighbour[37] algoritmus, más néven differenciáló éldetektálás, amely a medián szűrőhöz hasonlóan egy nem lineáris szűrés. Az algoritmus egy 3x3-as maszkkal dolgozik, ahol az átlóirányú komponensek pixelintenzitás-különbségeinek maximuma adja az új pixel intenzitás értékét. Az algoritmus előnye, hogy mindenhol jól kapcsolódó éleket produkál, továbbá eltünteti az egyes homogén régiókban felelhető hibákat.

A maszk:	Az új pixel intenzitás értéke:
P1 P2 P3 P8 *X* P4 P7 P6 P5	$X = \max(\lvert P1-P5\rvert,\lvert P2-P6\rvert,\lvert P3-P7\rvert,\lvert P4-P8\rvert)$ (Csebisev távolság)

Az algoritmus hátránya azonban, hogy egyes élek megvastagodva jelennek meg, amennyiben az adott élpixel környezetének grádiens nagysága is meghaladja a küszöbértéket, ezt többpixeles válasznak nevezzük.

4.2.1.5 Canny éldetektálás

A Canny[39] éldetektálás az előző módszerben is tapasztalható többpixeles válasz esetére próbál megoldást találni. Több pixeles válasz akkor jelentkezik, amikor a kép egyes pontjaiban található intenzitás különbségek a szomszédos pixeleken is jelentkeznek, vagyis adott esetben az objektumunk határoló vonala vastagként jelenik meg. Ez az algoritmus az él minél pontosabb lokalizációjára törekszik az által, hogy kiválasztja minden gradiens pontban, azok lokális maximumát. Első lépésben Gauss szűréssel készíti elő a képet, majd a gradiens pontok lokalizációja után, hiszterézises küszöböléssel tünteti el a keletkezett zajokat. Az utolsó lépés abból a feltevésből indul ki, hogy a fontos élek folytonosan helyezkednek el, és átlag intenzitás értékeik konvergálnak. Mivel magas számítási igénnyel rendelkezik, valós időben nem minden esetben alkalmazható.

4.2.1.6 Hough transzformáció

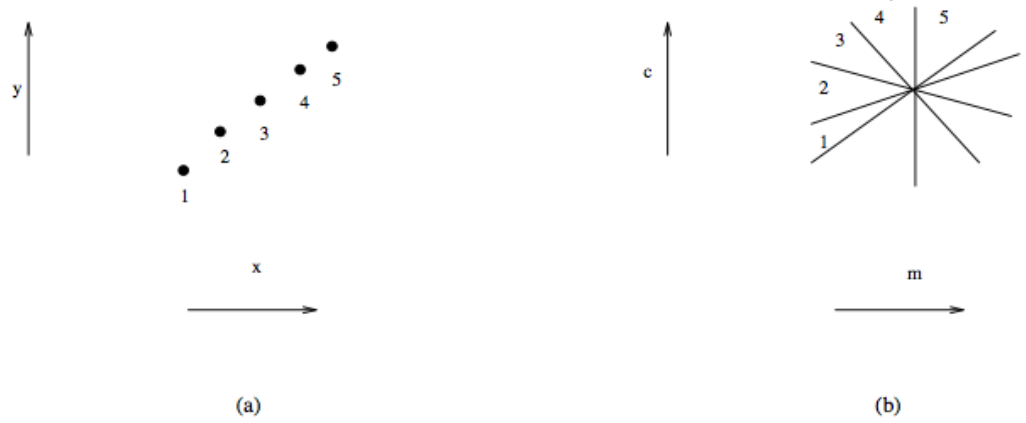
A Hough transzformáció [40] az egyenes egyenletéből indul ki:

$$y = mx + c \tag{12}$$

ahol y az egyenes y irányú komponense, m a meredekség, x az x irányú komponens és c az a konstans, amivel az egyenes y irányban el van tolva. Ezt az egyenletet átalakítva megkapjuk az alábbi összefüggést:

$$c = (-x)m + y \tag{13}$$

vagyis megkapjuk az egyenes c-m térbeli reprezentációját. Ez az egyenlet nem más, mint egy X-Y térbeli pont M-C térbeli transzformációja, amely egyenesként reprezentálódik. Ahol ezek az egyenesek találkoznak az M-C térben (M', C'), ott egy egyenes található az X-Y térben. A leképzést az alábbi ábra szemlélteti:



10. ábra, a) egy X-Y térbeli egyenes pontjai, b) ezen pontok M-C térbeli egyenesként leképzett reprezentációi, az X-Y térbeli valós egyenest ezen projekciók metszésponja adja [36]

A probléma egyedül ott áll föl, ahol olyan egyenes pontjait vetítjük az M-C paraméter-térbe, ahol az eredeti egyenes párhuzamos az Y tengellyel, hiszen ebben az esetben végtelen meredekségről beszélünk. Ezen hiba elkerülésére megoldás a következő, módosított egyenlet:

$$p = x\cos\theta + y\sin\theta \tag{14}$$

ahol θ az *p* egyenes x-tengellyel bezárt szöge. Ebben az összefüggésben *p* és θ kezdeti értéke végtelen, viszont θ értéke közvetlenül, az éldetektálás során előálló gradiens irányokból számolható.

Az algoritmus során először elő kell állítanunk a paraméter-teret, zérus kezdeti értékekkel. Második lépésben az előzetesen éldetektált kép minden élpontjára meg kell határoznunk az előző egyenletbeli *p* és θ értéket, amellyel frissíthetjük a paraméter-teret: $P[\theta, p] = P[\theta, p] + 1$.Utolsó lépésként pedig össze kell gyűjtenünk a paraméter-

tér lokális maximumait, ezek fogják reprezentálni az X-Y térben található egyeneseinket.

4.2.1.7 Otsu binarizálás

A valós térben elhelyezkedő, könyv alakú objektumok összefüggő szegmensként való kivágására kiváló algoritmus lehet az Otsu binarizálás [41], amennyiben az vizsgálandó objektumunk és a háttér között szignifikáns intenzitásbeli különbség fedezhető fel. Az algoritmus abból indul ki, hogy a homogén régiók varianciája kicsi. Az algoritmus első lépésben elkészíti a bemeneti, szürkeárnyalatos képünk intenzitás hisztogramját. Második lépésben ezt a vektort normalizálja, vagyis előállítja az egyes pixel intenzitás értékhez tartozó előfordulási valószínűséget tartalmazó új hisztogramot. Harmadik lépésben ezen az új hisztogramon végigiterálva megállapítja azt a küszöbszámot, amely maximalizálja az előtér és háttér közötti varianciát.

4.2.1.8 Otsu binarizálás javítása

Az Otsu algoritmus nem mindig képes jó megoldással szolgálni, az eredmény függ az előtér és a háttér közötti átlag intenzitás különbségektől. Az előtér jelen esetben az objektumunkat jelenti. Egy lehetséges javítás lehet, ha a binarizálás lépése előtt a képünk közepéről származó részkép vágásának felhasználásával megállapítjuk annak átlagos intenzitás értékeit és szórását, RGB csatornánként. Mivel követelmény, hogy a detektálandó könnyborítónak a kép középső régiójában kell elhelyezkednie, ezért egy olyan eredmény-vektort párt kapunk, amely jellemezheti az adott könyvet. A következő lépésben egy küszöbszám felhasználásával megvizsgálhatjuk ezen értékeket, amennyiben valamelyik csatornán a szórás meghaladja a küszöbszámunkat, az adott csatorna szürkeárnyalatúként felhasználva javíthatja az Otsu algoritmus eredményeként fennálló szegmentálásunkat, hiszen az adott csatorna jobban jellemzi a kivágandó objektumunkat, mint adott esetben a három csatorna átlagából előállított szürkeárnyalatú képünk.

4.2.2 Tengelyek becslése főkomponens analízissel

A főkomponens analízis [42] az osztályozás altípusaként definiálható regressziós algoritmusok közé tartozik. A regresszió célja az adatban rejlő sajátosságok modellezése. A főkomponens analízis során az adott mintánk dimenzióinak számát csökkentjük, térben végzett transzformációval.

Az algoritmusban első lépéseként a bemeneti vektorok értékeinek normalizálása történik, ez az adott dimenzió pontjainak és a dimenzió átlagos értékének kivonásával történik. A normalizált dimenzió-vektortok között a következő lépésben meghatároz egy kovariancia mátrixot. A kovariancia két különböző vektor közötti függőséget méri,

a két vektor együttes mozgását adja. Adott N dimenziós vektorok esetében a számítandó kovariancia mátrixok száma (N * (N-1) / 2). Számításának módja a következő:

$$\mathrm{Cov}(X,Y) = \frac{1}{N-1}\sum_{i-1}^{N}(X_i - \overline{X})(Y_i - \overline{Y}) \tag{15}$$

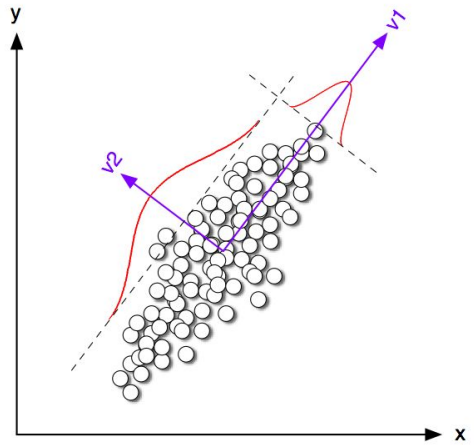
ahol N, az adott vektorok hossza. Könnyen belátható hogy egy adott X vektor önmagával alkotott kovarianciája a vektor szórásnégyzetét adja. A kovariancia mátrix megalkotása a következőképpen történik:

$$C^{n\times n} = (c_{i,j},\ c_{i,j} = \mathrm{Cov}(Dim_i,\ Dim_j)) \tag{16}$$

ahol *n* a bemeneti vektorok dimenziószáma.

A következő lépésben az előállt kovariancia mátrix sajátértékeinek és sajátvektorainak számítása történik, illetve e sajátvektorok sajátértékek szerinti rendezése. Az így előálló komponensek által alkotott lista jól jellemzi a bemeneti adathalmazunkat, a komponensekhez tartozó sajátértékek pedig azok szignifikanciáját határozzák meg. Ezen rendezett lista egy meghatározott küszöbérték alatt található vektorainak elhagyásával tudjuk a bemeneti adathalmazunkból nyert leíró dimenziószámát csökkenteni. Ugyan ez adatvesztéssel jár, de a szignifikancia figyelembe vételével az előálló, csökkent dimenziószámú halmazunk jól közelíti az eredetit, továbbá adott esetben jelentős számítási teljesítménynövekedést okozhat.

Az algoritmust számos helyen alkalmazzák mintafelismeréstől tömörítésig, jelen feladatban az a tulajdonsága kerül felhasználásra, hogy bemeneti, kép koordinátákat tartalmazó ponthalmazból számított első és második főkomponense megadja a ponthalmaz két tengelyét, így az objektum kivágása előtt képessé válik - annak orientációjának függvényében-, módosítani a leképzés során használt modellt.



11. ábra, főkomponens analízis eredménye kétdimenziós bemeneti vektorok esetében [47]

4.2.3 Jellemző pont detektálás

A rendszerben a könyvborítókhoz tartozó leírókat az azokon detektált jellemző pontok felhasználásával számítom. Ezen pontok megtalálására számos algoritmus létezik, ezek közül az egyik a Harris [48] operátor.

Harris [48] operátor esetében első lépésként a kép deriváltjainak közelítését kell elvégeznünk minden pontban, vagyis éldetektálnunk kell a képet. Javasolt ezt megelőzően egy Gauss vagy átlagoló szűrővel simítani a képet, hogy a környezeti zajokat minimalizáljuk. A következő lépésben egy gradiens momentum mátrixot kell számítanunk valamennyi pont adott 2n+1 x 2n+1-es környezetében az alábbi formulával:

$$M_H = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \tag{17}$$

ahol I_x az x irányú gradiens nagyság, I_y pedig az y irányú. A momentum mátrix sajátvektorai él irányt, saját értékei él nagyságot jelölnek, amennyiben mindkét sajátérték elég nagy, tároljuk el mint lehetséges sarokpont. Érdemes a tároláshoz egy rendezett listát használni.

A következő lépésben ezen listán végighaladva távolítsuk el azokat a pontokat, amelyek egy erősebb detektált pont közelében találhatóak, vagyis egyes detektált pontok környezetében válasszuk ki azok lokális maximumait. Utolsó lépésben az elkészült, pontokat tartalmazó képen végezzünk el egy küszöbölést.

4.2.4 Leíró kinyerés

A leírók kinyerésekor fontos, hogy mint a jellemző detektornak, mint a leírónak invariánsnak kell lennie rotációra, eltolásra és skálázásra. A legtöbb detektort már ezen elvek alapján tervezik, ugyanakkor a Harris esetében a skálázás problémás. Ahhoz, hogy ezt javítsuk, érdemes a kép Gauss piramisain is elvégezni a pontdetektálást, a korszerűbb módszerek már automatikusan megtalálják a skálázási faktort. Fontos, hogy maguk a leírók is invariánsak legyenek, mivel a párosítás ezek vizsgálatával történik.

4.2.4.1 SURF leíró

A rendszerben SURF [33] leírót alkalmazok, amely a SIFT leíróhoz képest nagyobb pontosságot, robusztusságot és megismételhetőséget kínál. A leíró invariáns forgatásra, skálázásra és eltolásra is. A jellemző pontdetektálás Hessian mátrix alapon történik, a keresés a kép integráltján számolt másodrendű parciális deriváltakkal történik .

Szemben a SIFT-el, ahol Gauss piramisokon történik a jellemző detektálás, itt egy növekvő méretű maszkot mozgatnak az integrált képen, ez minden iterációban hasonló eredményhez vezet, mint a SIFT esetében, viszont tekintve, hogy az integrált kép esetében alkalmazhatunk egy két dimenzós lookup table-t, jól párhuzamosítható lesz. A maszk mérete 9x9, 15x15, 21x21 és 27x27-es méretig nő. A leíró számolása előtt Haar wavelet transzformációval határozzák meg a pont orientációját, egy 6s sugarú környezetben, ahol s az a skála faktor, ahol a pont elhelyezkedik. A leírót ezután a pont 20s méretű környezetében számolják, amelyet további 4x4-es blokkokra bontják, és minden blokk esetében kiszámolják azok x és y irányú Haar wavelet válaszát. Egy-egy blokk leíró vektora a következő lesz:

$$\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \tag{18}$$

ahol d_x az x irányú, d_y az y irányú wavelet válasz. Ezek összevonásával végeredményként egy 64 dimenziós vektort kapnak. Mivel a wavelet transzformációk megvilágításra invariánsak és a leíró egységvektorként reprezentálódik, a kontrasztváltozás is jól kezelhető, továbbá a módszer zajtűrő képessége is jobb.

Összegségében kijelenthető, hogy ez a módszer a SIFT-el összehasonlítva jelentősen gyorsabb detektálást és jellemző párosítást tesz lehetővé, a kinyert jellemzők robusztusak, és a leírók dimenziószáma is a felé csökkent.

4.3 Adatbázis alrendszer

Mivel adott esetben az adatbázis több ezer könyvborítót és ahhoz tartozó metaadatot tartalmazhat, szükséges, hogy valamilyen előválogatást végezzek. A rendszerben szín-elrendezés alapú megközelítést alkalmazok az osztályozás során szükséges értékkészlet méretének csökkentésére, ezzel az osztályozás problémáját két részre bontom: előválogatás és felismerés.

4.3.1 Klaszterezés LAB színtávolság alapján, adaptív mintavételezéssel

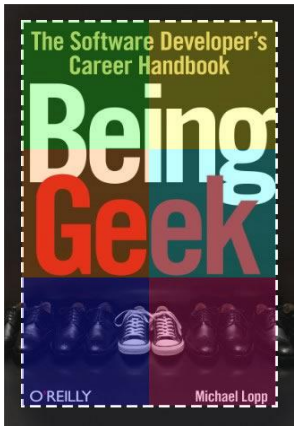
Az előválogatáshoz szín alapú jellemzőket hasonlítok össze. Ehhez minden tárolandó könyvborítót 6 darab szegmensre osztom, majd ezeknek veszem LAB szintértben számított átlagát. Minden könyvborító szélétől egy 40 pixeles behúzást alkalmazok, hogy az előfeldolgozás végén elvégzett transzformáció során a minták szélén jelentkező torzulás a lehető legkevésbé befolyásolja az eredményt. Az RGB színek LAB-be történő átszámításához először azt az XYZ színtérbe kell átszámítani az alábbi módon:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{b_{21}} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

(19)

Ez után történik meg az LAB térbe való átszámítás, vagyis a színtér normalizálása, két konstans felhasználásával [43].

Mivel az LAB színtérben az L csatorna csupán az adott szín fényességét tárolja, így figyelmen kívül hagyható a vizsgálat során. Emiatt csupán a 3 * 2 darab, két dimenziós vektort használok fel az értékkészlet leszűkítésére, a rendszerben a vektoriálisan legközelebb található 10 darab könyvet veszem ki, majd továbbítom a felismerő alrendszer felé. Az eljárás során alkalmazott maszkot a következő ábra szemlélteti:



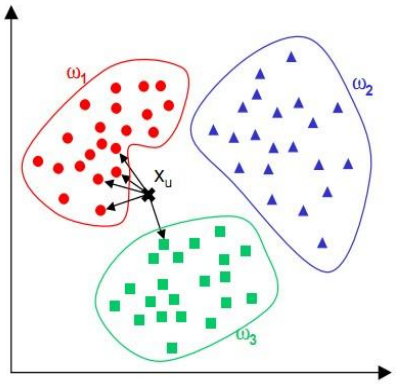
12. ábra, a színes téglalapok egy-egy vizsgált régiót reprezentálnak

4.4 Felismerő alrendszer

Ebben az alrendszerben történik meg a kinyert leíróvektor és az adatbázisban tárolt leíróvektorok párosítása. A párosítandó könyvek száma az előző pontban alkalmazott csoportosítás eredményeként csupán egy részhalmaza a teljes adatbázisban tárolt könyvek számának. A párosításkor egy lineáris keresés történik, a legjobb jósággal rendelkező minta kerül kiválasztásra, mint találat, amennyiben egy előre definiált küszöbszám szerint elfogadhatónak mondható.

4.4.1 K-nearest Neighbour osztályozás

A K-nearest Neighbour egy úgy nevezett lusta tanuló osztályozás, ahol a modellépítés folyamata kimarad. Az algoritmus abból a feltételezésből indul ki, hogy a hasonló tulajdonságokkal rendelkező objektumok hasonlóak, ezek vizsgálatára jellemző Euklideszi vagy szövegek esetében Levenshtein távolságot alkalmaz. Mivel a modell építés lépése kimarad, ezért a tanulási lépés kimarad, elegendő csupán a leíróink tárolása.



13. ábra, kNN, X_u a piros osztályba kerül [46]

A predikciós fázisban az algoritmus megkeresi a k legközelebbi szomszédot, és többségi szavazással választ. Az a címke lesz a nyertes, amely a legtöbb szavazatot kapja, vagyis: vesszi d dokumentum k legközelebbi dokumentumainak halmazát (N_k), majd vizsgálja ezek osztályait és kiválasztja azt az osztályt, amelyikhez a legtöbb dokumentum tartozik. Ez az algoritmus regresszióra is használható, a minta átlagát adja eredményül, osztályozás esetén pedig a relatív gyakoriságot. Ezen tulajdonsága miatt univerzális approximátornak is nevezik: végtelen sok tanító pont esetén bármilyen függvényt tetszőlegesen jól közelít.

Ugyanakkor az algoritmus érzékeny a független attribútumokra, ezért ügyelni kell, hogy ezeket lehetőleg külön-külön osztályozzuk. Másik probléma, hogy érzékeny különböző mértékekre, vagyis az algoritmus nem skála invariáns, ekkor alkalmazzunk normalizálást, illetve fontosság szerinti súlyozást. A *k* szám megválasztása is problémás lehet, amennyiben túl alacsony vagy túl nagy számot választunk, befolyásolhatja az osztályozásunk jóságát.

A kNN-ről elmondható tehát, hogy az osztályozási idő lineárisan aránylik a tanuló adat méretével, ugyanakkor jól skálázható, hiszen nem függ az osztályok számától, nem kell az egyes osztályokat előre betanítani. Sok tanítóadat esetén nincs szükség jellemző kiválasztásra, viszont az osztályok befolyásolhatják egymást. Mivel jelen esetben SURF leírók között végzek osztályozást, egyik probléma sem áll fent.

4.4.2 OCR alapú visszakeresés

Elegendő képi leíró esetén, illetve hibás osztályozásból adódó pontatlan találatok kezelésére kívánok egy optikai karakterfelismerésen alapuló korrekciót végezni. Ehhez szükséges egy hasítótábla megléte, amely tárolja az egyes kulcsszavakat, illetve azoknak a dokumentumoknak a halmazát, amelyek tartalmazzák a dokumentumot. Az egyes kulcsszavakat a BookSpineOCR rendszerben alkalmazott, illetve egyéb keresőrendszerek által is általánosságban alkalmazott terminus frekvencia és inverz dokumentum frekvencia felhasználásával kívánom súlyozni.

Fontos, hogy az OCR lépés nem mindíg vezet jó eredményre, így szükséges, hogy olyan szó egyezés vizsgálatot végezzek, amely az esetleges 1-2 hibás karakter osztályozásból adódó hibát képes kezelni. Erre egy ideális megoldás, ha Levenshtein távolsággal küszöbölök 1-1 azonosított szót, mégpedig 2-3 permutáció megengedésével.

4.5 API

Ez az alrendszer felelős a kliensalkalmazás, az adatbázis, továbbá a natív nyelven megírt, képfeldolgozó alrendszerek kommunikációjának közrefogására. A NodeJS egyik nagy előnye, hogy a Google v8-as JavaScript motorjában alkalmazott Just In Time fordító közel olyan sebességet képes biztosítani, mintha maga a modul is natív nyelven íródott volna, ugyanakkor a platformfüggetlenség, könnyű skálázhatóság és a dinamikus nyelvek előnyei mind megtalálhatóak benne. További előnyt ad, az ún. nem-blokkoló I/O, illetve az aszinkron programozás lehetősége.

Az API-nak csak egy feladata van: a bejövő képek feldolgozása, illetve az adatbázisban történő keresés vezérlése. A keresés során alkalmazott folyamatot a melléklet 1. pontja szemlélteti.

4.6 Kliens

A kliens egy két tabból álló alkalmazás, két menüponttal: Kedvencek és Keresés. Az alkalmazás indulása után a felhasználó a Keresés felületre jut. A kedvencek közé elmentett könyvek az iOS SDK Core Data nevű Objektum Gráf kezelő keretrendszere által elfedett perzisztens SQLite adatbázisába kerülnek elmentésre.

4.6.1 Keresés

Ezen a felületen tud a felhasználó egy adott könyvről fotót készíteni, majd beküldeni az API-nak, hogy az felismerje. Amennyiben a felismerés sikeres volt, a felhasználó a “könyv nézet” képernyőre kerül, ahol láthatja a könyv borítóját, címét, szerzőjét, kiadóját, illetve lehetősége van kedvencekhez adni a könyvet.

4.6.2 Kedvencek

Ez egy Table View lista, amelyben a felhasználó megtekintheti korábban detektált és kedvencnek elmentett könyveinek listáját. Egyes bejegyzés kiválasztása után a felhasználó a Keresés során is felhasznált “könyv nézet képernyőre kerül”, ahol lehetősége van a könyvről tárolt információkat megtekinteni, eltávolítani azt a könyvjelzők közül, továbbá kézzel bővítenie a listát. Új dokumentum felvitele esetén az API gondoskodik ezen objektum metaadatainak tárolásáról, az LAB klasztervektor kinyeréséről, továbbá a borító leíróinak fájlrendszerre történő mentéséről.

5. IMPLEMENTÁCIÓ

A rendszer dinamikus és natív nyelveket vegyít, a kommunikáció HTTP protokolon, illetve a standard kimeneten zajlik. Az egyes modulok jól elkülönülő komponensekre bomlanak. Az API esetében ez 4 komponensre bomlik, az azonosítási és felismerő alrendszerek esetében 3 osztály, egy program végrehajtó egység, illetve egy különálló program modul található.

Az implementáció során GIT nevű verziókövető rendszert alkalmaztam, a natív modulok kifejelsztése Xcode IDE környezetben, míg az API Sublime Text 2 nevű, moduláris szövegszerkesztőben készült.

5.1 Azonosítási és felismerő alrendszerek

Ez a két alrendszer egy C++-os modulban került implementálásra. A modulban négy alap funkciót különböztetek meg, amelyeket a program megfelelő paraméterekkel történő meghívásával lehet elérni. Ezen négy funkció paraméteres hívása a következő:

1. -ocr kép-útvonal
2. -cluster kép-útvonal
3. -recognize kép-útvonal [[lehetséges-könyv-azonosító] ...]
4. -store kép-útvonal új-azonosító

Ezen kívül az alrendszeren belül található swt könyvtárában elehelyeztem egy Stroke Width Transformation-n megvalósító programot, amely első paraméterként egy feldolgozandó képet, másodikként pedig a kimeneti kép címét várja.

5.1.1 Optikai Karakter Felismerés

Ahhoz, hogy ez a modul megfelelő minőségű karakterfelismerést végezzen, bemenetként már egy Stroke Width Transform-on átesett képet adok meg. Az API-ban implementált Wrapper már ez az elvet követi, a képről kivágott könyvborítót átadja az SWT-t végző programnak, majd ez után hajtja végre a karakterfelismerést. A karakter felismerést végül a Tesseract keretrendszer végzi.

5.1.2 Klaszterezés

Ezen funkció első lépésként megkeresi a borítót reprezentáló régiót a képen, kivágja, majd térbe transzformálja. Ez után a kivágott könyborító régiókra bontását végzi, előállítja az egyes régiók RGB színeinek LAB színtérbe történő átszámítását, végül és az átlagos A-B értékekből képez egy 3*2 darab 2 dimenziós vektort, és kiírja a standard kimenetre. Az algoritmus megvalósítása a Preprocessor osztály implementációs fájljában található. A rendszerben tárolt klasztervektorok értékei dimenziónként 0 és 255 közötti tartományra vannak normalizálva, a tárolás lebegőpontos formátumban történik.

5.1.3 Felismerés

A felismeréshez az előzőekben kivágott borítót használom, ezen borítókon végzek pont detektálást, majd leíró számolást. A képek párosítása egy cimkés, távolsággal küszöbölt kNN osztályozással zajlik, egy-egy leíró esetében két legközelebbi learó pontot veszem. A párosított leírók elfogadhatóságára alkalmazott küszöb értéket az első és második legközelebbi pont távolságának aránya adja, amely a rendszerben 0,7. Ha az arány fennáll, egy adott találatot felveszek a jó találatok listájába, A funkció meghívásakor a harmadik paramétertől kezdve kerülnek átadásra az előzetesen közeli objektumnak ítélt borítók azonosítói, ezen azonosítók felhasználásával kerülnek beolvasásra az előzetesen elmentett, borítókhoz tartozó leíró vektorok.

5.1.4 Tárolás

Az adatbázisba történő beszúrás után ezen funkció feladata, hogy a második paraméterként átadott kép jellemző pontjait és leíróit kinyerje, majd az azonosítóhoz tartozó XML fájlt létrehozza, és az adatokat elmentse. Mentés után a klaszterezésnél alkalmazott eljárással kiszámolja a borítóhoz tartozó vektort, és kiírja a standard kimenetre.

5.2 API

Az API feladata az azonosítási folyamat vezérlése, továbbá az adatbázisműveletek és a natív alrendszerek közötti kommunikáció vezérlése. Az API az Express nevű keretrendszerre épül. Ez a keretrendszer megvalósít egy ú. n. middleware tervezési mintát, amely a bejövő kéréseket és válaszokat egy csővezetékbe szervezi, ezáltal az útvonalválasztás előtt és után is lehetőség nyílik adott esetben adat transzformációkra, vagy feltételek vizsgálatára. A middleware-ek nagyon hasonlóak a Python nyelvben található dekorátorokhoz.

Én egy middleware-t alkalmazok, amelynek segítségével a válasz objektumot kiegészítem egy *sendJSON* nevű függvénnyel, amely a paraméterként kapott objektumot JSON formátumra alakítja, illetve beállítja - a válasz fejlécében - a tartalom típusát application/json-re.

A végpontok kezelését egy külön komponensben végzem, amelynek egy-egy dedikált funkciója végzi a kérés adatainak feldolgozását, és a válasz tartalmának összeállítását. Az API-ban három végpontot különböztetek meg:

1. /books
2. /books/{könyv-azonosító} (egész szám)
3. /books/recognize

5.2.1 Lekérdezés és beszúrás

Az első végpont feladata bejövő *GET* lekérés esetén, az adatbázisban tárolt könyvek listájának megjelenítése, *POST* lekérés esetén, pedig az adatbázis bővítése, a bejövő paraméterek függvényében. A *POST* lekérésre Multipart tartalmat feltételez, amelynek *cover* paramétere a könyv borítóját tartalmazó kép, a *title*, *keywords, author* és *publisher* paraméter esetében karakterlánc típust feltételez.

Új könyv beszúrásakor a feltöltött könyvborító kép először átmozgatásra kerül az API könyvtárán belüli a public/images könyvtárba, majd összeállításra kerül a könyvhöz tartozó kulcsszavak listája a metaadatok aggregálásával. Amennyiben egy ilyen kulcsszó nem létezik a globálisan alkalmazott hasító táblában, beszúrásra kerül, ellenkező esetben az új könyv azonosítója bekerül ezen kulcsszóhoz tartozó, könyvazonosítókat tartalmazó listába.

Ezután a natív réteg kezelésére szolgáló image-processing nevű wrapperen keresztül történik a képhez tartozó LAB klasztervektor kinyerése, majd ezen vektorok mentése a könyvet reprezentáló rekordban. A mentés során megtörténik a jellemző pontok detektálása és a leíró vektorok számítása is, amelyek az *api/storage* könyvtárban kerülnek tárolásra, XML formátumban.

5.2.2 Keresés

Könyv azonosítás esetében az API elvégzi a könyv klaszterezését a natív réteg megfelelő funkciójának meghívásával. Az így kapott vektor és az adatbázisban tárolt vektorok közötti vektoriális távolság számolásával elvégez egy előválogatást. Ez után az előválogatáskor jónak bizonyuló könyvek azonosítóival meghívja a natív réteg azonosító funkcióját.

Amennyiben az azonosítás sikertelen volt, vagyis a találatként vett minta konfidencia szintje nem éri el a globálisan alkalmazott küszöbszámot, a dokumentumon azonosított szavakkal történik egy további keresés. A felismerendő karaktereket először Stroke Width Transformation [32] alkalmazásával gyűjtöm össze a képről, majd az így keletkező, fehér alapon szürke karaktereket tartalmazó képet továbbítom a karakter detektálást megvalósító funkció felé.

Minden adatbázisban tárolt könyv esetén tárolok egy kulcsszó vektort, amelyet a könyv metaadataiból és a borítón található egyéb szavak felhasználásával készítek el, továbbá tárolok egy hasító táblát minden kulcsszóra, amely tartalmazza azon dokumentumok azonosítóit, amelyek tartalmazzák az adott kulcsszót.

Adott könyv esetében az azonosított szavakat ezen hasítótábla tartalmának összehasonlításával végzem, a hibás detektálást Levenshtein távolsággal koorigálom, majd minden könyvre számolok egy összegzett jóságot az adott kulcsszavak inverz

dokumentum gyakoriságának és terminus frekvenciájának alkalmazásával. Egy kiegészítésként alkalmazok egy “stopszó listát”, hogy adott esetben egy kulcsszó csupán egyszer kerülhessen be egy adott könyv jóságágát reprezentáló változóba.

5.3 Adatbázis

Az adatbázisban az adatok JSON formátumban kerülnek tárolásra, futás időben a memóriában, egyébként a fájlrendszeren. Az adatbázis motorját is JavaScript nyelven írtam, mivel a JSON objektumok kezelése így tűnt a legkézenfekvőbbnek. Az adatbázis gyakorlatilag egy gyenge sémát alkalmaz, egyes könyvek rekodjánál a kulcsszavakat képző tömb dinamikusan bővíthető, erős relációkat nem alkalmazok. Mivel egy könyv feltöltésénél annak adatai két lépésben kerülnek összeállításra (metaadatok és klasztervektor), ezért is ideális ez a megközelítés. Az adatbázisban egyrészt tárolok könyv objektumokat, illetve kulcsszavakhoz tartozó kulcs-érték párokat, ebből kifolyólag a motor 5 funkcióval rendelkezik:

- beszúrás
- kulcs-beszúrás
- frissítés
- következő-azonosító
- commit
- rollback

Commit esetén történik meg az aktuálisan memóriában tárolt állapot merevlemezre mentése, a következő-azonosító funkció pedig, a könyvek esetében alkalmazott szekvenia aktuális értékét adja vissza.

5.4 iOS Kliens

A kliens implementációjában alapvetően - a menüpontoknak megfelelően - két csoportot különböztettem meg. Az alkalmazás implementációjában az első belépési pont egy TabBarController, amely további két NavigationControllert tartalmaz. A két NavigationController egy közös, DetailViewControllerben találkozik, ez felelős az egyes könyvek metadatainak és borítójának megjelenítésére.

Az adatkapcsolat, illetve a perzisztens tárolóhoz tartozó CoreData keretrendszer elfedésére két singleton osztályt valósítottam meg, ezek az APIAgent és StorageManager. A fotók készítése egy külső YCameraViewController nevű nyilt forráskódú komponens módosításával készült, megjelenítése modálisan történik. Az eredményként előálló UIImage objektum egy delegált metóduson keresztül kerül vissza az azt megjelenítő SearchViewController osztályba, amely ez után elküldi ezt az API-nak és elindítja a felismerés folyamatát.

A fejlesztés megkönnyítéséhez három egyéb könyvtárat alkalmaztam, az egyik az AFNetworking. Ez a könyvtár az iOS SDK alap HTTP kommunikációra szolgáló osztályaira ráülve valósít meg egy sokkal fejlesztőbarátabb réteget. A könyvtár tartalmaz egy AFJSONRequestSerializer és AFJSONResponseSerializer nevű funkciót, amely nagyban megkönnyíti az API felől visszaérkező adatok kezelését, elvégez egy automatikus átalakítást NSDictionary típusú objektumra.

A második ilyen könyvtár a MagicalRecords, amely a CoreData alap funkcionalitására ül rá, és ugyancsak a fejlesztői munka megkönnyítésére szolgál. Használatával nagymértékben csökkenthető az egyes entitásokat reprezentáló NSManagedObject példányok létrehozása majd mentése a perzisztens tárolóra. A StorageManager osztály ezt alkalmazza a CoreData stack beállításához, illetve az alkalmazás terminálásakor, továbbá a háttérbe helyezésekor szükséges adat perzisztens tárolóba történő mentésére.

Attól függően, hogy melyik ágról kerülünk a DetailViewControllerre, megjelenik egy új gomb, amely felkínálja a felhasználónak, hogy elmentse a felismert könyvet a kedvencek közé. Ehhez a Book osztály példányaként létrehozott entitást egy ú.n. gyermek kontextusra helyezem, amely mintegy ideiglenes tároló fogja tartalmazni a könyvet. Ha a felhasználó hozzá kívánja adni az így létrehozott objektumot a kedvenceket tartalmazó listához, akkor kerül átírásra a szülő kontextusként beállított, fő szálon futó kontextusra.

A kedvencek nézetnél ezen kívül lehetőség van új könyvvel bővíteni az adatbázist. Ehhez egy AddNewBookViewController nevű osztályt valósítottam meg, amelynek megjelenítése modálisan történik, az űrlap mezőiben megadott adatok, egy delegált metóduson keresztül kerülnek vissza azt azt megjelenítő MasterViewController-re. Ebben a nézetben megvalósítottam a UIImagePickerControllerDelegate-et, ennek segítségével tudja a felhasználó betallózni az adott könyvhöz tartozó borítóképet. Hozzáadás után a MasterViewController végzi el a könyv feltöltését az APIAgent-en keresztül küldött lekéréssel.

Az alkalmazás képernyőit és használatát a melléklet 9.5-ös pontja részletezi.

6. TESZTELÉS

A teszteket egy Ubuntu 12.04-es verziójú operációs rendszerrel rendelkező virtuális szerveren (linux kernel verzió: 3.2.24) futtattam. A tesztelés három lépésben történt. Az első lépésben egy 10 képet tartalmazó, jól szegmentálható, saját magam által készített mintán teszteltem a klaszterezés eredményeit. A második lépésben ugyanezen a mintán végeztem el SWT transzformációt, optikai karakterfelismerést, majd a kinyert kulcsszavak felhasználásával végeztem el keresést. A harmadik lépésben egy 100 darab könyvborítót [44], illetve ezekhez tartozó 100 darab iPhone 3GS kamerával készített fotót tartalmazó adatbázist használtam a képi leírók robusztusságának és az osztályozás jóságának tesztelésére. A korai prototípus teszteléseinek néhány eredménye melléklet 2. pontjában látható.

Az értékeléshez IR [45] mértékeket (pontosság/fedés) számoltam, a mérést minden könyvborító mintára elvégeztem.

Az eredményeket mérésenként egy 2x2-es táblázatban foglalom össze, az első sor tartalmazza a mérés során eredménynek elfogadott találatokat, a második sor, pedig az elvetett találatokat, az alábbi formában:

	Igaz	Hamis
Találat	Helyes találatok száma (TP)	Helytelen találatok száma (FP)
Nem találat	Hiányzó helyes találatok (FN)	Hiányzó negatív találatok (TN)

Pontosságot a következő képlet adja:

$$P = \frac{TP}{TP + FN}$$

ahol TP a helyes és elfogadott, FN a helyes, de nem elfogadott találatok. A fedés pedig:

$$R = \frac{TP}{TP + FP}$$

ahol FP a hamis és találatnak elfogadott eredmény. A két görbe általában egymással szemben mozog.

6.1 Eredmények

6.1.1 Klaszterezés LAB színtérben

A mérés során két féle távolság mértéket különböztettem meg:

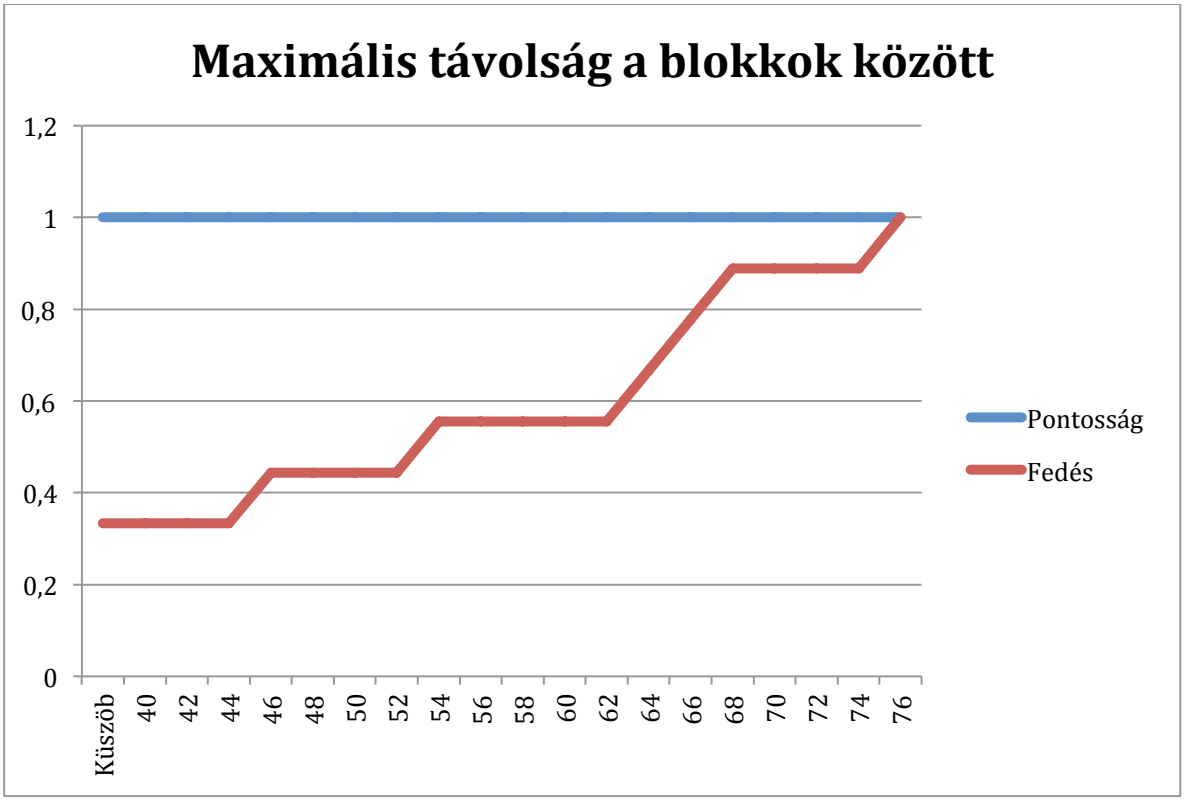
$$D_{Csebisev}(b,s) = \max\{ f(b_i, s_i) : i \in \{1,2,3,4,5,6\} \}$$

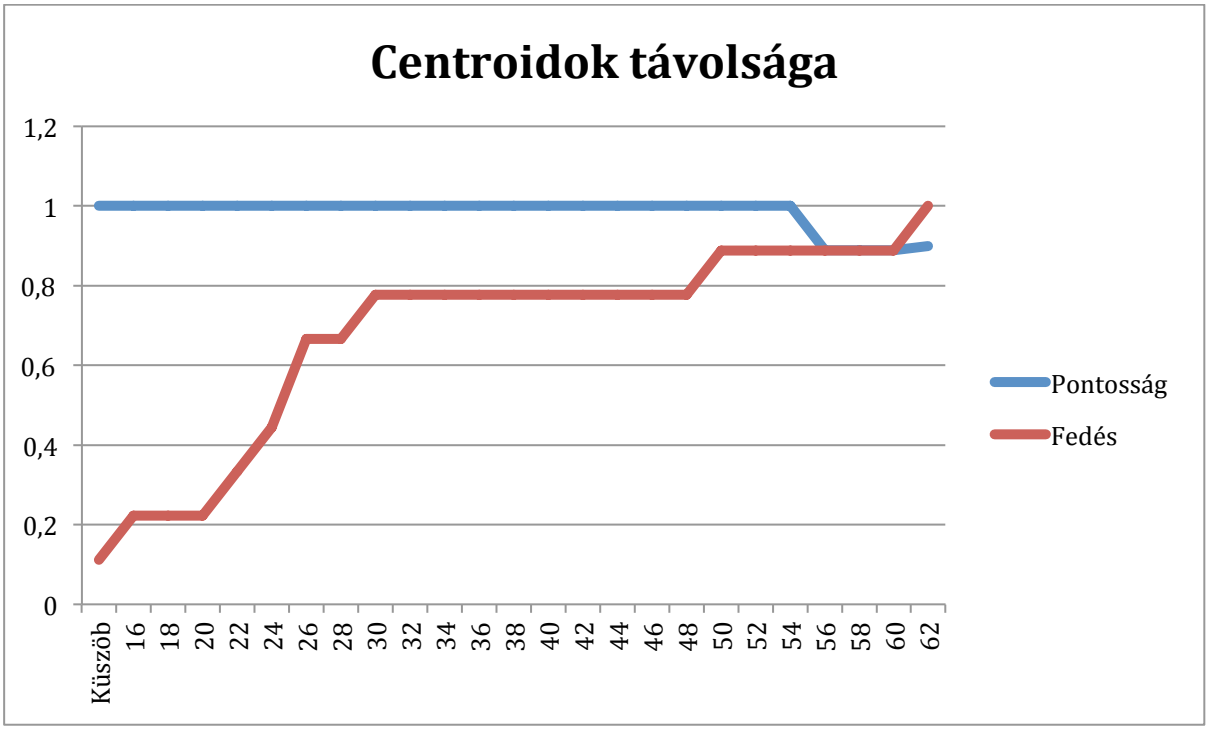
ahol b_i a kinyert klasztervektor i-edik blokkjának értéke, s_i a tárolt klasztervektor i-edik blokkjának értéke, f pedig ezen pontok vektoriális távolsága.

A második távolság mérték a kinyert és tárolt blokkok értékeiből számolt centroidok távolsága:

$$D_{centroid}(b,s) = f\left(\sum_{i=1}^{6}\frac{b_i}{6}, \sum_{i=1}^{6}\frac{s_i}{6}\right)$$

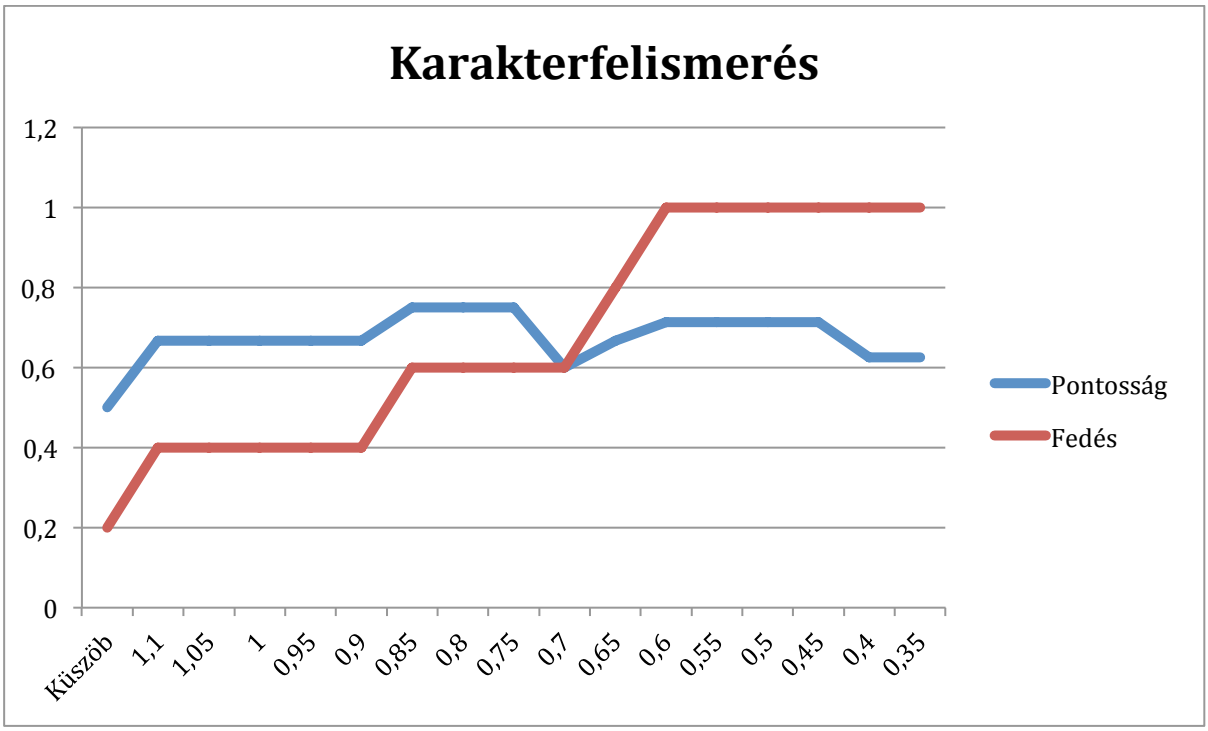
Az eredmények a következőek:





6.1.2 Optikai karakterfelismerés

Az optikai karakterfelismeréssel történő keresés esetén minden dokumentum esetében meghatározok egy pontszámot, amely a találatként felvehető kulcsszavak idf és tf álatal súlyozott értékeinek összege ad meg.



6.1.3 Képi jellemzők vizsgálata

A képi jellemzők vizsgálatához használt adatbázis montázsát a melléklet 21. ábrája szemlélteti. A rendszerben a találatok kiértékelésére egy arányszámot alkalmazok, amely a detektált könyvborító leíró vektorainak és az éppen vizsgált borító leíró vektorainak párba állításkor előállt, találatokat tartalmazó vektorok száma, illetve az ezekből jó találatként elfogadható leíró vektorok számának aránya határoz meg. Az elfogadhatóság vizsgálatához egy további távolság küszöböt alkalmazok, amelyhez egy vizsgált leíró vektor két legközelebbi leíró vektorának távolsága által meghatározott arányát hasonlítom, ennek értéke globálisan 0,7.

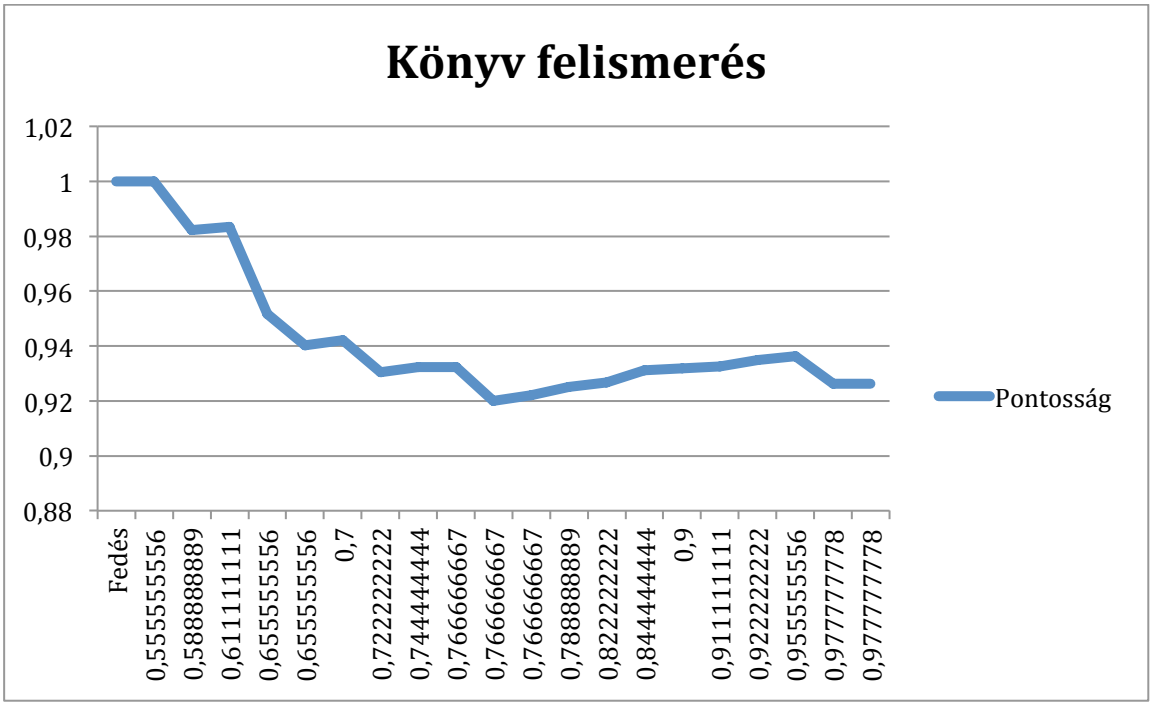
Az eredmények vizsgálatakor célom volt annak az optimális arányszámnak a megállapítása, ahol a fedés és a pontosság a lehető legnagyobb. Ehhez az arányszámhoz viszonyított két érték által alkotott görbe metszéspontját kerestem.

Részeredmények 0,22-os arányszámnál:

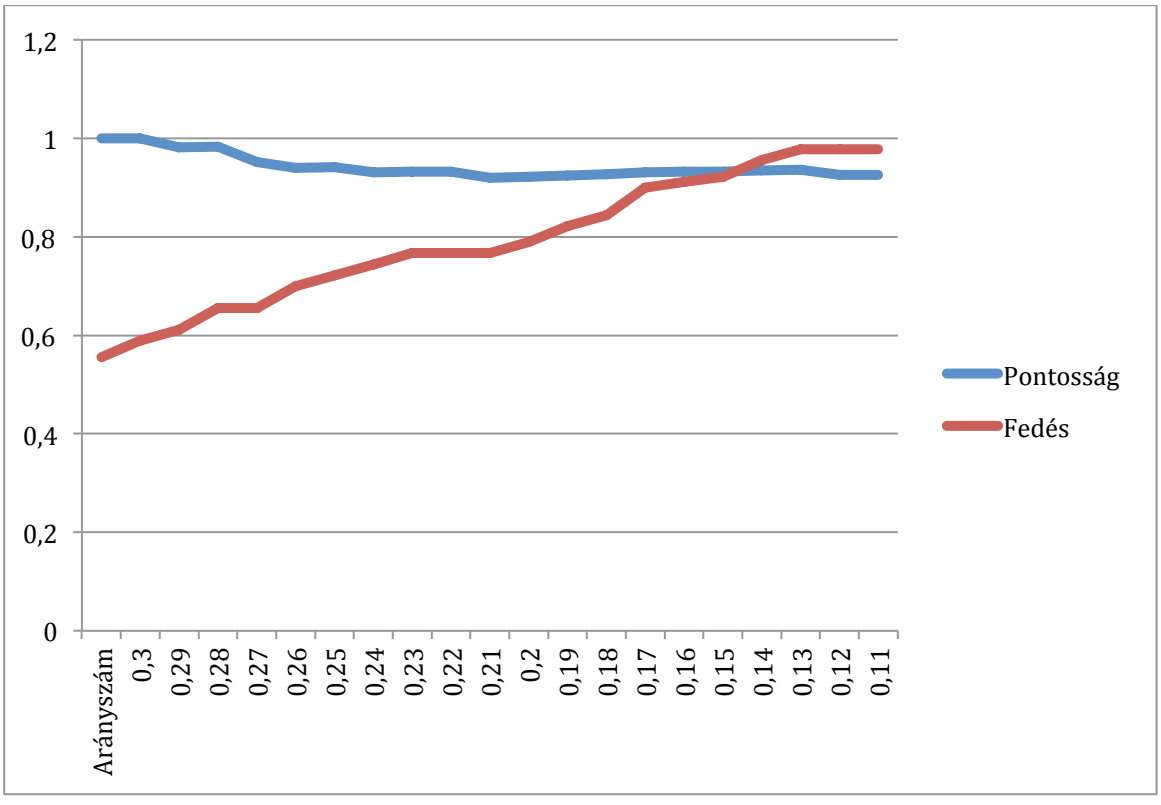
	Igaz	Hamis
Találat	69	5
Nem találat	21	5

Ezen értékek csökkentett, 0,16-os arányszámnál:

	Igaz	Hamis
Találat	81	6
Nem találat	9	4



Az arányszám 0,3 és 0,1 közötti változtatásával kapott precízió és fedés arányát a következő diagram szemlélteti:



6.2 Értékelés

Az első két mérést egy kisméretű adathalmazon végeztem. Az előválogatás során elvégzett klaszterezésnél két féle távolság mértéket alkalmaztam, az egyik az egyes blokkok között fellelhető maximális távolság, a másik pedig a blokkok átlagintenzitásai által képzett pontok centroidjainak távolsága. Megfigyelhető, hogy az első esetben a rendszer fedése közel lineárisan növekszik, a két mérték metszéspontjában található harmonikus közép valahol 100-as érték környékén található. Könnyen belátható, hogy ez a megközelítés, alacsonyabb számítási igénnyel rendelkezik, hiszen adott esetben egy blokk vizsgálat során teljesülhet a terminálási feltétel. Az is belátható hogy ez a megközelítés nem vezet optimumra, mivel a pontok környezetében gyakorlatilag a vektortér közel negyedét kellene venni a helyes eredményhez. A második esetben viszont már jól megfigyelhető egy meredek emelkedés 16 és 40 rádiuszú körben. Ez azt mutatja, hogy még az egyes blokkokban kiúgró zajként jelentkező szín torzulások sem módosítják oly mértékben a pontokból képzet klaszter középpontját, hogy az ne a tárolt minta centroidjának egy kissebb környezetében képződjön le.

Az Optikai Karakterfelismerés esetén úgy gondolom, hogy még nem értem el a kívánt eredményt. Ugyan a rendszer az esetek közel felében képes a helytelenül detektált szavak által okozott hibás találatok kezelésére, illetve a idf és tf alapú súlyozás is a várakozásoknak megfelelően működik, szükséges, hogy további munkálatokat végezzek el ezzel kapcsolatban. A grafikonon jól látható, habár a fedés monoton emelkezést mutat, a precízió végig az 50-80%-os sávban ingadozik, amely mutatja a bizonytalan működést.

A képi jellemzők jóságának mérése során változtatott arányszámról fontos megjegyezni, hogy ennek értékét a találatok jóságának megállapításkor alkalmazott távolság küszöb nagyban befolyásolja. Általában a találatként elfogadott könyv egyezésnél ez az arányszám szignifikánsan kiugró értékkel rendelkezik, szemben a többi könyv esetében tapasztalt 1-3% közötti intervallumba eső jónak vélt vektor-pár aránynál.

Az első diagramról leolvasható, hogy a pontosság fedés függvényében alkotott görbéjének 0,76 körüli értékében lokális minimuma van, a pontosság e fölött monoton növekedést mutat egészen 0,93-ig, ahol úja csökkenni kezd. A második diagram esetében a fedés és a pontosság által alkotott görbe 0,15-ös aránynál metszik egymást, a fedés 0,13-ös arányszám után már nem mutat növekedést, illetve a pontosság is csökkenni kezd.

Ebből kifolyólag az új arányszám legfeljebb a fedés 0,76-os értékéhez tartozó 0,22, legalább 0,13 –ös értékkel kell rendelkeznie, a minta globális optimuma 0,15.

A globális optimum alkalmazásával megállapítható, hogy a rendszer csupán képi jellemzők vizsgálatával 93%-os pontossággal tudja azonosítani az egyes könyveket

borítójukról készített fotója alapján, az optikai karakter felismerés továbbfejlesztésével ez az eredmény tovább növelhető.

6.3 Továbbfejlesztési lehetőségek

A klaszterezés lépésében érdekes lehet, ha az eredetileg alkalmazott 2*3 blokkos felosztás helyett ennél több blokkos felosztást alkalmaznék. A mérésből kiderült, hogy ez a megközelítés elég robusztus lehet, esetlegesen a szín pontok környezetének vizsgálata során alkalmazott kör elipszoid formára módosításával további javulás lehetne elérhető. Ez a módosítás csupán az a és b csatorna különböző mértékű súlyozásával történhetne, az LAB színtér alakjának figyelembe vételével.

Másik továbbfejlesztési lehetőség lehet a rendszerben alkalmazott Stroke Width transzformáció valamilyen egyéb megoldással helyettesítése vagy ötvözése. Tapasztalataim szerint ez a megközelítés nem mindig ad helyes eredményt, adott esetben valamilyen küszöböléses megoldás nagyobb pontosságra vezethet. A továbbfejlesztés szepontjából fontos lehet valamilyen egyéb utófeldolgozási lépés bevezetése a detektálás során szétszakadt kulcsszavak összevonásának kezelésére.

Egy lehetőség lehetne a rendszer Amazon web-áruházával történő összekötése, ezáltal a felhasználói interakciók kibővítése vásárlással. Beépítésre kerülhetne az alkalmazásban különböző szociális interakcióra szolgáló funkció is, így mint adott könyv megosztása különböző csatornákon, hozzászólás, értékelések írása.

A rendszer webre történő kiterjesztése is célszerű lenne, egy web-alkalmazás lefejlesztésével. Ez az alkalmazás használhatná a már meglévő API alrendszert, nem lenne szükséges egy új implementálására. A dokumentumok menedzselésére is történhetne egy webes felületen, az adatok felvitelének egyszerűsítése szempontjából.

Egy jóval összetettebb továbbfejlesztés lehet egy ajánlórendszer implementálása és beépítése, amely képes lenne a dokumentumok között fennálló rejtett asszociációk feltárására, és ezáltal a felhasználói élmény javítására. Ezen dokumentumok megjelenítése történhetne az alkalmazás dokumentum-nézet képernyőjének alján elhelyezett kapcsolódó művek szekcióban.

7 ÖSSZEGZÉS

A dolgozatban részletesen bemutattam a könyvborító alapú keresés problematikáját digitális könyvtári katalógusokban. Ismertettem a CBIR rendszerek általános megközelítéseit, illetve az ott alkalmazott módszereket. Kifejtettem a könyvborítók kontextusból való kiemeléséhez szükséges lépéseket és algoritmusokat. Szemléltettem a könyvborító felismerés több megközelítését, illetve keresés gyorsítására szolgáló több módszert. Részletesen ismertettem három hasonló rendszert, az ott alkalmazott algoritmusok nagy részét.

Vázoltam az optikai karakterfelismerés könyvborítók vizsgálatánál előálló problematikáját, a szótárépítés, a nyelvfelismerés és a szótövezés segítségével felépíthető megközelítését a döntési tér szűkítésére. Kifejtettem az inverz dokumentum frekvencia és terminus frekvencia alapú súlyozási módszert, továbbá a Levenshtein távolság alkalmazásával elérhető javítási lehetőségeket.

Az irodalom kutatás eredményeit felhasználva, saját ötleteimmel kiegészítve megterveztem és implementáltam egy Three-trier megközelítésű architektúrán alapuló rendszert a feladat megoldására. A megvalósítás során számos saját megoldást implementáltam, igyekeztem ezek működését részletesen bemutatni. A döntési tér sűkítését LAB színtérben végzet klaszterezéssel végeztem.

A megvalósított rendszer architekturális megközelítés szempontjából könnyen skálázhatónak mondható. A rendszer implementációja elkészült, tesztelés és mérés megtörtént. A képi jellemzők felhasználásával készült mérésem eredménye 93%-os pontosságot mutat, amely bizonyítja a SURF leírók robusztusságát és az osztályozási megközelítésem jóságát. Az Optikai Karakterfelismerés továbbfejlesztésével a rendszer pontossága tovább növelhető lehet.

8 REFERENCIÁK

[1] Könyvtári katalógusok
(http://www.bdf.hu/konyvtar/informaciokereses/katalogus.htm),
utoljára megtekintve: 2014. árpilis 13.

[2] 3D Projection (http://en.wikipedia.org/wiki/3D_projection),
utoljára megtekintve: 2014. árpilis 13.

[3] M. Brown, R. Szeliski, and S. Winder, "Multi-Image Matching using Multi-scale Oriented Patches," *Technical Report, Microsoft Research*, 2004.

[4] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom et al. "Query by Image and Video Content: The QBIC System," *IEEE Computer*, vol. 28, no. 9, 1995.

[5] A. Pentland, R.W. Picard, and S. Sclaroff, "Photobook: Tools for Content-Based Manipulation of Image Databases," *Proc. SPIE*, vol. 2185, pp. 34-47, Feb. 1994.

[6] J.R. Smith and S.-F. Chang, "VisualSEEk: A Fully Automated Content-Based Image
Query System," *Proc. ACM Multimedia*, pp. 87-98, Nov. 1996.

[7] S. Stevens, M. Christel, and H. Wactlar, "Informedia: Improving Access to Digital Video," *Interactions*, vol. 1, no. 4, pp. 67-71, 1994.

[8] J.Z. Wang, G. Wiederhold, O. Firschein, and X.W. Sha, "Content-Based Image Indexing and Searching Using Daubechies' Wave-lets," *Int'l J. Digital Libraries*, vol. 1, no. 4, pp. 311-328, 1998.

[9] W.Y. Ma and B. Manjunath, "NaTra: A Toolbox for Navigating Large Image Databases," *Proc. IEEE Int'l Conf. Image Processing*, pp. 568-571, 1997.

[10] S. Jeong, "Histogram-Based Color Image Retrieval", *Psych221/EE362 Project Report*, 2001

[11] A. Natsev, R. Rastogi, and K. Shim, "WALRUS: A Similarity Retrieval Algorithm for Image Databases," *SIGMOD Record*, vol. 28, no. 2, pp. 395-406, 1999

[12] C. Carson, M. Thomas, S. Belongie, J.M. Hellerstein, and J. Malik, "Blobworld: A System for Region-Based Image Indexing and Retrieval," *Proc. Visual Information Systems*, pp. 509-516, June 1999.

[13] J.R. Smith and C.S. Li, "Image Classification and Querying Using Composite Region Templates," *Int'l J. Computer Vision and Image Understanding*, vol. 75, nos. 1-2, pp. 165-174, 1999

[14] J. Z. Wang, J. Li and G. Wiederhold, “SIMPLIcity: Semantics-Sensitive Integrated Matching for Picture Libraries,” *IEEE Transactions on Pattern Analytis and Machine Intelligence*, vol 23, no 9, September 2001

[15] K.Velmurugan, Lt.Dr.S. Santhosh Baboo, “Content-Based Image Retrieval using SURF and Colour Moments” *Global Journal of Computer Science and Technology*, vol 11, May 2011

[16] P. Indyk and R. Motwani, “Approximate Nearest Neighbours: Towards Removing the Curse of Dimensionality” *Proceedings of the 30th Symposium on Theory of Computing*, pp. 604-613, 1998

[17] Levenshtein, Vladimir I., “Binary codes capable of correcting deletions, insertions, and reversals,” *Soviet Physics Doklady*, vol 10 (8), pp. 707–710, February 1966

[18] R. Smith, D. Antonova and Dar-Shyang Lee, “Adapting the Tesseract Open Source OCR Engine for Multilingual OCR,” *Proceedings of the International Workshop on Multilingual OCR*, 2009

[19] G. Nagy, “Chinese character recognition: a twenty-five-year perspective,” *9th Int. Conf. on Pattern Recognition*, pp. 163-167, November 1988

[20] Porter, Martin F., “An Algorithm for Suffix Stripping,” *Program*, 14(3): 130–137, 1980

[21] Snowball (http://snowball.tartarus.org/index.php),
utoljára megtekintve: 2014. árpilis 13.

[22] W. B. Cavnar , J. M. Trenkle, “N-grambased text categorization,” *Proc. of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, 1994

[23] T. Gottron and N. Lipka, “A Comparison of Language Identification Approaches on Short, Query-Style Texts”,

[24] G Windisch, L. Csink, “Language Identification Using Global Statistics of Natural Languages”, *Proceedings of the 2nd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence (SACI)*, pp. 243-255, 2005

[25] C. Galleguillos, “Book Cover Recognition Project,” *University of California San Diego* (http://cseweb.ucsd.edu/classes/wi06/cse190a/proposals/cgallegu.pdf), utolsó megtekintés: 2014 április 14.

[26] Lowe, David G., “Object recognition from local scale-invariant features,” *Proceedings of the International Conference on Computer Vision 2*., pp. 1150–1157, 1999

[27] MacQueen, J. B., “Some Methods for classification and Analysis of Multivariate Observations,” *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1. University of California Press*. pp. 281–297, 1967

[28] Shao-Chuan Wang, “Book Cover Recognition”, Columbia University (http://www.slideshare.net/shaochuan/book-cover-recognition), 2010

[29] C. Cortes, V. Vapnik, “Support-vector networks,” Machine Learning 20 (3): 273., 1995

[30] S. S. Tsai, D. Chen , H. Chen , Cheng-Hsin Hsu , Kyu-Han Kim , J.P. Singh , B. Girod, “Combining image and text features: a hybrid approach to mobile book spine recognition,” *Proceedings of the 19th ACM international conference on Multimedia*, 2011

[31] J. Matas, O. Chum, M. Urban, and T. Pajdla., “Robust wide baseline stereo from maximally stable extremal regions,” *Proc. of British Machine Vision Conference*, pp 384-396, 2002

[32] B. Epshtein, Y. Wexler, and E. Ofek, “Stroke Width Transform,” *IEEE International Conference on Computer Vision and Pattern Recognition,* 2010

[33] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, “SURF: Speeded Up Robust Features,” *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346-359, 2008

[34] D. Nister and H. Stewenius, “Scalable recognition with a vocabulary tree,” *CVPR*, 2006

[35] Altman, N. S., “An introduction to kernel and nearest-neighbor nonparametric regression,” *The American Statistician 46*, pp 175–185, 1992

[36] Palágyi K., “Képfeldolgozás haladóknak,” *Szegedi Tudományegyetem*, 2011

[37] V. Hong, H. Palus, D. Paulus, “Edge Preserving Filters on Color Images,” *Universität Koblenz-Landau*, 2004

[38] Jan Eric Kyprianidis, “Image and video abstraction by multi-scale anisotropic Kuwahara filtering,” *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, August 2011

[39] R. C. Gonzales, R. E. Woods, “Digital Image Processing. 2nd,” *Prentice Hall*, 2001

[40] P. V. C. Hough, “A Method and Means for Recognizing Complex Patterns,” *US Patent 3,069,654*, 1962.

[41] N. Otsu, “A threshold selection method from grey-level histograms,” *IEEE Transactions on Systems, Man, and Cybernetics*, pp. 62-66, 1979

[42] Pearson, K., “On Lines and Planes of Closest Fit to Systems of Points in Space,” *Philosophical Magazine 2*, pp. 559–572, 1901

[43] XYZ to LAB (http://www.brucelindbloom.com/index.html?Eqn_XYZ_to_Lab.html), utolsó megtekintés: 2014 április 14.

[44] D. Chen, S. Tsai, C.-H. Hsu, K.-Y. Kim, J. P. Singh, and B. Girod, “Building book inventories using smartphones,” *ACM Multimedia (MM)*, October 2010

[45] D. Tikk, “Szövegbányászat,” *Typotex*, 2007

[46] Chapter 11 Non-Parametric Techniques (https://www.byclb.com/TR/Tutorials/neural_networks/ch11_1.htm), utolsó megtekintés: 2014 április 14.

[47] CHAPTER 5 Learning Music Signals (http://web.media.mit.edu/~tristan/phd/dissertation/chapter5.html), utolsó megtekintés: 2014 április 15.

[48] Z. Vámossy, “Vonalak, görbék, sarokpontok, ”, Budapest Tech, 2004

9 MELLÉKLETEK

9.1 Az ETO főbb osztályai

Általános művek

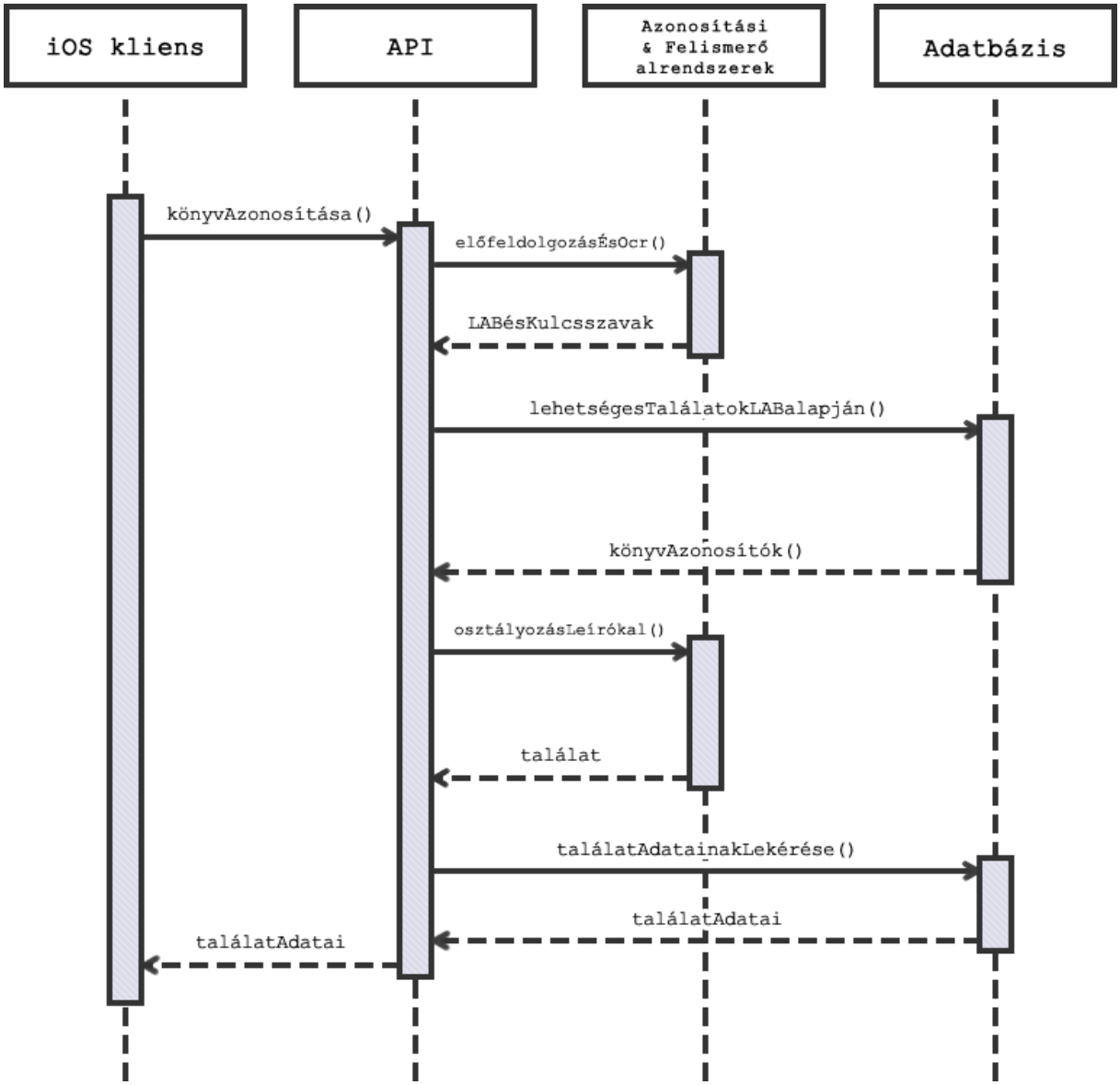
1. Filozófia
2. Vallás
3. Társadalomtudományok
4. (jelenleg betöltetlen)
5. Természettudományok

    51. Matematika
        - 511. Számelmélet
        - 512. Algebra
        - 513. Geometria
        - 514. Analízis, függvénytan
        - 515. Kombinatorika
    52. Csillagászat, asztrofizika
    53. Fizika
    54. Kémia, ásványtan
    55. Földtudományok, geológia
    56. Őslénytan
    57. Biológia
    58. Botanika, növénytan
    59. Zoológia Állattan

6. Alkalmazott tudományok
7. Művészetek, sport
8. Nyelv és irodalom
9. Történelem, földrajz

9.2 Azonosítási folyamat

Az azonosítási folyamatot az alábbi flow-diagram szemlélteti:

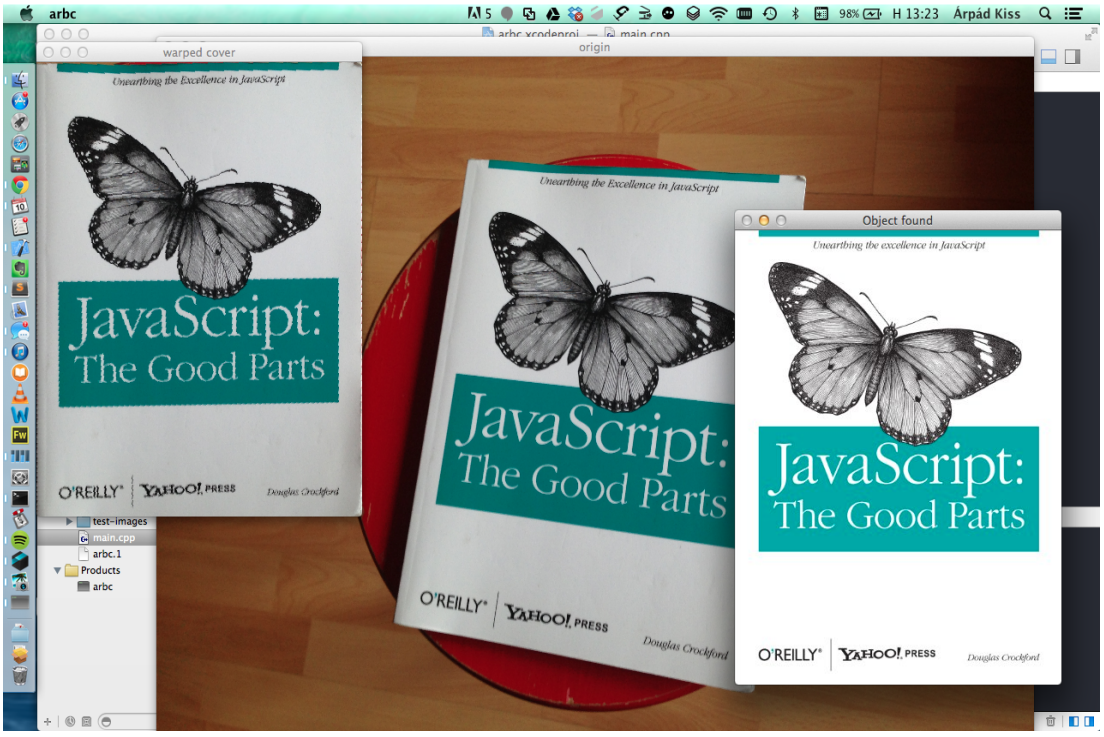


14. ábra, az azonosítási folyamat flow diagramja

9.3 Korábbi teszteredmények a prototípusból

Az háttérben a felismeréshez használt fotó, bal oldalt a vágás és térbe transzformálás eredménye, míg jobb oldalt a találat látható.

15. ábra, korai teszteredmény

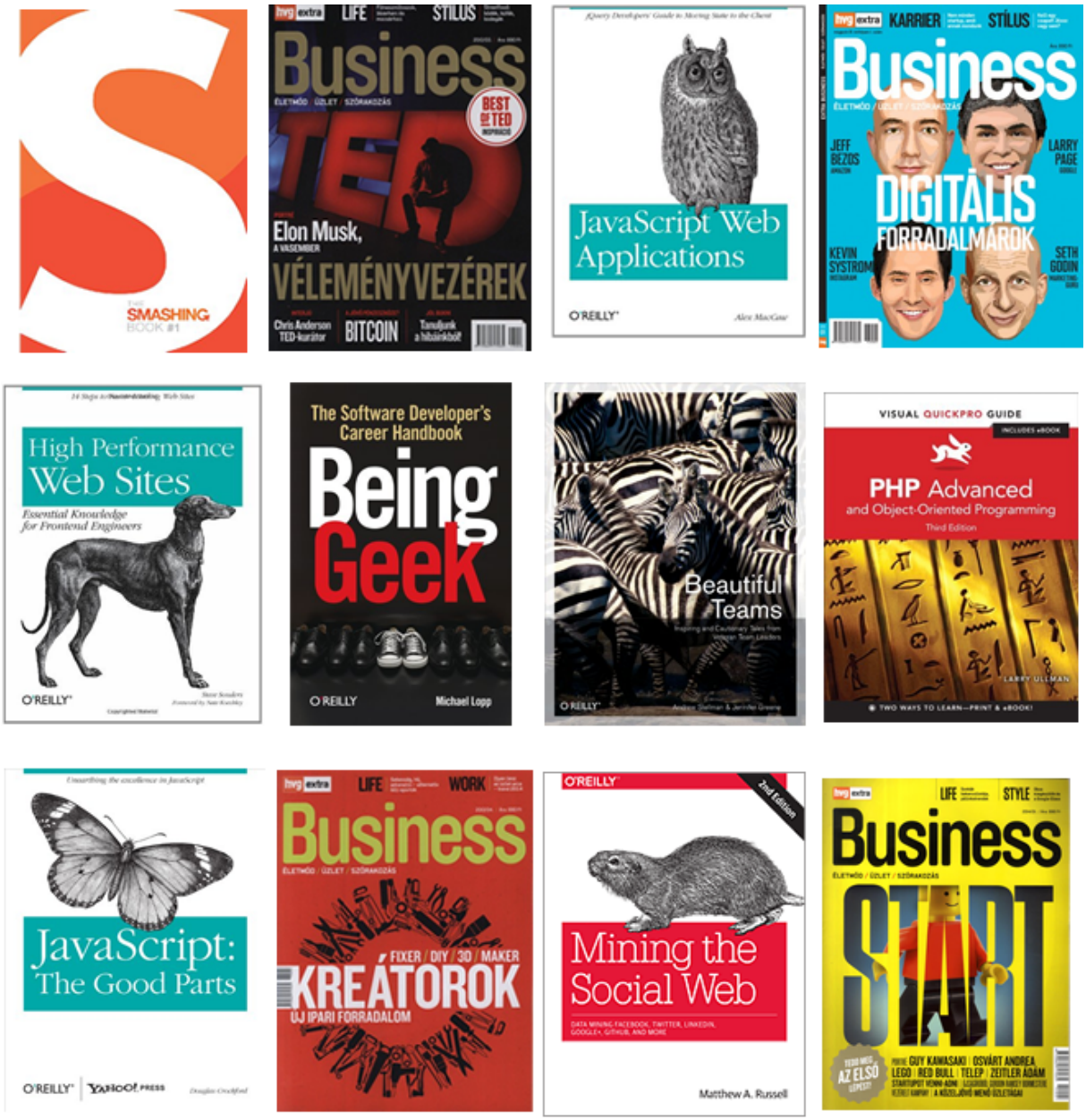


16. ábra, korai teszteredmény

17. ábra, korai teszteredmény

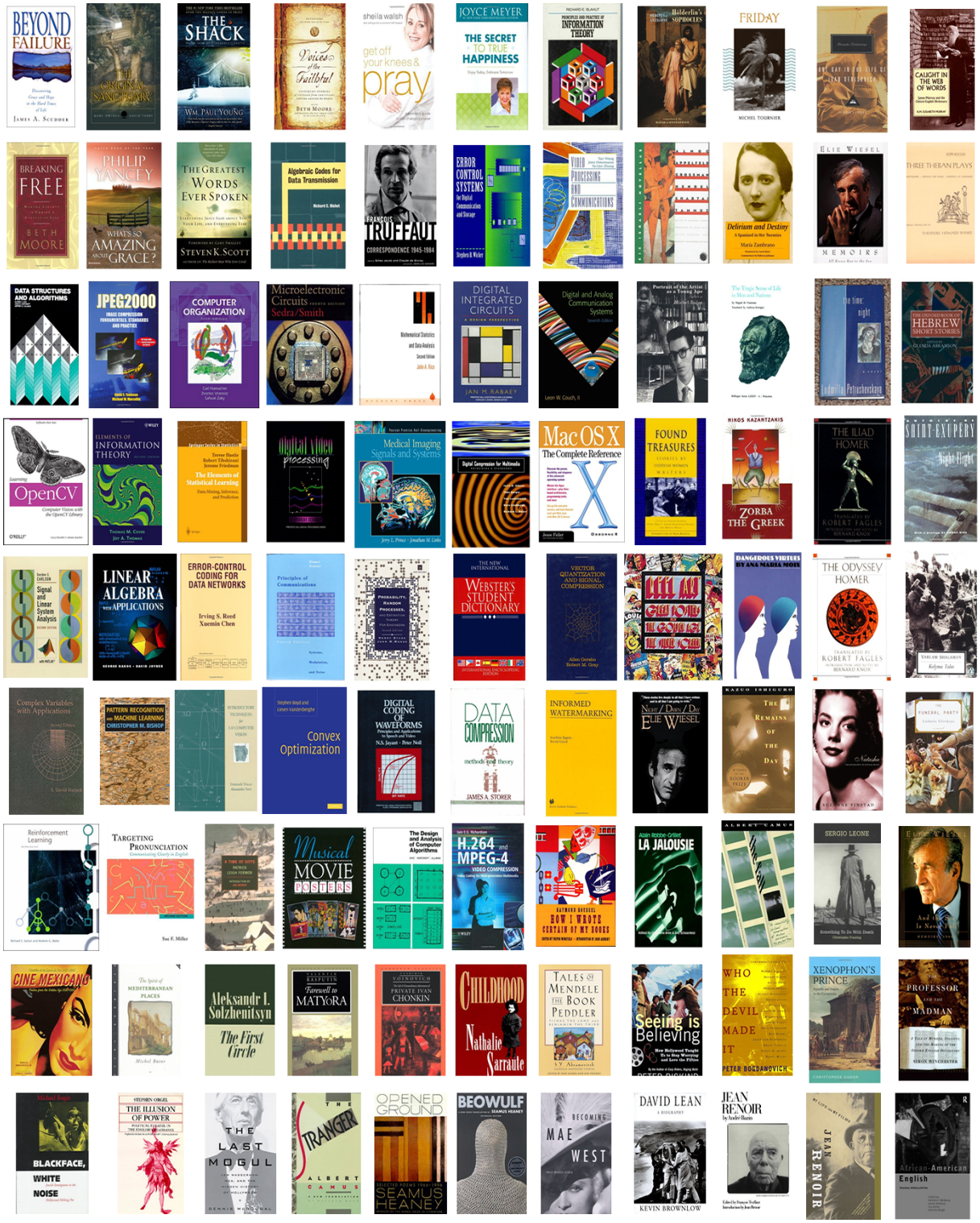
18. ábra, korai teszteredmény

A 19.-es ábránál jól megfigyelhető az objektum főtengely meghatározásának hiányából adódó torzulás a modellbe való leképzés során. A rendszer ettől függetlenül helyes eredményt ad.



19. ábra, a korai prototípus tanító adatai

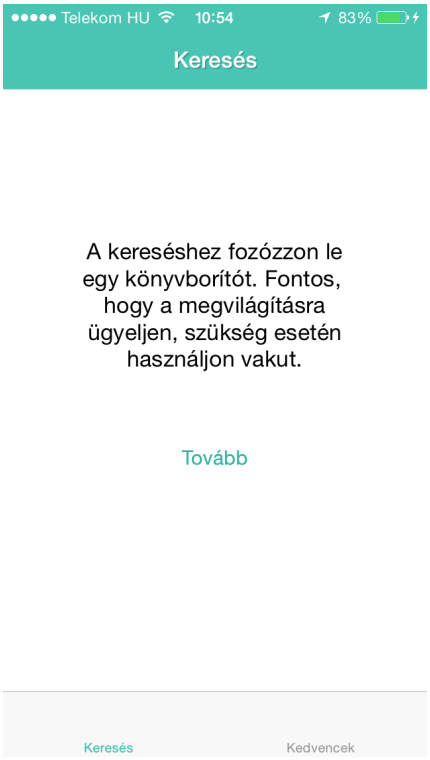
9.4 A teszteléshez használt adatbázis



20. ábra, a teszteléskor felhasznált adatbázis montázsa

9.5 Felhasználói kézikönyv

9.5.1 Kezdőképernyő



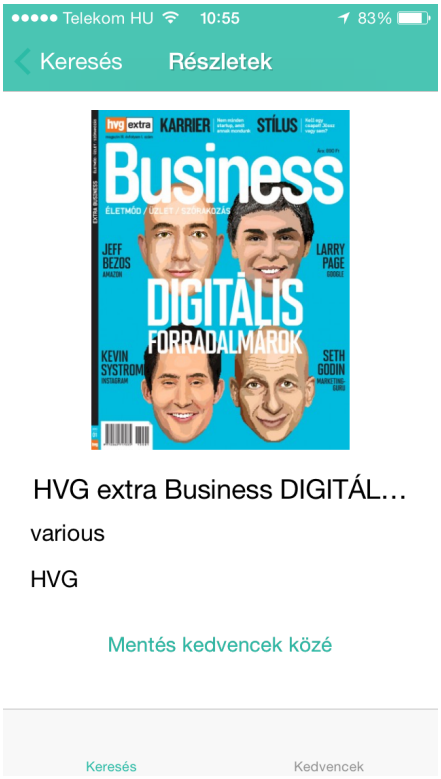
A felhasználó az alkalmazás indításakor erre a képernyőre jut. Az alsó sávban található két menüpont segítségével juthat el az alkalmazás két szekciójára.

A képernyő közepén található *Tovább* gomb segítségével juthat el a felhasználó a keresés felületére.

9.5.2 Fotó készítés

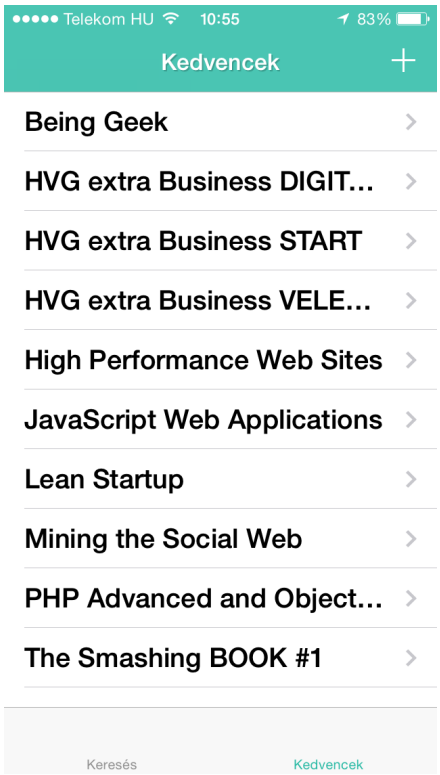
Erre a képernyőre jutva a felhasználó elkészítheti a könyvborítót tartalmazó képet. A felső sávban található akció gombok segítségével képes a vaku ki- és bekapcsolására, a segédrács ki- és bekapcsolására, továbbá az első és hátsó kamera közötti váltásra. A képernyő alján található két négyzetet tartalmazó ikon segítségével, a felhasználó képes a korábban készített fotói között válogatni, majd egyet kiválaszva, azzal indítani keresést. A nagy cián kör megnyomásával történik meg a fotó készítés.

9.5.3 Eredmény



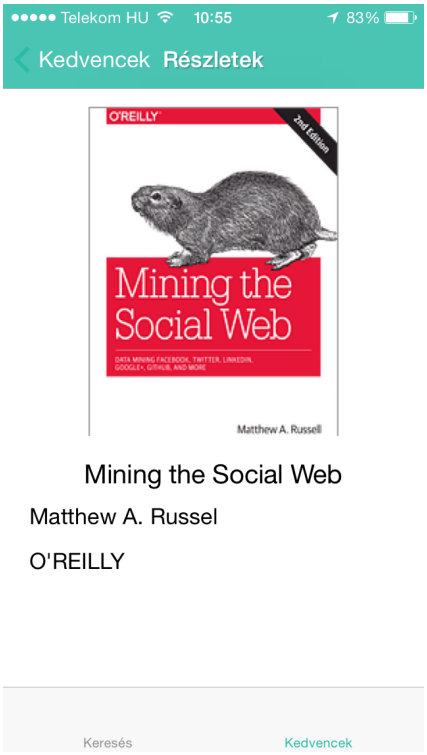
Amennyiben a felhasználó által elküldött fotón lévő könyvborító azonosítása sikeres volt, a felhasználó erre a képernyőre jut. Lehetősége van az alul található *Mentés kedvencek közé* gomb segítségével elmenteni a találatként megjelenített könyvet a kedvencei közé.

9.5.4 Kedvencek



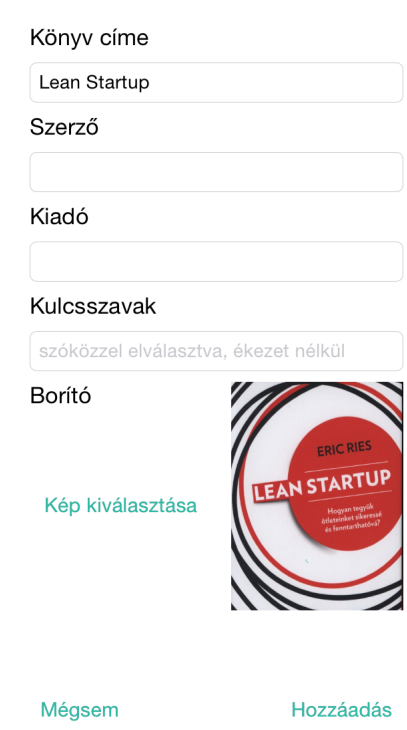
Ez a képernyő tartalmazza a felhasználó által elmentett és hozzáadott könyvek listáját. Egy cellasor balra húzásával a felhasználónak lehetősége van az adott művet törölni a kedvencei listájából. A jobb felső sarokban található + gomb segítségével a felhasználó az Új könyv felvitele képernyőre juthat, egy cella kijelölésével pedig az adott mű adatlapjára.

9.5.4 Kedvenc megtekintése



Ez a képernyő lényegében megegyezik a keresés után megjelenített találati képernyővel, a különbség csupán a *Mentés* gomb hiánya.

9.5.5 Új könyv felvitele



Ez az új könyvek felvitelére szolgáló képernyő. A felhasználó itt adhatja meg a feltöltendő műhöz tartozó metaadatokat, illetve a borítón szereplő egyéb kulcsszavakat, továbbá lehetősége van a borítókép betallózására.

9.6 telepítési útmutató

9.6.1 Rendszerkövetelmények

- Operációs rendszer Ubuntu 12.04 vagy annál újabb, (3.2.24-es kernel legalább)
- Memória: 256 MB
- Háttértár: 512 MB (1 GB javasolt)
- Processzor: 1 mag

9.6.2 A forráskód letöltése

A rendszer két különálló GIT gyűjteményben található, melyet egy harmadik gyűjtemény fog közre. A forráskód letöltéséhez csupán ezt a harmadikat kell rekurzívan klónozni, az algyűjtemények automatikusan kicsomagolódnak a megfelelő helyre:

```
git clone --recursive https://github.com/arpad1337/ARBC.git
```

Az iOS kliens forráskódját is egy GIT gyűjteményben mentettem el:

```
git clone https://github.com/arpad1337/ARBC-APP.git
```

9.6.3 Dependenciák telepítése

A rendszer a következő dependenciákkal rendelkezik:

- OpenCV 2.4.9
- Leptonica 1.7
- Tesseract 3.0.2
- NPM 1.4.3
- Node 0.10.26

Az alábbi telepítési lépések Ubuntu 12.04-es verzión let tesztelve, egyéb disztribúciók esetén eltérhetnek a parancsok.

9.6.3.1 OpenCV 2.4.9

```
cd

sudo apt-get install build-essential

sudo apt-get install cmake git libgtk2-dev pkg-config
libavcodec-dev libavformat-dev libswscale-dev

sudo apt-get install python-dev python-numpy libtbb2
libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev
libdc1394-22-dev

mkdir opencv-working-dir
```

```
cd opencv-wokring-dir

git clone https://github.com/Itseez/opencv.git

cd opencv

mkdir release

cd release

cmake -D CMAKE_BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/usr/local -D WITH_TBB=ON -D
BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D
INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D
BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON
BUILD_opencv_nonfree=ON ..

make -j2

sudo make install

sudo sh -c 'echo "/usr/local/lib" >
/etc/ld.so.conf.d/opencv.conf'

sudo ldconfig

cd ../../
```

9.6.3.2 Leptonica 1.70

```
cd

mkdir leptonica

cd leptonica

wget http://www.leptonica.com/source/leptonica-1.70.tar.gz

tar –zxvf leptonica-1.70.tar.gz

cd leptonica-1.70

./configure --prefix=/usr/local && make && make install

sudo ldconfig

cd ../../
```

9.6.3.3 Tesseract 3.02.02

```
cd

sudo apt-get install libpng-dev libjpeg-dev libtiff-dev
zlib1g-dev

sudo apt-get install gcc g++

sudo apt-get install autoconf automake libtools
checkinstall

mkdir tesseract

cd tesseract

wget https://tesseract-ocr.googlecode.com/files/tesseract-
ocr-3.02.02.tar.gz

tar -zxvf tesseract-ocr-3.02.02.tar.gz

cd tesseract-ocr

./autogen.sh

./configure

make

sudo make install

sudo ldconfig

cd ../

wget https://tesseract-ocr.googlecode.com/files/tesseract-
ocr-3.02.eng.tar.gz

tar –zxvf tesseract-ocr-3.02.eng.tar.gz

mkdir /usr/local/share/tessdata

cp ./tesseract-ocr/tessdata/* /usr/local/share/tessdata/

cd ..
```

9.6.3.4 Node és NPM

```
cd
```

```
sudo apt-get install python-software-properties

sudo apt-add-repository ppa:chris-lea/node.js

sudo apt-get update

sudo apt-get install nodejs

sudo apt-get install npm
```

9.6.3.5 Boost 1.55

```
cd

wget -O boost_1_55_0.tar.gz
http://sourceforge.net/projects/boost/files/boost/1.55.0/bo
ost_1_55_0.tar.gz/download

tar xzvf boost_1_55_0.tar.gz

cd boost_1_55_0/

wget https://dl.dropbox.com/u/88131281/install_boost.sh

chmod +x install_boost.sh

./install_boost.sh

cd ../../
```

9.6.4 Image-processing modul fordítása

Ehhez készült egy fordító script, amely COMPILE névre hallgat. Ennek futtatásával történik meg az image-processing modul illetve az SWT-t végző program fordítása.

```
cd image-processing/arbc

chmod +x COMPILE

./COMPILE
```

9.6.5 API telepítése és elindítása

Az API elindításához először annak dependenciáit kell feldoldani, melyek listáját a package.json fájl tartalmazza. Telepítés:

```
cd api

npm install
```

Indítás:

```
export NODE_PORT=1337; node app.js
```

Az API az indítást követően a http://localhost:1337-es címen lesz elérhető.

9.6.6 Az alkalmazás átállítása saját API példány címére

Ahhoz, hogy az alkalmazás már az új címen elérhető API-val kommunikáljon, csupán át kell írni ennek URL címét az ARBC/APIAgent.m fájlban található *init* függvényben.