Vizuális információkinyerés és tartalomalapú képkinyerési technikák képadatbázisokban

doktori (PhD) értekezés

Veréb Krisztián

Debreceni Egyetem

Debrecen, 2004

Ezen értekezést a Debreceni Egyetem Matematika és Számítástudományok doktori iskola Informatika programja keretében készítettem 2000–2003 között és ezúton benyújtom a Debreceni Egyetem doktori Ph.D. fokozatának elnyerése céljából.

Debrecen, 200.. ... ...

.....
Veréb Krisztián
jelölt

Tanúsítom, hogy Veréb Krisztián doktorjelölt 2000–2003 között a fent megnevezett doktori program keretében irányításommal végezte munkáját. Az értekezésben foglaltak a jelölt önálló munkáján alapulnak, az eredményekhez önálló alkotó tevékenységével meghatározóan hozzájárult. Az értekezés elfogadását javaslom.

Debrecen, 200.. ... ...

.....
Dr. Kormos János
témavezető

Köszönetnyilvánítás

Ezúton is szeretnék köszönetet mondani mindazoknak, akik közvetlenül vagy közvetve hozzájárultak a disszertációm elkészítéséhez.

Témavezetőmnek, Dr. Kormos Jánosnak, aki nagy segítséget nyújtott a statisztikával kapcsolatos területeken és végig kísérte PhD tanulmányaimat,

Szüleimnek, akik mindenben mellettem álltak,

Tanáraimnak, Dr. Arató Mátyásnak, aki elindított a tudományos pályán, és sokat segített a publikációimmal kapcsolatosan,

Külön köszönet még Dr. Bognár Katalinnak, Dr. Várterész Magdának, Dr. Dömösi Pálnak, Dr. Fazekas Gábornak, Dr. Juhász Istvánnak és mindenkinek, azoknak is, akiket itt most nem soroltam fel.

Szeretnék köszönetet mondani Sipos Henriettának és Baráz Ákosnak a nyelvi lektori és nyomdai munkálatokért.

Végezetül barátaimnak akik végig segítkeztek a dolgozat és a hozzá kapcsolódó fejlesztések kivitelezésében.

Tartalomjegyzék

1. fejezet

Bevezetés

A képadatbázisok a multimédiás, képet, hangot, mozgófilmet tároló adatbázisok egy típusa. Kialakulásuk két különféle igénynek köszönhető. Az egyik a képfeldolgozás oldaláról merült fel, hogy a nagy mennyiségű feldolgozott, illetve feldolgozandó képet valamilyen egységes, szabványos módon tárolni lehessen. A képfeldolgozás oldaláról így adottak voltak az algoritmusok, matematikai modellek, csak valamilyen tároló mechanizmusra volt szükség. A másik oldalról az adatbázis-kezelésben merültek fel olyan igények, hogy ne csak primitív adattípusokat tároljunk adatbázisokban, hanem bonyolultabb, összetett adatokat is kezelhessünk. Így születtek meg kezdetben a LOB-ok (Large OBject), BLOB-ok (Binary LOB), CLOB-ok (Character LOB) tárolására is alkalmas relációs adatbázis-kezelő rendszerek, majd később az objektumok elterjedésével az objektumrelációs vagy teljesen objektumorientált rendszerek, melyekben a tárolt objektumok már lehettek multimédiás tartalommal rendelkező adattípusok is. Mivel egy adatbázis-kezelő rendszernek nemcsak az adatok tárolását kell megoldania, hanem a visszakeresésüket is, így azonnal megjelent az igény valamilyen matematikailag megalapozott visszakereső mechanizmusra. A kétfajta igény tehát adott volt, azok egyértelműen megteremtették a képi adatbázisok iránti kutatási igényeket is.

Dolgozatom első részében bemutatom azokat a technikákat, melyek megjelentek a képadatbázisokból történő tartalomalapú képkinyeréseket illetően. Az irodalomban található technikák valójában három nagyobb csoportba oszthatók. Az első csoport a szűkebb értelemben vett CBIR (Content-Based Image Retrieval), azaz a tartalomalapú képkinyerés. Ide tartoznak a különféle illesztő algoritmusok. A második csoport az indexelési technikák csoportja, mely már jóval kevesebb irodalommal rendelkezik. A harmadik csoport pedig a keresési interfészek, illetve lekérdezőnyelvek csoportja.

A második részben a fenti problémakörökhöz kapcsolódó saját kutatási eredményeimet mutatom be. A keresési algoritmusok tekintetében a statisztikus alakfelismerésben, az indexeléshez a szemantikus indexelési technikák területén, míg a lekérdezőnyelvek tekintetében a fuzzy nyelvek felhasználásával végeztem kutatásokat. Ez utóbbi nemcsak mint lekérdezőnyelv szolgál, hanem a Cut-And-Or-Not technika segítségével lehetőséget teremt a komplex kérdésfeltevésre is.

A dolgozat harmadik részében egy alkalmazási példát mutatok be, mely Oracle $9i$ objektumrelációs környezetben mutat példát a képek tárolására és visszakeresésére.

Alapok

2. fejezet

Fogalmak, jelölések

A multimédiás, képet, hangot, videót tároló dokumentumok széleskörű elterjedése becsempészte az informatika mindennapjaiba a multimédiát (és vice versa, a médiákba az informatikát). A dokumentumok kezelésén túl megjelentek a tisztán multimédiás alkalmazások is, de felmerültek az anyagok tárolásának és visszakeresésének problémái is. Ez utóbbi két részproblémával foglalkoznak a multimédiás adatbázisok. Maga a probléma olyan kiterjedt, hogy számos kapcsolódó informatikai területet is nagymértékben érint. Ha meggondoljuk, már csak a médiatípusok osztályozásával is (kép, hang, mozgófilm) hatalmas területeket érintünk (képfeldolgozás, hangfelismerés, stb.).

A képadatbázisok az egyik legelterjedtebb formái a multimédiás adatbázisoknak. Első megközelítésben feladatuk a képi információk és a hozzájuk kapcsolódó szöveges információk tárolása, és visszakeresése. Absztrakt szinten a visszakeresés a problematikusabb és fontosabb. A visszakeresés működése a következő: valamilyen ismérvek alapján leírjuk a képeket, majd ezen ismérvek alapján keresünk a tárolt képek között. Ez történhet egy hasonló kép megadásával, vázlattal, szöveges leírással, stb. A fontos az, hogy valamilyen tulajdonságait adjuk meg a képnek. Ezek a tulajdonságok általában a képből, vagy környezetéből kinyerhetők. Ezek után valamilyen hasonlósági mérték szerint össze kell hasonlítani a tulajdonságokat, hogy melyek a számunkra fontosak. Itt jelennek meg a különféle távolság- és hasonlóság fogalmak, illetve metrikák. Mivel nagy mennyiségű információról van szó, a tulajdonságokat, illetve magukat a képeket valamilyen indexszerkezettel indexelik, hogy elősegítsék a visszakeresést. Mivel ezek mind elég erős szakmai ismereteket kívánnak, valamilyen interfészt kell a felhasználó felé nyújtani, amelyen keresztül felvázolhatja a konkrét adatbázis lekérdezést, majd megkapja az eredményt. Gyakori, hogy ekkor a rendszer valamilyen visszacsatolást vár a felhasználótól, hogy mennyire jó az eredmény, több eredmény esetén rangsor felállítását is kérheti. Ezek a visszacsatolási információk (amennyiben a rendszer képes rá) betanítási funkciókat is elláthatnak. Általános, mindent átfogó modellekről bővebben olvashatunk a [37] [53] [26] [19] [49] és [31]-ben.

Ha nem absztrakt szemszögből nézzük a képadatbázisokat, akkor már maga a tárolás is kérdéseket vet fel. Sőt, a kereső algoritmusok megvalósíthatósága is előtérbe kerül. Az irodalom általában két nagyobb részre szakad a tárolást illetően. Vannak, akik azt hangoztatják, hogy a képadatbázis maga az Internet. A HTML lapok a dokumentumok, melyek beágyazva képeket tartalmazhatnak. A kereső rendszerek-

nek tehát az internetes szövegkeresőket kell kiegészíteniük, melyek implementációja – a CGI és egyéb megoldásoknak köszönhetően – nem ütközhet túl nagy problémába. A másik nagyobb csoportba azok tartoznak, akik a meglévő adatbázis-kezelő rendszereket próbálják meg kiegészíteni képek tárolására és visszakeresésére alkalmas algoritmusokkal. Itt kérdéses, hogy milyen formában tárolható a kép, és az is, milyen nyelven lehet (vagy lehet-e egyáltalán) implementálni a szükséges algoritmusokat, ugyanis számos adatbázis-kezelő rendszer nem nyújt procedurális megközelítési módot adatai eléréséhez.

Most tekintsünk át néhány, a képfeldolgozás témakörébe tartozó definíciót, melyek elengedhetetlenek számunkra a képadatbázisok további vizsgálataihoz.

1. Definíció. *Az $X \subseteq \mathbb{Z} \times \mathbb{Z}$ halmazt, ahol $\mathbb{Z}$ az egész számok halmaza, kétdimenziós digitális halmaznak nevezzük.*

2. Definíció. *Legyen adott egy X digitális halmaz és egy $f : X \to \{0, \ldots, m\}$ leképezés, ahol $m \geq 1$. Ekkor f egy $m + 1$ szintű digitális kép. Ha $m = 1$, bináris kép.*

3. Definíció. *Legyen adott egy X digitális halmaz, és egy $f : X \to \{0, \ldots, m\}$ digitális kép. Ekkor ez (x, p) párost, ahol $x \in X$ és $p \in \{0, \ldots, m\}$ pixelnek nevezzük. x a koordináta és p a pixelintenzitás.*

A továbbiakban gyakran alkalmazom a szignatúra kifejezést, mely a kép tulajdonságainak valós vektorok formájában kifejezett alakját jelzi (szinonimája a tulajdonságvektor kifejezésnek). A dolgozat második felében szintén gyakran előfordul az objektum kifejezés. Ez egyrészt a képeket, másrészt a képen található ábrákat, foltokat jelöli. Amennyiben a kép objektumorientált absztrakciójáról, illetve tárolásáról van szó, az objektum az objektumorientált fogalmaknak megfelelő objektumot jelöli (lásd SIMULA, SMALLTALK programozási nyelvek, a [9] illetve a [13] objektumorientált adatbázisokra vonatkozó részeit).

Az alkalmazott programozási alapfogalmak tekintetében a [50], a képfeldolgozási alapok tekintetében a [52] és [21], az adatbázis-kezelő rendszerekkel kapcsolatosan a már említett [13], metamatikai statisztikai és matematikai logika tekintetében pedig a [17] [51] és [54] [8] tárgyalt fejezeteit tekintem mérvadónak.

A dolgozat során mindenütt törekedtem az egységes jelölésekre. Mivel a dolgozat anyaga eléggé szerteágazó, így rengeteg jelölést kellett bevezetnem. Ahol külön nem említem, egy adott jelölés mit azonosít, ott az alábbiak szerint jártam el:

X, Y, Z, $\ldots$: digitális halmazok

x, y, z, a, b, $\ldots$: a d-dimenziós tér vagy digitális halmaz pontjai

f, g, $\ldots$: képek (például digitális kép)

Q, q: illesztés, illesztés eredménye (kiértékelt illesztés)

$\underline{x}$, $\underline{y}$, $\underline{z}$, $\ldots$: vektorok

l, l^f, l_i^f, $\mathfrak{L}$: lánckód, egy f objektum lánckódja, a lánckód i-edik eleme, illetve a lánckódot képező transzformáció

d, d^f, d_i^f, $\mathcal{D}$: differencia kód, egy f objektum differencia kódja, a kód i-edik eleme, illetve a differencia kódot képező transzformáció

s, s^f, s_i^f, $\mathcal{S}$: alak kód, egy f objektum alak kódja, a kód i-edik eleme illetve az alak kódot képező transzformáció

D, D_i: előjeles differencia kód, illetve annak i-edik eleme

$\delta(x, y)$: x és y valamilyen értelemben vett távolsága

$\mathcal{C}$: a vágás (cut) operátor

$\boldsymbol{C}$: osztályhierarchia

Obj: objektumok halmaza (típusa)

obj: objektum (képobjektum)

$\mathcal{F}$: a tulajdonságvektorokat kinyerő transzformáció

$\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{R}^+$, $\mathbb{R}_0^+$: a természetes számok halmaza, az egész számok halmaza, a valós számok halmaza, a pozitív valós számok halmaza és végül a nemnegatív valós számok halmaza

3. fejezet

Általános megközelítés

3.1. Miért van szükség képadatbázisra?

Középiskolában a rajztanárom a művészettörténet keretében minden év végén arra kért, hogy egy adott művészettörténeti korszakról készítsek prezentációt. Ez abban az időben fényképgyűjteményt jelentett kevéske magyarázó szöveg kíséretében, úgymint besorolás (festmény, szobor, épület, egyéb), szerző, mű címe, kora stb. Nos, azok a prezentációk (a szó általánosabb értelmében) kezdetleges képadatbázisok voltak. Az [58]-ban kifejtem, hogy miket kell tudnia egy képadatbázisnak. Nem elég a képek tárolását megoldania, hanem azok visszakereshetőségét is biztosítani kell. Ahogy teltek az évek, a gyűjtemény egyre bővebb lett, és az új elemeket a régiek közé kellett beilleszteni. Na, ekkor a visszakeresés abból állt, hogy az elejétől el kellett kezdenem végignézni az összes képet, és a szöveges adatok alapján kikeresni az adott kort. Onnantól viszont a képeket kellett átnézni, hogy nehogy valamit kétszer tegyek bele a prezentációba. Gyakran ezt a sok munkafázist elkerülendő, emlékezetből próbáltam meg eldönteni, egy adott kép szerepel-e az „adatbázisomban" vagy nem. De mit „kell" megjegyezni egy képen a visszakereséshez?

Biztos mindenki emlékszik az általános iskola első osztályában eltöltött napokra. Ki így, ki úgy, de azért abban megegyezhetünk, ha visszagondolunk egy konkrét napra, hogy a megmaradt emlékek két nagyobb csoportba oszthatók. Az egyik csoport a konkrét emlékek osztálya. Emlékszünk a padon a feliratokra. Szinte magunk előtt látjuk a vésett betűket. Emlékszünk az osztálytársaink nevére (vagy legalábbis nagy részükre). Viszont sok meghalványult emlékünk van, amit nem tudunk konkrétan, csak sejtésünk van róla. Például tudjuk, hogy voltak ablakok. De hogy mennyi, és hogy hány részre voltak az ablaktáblák osztva, már nem biztos, hogy meg tudnánk mondani.

Valahogy hasonlóan van ez a képekkel is. Ha megpróbálunk felidézni a családi fotóalbumból egy képet, például szüleink, nagyszüleink házassági képét, emlékszünk az alakokra. Hogy a menyasszony a vőlegénytől jobbra állt, fehér ruhában, a vőlegény fekete hosszanti csíkos öltönyben volt. De hogy milyen a háttér, arra biztos kevesen emlékeznek. Vagy nézzük meg Picasso galambját (3.1 ábra). Majd csukjuk be a szemünket, és emlékezzünk vissza a képre. Hány toll ábrázolás van a galamb bal szárnyán? Három vagy négy? És a jobb szárnyán? De hogy volt valami a csőrében

abban biztosak lehetünk. Hány ága volt?

A vizuális információk több nagyobb csoportba oszthatók. A fentiekből is látszik, hogy a legtöbb ember egy képet szemlélve megjegyzi a képen lévő objektumok egymáshoz viszonyított elhelyezkedését. Az objektumok színét, esetleg textúráját. Az objektumok alakját. Ezek az alapvető, nem szemantikus, alacsony szintű tulajdonságok. Ezen felül megjelennek a magas szintű tulajdonságok is, melyek már több szemantikával rendelkeznek. Ilyenek például, hogy tudjuk, a képen az alakok emberek, sőt, tudjuk, hogy az egyik vőlegény, a másik pedig menyasszony. Sőt, azt is tudjuk, hogy Picasso ominózus képén egy galamb van.

Amint láthatjuk egy képadatbázis létrehozása fontos abban az esetben, ha a képeket nemcsak tárolni, hanem visszakeresni is akarjuk. A visszakeresés alapja pedig az alacsony- és magasabb rendű tulajdonságok halmaza. Már csak az a kérdés, miért akarunk képadatbázisokból képeket visszakeresni?

3.2. Keresési modellek

A képek tárolása és a visszakeresése két fontos dolgot igényel. Egy tároló mechanizmust és egy visszakereső mechanizmust. A tárolás nemcsak adatbázis-kezelő rendszerek segítségével valósítható meg, hanem – a képek állományjellegéből adódóan – arra felhasználható az Internet is, ahol a weboldalak tárolását illetően a gépek sebessége nem játszik igazán fontos szerepet. A visszakereső mechanizmus viszont mindenképp olyan munkaállomást kíván meg, amelynek számítási kapacitásai nagyobbak. Amennyiben tehát van egy erősebb számítási kapacitásokkal rendelkező illesztő egységünk, és egy tároló egységünk, egy keresés esetén a kettő közötti kommunikációban a tárolt képeket minden egyes illesztés esetén át kell küldeni a hálózaton. Ez meglehetősen lassú. Gyakori, hogy nem magukat a képeket küldjük át a hálózaton, hanem őket leíró, általános esetben kisebb ún. tulajdonságvektorokkal helyettesítjük a képeket. Ez a megoldás több problémát is rejthet magában. Előfordulhat ugyanis olyan eset, hogy az illesztő algoritmus megköveteli a teljes kép megadását, vagy a képből kinyert tulajdonságvektorok mérete megegyezik, vagy meghaladja a képek méretét. Ilyen esetekben érdemes valamilyen módon a tároló és visszakereső állomásokat valamilyen módon közelebb hozni a hálózati kommunikáció minimalizációja érdekében. Ahogy azt [57]-ben is kifejtem, adatbázis-kezelő rendszert alkalmazó megoldások esetén ez úgy valósítható meg, hogy egy adatbázisközeli nyelven, magán az adatbázis-kezelő szintjén hajtjuk végre az illesztő algoritmusokat. Így adott nekünk egy komplex tároló és visszakereső egység, egy „valódi” képadatbázis. A kérdés viszont továbbra is megmaradt, miért fordulunk kérdéssel egy ilyen egységhez?

Santini a [49] könyvében kifejti, hogy Ornager [42], Makkula és Sormounen [34] újságírókat figyeltek meg, hogyan és milyen képeket keresnek cikkeikhez a képarchívumokban. Ezek öt nagyobb csoportba oszthatók.

Van aki pontosan tudja mit keres. Megy, és megmondja az archívum kezelőjének, hogy ő életrajzot ír, és ezért Csontváry Kosztka Tivadar, 1907-ben készült Magányos Cédrus című festményét keresi (3.1 ábra).

A második csoportba azok tartoznak, akik már nem egy adott képet keresnek, hanem inkább több hasonlót, melyből majd kiválasztanak egyet. Például Ronald Reagen-ről keresnek képeket, ahol éppen napi teendőjét végzi, vagy éppen az elnökválasztásról keresnek egy tucat képet.

3.1. ábra. Csontváry és Picasso képei.

A harmadik csoportba a történetet elmesélők kerülnek. Ők elmesélik a történetet az archívum vezetőjének, és a segítségét kérik, adjon meg olyan képcsoportokat, melyek illusztrálhatják az adott történetet. Gyakran csak hangulatokat, érzéseket akarnak kifejezni.

A negyedik csoport a lusta témamesélő. Ő is elmondja a cikk tartalmát, de nem foglalkozik a képpel, annak kiválasztását egy az egyben az archívumban dolgozóra bízza.

Az ötödik csoport pedig csak helykitöltésre használja a képeket. Ők már csak a kép méretével foglalkoznak.

Persze nemcsak az újságírók keresgélnek képadatbázisokban. Íme a teljesség igénye nélkül néhány terület, ahol képadatbázisokra lehet szükség: orvosi képfeldolgozás (mikrobák alakja alapján történő keresés), térinformatika (járulékos információk alapján történő térképkeresés), művészettörténeti katalógusok (festmények keresése), azonosító/beléptető rendszerek (ujjlenyomat visszakeresés) stb. Ezekről többet olvashatunk Michael S. Lew és Thomas S. Huag leírásában [31]-ben.

A cél most már adott, már csak a mikéntet kell osztályozni. Megfigyelhető, hogy a keresések, a keresési modellek három nagyobb csoportba oszthatók a képekről rendelkező információk tekintetében. Van ahol a képről mindent tudunk. Pont az adott képet keressük, pontosabban nem is a képet, hiszen az már meg van nekünk, hanem a hozzá kapcsolódó járulékos információkat. A másik csoportba azok a keresések tartoznak, ahol szintén rendelkezünk egy többé-kevésbé pontos képpel, és a hozzá hasonlókat szeretnénk leválogatni (esetleg a legpontosabbat megkapni). A harmadik csoport, ahol egyáltalán nincs kép, járulékos információk (például szöveges leírás) alapján keressük a leírásnak megfelelő képeket.

Mindegyik keresés esetében észrevehető, hogy létezik valamilyen archívumvezető, aki elvégzi a megkapott információk alapján a keresést. Ez nem más, mint a visszakereső mechanizmusunk, az illesztő egységünk. Az a nyelv, amelyen ő kommunikál velünk, az az interfész. És maga az archívum az adatbázis. A keresési interfészekről a későbbiekben még szó lesz, most tehát összegezzük a keresések menetét.

Adott tehát egy keresési kritérium, és a legtöbb esetben egy kereső kép, amihez hasonlót keresünk. Majd adottak az illesztendő képek az adatbázisban. Eljuttatjuk a kereső képet a képadatbázishoz, ahol a kereső képből megtörténik a szükséges tulajdonságvektorok kinyerése, majd megtörténik a tárolt illesztendő képekből a tulajdonságvektorok kinyerése (ha azok még nem lennének kinyerve és letárolva, mint

ahogy azt [58] és [59]-ben is ajánlom), majd végrehajtódik a vektorok egyezésének vizsgálata az adott keresési kritériumok alapján. Itt ezen a ponton újabb kérdések merülnek fel. Például az, hogy mik ezek a tulajdonságvektorok?

3.3. Tulajdonságok

Egy adott keresés esetében, tekintsünk egy adott g képet. Ebből a képből szeretnénk valamilyen tulajdonságokat kinyerni. A kinyerhető tulajdonságvektorokat két nagyobb csoportba szokás sorolni [49]. Az első csoportba azok a vektorok tartoznak, melyekből a kép kis hibával teljes mértékben visszaállítható. Ez a csoport a reprezentáció. A másik csoportba azok a vektorok tartoznak, melyekből a kép nem állítható vissza, de a kép, vagy a képen található objektumok valamilyen mérhető tulajdonságait reprezentálják. Ezek a jellegzetességek. Az illesztések gyakran mindkét fajta tulajdonságvektor meglétét is igénylik egy-egy hasonlóság eldöntéséhez. A vektor szó maga mindkét osztály vektorait jelenti. Az irodalomban legtöbbször csak a reprezentációt szokás külön nevesíteni, ha az szükséges.

Így egy képreprezentációt, illetve a kép egy kinyert tulajdonságát egy d dimenziós

$$\mathcal{F}(g) = \underline{x} = \langle x_1, \ldots, x_d \rangle \tag{3.1}$$

vektornak tekintjük, ahol $x_i \in \mathbb{R}$, $i = 1, \ldots, d$.

A tulajdonságvektorokat másként is lehet osztályozni. Ez az osztályozás az attentív/pre-attentív osztályozás.

Legyen egy U felhasználó, aki kiadott egy Query lekérdezést egy g képre. A hasonlóság becslése a Query és az g között egy $P(g|\text{Query}, U)$ feltételes kifejezéssel adható meg [31]. Bayes tétele alapján

$$P(g, \text{Query}, U) = P(g|\text{Query}, U)P(\text{Query}, U) = P(g|\text{Query}, U)P(\text{Query}|U)P(U) \tag{3.2}$$

$$P(g, \text{Query}, U) = P(\text{Query}|g, U)P(g, U) = P(\text{Query}|g, U)P(g|U)P(U) \tag{3.3}$$

azaz

$$P(g|\text{Query}, U) = \frac{P(\text{Query}|g, U)P(g|U)}{P(\text{Query}|U)}. \tag{3.4}$$

Most keressük a maximum értékét ennek a valószínűségnek az összes g kép felett. Tehát a $P(\text{Query}|U)$ kifejezés nem szól ebbe bele, azaz a $P(\text{Query}|g, U)P(g|U)$ szorzatot kell maximalizálni. A $P(\text{Query}|g, U)$ *a posteriori* kifejezés valamilyen attentív mechanizmust tételez fel, azaz az összehasonlított stimulusoknak létezik előtte valamilyen értelmezése. Ellenben a $P(\text{Query}|g)$ pre-attentív rész nem tételez fel előzetes értelmezést (a felhasználótól független érték). Így érdemes elkülöníteni a $P(\text{Query}|U)$ és $P(\text{Query}|g)$ értékeket, azaz az alábbi szorzat maximalizálandó:

$$P(\text{Query}|U)P(\text{Query}|g). \tag{3.5}$$

Ahogy Santini is tette, azt tételezzük fel, hogy az attentív hasonlóság tehát nemcsak a tulajdonságokon, hanem az illesztési folyamaton is alapszik, míg a pre-attentív hasonlóság csak magukon a tulajdonságokon. A klasszikus illesztési algoritmusok

általában keverik a kettőt, azaz a kinyert, pre-attentív hasonló tulajdonságokkal rendelkező képeket visszaadják a felhasználónak, rangsorolja be őket attentív hasonlóság szerint (visszacsatolás).

Az illesztés absztrakt esetben nem más, mint távolságmérés a kinyert vektorokon.

Legyen adottak nekünk a képek, azok tulajdonságvektorai (reprezentációi), és szükségünk van egy metrikára hogy az f adatbázisbeli képeket összehasonlítsuk a Query lekérdezéssel. Használjunk egy

$$\delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_0^+ \tag{3.6}$$

alakú δ távolságot. Biztos, hogy erre van szükségünk? Nézzük meg, miket kell kielégíteni egy δ távolságnak f_1, f_2 és f_3 képek esetén.

$$\begin{array}{ll} p_1 : \delta(\mathcal{F}(f_1), \mathcal{F}(f_1)) = \delta(\mathcal{F}(f_2), \mathcal{F}(f_2)) & \text{önhasonlóság} \\ p_2 : \delta(\mathcal{F}(f_1), \mathcal{F}(f_2)) \geq \delta(\mathcal{F}(f_1), \mathcal{F}(f_1)) & \text{minimalitás} \\ p_3 : \delta(\mathcal{F}(f_1), \mathcal{F}(f_2)) = \delta(\mathcal{F}(f_2), \mathcal{F}(f_1)) & \text{szimmetria} \\ p_4 : \delta(\mathcal{F}(f_1), \mathcal{F}(f_3)) + \delta(\mathcal{F}(f_3), \mathcal{F}(f_2)) \geq \delta(\mathcal{F}(f_1), \mathcal{F}(f_2)) & \text{háromszög egyenlőtlenség} \end{array} \tag{3.7}$$

Azok, melyek kielégítik p_1, p_2 és p_4 tulajdonságot, azok a metrikák, amelyek pedig a p_1, p_2 és p_3 tulajdonságokat elégítik ki azok a hasonlóságok (különbözőségek). Bár meg kell említeni, hogy az emberi érzékelés ezeket nem mindig támasztja alá. A háromszög egyenlőtlenséget pedig szinte lehetetlen értelmezni az emberi érzékelés szerint.

A hasonlóság legminimálisabb követelményei a p_1 és a p_2. Persze ha valaki mindegyik tulajdonságot megcáfolja, annak új megszorításokat kell bevezetnie. (Tversky [55] próbálkozott hasonlóval).

Napjaink hasonlósági mértékei a képekből már korábban kinyert tulajdonságvektorok összehasonlításán alapszik. Ezek a vektorok főleg az alábbiakat jellemzik:

- Szürkeségi szintek,
- Színek hisztogramokkal, vagy momentumokkal
- Textúrák (Fourier, Gabor, együtthatók megadásával)
- Alakok, geometriai tulajdonságok (görbék)
- Struktúra

A tulajdonságvektorok hasonlóságának vizsgálata esetén az is felmerül, hogy vajon melyik vektor a jó? Az amelyik a teljes képre vonatkozó információkat hordozza, vagy az, amelyik csak egy részére vonatkozókat (és ekkor más-más részhez más-más vektor tartozik).

Carson és társai [3] ajánlották először, hogy szegmentáljuk a képet régiók egy halmazára. Ez a megközelítési mód eléggé összetett, ugyanis egy jól működő szegmentációs lépést kíván meg (és mint tudjuk, a szegmentáció sosem működik helyesen). Sőt, azt feltételezi, hogy a szegmentáció által kinyert régiók közeli kapcsolatba hozhatók a jelenetben található objektumokkal. Ez leginkább csak homogén szín/textúra esetén lehetséges. Ezután a régiók hasonlóságát kell vizsgálni. Ez lehetséges például középvonaluk alapján [48], vagy úgy, hogy kinyerjük a régiók határait [25], majd így

azokat, mint foltokat vizsgáljuk. Kormos Jánossal végeztem hasonló vizsgálatokat a [27] és [28]-ban.

Egy másik megközelítési mód a kulcspontokon alapszik. Két jel akkor hasonló, ha bizonyos tulajdonságértékeik helyileg egymáshoz konzisztensen helyezkednek el. Ezek a bizonyos értékek a kulcspontok. Ebben a megközelítési módban nincs szükség minden tulajdonságra, csak bizonyos (számunkra fontos) tulajdonságértékekre.

Egy tulajdonságérték lehet globális, ha minden pixelt figyelembe vesz a képen, illetve ellenkező esetben lokális. Például a pixelintenzitásoknak használhatjuk valamilyen átlagát, mint globális értéket, illetve pixelről-pixelre is vizsgálhatjuk azokat. A lokális tehát azt jelenti, hogy meghatározásához a képnek csak egy részét használjuk fel (általában egy pont valamilyen szomszédjait).

A legtöbb rendszer úgy működik, hogy veszi sorra a tulajdonságokat, azokat összehasonlítja, majd az eredményeket összekombinálja. Ez abból fakad, hogy a különböző tulajdonságok egymással nem összemérhetőek (a szín nem textúra stb.). Tehát a hasonlóság az valamilyen többdimenziós dolog, míg nekünk egy egydimenziós mérőszámot kell adni, így a részeredmények valamilyen lineáris kombinációját tekintjük. Probléma a súlyok megadásával lehet, így több rendszer ebből a szempontból interaktívan működik. Viszont ha a felhasználó nem tudja értelmezni ezeket a súlyokat, akkor több lehetséges iterációs lépésen keresztül kell finomítani egyesével a vektorok közti hasonlóságokat, hogy jó eredményt kaphassunk.

A következő fejezetben azokat a keresési algoritmusokat mutatom be, melyek a fent említett kinyert tulajdonságvektorok hasonlóságát vizsgálják. Így bemutatásra kerül a szürkeségi szintek (pixelintenzitások) vizsgálati módszere, a színek hisztogramjainak vizsgálata, a textúra és az alakok, görbék illeszkedésének vizsgálata.

4. fejezet

Keresési módszerek

Ebben a fejezetben a képadatbázisokhoz használatos főbb illesztő algoritmusok működéséről lesz néhány szó. Mivel az algoritmusok mindegyike elég nagy témakör, több könyvet is kitenne, így csak ismertető jelleggel tekintjük át őket.

4.1. Bináris illeszkedés

Bár ezt tekintjük át elsőként, általában ez a „végső menedék" egy illesztés során [56]. Ez a mezítlábas mintaillesztés kétdimenziós formája. Végül is nem más, mint a képek két szintre vágása után a 0 és 1 pixelértékek százalékos egyezésének vizsgálata a két képen. A képek két szintre vágása a küszöbölés. A színes képek küszöbölése visszavezethető a szürkeskálás képek küszöbölésére, így mi most csak ezzel foglalkozunk.

Tekintsük át az alapkoncepciókat [20] [21] [43]. Legyen $f(x, y)$ egy k szintű digitális $M \times N$ méretű kép. Célunk, hogy bináris képet készítsünk ebből a képből egy küszöb megadásával. A legegyszerűbb küszöbérték a $\frac{k}{2}$. Az eredmény $g(x, y)$ bináris kép ekkor a következő formában áll elő

$$g(x,y) = \begin{cases} 0, & \text{ha } f(x,y) \leq \frac{k}{2} \\ 1, & \text{egyébként} \end{cases} \tag{4.1}$$

Alacsony kontrasztú vagy alul/túlexponált képek esetén ez a módszer nem megbízható. Egy jobb megoldás az átlagérték használata. Ekkor az eredménykép:

$$g(x,y) = \begin{cases} 0, & \text{ha } f(x,y) \leq \overline{f(x,y)} \\ 1, & \text{egyébként} \end{cases} \tag{4.2}$$

ahol

$$\overline{f(x,y)} = \sum_{i=0}^{M} \sum_{j=0}^{N} \frac{f(i,j)}{M \cdot N}. \tag{4.3}$$

Természetesen még ez a módszer sem kielégítő. Léteznek lokálisan működő eljárások, melyek egy képpont 0 vagy 1 értékének meghatározása esetén csak adott szomszédjait vizsgálják a pixelnek (szinte pixelenként változik a küszöbérték), illetve olyan globális módszerek, melyek a kép intenzitás hisztogramjának elemzésén alapszanak (hisztogramsimítás, bipolarizáció).

Miután megkaptuk a keresett bináris képeinket, két azonos méretű bináris kép százalékos bináris illesztésének eredménye

$$1 - \sum_{i=0}^{M} \sum_{j=0}^{N} \frac{|f(i,j) - g(i,j)|}{M \cdot N} \tag{4.4}$$

ahol f és g a két $M \times N$ méretű bináris kép.

A módszer általánosításaival, kiterjesztésével szürkeskálás képekre, eltolás- és nagyítás invariáns változataival [56]-ban bővebben foglalkozom.

Ha ki akarjuk küszöbölni a szintrevágás okozta problémákat, az illesztést világosságkód invariánssá kell tenni. Ez aránylag még egy egyszerű feladat. De mi történik akkor, ha a két kép mérete nem megegyező. Sőt, ha a kereső kép mérete jóval kisebb, mint a keresett kép mérete, és a keresés úgymond eltolás invariáns, azaz a kereső kép a keresett képen saját méretével azonos területen található meg. Ilyenkor az is kérdéses, mely pozíción található meg a kereső kép. Ha az eltolás invarianciához hozzávesszük a nagyítás invarianciát is, akkor az is kérdéses, mekkora nagyítással található meg az adott pozíción a kereső kép. Ilyenkor nagy figyelmet kell szentelni a nagyításból eredő hibákra, zajokra is.

Ha a képeink $f(x,y)$ és $g(x,y)$, ahol azok méretei $M \times N$ és $K \times L$, $K \ll M$ és $L \ll N$ akkor legyen $f'(x,y) = f(x,y) - \min\{f(x,y)\} + t_2$, és $g'(x,y) = g(x,y) - \min\{g(x,y)\}$, ahol $t_2 \geq 0$ egy tetszőleges egész. Tegyük fel, hogy egy nagyított g található valahol az f képen.

Az f mint függvény elmozgatható a $z = t_2$ síkra az x, y, z koordináta rendszerben. A g pedig leszállítható az x és y által kifeszített síkra. Ha egy megfelelő helyen kijelölünk egy pontot a z tengely negatív tartományában az x, y síkja alatt, az a pont tekinthető egy vetítési pontnak. Legyen ez a pont a $(0, 0, -t_1)$ pont. Minden egyes illesztéskor jelöljünk ki egy k számot, ahol $k = 0, \ldots, t_2$.

Általános esetben ha n-szeresére nagyítunk egy $f(x,y)$ képet, az nem $nf(x,y)$ lesz. A háromdimenziós térben egy nagyított kép $nf(x/n, y/n)$ alakban adható meg. A képfeldolgozásban viszont ez a világosságkód értékek miatt csak $f(x/n, y/n)$ alakban vihető végbe. Ezért tehát egy (x_0, y_0) g'-beli pont f'-beli (x_1, y_1) megfelelője [56]-szerint az $x_1 = \frac{x_0(t_1+t_2)}{t_1+k}$ és $y_1 = \frac{y_0(t_1+t_2)}{t_1+k}$ alakban adható meg.

Legyen

$$D_{i,j,k}(x,y) = |f'(x_1 + i, y_1 + j) - g'(x,y)|, \tag{4.5}$$

ahol $i = 0, \ldots, M - K$, $j = 0, \ldots, N - L$, $x = 0, \ldots, K$, $y = 0, \ldots, L$, $k = 0, \ldots, t_2$ és $x_1 = \frac{x(t_1+t_2)}{t_1+k}$ egész része, és $y_1 = \frac{y(t_1+t_2)}{t_1+k}$ egész része. Ha $x_1 > M$ vagy $y_1 > N$, akkor D nem definiált.

A függvényértékek abszolút értékben vett különbségének tulajdonságai miatt az f és g függvény nyugodtan használható az f' és g' függvények helyett. Legyen tehát

$$D_{i,j,k}(x,y) = |f(x_1 + i, y_1 + j) - g(x,y)|. \tag{4.6}$$

Vezessünk be egy r mérőszámot,

$$r_{i,j,k} = \sum_{m=0}^{K} \sum_{n=0}^{L} |D_{i,j,k}(m,n) - \overline{D_{i,j,k}}|, \tag{4.7}$$

ahol $\overline{D_{i,j,k}}$ a D átlagértéke.

A hibakezelés miatt a $g(x_0, y_0)$ megfelelő pontja nem biztos, hogy $f(x_1+i, y_1+j)$, annál inkább $f(x_1 + i \pm \Delta, y_1 + j \pm \Delta)$. Ebben az esetben a D az alábbi módon számolható:

$$D_{i,j,k}(x, y) = |\overline{f(x_1 + i, y_1 + j)} - g(x, y)|, \tag{4.8}$$

ahol

$$\overline{f(x_1 + i, y_1 + j)} = \frac{1}{2\Delta} \sum_{m=0}^{\Delta} \sum_{n=0}^{\Delta} f(x_1 + i - \frac{\Delta}{2} + m, y_1 + j - \frac{\Delta}{2} + n) \tag{4.9}$$

ha az létezik. Ha nem, $\overline{f(x_1 + i, y_1 + j)}$ legyen $f(x_1 + i, y_1 + j)$.

Így a fenti (4.4) képlet általánosítható. Ekkor ahol $r_{i,j,k}$ minimális, abban az i, j pontban a g kereső kép megtalálható az f képen. A nagyítás mértéke ekkor $(t_1 + t_2)/(t_1 + k)$, az illeszkedés mérőszáma pedig maga az r.

4.2. Hisztogramok távolsága

A hisztogramok (vektorok), mint eloszlások távolságának meghatározása egy nagyon jól alkalmazható terület a képadatbázisokban történő keresésekben. Most nézzük meg, mit nevezünk hisztogramnak, és hogyan mérhető meg két hisztogram távolsága. Tekintsük értékek egy halmazát $X = \{x_1, \ldots, x_m\}$, és legyen adott binek (osztályok) egy $B = \{B_1, \ldots, B_n\}$ halmaza. Vezessünk be egy b_k indikátor függvényt, mely

$$b_k(x) = \begin{cases} 1 & \text{ha } x \in B_k \\ 0 & \text{egyébként} \end{cases} \tag{4.10}$$

alakú. Ekkor egy $H = (H(1), \ldots, H(n))$ hisztogram úgy áll elő, hogy

$$H(k) = \frac{1}{m} \sum_{x \in X} b_k(x) \tag{4.11}$$

ahol $k = 1, \ldots, n$

A leggyakrabban alkalmazott módszer hisztogramok összehasonlítására a klasszikus távolság. Tekintsük két eloszlás távolságát (ahol a két eloszlás nem más, mint két hisztogram). Feltételezzük, hogy a hét hisztogramnak, H_0-nak és H_1-nek ugyanannyi binje van (n). Ilyenkor alkalmazható például a Minkowski távolság

$$\delta_{L_p}(H_0, H_1) = \left[\sum_{i=1}^{n} |H_0(i) - H_1(i)|^p \right]^{\frac{1}{p}} \tag{4.12}$$

ahol, ha $p = 2$, akkor az Euklideszi távolságot kapjuk. Ha $p = 1$, akkor a Manhattan (city block) távolságot kapjuk. Ha $p \to \infty$, akkor a határérték alkalmazható. Ezek a távolságok viszont nagyon érzékenyek az eltolásra (az értékek elléptetésére), illetve a zajra. A Kullback-Leiber távolságra építve a zajjal szemben robosztusabb távolságot kaphatunk (Jeffrey távolság):

$$\delta_J(H_0, H_1) = \sum_{i=1}^{n} \left[H_0(i) \log \frac{H_0(i)}{m_i} + H_1(i) \log \frac{H_1(i)}{m_i} \right] \tag{4.13}$$

ahol $m_i = \frac{H_0(i)+H_1(i)}{2}$. Ez már kevésbé zajérzékeny, viszont az eltolást ez sem szereti. Ilyenkor a Bhattacharyya távolság alkalmazható (korreláció):

$$\delta(H_1, H_2) = -\log \sum_{\alpha} \sqrt{H_1(\alpha) H_2(\alpha)} \tag{4.14}$$

ahol a hisztogramok természetesen normalizáltak, azaz $\sum_{\alpha} H_i(\alpha) = 1$. Ez rendelkezik a szimmetria tulajdonsággal, de nem teljesíti a háromszög egyenlőtlenséget.

Jól alkalmazható még a metszet távolság:

$$\delta_{\cap}(H_1, H_2) = 1 - \sum_{k=1}^{n} \min\{H_1(k), H_2(k)\} \tag{4.15}$$

illetve az L_1 és L_2 távolságok (az L_p távolságból származtatva).

A hisztogramtávolságok leggyakoribb alkalmazási területe a színek illeszkedésének vizsgálata.

4.3. Színek illeszkedése

El tudjuk képzelni az életet színek nélkül? Habár nem (csak) a színek adják a tárgyak szépségét, azért jelentős információt hordoznak az objektumokról magukról. A színek megkönnyítik az életet a forgalomban (piros lámpa), sportban (kedvenc csapat színe), stb.

Azután háromszáz évvel, hogy Newton megadta a színek, a színképzés alapjait az "Opticks" című művében 1704-ben, a színek sok egyéb területen megjelentek mint kutatási terület, a művészetektől kezdve a növények viselkedésén át a kvantummechanikáig.

A színeknek rengeteg modellje létezik, mindazonáltal nincs univerzális színmodell. Minden egyes színrendszer felépíti a saját színmodelljét. A színrendszerek különböző célok érdekében jöttek létre. Ilyenek például a megjelenítés, nyomtatás (RGB, CMY), a televízió, videó átjátszás (YIQ, YUV), a színszabványosítás (XYZ) vagy a színérzékelés (U*V*X, L*u*v). Felmerül a kérdés, mely modell mely képkinyerési technikához alkalmazható? Ehhez először tisztázni kell néhány dolgot. Ilyen például az, hogy az alkalmazott színmodell független-e a felhasznált képalkotó eszköztől (mindegy hogy kamerából, scannerből vagy az Internetről jött a kép). A másik ilyen fontos dolog, hogy mindegyik modell az emberi érzékelés szempontjából uniform, az adott modellekben a távolságfogalom illeszkedik az emberi színhasonlóság fogalomhoz. Fontos, hogy a színrendszerekben használt transzformációknak lineárisoknak kell lenniük. A nem lineáris színtranszformációk a zaj kezelésének szemszögéből instabilitási és hatékonyságromlási problémákhoz vezethetnek. Fontos még, hogy a színmodelleknek a felhasználó által érthetőnek és ösztönösnek kell lennie, valamint robosztusnak és sok értelemben invariánsnak kell lennie.

Alapvetően a színek az elektromágneses spektrum egy részei a 380-tól 780 nm-ig terjedő hullámhosszban. A spektrumnak ez az a része, amelyet az emberi szem érzékel. Ezt gyakran leszorítják a 400-700 nm-es sávra. Ez a látható rész, amely az ibolya színtől a kéken, zöldön, sárgán át a vörösig tart. Ez a folytonos spektrum előállítható egy egyszerű prizmával fénytörés segítségével. A hullámhossz az, amely

fizikai különbséget tesz a spektrum különböző régiói között. A hullámhossz mértékegysége a nanométer (nm). Minden egyes adott hullámhossz egy különálló színhez tartozik. Az általunk látott színek legtöbbje nem egy adott hullámhosszhoz tartozik, hanem azok keverékéből adódik (például a fehér szín mindegyik hullámhosszból azonos mennyiséget tartalmaz).

A leggyakrabban alkalmazott három színtulajdonság a színárnyalat (hue), a telítettség (saturation) és a fényesség (lightness). A színárnyalat a domináns hullámhosszt adja meg, a telítettség az alapszín tisztaságára vonatkozik, a fényesség pedig az intenzitásra (energia) vonatkozik, amivel a fény visszaverődik a tárgyakról. A fényerő (brightness) a fényforrásból kibocsátott energiát (intenzitást) adja meg.

Az alap fény-objektum-megfigyelő hármasunk standardizálása megköveteli, hogy objektív eszközökkel szemléljük a képalkotást. A különféle fényforrásoknak különféle spektrál energia eloszlása van $E(\lambda)$. A tárgyakról visszaverődött $S(\lambda)$ fény mérhető. Így a visszavert fény színe meghatározható $P(\lambda) = E(\lambda)S(\lambda)$. Az emberi szemben viszont három receptor van (trikromatika teória), így az érzékelt színek három szín szignálból tevődnek össze:

$$R = \int_{\lambda} E(\lambda)S(\lambda)f_R(\lambda)d\lambda \tag{4.16}$$

$$G = \int_{\lambda} E(\lambda)S(\lambda)f_G(\lambda)d\lambda \tag{4.17}$$

$$B = \int_{\lambda} E(\lambda)S(\lambda)f_B(\lambda)d\lambda \tag{4.18}$$

ahol az f-ek a színeket illesztő függvényei a szemnek, vagy a kamerának a látható spektrum hullámhosszait tekintve.

A rendszerek nagy többsége kiválaszt egy színmodellt (leggyakoribb az RGB), majd a kiválasztott modellben található változókból felépít egy-egy gyakorisági hisztogramot, és azok távolságait vizsgálja a hasonlóság eldöntése érdekében.

4.4. Textúra illeszkedése

A textúra egy nagyon intuitív fogalom. Minden gyerek tudja, hogy a leopárd pöttyös, de a tigris csíkos. Ebből a példából is látszik, hogy a textúra az valamilyen intenzitások és színek ismétlődése. No persze ez csak egy megközelítés. A textúrát (vizuális textúrát) befolyásolja az anyag fizikai felülete is (érdes, tükröződő, stb.). Érződik az intuitív megközelítés, ugyanis a textúrát pontosan definiálni meglehetősen nehéz (ez abból is látszik, hogy számos eltérő definíciója létezik már az irodalomban).

Az univerzális textúra definíció hiányától eltekintve minden kutatás megegyezik néhány közös pontban. Ezek: (1) egy adott textúrán belül fontos a pixelintenzitások változását követni az egymáshoz közel álló pixeleken, azaz limitált a felbontás alulról ilyen értelemben, (2) másrészt a skálázás is közrejátszik a textúrában, mert más-más skálázás esetén mást lehet ismétlődőnek tekinteni.

Mikor különböztethető meg két textúra, ha ugyanazokkal a fényerő, kontraszt és szín tulajdonságokkal rendelkeznek? Ha beágyazzuk az egyik textúrát a másik textúrába, és a beágyazott vizuálisan elüt a befogadótól, akkor a két textúra nem

tekinthető hasonlónak. Abból a célból, hogy ezt eldönthessük, első- és másodrendű statisztikákat alkalmazhatunk.

Az elsőrendű statisztikák egy véletlenül kiválasztott pont szürkeségi értékének valószínűségét mérik. Ezek a statisztikák a hisztogramból számíthatók. Ezek persze csak különálló pixelektől függnek, nem a szomszédosan előforduló pixelek kölcsönhatásaitól. Az átlagos szürkesége egy képnek egy ilyen elsőrendű statisztika. A másodrendű statisztikák szürkeskála párok valószínűségei, mely pontpárok véletlenszerűen elhelyezett véletlen hosszúságú, véletlen irányú dipólusok végpontjai. A hasonló elsőrendű statisztikával, de eltérő másodrendű statisztikákkal rendelkező textúrák könnyen elkülöníthetők.

Az ilyen jellegű textúrák felismerésének kutatásában a másodrendű statisztikákat gyakran a textonokra alkalmazzák. A texton a textúra építőeleme. Három csoportja létezik, ezek a szín, az elnyújtott pacák, illetve a terminátorok (a pacák végpontjai).

A textúrák illeszkedésének vizsgálata egy meglehetősen kiterjedt, bonyolult terület, mi csak a megemlítés szintjén foglalkozunk vele. A teljesség igénye nélkül megadok néhány megközelítési módot, melyek jól alkalmazhatóak. Ilyenek például, mikor a textúrát Gauss-Markov véletlen mezőnek tételezzük fel [33], vagy például egy másik megközelítés, amikor fa automatákkal próbálunk textúrát generálni, illetve felismerni [18].

Mint említettük, a textúra vizsgálata meglehetősen nehéz feladat, hasonlóan nehéz, ha nem nehezebb a képen látható alakok felismerése, a „foltok" illeszkedésének vizsgálata.

4.5. Alak illeszkedése

Egy objektum egy képen meghatározható teljes területével, vagy kontúrjával. Nagyon fontos, hogy az objektumoknak tisztán szegmentálódniuk kell környezetüktől a jó illesztéshez. Amennyiben adott egy képünk, azon az objektumok jól jellemezhetők momentumok egy halmazával. Egy $O \subset \mathbb{R}^2$ objektum (p, q) momentuma $(m_{p,q})$ az alábbi módon adható meg:

$$m_{p,q} = \int_{(x,y)\in O} x^p y^q dx dy \tag{4.19}$$

vagy $n \times m$ méretű bináris képek esetében

$$\sum_{x=1}^{n} \sum_{y=1}^{m} x^p y^q f(x, y) \tag{4.20}$$

ahol a háttér pixelintenzitása nulla, az előtéré egy. A p, q momentumok egy végtelen sorozata egyértelműen meghatározza a képen levő alakot, és ugyanez fordítva, azaz minden egyes alakhoz egy egyértelmű sorozat tartozik.

Egy másik megközelítési mód esetében az alakot sajátvektorok egy halmazára bontjuk fel (hívhatjuk őket főkomponenseknek is). Az ötlet az, hogy tekintsünk n pontot a kontúron és definiáljunk egy D mátrixot úgy, hogy annak D_{ij} eleme megadja, hogy hogyan vannak kölcsönhatásban az i és j pontok. (Tipikusan például távolságmátrix.) A D e_i sajátvektorai kielégítve a $De_i = \lambda e_i$-t a D módjai, vagy más

szóval sajátalakjai. Két alak illesztéséhez tekintsük a kereső kép e_i sajátvektorait illetve a célkép e'_j sajátvektorait majd alkalmazzunk egy $m(e_i e'_j)$ illesztő függvényt. Az egyszerűség kedvéért legyenek a vektorok azonos hosszúak, és egy rögzített $i = i_0$ esetén határozzuk meg j azon j_0 értékét, melyre az $m(e_{i_0} e'_j)$ minimális. Ha azon i, melyre $m(e_i e'_{j_0})$ minimális megegyezik az i_0 értékkel, akkor a kereső kép i pontja és a célkép j pontja illeszkedik egymásra.

A görbék legközvetlenebb reprezentálási módja a helyfüggvényük megadása. Egy paraméteres görbe általánosságban $A(t) = ((x(t)), (y(t)))$ alakban adható meg. Sok parametrizáció adhatja ugyanazt a görbe alakot, de azok görbe menti deriváltjai különbözni fognak. Egy standard parametrizáció az s ívhossz segítségével adható meg. A poligon görbék (polivonalak) a csúcspontjaik sorozatával reprezentálhatók.

A poligonok gyakran tartalmaznak hamis csúcspontokat, melyek a poligon approximációjánál elhagyódnak. Számos heurisztika alkalmazható ilyen esetekben az approximációhoz. Két általános közelítés az alábbi:

- Adott egy A polivonal és egy k szám. Konstruáljunk egy approximációs A_k polivonalat k csúccsal, mely minimalizálja az approximációs hibát, vagy a $\delta(A, A_k)$ eltérést.

- Adott egy polivonal és egy ϵ hiba. Konstruáljunk egy A_ϵ polivonalat, mely esetében $\delta(A, A_\epsilon) < \epsilon$, és a csúcsok száma minimális.

Nemcsak a helyfüggvény-szerű reprezentációk alkalmazhatók jól az illesztésekhez, hanem más, származtatható függvények is. A kumulatív szög, vagy más néven a forgató függvény $\Theta_A(s)$ egy A poligon esetében megadja az óramutató járásával ellentétes irányban mérve az x tengely és az s ívhosszhoz érintőjének szögét.

Az ezen alapuló illesztések esetében az egyszerűség kedvéért tegyük fel, hogy a két görbének azonos a hossza. Az L_p metrika a függvények terén alkalmazva a Θ_A és Θ_B függvényekre egy különbözőségi mértékként alkalmazható A-ra és B-re, azaz

$$\delta_{A,B} = \left(\int |\Theta_A(s) - \Theta_B(s)|^p ds \right)^{1/p}. \tag{4.21}$$

Ennek minimalizálása egy θ elforgatás mellett nem más, mint a $\delta_{A,B} = \int |\Theta_A(s) - \Theta_B(s) + \theta|^p ds$ minimalizálása. A minimumhely a $\theta = \int \Theta_B(s) ds - \int \Theta_A(s) ds$-nél fordul elő.

A görbék reprezentációjára jól alkalmazható még a lánckód, illetve annak származékai. Ezzel a későbbiekben még foglalkozom.

5. fejezet

Képek indexelése

Maga az a fogalom, hogy indexelés, azt jelenti, hogy a nagymennyiségű, de közvetlen elérésű információt (pontosabban azok kulcsait) valamilyen reláció, elv segítségével berendezzük, majd az így létrejött rendezett állományt a gyorsabb elérés érdekében közvetlenül elérhető részekre csoportosítjuk. Így egy adat keresésekor először a csoportokon végzünk keresést, majd csak aztán a csoportokon belül. Ha a régi könyvtárak papíralapú nyilvántartására gondolunk, ott az elsődleges index az abc-rend volt. Minden betű más fiókban volt, így először a fiókok közt kerestünk, és csak aztán a fiókban. Természetesen a kor előrehaladtával felmerült az adatbázisokban tárolt adatok indexelésének problémája is. Most tekintsünk át egy két alapfogalmat az indexeléssel kapcsolatban [49], [31], [13].

5.1. Általában az indexelésről

Az adatbázisokban tárolt információk visszakeresése érdekében (azok rendezettségét feltételezve) több különféle indexszerkezet hozható létre (ld. [13]). Az alapállományra (táblára) felépített indexet elsődleges indexnek nevezzük. Az indexekre épített újabb indexeket másodlagos, stb. indexeknek nevezzük.

A képek indexelése nagy adatbázisokban egy ismert terület [22], [29], [14]. Mint tudjuk, az indexek szerkezete legtöbbször a rendezett táblázat szerkezetét követi, mely – az elterjedt relációs adatbázisoknak köszönhetően – egy adatbázisbeli tábla sorait indexeli valamilyen attribútuma alapján. Az analógiát tovább alkalmazva – mivel kvázi az index is egy tábla – annak indexelése is megoldható. Ez a technika, a többszintű indexelés technológiája már évtizedek óta ismert az adatbázis-kezelés területén.

Az adatbázisokban tárolt képek indexelése azok tulajdonságvektorain alapszik. A vektorok leggyakrabban valós koordinátájú többdimenziós $\langle x_1, \ldots, x_d \rangle \in \mathbb{R}^d$ alakú vektorok alakjában írhatók fel, ahol d a dimenziók száma. Adott illesztő algoritmusok és konkrét megvalósítások esetén természetesen a d rögzített. Az is belátható, hogy így egy adott illesztés esetén a keresett kép szignatúrája is a d dimenziós valós koordinátájú vektorok teréből adódik.

Az indexelés egyik nagy problémáját az adja, hogy a vektortér elemeinek száma, azaz az elméletileg előforduló kép szignatúrák száma végtelen, míg a gyakorlatban előforduló szignatúrák száma igencsak véges.

5.2. Multimédiás indexelés

Mint azt már láthattuk, a multimédiás anyagok indexelése tehát azok szignatúráján, azaz valamilyen valós vektortérbe leképzett vektoraikon alapszik. Tehát a feladat nem más, mint a vektortér elemeit indexelni valamilyen módszer segítségével. A multimédiás indexelési technikák két nagy csoportba oszthatók. Az első az adatpartíció indexelés, a másik pedig a térpartíció indexelés. Az első az adatok eloszlása alapján osztja fel a teret, a másik pedig előre meghatározott vonalak mentén osztja fel a teret, függetlenül az adatok előfordulásától [1]. Ebbe a csoportba tartozik a négyfa (quadtree) indexelés, mely a térinformatikában gyakran alkalmazott technika. A térpartíciós indexelés nem tud túl hatékony lenni azon esetekben, mikor a képek közel azonosak, azaz az indexeik távolsága nem túl nagy, és egy nagyobb csoportba csoportosulva nem töltik ki az elméleti teret. Ilyen esetek elkerülése érdekében érdemesebb az adatpartíciós indexelést alkalmazni.

Az adatpartíció indexelés az R-fából származtatható [23], mely eredetileg kétdimenziós adatok indexelésére szolgált a GIS-ben. Később az R-fákat kiterjesztették többdimenziós adatokra is. Az SS-fa például egy kiterjesztés [63]. De nagyon sok egyéb kiterjesztés is létezik, melyek mind azon alapulnak, hogy nem minden régiónak van ugyanakkora szerepe a visszakeresésekkor. Erre építkezik a szemantikus indexelés, mellyel a későbbiekben még foglalkozom.

Nem adatbázis szemszögből tekintve a multimédiás anyagok indexelése az alábbi három indexelési sémán alapul (ld. [13]).:

- Osztályozási rendszer. Egy hierarchikus osztályozással osztályozzuk a dokumentumokat. Hátránya az, hogy a multimédiás anyagok közötti kapcsolatok kevésbé kezelhetők.
- Kulcsszóalapú rendszerek. Ekkor minden dokumentumot kulcsszavakkal látunk el, és mint szöveges dokumentumok, úgy indexeljük őket.
- Egyed-attribútum relációk. Ekkor minden dokumentum egy egyed, és az általa azonosított fogalmakkal, attribútumokkal kapcsolatot alkot. Ezeket a kapcsolatokat indexeljük ilyenkor.

Ezek közül az első, az osztályozási rendszer az, amely — ha a képek szemantikáját is tükrözi — alkalmazható adatbázis-kezelő rendszerekben is. A kulcsszó alapú rendszereknél már eltűnik az adatbázis-kezelő rendszerek szigorú értelmében vett indexelés (ld. [39] CREATE INDEX szakasza). A harmadik esetben a kapcsolatok leginkább predikátumok segítségével írhatók le (például Prolog-szerűen), így ez a rendszer sem tud igazán kapcsolódni az adatbázis-kezelő rendszerek meglévő indexelési technikáihoz.

6. fejezet

Keresési interfészek

Az eddigiekben láthattuk, a képekből milyen információkat lehet kinyerni a későbbi visszakeresésekhez. Láthattuk magukat a keresési algoritmusokat, megvizsgáltuk a keresést elősegítő indexelési technikákat, már csak annyi hiányzik ahhoz, hogy minden fontosabb területet érintsünk, hogy megvizsgáljuk a képadatbázisok legkülsőbb rétegét, az interfész réteget, ahol a felhasználó felteheti kérdését az adott képadatbázis felé.

A keresési interfészeket két nagyobb csoportba lehet osztályozni. Az első csoportba a konkrét, vizuális interfészt kell érteni, mely megkönnyíti a felhasználónak az adatbázisból való lekérdezés megadását, illetve megjeleníti a visszaadott eredményeket. A másik csoportba az úgynevezett kérdés formalizációk tartoznak, melyek a felhasználó által feltett kérdések matematikai eszközökkel történő formalizációját szolgálják. Úgy is lehet mondani, hogy az első csoportba a magas szintű, míg a második csoportba az alacsonyabb szintű interfészek tartoznak.

6.1. Query By Example

A magas szintű interfészek egyik fajtája a minta alapján történő lekérdezés [45]. Természetesen a legtöbb keresés mindig feltételez valamilyen mintát, de nem mindegy, az a minta hogy van megadva. A minta alapján történő lekérdezések három nagyobb csoportba sorolhatók. Ezek a hasonlókép-alapú, a vázlat alapú és az ikon alapú lekérdezések. Mindhárom lekérdezés esetén nulladik lépésben szükséges egy keresési alap (kereső kép) elkészítése.

A hasonlókép-alapú lekérdezések esetében a felhasználónak össze kell valamilyen módon állítania egy olyan képet, melyhez hasonlót keres az adatbázisban. Megfelelő eszköz nélkül ez elég nehézkes feladatnak tűnik. Ez a megoldás azokban az esetekben használható, ha az adott kép már rendelkezésünkre áll, csak a csatolt egyéb információkra vagyunk kíváncsiak, illetve akkor, amikor a kép rendelkezésre áll, de nem megfelelő minőségben. Ide szinte bármilyen alapvető képkeresési technika alkalmazható.

A vázlat alapú lekérdezések esetében szintén össze kell állítani egy kiindulási képet, de erre a rendszer különféle rajzolási segítséget nyújt (megrajzolunk, felvázolunk egy képet) [11] [35]. A rajzolási folyamat támogatásától eltekintve ez a megközelítési mód

nem sokban különbözik az előbbi megközelítési módtól. (Leginkább abban, hogy a kézi rajz sajátosságait kihasználó algoritmusok is használhatók.)

Az ikon alapú módszer nagyobb mértékben különbözik az előbbi két módszertől, bár itt is a felhasználónak kell összeállítani a kiindulási képet, de nem rajzeszközökkel, hanem különféle előre definiált ikonok segítségével [31]. Ekkor tehát különféle speciális ikonok által jelölt etalonokat feltételezve a háttérben a felhasználó megadja, a kép mely részén milyen ikonhoz hasonló elemnek kell lennie az eredményképen. Ekkor az a fontos, hogy az ikonok által jelölt illesztendő elemek milyen térbeli viszonyban helyezkednek el a képen [5]. Az ikon alapú kereséseknél is az a kellemes, hogy több különféle speciális algoritmus is alkalmazható.

A QBE interfészekhez nagyban hasonlít Yang hypertext megoldása [64], mely viszont már átmenetet képez a lekérdezőnyelvek felé.

6.2. Lekérdezőnyelvek

Az alacsonyszintű interfészek nemcsak a fenti interfészek által leírt kérdések matematikai eszközökkel való formalizálását, hanem önálló kérdésformalizációt is ellátnak. Igazi feladatuk a hagyományos adatbázis-kezelő rendszerekből kiindulva, a lekérdezőnyelv multimédiás információk lekérdezésére alkalmassá tétele, kibővítése. Számos, az SQL nyelv kibővítésével létrejött képadatbázis lekérdezőnyelv jött ezáltal létre. De a matematikailag pontosabb kalkulusok és algebrák is megtalálhatók a lekérdezőnyelvek sorában, és nemcsak mint absztrakt multimédia lekérdezőnyelvek. Az objektumrelációs modellről szóló könyvében Date [9] is leír egy OR lekérdezőnyelvet, mely alkalmas multimédiás adatok kezelésére.

A heterogén multimédiás rendszerek lekérdező nyelveinek fejlesztése még egy elég új terület. Az ilyen nyelvek legtöbbször az SQL kiterjesztéseire épülnek rá (például PSQL-re [47] és Spatial SQL-re [12]). Az SQL-szerű lekérdező nyelvek mellett megjelentek a videó lekérdező nyelvek is, melyek a tartalomalapú kinyerésekre és a temporális megszorításokra fókuszálnak a szöveges reprezentációk mellett. Az OL/G nyelv [7] már geometriai adatbázisokhoz készült, szöveges és geometriai adatokat támogat. A multimédia objektum lekérdező nyelv (MOQL [32]) kiterjeszti a az OQL-t multimédiás adatokra. A listát lehetne még tovább folytatni, hisz számos fejlesztés indult már el e téren.

Mivel a fent említett megközelítések mind valami fontos kérdést helyeznek a középpontba, hiányzik belőlük a lekérdezések egységes kezelése, amely mind a térinformatikai, mind a temporális megszorításokkal foglalkozna nemcsak az adatbázisban tárolt adatok, hanem az élő adatforrások tekintetében is. Mivel a háttérben elfekvő adatbázisok meglehetősen komplexek, a felhasználónak is bonyolult lekérdezéseket kell írnia, hogy visszanyerje az integrált multimédia adatokat.

A [31]-ben Shi-Kuo Chang és Erland Jungert leírnak egy olyan nyelvet, mely nem speciális területre koncentrál, hanem kezeli a több adatforrással rendelkező adatbázisokból történő információkinyerést. Az erőssége abban rejlik, hogy minden lekérdezés egy egyszerű operátoron, a σ operátoron alapszik. A koncepció nagyon egyszerű, és segítségével könnyen készíthetünk SQL-szerű lekérdező nyelvet. A σ-query nyelv nagyon hasznos a teoretikus megközelítésekben, míg az SQL-szerű nyelvek a könnyű implementálhatóságukról ismeretesek.

Gyakran alkalmaznak különféle logikai nyelveket a kérdések formalizációjához. Itt nemcsak a kalkulus-szerű megoldásokra kell gondolni, hanem például külön definiált predikátumokra, stb. A fuzzy logika segítségével történő speciális kérdésformalizációról még lesz szó a későbbiekben.

Saját eredmények

7. fejezet

Alapelvek

Az előző részből megtudhattuk, hogy a képadatbázisok képek tárolására és visszakeresésére alkalmas rendszerek. A képek visszakeresése legtöbbször egy kereső kép alapján történik, amelyhez hasonlót vagy vele egyezőt szeretnénk az adatbázisból kinyerni. A kinyerés legtöbbször a képet leíró tulajdonságokon, tulajdonságvektorokon alapszik. A vektorok közti hasonlóság eldöntése egy távolságfogalom bevezetésével oldható meg. Az összehasonlított eredményekből valamilyen módon egy hasonlósági mérőszámot képezünk (például súlyozott összeg). A mérőszám alkalmas igaz/hamis válasz megadására is valamilyen küszöb megadásával (küszöbtől nagyobb vagy kisebb eredmény).

A fenti keresési módszereken túlmenően bepillantást nyerhettünk a képek indexelése és a kérdések formalizációja tekintetében. Ebben a részben a fenti eredményekhez kapcsolódó saját eredményeimet tekintjük át. Az indexelés tekintetében a szemantikus indexelés területéről, a keresési módszerek tekintetében az alakfelismerés területéről, valamint az összetett lekérdezések és a kérdésformalizáció tekintetében a Cut And Or Not módszer eredményeibe kaphatunk bepillantást.

Ebben a fejezetben pedig néhány fontos kiegészítő alapelvet ismertetek, melyek meghatározták kutatásaim irányát.

7.1. Képi adatbázisok

Mint már korábban említettem, léteznek speciális, képek tárolására szolgáló adatbázisszerű megoldások, de én a továbbiakban csak olyan általános adatbázis-kezelőkben megvalósítható megoldásokkal foglalkozom, ahol felmerül nagyobb képadatbázisok létrehozásának kérdése (például Oracle9i ORDBMS). Általánosan elmondható, hogy egy adatbázis-kezelő rendszer jónak tekinthető, mikor az adatbázis a képek BLOB-kénti tárolásán felül már többet is nyújt. Például megoldja a képek méretezését, bináris le- és feltöltését stb. A képi adatbázisokkal és lekérdezésekkel már több aspektusból is találkozhattunk [4] [19] [5], sőt, irányelveket kaptunk az illesztés stratégiájára is [10], de az igények növekedése új irányelveket követel. Az igényekkel egyetemben az adatbázis-kezelő rendszerek képességei is növekednek, ezáltal lehetővé téve a fejlődést, és a nyitást az új, tisztán OO szemléletű adatbázis-kezelő rendszerek felé.

Első alapelvként tehát azt kell szem előtt tartani, hogy kutatásaim objektumrelációs adatbázis-kezelő rendszerek kép-visszakeresési technikákkal történő felruházását célozzák meg.

7.2. Illesztési technikák

Most tekintsük át, milyen lekérdezéseket különböztethetünk meg a továbbiak folyamán.

A képi adatbázisokból történő képkinyerés alapsémája a következő: Adott egy adatbázis, mely képeket tartalmaz. Adott nekünk egy kérdező kép, kereső kép (query image), egy minta, és azt szeretnénk tudni, található-e az adatbázisban olyan kép, mely legjobban hasonlít a minta képünkre. Mivel azon képek száma, melyek feltehetően identikusak, megegyezők a mintánkkal meglehetősen kicsi, így valamilyen

$$\delta : \mathrm{Obj} \times \mathrm{Obj} \to \mathbb{R}_0^+ \tag{7.1}$$

távolságfogalmat kell bevezetni (ld. (3.6)), ahol Obj az adatbázisban tárolható képobjektumokat jelöli, és azon képeket leválogatni az adatbázisból, melyek távolsága a query image-től minimális ($\mathbb{R}_0^+$ a nemnegatív valós számokat jelöli).

Ezek alapján a következő illeszkedéseket szokás megkülönböztetni [2]:

- Identikusság, azaz totális egzakt illeszkedés, amikor $\delta(\mathrm{obj}_1, \mathrm{obj}_2) = 0$.
- ϵ hasonlóság, amikor $\delta(\mathrm{obj}_1, \mathrm{obj}_2) < \epsilon$, ahol $\epsilon \in \mathbb{R}^+$.
- NN-hasonlóság, avagy legközelebbi szomszéd, ha $\forall \mathrm{obj} \in \mathrm{DB}$, $\mathrm{obj} \neq \mathrm{obj}_2$, $\delta(\mathrm{obj}_1, \mathrm{obj}_2) \leq \delta(\mathrm{obj}_1, \mathrm{obj})$.

A DB az adatbázist jelöli.

Ezek alapján tehát az adatbázisban az olyan keresések végezhetők el egy adott obj_q kereső képpel, mint például az identikus keresés, mikor az alábbi objektumokat keressük

$$\{\mathrm{obj} \in \mathrm{DB} | \delta(\mathrm{obj}, \mathrm{obj}_q) = 0\}, \tag{7.2}$$

vagy az ϵ keresés, ahol

$$\{\mathrm{obj} \in \mathrm{DB} | \delta(\mathrm{obj}, \mathrm{obj}_q) < \epsilon\} \tag{7.3}$$

a keresés által eredményül adott képek halmaza, illetve az NN keresés, ahol az alábbi képeket keressük

$$\{\mathrm{obj} \in \mathrm{DB} | \forall \mathrm{obj}_p \in \mathrm{DB}, \mathrm{obj} \neq \mathrm{obj}_p, \delta(\mathrm{obj}, \mathrm{obj}_q) \leq \delta(\mathrm{obj}_p, \mathrm{obj}_q)\}. \tag{7.4}$$

A következő lépés az, hogy olyan távolságmérést — illesztési stratégiákat — keressünk, melyek a képeknek valamilyen kinyerhető tulajdonságain alapszik [5] (ld. (3.1)). Ily módon kinyerjük a minta tulajdonságvektorát, majd az adatbázisban található képek tulajdonságvektorait, és ezen vektorok távolságait vizsgáljuk a továbbiakban. Alkalmazhatjuk akár speciálisan Eakins [10] háromszintű modelljét is, a fontos, hogy valamilyen tulajdonságvektorokat kell kinyernünk a képekből, mint azt már a korábbi fejezetekben láttuk.

Ilyenkor tehát be kell vezetni egy

$$\mathcal{F} : \mathrm{Obj} \to \mathbb{R}^d \tag{7.5}$$

tulajdonságvektor kinyerést (ld. (3.1)), ahol $\mathbb{R}^d$ a d-dimenziós vektorok halmazát jelöli. Ekkor például a δ távolság az alábbi módon alakul:

$$\delta(\mathrm{obj}_1, \mathrm{obj}_2) = \delta_{vectors}(\mathcal{F}(\mathrm{obj}_1), \mathcal{F}(\mathrm{obj}_2)) \tag{7.6}$$

ahol $\delta_{vectors}$ a vektorokon értelmezett (akár Euklideszi) távolság (ld. (3.6)).

Ezzel a megközelítéssel tehát elkerüljük a képek többszöri végigolvasását, főleg ha feltesszük, a tulajdonságvektorok szintén letárolhatók.

A keresés gyorsítását szolgáló indexek felépítéséhez a képeket valamilyen rendezési elv szerint be kellene rendezni. Ez egy elég bonyolult feladat. Legtöbbször nem magukat a képeket rendezik, hanem azok tulajdonságvektorait, ugyanis azok valós vektor formában már könnyebben rendezhetők. Az így felépített indexekre esetleg további másodlagos stb. indexek építhetők. Így sikeresen felindexeltük a képeket vektoraik alapján. Léteznek indexelési technikák, melyek figyelembe veszik a belső szerkezetet is [6]. Számomra is fontos alapelv, hogy az indexelést az is meghatározhatja, mi van a képen. Mivel a képen lévő objektumok felismerésében az emberi tényező, az emberi szemantika nem elhanyagolható, ezek csoportját szemantikus indexelési technikáknak nevezzük. Így az is fontos alapelv volt számomra, hogy szemantikus indexelési technikát érdemes használni a meglévő tulajdonságvektor indexelések kiegészítőjeként.

7.3. Műveletek adatbázis szinten

Amennyiben a műveleteket adatbázisszinten implementáljuk, úgy elkerülhetjük a képek felesleges letöltését az illesztésekhez, hiszen a sok kép letöltése helyett csak egy képet, az illesztő képet kell feltölteni az adatbázisba. (Bizonyos esetekben, amennyiben a kinyert vektorok nem nagyobb méretűek, mint maga a kép, vagy nincs szükség a képre, lehetséges, hogy csak az illesztő képből kinyert tulajdonságvektort kell feltölteni.) Ily módon nagyban megnő az illesztés, az adatbázisban történő tartalomalapú képkinyerés sebessége. Ugyanilyen előnyként említhető az is, hogy ha több helyről szeretnénk az adatbázist elérni (mint központi adatszolgáltató szervert), úgy az illesztések determinisztikussá válnak, hiszen garantált, hogy az illesztő metódusok minden kliens gépről ugyanazok, s nem esetlegesen eltérő módon helyileg implementált változatok. Ez a központi megoldás azért is jó, mert az algoritmusok fejlődése, frissítése egy helyen, központilag lehetséges. A mai adatbázis-kezelő rendszerek többsége rendelkezik olyan nyelvvel, mely az ilyen feladatokra jól alkalmazható.

Ez is egy nagyon fontos alapelv kutatásaimban, mert a létező adatbázis-kezelő rendszerek csak akkor bővíthetők ki képadatbázis kezeléséhez használható eszközökkel, ha maguk az eszközök jól implementálhatók az adatbázis-kezelő rendszer által támogatott nyelven illetve nyelveken. Én fejlesztéseimhez a Oracle PL/SQL [38] nyelvét használtam.

7.4. Az illesztési modell

Van még egy fontos alapelv, mely az irodalomban használt módszerek hiányosságaiból egyértelműen következik. Az, hogy gyakran csak a kép bizonyos részein van szignifikáns információ. Így biztosítani kell a részképeken alapuló illesztéseket, illetve biztosítani azt, hogy összetett kérdéseket lehessen ezáltal feltenni. Ez alatt azt értem, hogy a kérdést feltevő személy azért kérdez, mert lehet, hogy nem emlékszik pontosan. Lehet, hogy keveri a képeket a fejében. Úgy emlékszik például, hogy vagy volt piros folt a képen, vagy nem, de ha volt, akkor biztos nem volt kék a háttér. Az ilyen jellegű összetett kérdéseket is lehetővé kell tenni.

Amennyiben követjük a [57] [58]-ban és az itt említett megközelítési módokat, mindent összevetve az alábbi alapelvek gyűjthetők össze:

- Objektumrelációs illetve objektumorientált megközelítést kell alkalmazni.
- A képekre szemantikus indexelést kell alkalmazni.
- A műveleteket adatbázis szinten kell implementálni és alkalmazni, ezáltal elérve a mobilitást a rendszerekben, biztosítva a több helyről történő egységes elérhetőséget.
- Biztosítani kell a részképen alapuló összetett lekérdezéseket.

Ezek azok az alapelvek, melyek a kutatásaim alapjait képezik, részben ezekkel az elvekkel foglalkozom a [62]-ben is. Minden általam véghezvitt fejlesztés megfelel ezeknek az alapelveknek. Az alapelvek természetesen nemcsak a szigorú értelemben vett képadatbázisok esetén kamatoztathatók. Amennyiben például egy képfeldolgozó rendszert támogató adatbázisunk van (mely a képek mellett mondjuk a képfeldolgozó operátorokat is tartalmazza) szintén alkalmazhatók a fenti alapelvek, hiszen a különbség csak annyi, hogy az operátorok nem a kinyerést, visszakeresést szolgálják, hanem a feldolgozást. Ilyen alkalmazott rendszerre láthatunk példát a [24]-ben.

8. fejezet

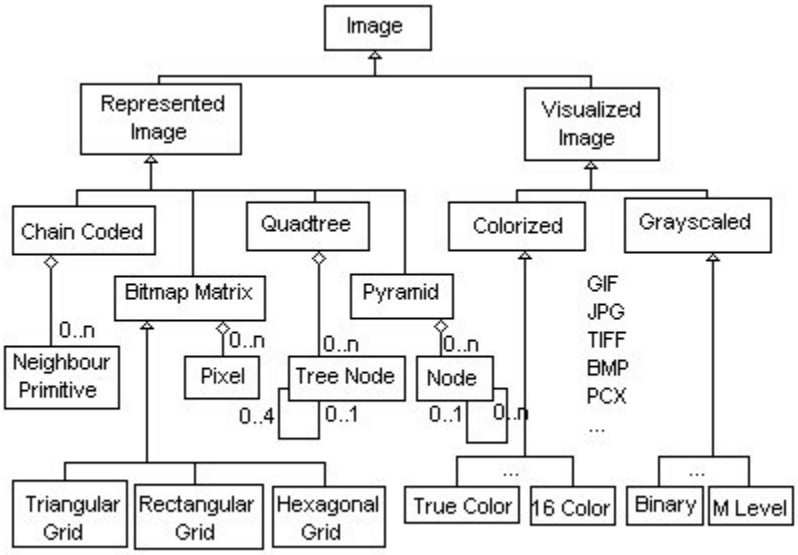
Szemantikus képindexelés

Ebben a fejezetben bemutatok egy objektumorientált alapú technikát az adatbázisokban tárolt képobjektumok indexelését, illetve tipizált kérdésekkel történő visszanyerését illetően.

8.1. Az OO technika használata

A képek, a képek alkotóelemei mind objektumok. Ez most első megközelítésben annyit jelent, hogy egy képobjektum tartalmazza a saját tulajdonságait, és a saját kezelő metódusait is. Ez a bezárás. Ha megfigyeljük, ez a megközelítési mód igenis létjogosult, ugyanis másként kell illeszteni egy négyfa-reprezentált képet mint egy lánckódolt poligont, vagy egy bináris rendszámtáblát és egy truecolor tájképet. Viszont ha sok különálló objektumot alkalmazunk, akkor fenn áll a veszélye annak, hogy az esetleges reprezentációváltások esetén inkompatibilitási problémák léphetnek fel. A megoldás az lenne, ha a képeknek lenne valamilyen közös interfész felületük, vagy valamilyen szabványos reprezentációjuk. Természetesen erre a problémára is van megoldás. A neve öröklődés. Minden lánckódolt, négyfa-reprezentált stb. kép egy Kép. Azaz vannak közös műveleteik, amiket örökölnek és természetesen önspecifikusan megváltoztathatnak. Tehát az ős szintjén és a modellben mindenütt például a tulajdonságvektor ugyanazt jelenti — egy olyan vektor, szignatúra, mely jellegzetességeket tárol —, de a konkrét alosztályok példányai el tudják magukról dönteni, hogy milyen vektort, milyen konkrét jellegzetességeket kell kinyerni magukból az illesztésekhez. Természetesen a kinyerés után ezek példányszinten tárolódnak is. Ugyanilyen módon, a modellben örökölt viselkedésmód az illesztés metódusa is. Itt is az alosztályok szintjén konkretizáljuk, milyen illesztési eljárást alkalmazunk. Tehát — mondhatni — az ősosztály szerkezete adja számunkra az interfészt, az egységes megjelenést.

A 8.1. ábra egy lehetséges objektummodellt mutat be [61]. Természetesen a modellezést más oldalról is meg lehet közelíteni. Maga az objektummodell öröklődési fája pedig nem más, mint a képek szerkezeti fa indexelése, ugyanis a levélelemek speciális képekként már egyértelműen indexelhetők. Ha minden egyes táblában tárolt levélelemre felépítünk egy indexet, akkor egy speciális többszintű asszociációs indexelési technikához jutunk. Vizsgáljuk ezt meg egy kicsit közelebbről.



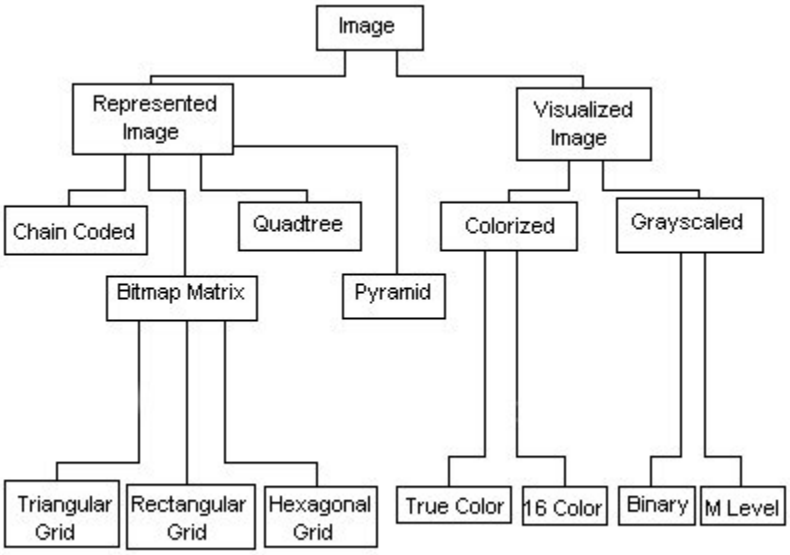
8.1. ábra. Az objektummodell.

8.2. Az objektumok indexelése

A technika az objektumok hierarchiában betöltött „szemantikájuk" szerinti indexelésén alapszik. Az indexelendő tulajdonságvektorok alakja a leggyakrabban egy többdimenziós vektor. Az indexelés lelke ezen vektorok valamilyen rendezési elv alapján történő rendezése a többdimenziós térben (ld. Indexelés szakasz). Mivel a vektorok mérete is nagy elemszámú kép esetén már jelentőssé válik, így szükségessé válik a vektorok többdimenziós terének valamilyen elven történő felosztása, és ezen klasszifikáció segítségével többszintű indexszerkezet felépítése.

Az első, és legfontosabb dolog, hogy az adatbázisban tárolni kívánt képek típusait meg kell határozni. Ez a tipizálás valójában egy hierarchikus osztályozás, mely a képek nem mérhető, ún. asszociatív tulajdonságain alapszik. Ez az osztályozás az adatbázis-tervezők feladata. Lehet az általunk már említett képreprezentáció alapú osztályozás is, vagy — ha az nem lehetséges — akár bonyolultabb, a kép által ábrázolt motívum alapján történő osztályozás is. A fontos, hogy ennek az adott képhez tartozó adott típusnak ismertnek kell lennie az adatbázisba történő beszúráskor, illetve a visszakereséskor. Mint említettük, ez a klasszifikáció valójában egy hierarchikus osztályozás, azaz az adatbázisban tárolható összes létező kép típusára fel kell készülni, és azok típusait egy öröklődési fában reprezentálni.

Tekintsük például a már többször említett objektumdiagramunknak egy egyszerűsített változatát (8.2. ábra). Tekintsük ezt a diagramot úgy, mint egy típusfát. Megjegyezzük, hogy esetünkben a típusfa egy olyan osztályhierarchia, melyben az osztályok között csak egyszeres öröklődési kapcsolat létezik. Amint látszik, minden képre felkészültünk, hiszen a gyökér szinten (Image) minden képobjektum elhelyezhető. A lentebbi szinteken egyre jobban pontosítjuk, milyen képeket tárolunk. Kü-



8.2. ábra. A típusfa.

lönbséget teszünk például megjelenítésre tervezett (például weblapokon) képek (Visulaized Image) illetve további feldolgozásra váró képek között (Represented Image). Egyértelmű például, hogyha egy lánckódolt foltot ábrázoló képet keresek az adatbázisban, akkor azt nem a megjelenítésre szánt design jpg-k között kell keresnem. Most vizsgáljuk meg egy picit jobban, milyen tulajdonságokat kell kielégíteni egy ilyen típusfát reprezentáló objektumhierarchiának.

Először is, legyenek adottak az adatbázisban elméletileg maximálisan előforduló $\text{obj}_i \in \text{Obj}$ képek. Létre kell hozni egy osztályhierarchiát, és minden egyes képet a hierarchiában levő osztályokhoz hozzárendelni. Legyen N a $\boldsymbol{C}$ hierarchiában levő osztályok száma.

Jelöljük a $\boldsymbol{C}$ osztályhierarchia osztályait C_j-vel, ahol $j = 1, \ldots, N$. Az osztályozás eredményeként minden obj-hoz, $\text{obj} \in \text{DB}$, létezik egy C_j, $j \in 1, \ldots, N$ osztály úgy, hogy obj egy példánya (eleme) a C_j osztálynak. Ezt jelöljük $\text{obj} \in C_j$-vel, azaz

$$\forall \text{obj}, \text{obj} \in \text{DB}, \exists C_j, j \in 1, \ldots, N, \text{obj} \in C_j. \tag{8.1}$$

Ha megfigyeljük, nem az Obj összes eleméhez határozzuk meg az osztályokat, hanemcsak azokhoz, amelyek bekerülnek az adatbázisba. Ez csak elméleti megfontolás. Ha lenne olyan adatbázis, amelyben a Obj minden eleme elhelyezhető, akkor azonnal mindegyikhez hozzárendeltünk egy osztályt, tehát semmivel sem csorbul a fenti megfontolással az elmélet. Azt viszont már most meg kell említenünk, hogy a képadatbázisok gyakorlati megvalósításaiból kifolyólag a kereső képet is adatbázisbeli tárolt képnek tekintjük, tehát annak osztálya is meghatározható.

A jobb érthetőség kedvéért vezessünk be néhány jelölést. Egy C_2 osztály közvetlen leszármazottja (gyereke) egy C_1 osztálynak, ha a típusfában közvetlen vonal köti őket össze (közvetlen öröklődés). Ezt jelölje $C_1 \rightarrow C_2$. Ebben az esetben a C_1 osztály

a közvetlen szülője a C_2 osztálynak. Azt is mondhatjuk, a C_p osztály alacsonyabb szinten van a fában, mint C_q, ha $\exists C_1, \ldots, C_n$, ahol $C_p = C_1$, $C_q = C_n$ és $C_i \rightarrow C_{i+1}$, $i = 1, \ldots, n-1$. Ezt jelöljük $C_p < C_q$ alakban. Ez egy szimmetrikus reláció, azaz használhatjuk a $C_q > C_p$ jelölést is. Azt a terminológiát szokás használni, hogy ha C_p alacsonyabb szinten van a fában, mint C_q, akkor C_q magasabb szinten található, mint C_p. (Ha $C_1 < C_2$, akkor C_1 egy (közvetlen vagy közvetett) szülője C_2-nek, és C_2 egy (közvetlen vagy közvetett) leszármazottja (gyereke) C_1-nek.)

$\boldsymbol{C}$-nek ki kell elégítenie az alábbiakat:

- $\boldsymbol{C}$-nek csak egy gyökere lehet, azaz $\exists C_k \forall C_i$, $i, k \in \{1, \ldots, N\}$, $i \neq k$, $C_k < C_i$, és $\nexists C_j$, $C_j < C_k$, $j \in \{1, \ldots, N\}$. Jele C_0.
- Minden osztálynak csak egy közvetlen őse lehet, kivéve a gyökeret, azaz $\forall C_i \exists C_k$, $i, k \in \{1, \ldots, N\}$, $i \neq k$, $C_i \neq C_0$, $C_k \rightarrow C_i$, és $\forall C_l$, $C_l \rightarrow C_i$, $C_l = C_k$.

Ezekkel a feltételekkel a $\boldsymbol{C}$ egy általános fa.

8.3. Tipizált keresések

Ismeretes, hogy bármely kereső rendszer esetében az általános keresésektől sokkal hatékonyabbak, célravezetőbbek azok a keresések, mikor a keresési kritériumok némelyikét, vagy mindegyikét konkrétan meg tudjuk határozni. Egy az általunk elkészített típusfa típusaiba tartozó képeket tároló adatbázisban ilyen specializáció az, hogy meghatározzuk, a fa mely csomópontjához rendelt osztályba tartozik a képünk. Mivel az öröklődési fa ISA kapcsolatokból áll, ha a gyökér szintre helyezzük a képünket, akkor az bármelyik kép lehet az adatbázisból, viszont ahogy haladunk a fában a levélelemek felé, úgy pontosítjuk, mely képek tartoznak bele egy adott keresésbe, s melyek nem. (Ez végül is abból az OO szemléletből adódik, hogy egy gyermek mindig szerepelhet szülője helyett, hiszen örökli annak tulajdonságait.) Így tehát bevezetünk egy olyan keresést, melynél a kereső kép típusa (osztálya) meghatározza a keresendő képek osztályait. Nevezzük ezt tipizált keresésnek.

Így az alábbi tipizált kereséseket különböztethetjük meg:

- Az identikus tipizált keresést, ahol a keresett objektumok halmaza

$$\{\text{obj} \in \bigcup_{C_i < C_j} C_j | \text{obj}_q \in C_i, \delta(\text{obj}, \text{obj}_q) = 0\}, \tag{8.2}$$

- Az ϵ tipizált keresést, ahol a keresett objektumok halmaza

$$\{\text{obj} \in \bigcup_{C_i < C_j} C_j | \text{obj}_q \in C_i, \delta(\text{obj}, \text{obj}_q) < \epsilon\}, \tag{8.3}$$

- Az NN (nearest neighbour, legközelebbi szomszéd) tipizált keresést, ahol a keresett objektumok halmaza

$$\{\text{obj} \in \bigcup_{C_i < C_j} C_j | \\ \text{obj}_q \in C_i, \forall \text{obj}_p \in \text{DB}, \text{obj} \neq \text{obj}_p, \delta(\text{obj}, \text{obj}_q) \leq \delta(\text{obj}_p, \text{obj}_q)\}. \tag{8.4}$$

Belátható, hogy a tipizált keresésből kifolyólag az indexszerkezetet is érdemes a tipizált képekre korlátozni. Tehát minden egyes C_i, $i = 1, \ldots, N$ osztályhoz rendeljünk hozzá egy már korábbról ismert indexelési technikát, melyek akár különbözőek lehetnek. Ami fontos, hogy így elméletileg van N darab indexünk $(I_1 \ldots, I_N)$. Ami viszont fontos, hogy ezek az indexek nemcsak a hozzájuk tartozó C_i osztály elemeit indexelik, hanem az öröklődési fából adódóan (ISA kapcsolatok) a leszármazott osztályok elemeit is. Azaz C_0 szintjén a teljes képállományunkat indexeljük, s ahogy haladunk a levélelemek felé, úgy specializálódnak egyre a képek és csökken az adott indexekben azok száma. Tehát egy I_i, $i = 1, \ldots, N$ index azokat a képeket indexeli, melyekre

$$\forall \text{obj}, \text{obj} \in \bigcup_{C_i < C_j} C_j. \tag{8.5}$$

Amennyiben a tipizált keresés által visszaadott halmaz üres halmaz, úgy nagyon egyszerűen — akár lépésenként — általánosítható a keresés, hiszen a C_i osztálytól a fában az út egyértelműen meghatározható C_0-ig, ahol az I_0 a teljes képadatbázist indexeli. Ha a C_0 szintjén is üres halmazt kapunk eredményül, akkor az adott identikussági, NN vagy ϵ tulajdonsággal a kép nem található az adatbázisban.

8.4. Nyitott kérdések

Nos, maga a hierarchia használata indexelésre már ismerős lehet mindenkinek az információkinyerés elméleteiből. Viszont az az ötlet, hogy a szemantikus hierarchia, mint másodlagos index több különféle elsődleges indexet fog össze, és az objektumok öröklődéséből adódóan több keresési algoritmust tesz elérhetővé (művelet polimorfizmus) már újdonságnak számít. Épp ezért ebben a témakörben is felmerültek olyan kérdések, melyek megválaszolására még nem került sor. Ezek például, hogy létezik-e ideális objektummodell. Lehet-e úgy modellezni szerkezetük alapján a képeket OO módon, hogy a modell ne legyen túl bonyolult, mégis a legjobban tükrözze a képobjektumok hierarchiáját. Ha a modell túl bonyolult, emberidegenné válhat, s ezáltal elveszíti a végfelhasználók általi alkalmazhatóságát. Hasonlóan nyitott probléma, hogy hogyan lehet kiterjeszteni a fenti technikát olyan esetekre, amikor az öröklődésben a többszörös öröklődést is megengedjük, s az öröklődési fa ezáltal nem fa, hanem gráf.

9. fejezet

Összetett mintaillesztési stratégiák

Mint már korábban említettem, napjaink képi adatbázisaiban a képek visszakeresése közben felhasznált illesztési algoritmusok és stratégiák nem teszik igazán lehetővé az összetett illesztési kérdések alkalmazását. Most bemutatok egy olyan lehetőséget, mely segítségével a már meglévő illesztések kiegészíthetők, és lehetővé válik összetett mintaillesztési stratégiák alkalmazása képi adatbázisokban. A módszer a logikai formulák, illetve a fuzzy logika használatán, és néhány speciális műveleten alapszik. Ebben a fejezetben megadom a módszer teljes leírását és a szükséges formalizmust.

9.1. A naiv Cut-And-Or-Not megközelítési mód

A legtöbb képadatbázisból történő lekérdezés esetén maga a kérdés általában úgy hangzik, hogy „keresek egy olyan képet, ami hasonlít egy adott másik képhez”. Ezt a legtöbb illesztési modell úgy kezeli, hogy veszi az illesztő képet, majd sorra veszi az illesztendő képeket és temporálisan minden egyes illesztéskor azonos méretűekre hozza őket [40]. A tulajdonságvektor kinyerése és az illesztés többi része csak ezek után következhet. Persze a felhasználó oldaláról felmerülhetnek olyan igények, hogy a képeknek csak egy részét illesszük. Sőt, a kérdéseket akár kombinálhatják is egymással. Például „keresünk egy olyan képet, melynek a felső felében nem kék a domináns szín (azaz valószínűleg nem tájkép), de a jobb alsó negyedében olyan részlet van, mint ezen a másik képen, vagy ha mégis kék a domináns szín a felső felében, akkor a bal alsó negyedben legyen a kereső képhez hasonló motívum”. Beláthatjuk, ez már egy elég összetett kérdésnek feleltethető meg. Az ilyen irányú kérdésekre próbál megoldást találni az általam fejlesztett Cut-And-Or-Not megközelítés.

A Cut-And-Or-Not megközelítés mindössze annyit mond, hogy mind a kereső-, mind az illesztendő képek részeit kivághatjuk, és az így nyert kis képekkel történő hagyományos illesztési eredményeket (illeszkedik=true, nem illeszkedik=false) logikai összekötőjelek alkalmazásával formulába gyűjthetjük. Természetesen, amennyiben a részképek illesztési eredményei nem igaz/hamis válaszok, hanem mondjuk számszerűleg megfogalmazható (például százalékos illeszkedési arány, stb.) eredmények, akkor

az ilyen eseteket a logikai összekötőknek, műveleteknek is le kell tudni kezelni. Ilyen esetekben érdemes a fuzzy logikákat segítségül hívni.

A Cut-And-Or-Not megközelítés igaz/hamis illesztési válaszok esetén az alábbi módon formalizálható. (Hasonló módon lehet formalizálni a fuzzy logikán alapuló megközelítést is, abban az esetben az utolsó lépést kell a fuzzy logikának megfelelően megváltoztatni.)

- Adottak f és g digitális képek.
- Képezzük ezekből a képekből téglalap alakú vágások végrehajtásával a kívánt $f_1, \dots f_n$ és $g_1, \dots g_m$ részképeket.
- Végezzük el a megfelelő részképekkel az illesztéseket, így előáll p számú részkép illesztése esetén $Q_1, \dots Q_p$ logikai igaz/hamis érték, illesztés.
- Készítsünk ezekből a Q értékekből logikai formulát és értékeljük ki. Amennyiben a kiértékelt formulánk igaz, akkor g a keresett kép, ellenkező esetben nem.

Amennyiben fuzzy logikákat alkalmazunk, úgy az utóbbi lépés módosítandó. Ez utóbbi lépésben felmerül a kérdés, mit nevezünk formulának. Erre az alábbi induktív definíció ad választ:

- A $\top$ (igaz) és a $\bot$ (hamis) formulák.
- Minden Q illesztés formula.
- Ha Q_1, Q_2 és Q formulák, akkor a
  - $Q_1 \vee Q_2$ (vagy)
  - $Q_1 \wedge Q_2$ (és)
  - $\neg Q$ (nem) is formulák.

A logikai összekötők (vagy, és, nem) értelmezése egyértelmű, igazságtáblázatuk megadásával most nem foglalkozom.

Természetesen a fenti módszer általánosítható több mint két kép esetére is.

Látható, hogy a naiv megközelítés csak a koncepciókat tartalmazza, nem közöl olyan technikai részleteket, melyek egy konkrét implementáció esetén biztosan felmerülnek. Ilyenek például azok, hogy a konkrét illesztéseket mikor kell meghatározni, és hogy egyáltalán ki határozza meg őket. Ha például objektumorientált megközelítést alkalmazunk, akor egyértelmű, hogy minden egyes objektum példány ismeri a saját típusát, osztályát, és rendelkezik karakterisztikáján túl valamilyen viselkedésmóddal, azaz adott illesztési metódusokkal. A kérdés az eltérő reprezentációk esetén merül föl igazán, hogy mely példányhoz tartozó metódust kell alkalmazni?

Ugyanilyen kérdés az, hogy szinte minden egyes illesztésnek létezik invariáns illetve nem invariáns módja. Azaz az általános illeszkedésen túl, mikor temporálisan adott közös méretűekre konvertáljuk a képeket az illesztéshez, olyan illeszkedést is vizsgálhatunk, mikor a minta más méretű mint a vizsgált kép, és arra vagyunk kíváncsiak hol, milyen mértékű nagyítással vagy kicsinyítéssel található meg a minta a képen. Sőt, azt is vizsgálhatjuk, hány fokkal van elforgatva. Így tehát azt mondhatjuk, hogy az egzakt illeszkedésen túl vizsgálhatunk eltolás-, nagyítás-, illetve elforgatás invariáns

illeszkedéseket is. Ezeket mind olyan tényezők, melyeket érdemes előre meghatározni. A továbbiakban tehát megadjuk a Cut-And-Or-Not megközelítési mód pontos leírását és formalizmusát.

9.2. Formalizmus

Ahhoz, hogy megadhassuk a Cut-And-Or-Not megközelítés pontos formalizmusát, tekintsük az alábbi jelöléseket [52].

Tekintsünk egy X digitális halmazon értelmezett f, $m-1$-szintű digitális képet. Ez csak egy egyszerű leképezés, az, hogy a színek hogyan képződnek le a $0, \ldots, m$ halmazra jelen pillanatban nem fontos.

Most pedig definiáljuk a $\mathcal{C}$ vágás (cut) műveletét.

4. Definíció. *Legyen $f : X \to \{0, \ldots, m\}$ egy digitális kép. Ekkor*

$$\mathcal{C}_{x_1,y_1,x_2,y_2}(f)(x,y) := \begin{cases} f(x,y), & ha \begin{cases} (x,y) \in X, \ and \\ x_1 \leq x \leq x_2, \ and \\ y_1 \leq y \leq y_2 \end{cases} \\ nem\ definiált, & egyébként \end{cases} \tag{9.1}$$

ahol $x_1, y_1, x_2, y_2 \in \mathbb{Z}$.

Figyeljük meg, hogy $\mathcal{C}_{x_1,y_1,x_2,y_2}(f)$ végül is nem más, mint egy olyan $q : Y \to \{0, \ldots, m\}$ digitális kép, ahol $Y \subseteq X$, melyet a x_1, y_1, x_2, y_2 paraméterek határoznak meg.

A korábbi fejezetekben a tulajdonságvektorokat mindig valós vektoroknak tekintettük. Valójában koncepcionális szinten a tulajdonságértékek nem valósak (hanem színek, textúrák, stb.). Tehát most logikailag alakítsuk át egy picit az eddigi tulajdonságvektor fogalmunkat. Logikai szinten tehát nincsenek valós értékek. Azok csak egy leképezéssel jelennek majd meg. (És ez a leképezés eredményez valós vektort). Tehát minden f kép esetén létezik F_i tulajdonságok egy véges halmaza, ahol $i = 1, \ldots, l$. Ezek a tulajdonságok a valós életben mindig véges Dom_{F_i} tartománnyal rendelkeznek. Így a korábbi, általános tulajdonságvektor fogalom minden további nélkül megszorítható az alábbi módon:

$$\underline{v} = (v_1, \ldots, v_l), \text{ahol } v_i \in \mathrm{Dom}_{F_i}. \tag{9.2}$$

Ezek a vektorok leképezhetőek egy k-dimenziós $\mathbb{R}^k$ vektortérbe, ahol annak $\underline{x} \in \mathbb{R}^k$ elemei,

$$\underline{x} = (x_1 \ldots, x_k), \tag{9.3}$$

alakúak. Ehhez tehát módosítva a korábbi tulajdonságvektor leképezésünket egy

$$\mathcal{F} : \mathrm{Dom}_{F_1} \times \cdots \times \mathrm{Dom}_{F_l} \to \mathbb{R}^k \tag{9.4}$$

tulajdonságvektor leképezést alkalmazhatunk (ld. (3.1)). Ez egy nagyon fontos lépés, ugyanis mint már korábban is láthattuk, az illesztések gyakran távolságmérésekként jelentkeznek.

A $\mathbb{R}^k$ elemei vektorok. Alkalmazhatjuk rajtuk a vektorösszeadást (mint minden vektortérben), és ezáltal definiálhatjuk a normát

$$\|\underline{x}\| = \sqrt{x_1^2 + \cdots + x_k^2}, \tag{9.5}$$

alakban. Amennyiben a normát belsőszorzattal definiáljuk, akkor használhatjuk a háromszög-egyenlőtlenséget (lásd távolság vs. hasonlóság, ill. (3.7))

$$\|\underline{x} - \underline{y}\| \leq \|\underline{x} - \underline{z}\| + \|\underline{z} - \underline{y}\|, \tag{9.6}$$

ahol $\underline{x}, \underline{y}, \underline{z} \in \mathbb{R}^k$.

Ha a leképezés $\underline{d}$-ből a $\underline{x}$-be *a priori* ismereteket is felhasznál a vektorok eloszlását illetően (vagy az eredeti vektorok nem biztosítják a háromszög-egyenlőtlenség értelmezését, lásd [49]), nem alkalmazható a belső szorzat. Ebben az esetben nem támaszkodhatunk a háromszög-egyenlőtlenségre, azaz a normát más módon kell definiálnunk.

Mivel a tulajdonságok és azok tartományai mind végesek, a lehetséges vektorleképezések biztosan véges vektortérbe képeznek. Véges vektorterek esetén bizonyítható, hogy létezik egy maximum távolság, melytől mindegyik vektor közelebb van egymáshoz. Ezt hívjuk a vektortér határának. Ez a maximum norma érték.

$$N = \max_{\underline{x},\underline{y} \in \mathbb{R}^k} \{\|\underline{x} - \underline{y}\|\}. \tag{9.7}$$

A $\mathcal{F}$ leképezés nagyon fontos, mert ez biztosítja számunkra az N létezését. Ezáltal számos egyéb norma is alkalmazható a kinyert tulajdonságok függvényében (például a Banach terek vagy az információelmélet normája, stb.). A fuzzy megközelítések esetén jól alkalmazható a súlyozott norma is

$$\|\underline{x} - \underline{y}\| = \sum_{j=1}^{k} w_j |x_j - y_j|, \tag{9.8}$$

amennyiben vannak *a priori* ismeretek az értékek eloszlását illetően. A

$$\underline{w} = (w_1, \ldots, w_k) \in \mathbb{R}^k, \quad w_i \geq 0, \quad i = 1, \ldots, k$$

súlyvektor reprezentálja az *a priori* ismereteket ekkor.

Most már definiálhatjuk magát az illesztést is.

5. Definíció. *Legyen $Q_N(f, g)$ egy norma két digitális kép $f : X \to \{0, \ldots, m\}$ és $g : Y \to \{0, \ldots, n\}$ között a korábban említett véges vektortereken alapulva, ahol $m, n \in \mathbb{N}$, és legyen N a vektortér véges határa úgy, hogy $0 \leq Q_N(f, g) \leq N$, és $N > 0$ minden egyes f és g képre. Ha $Q_N(f, g) = 0$, a két digitális kép f és g identikus, azaz a köztük lévő távolság nulla. A határ definíciójából adódóan a két kép közötti maximális távolság N. A Q_N norma függvényt ekkor* illesztés*nek nevezzük.*

Ha megfigyeljük, a fenti definíció nem tér ki arra, milyen metrika szerint kell értelmezni a távolságot. Igazából ez számunkra teljesen mindegy. Mi nem foglalkozunk a Q illesztés jóságával, sem technikai paramétereivel, így az alkalmazott metrikával sem.

Mindössze annyi kikötésünk van, identikus képek esetén definíciójához híven értéke nulla legyen. Konkrét implementációkban bármilyen jellegű (statisztikai, szintaktikai, stb.) illesztések alkalmazhatók például [30] [35] stb.

Az is észrevehető, hogy azt sem adtuk meg, hogy az illeszkedés invariáns-e. Feltesszük, hogy ha az, akkor melléktermékként előállítja azt az (i, j) pontot, l nagyítási mértéket és r elforgatási szöget (illetve ezek tetszőleges részhalmazát), melyek segítségével meghatározható az f kép azon részképe, melynek g-től vett távolsága épp Q_N. Feltesszük, hogy az illesztések invarianciájuk szerint osztályozhatók, így mi csak jelöljük, hogy invariáns illesztésre gondoltunk-e, avagy sem. Így tehát az alábbi jelöléseket fogjuk alkalmazni:

- $Q_N^{\emptyset}$: invariancia mentes illesztés
- Q_N^T: eltolás invariáns illesztés
- Q_N^R: elforgatás invariáns illesztés
- Q_N^S: nagyítás invariáns illesztés

Illetve értelmezhető ezek kombinációja is, tehát például a $Q_N^{T,R,S}$ egy nagyítás- eltolás- és elforgatás invariáns illesztést fog jelenteni. Annyi kiegészítést azért tennünk kell, hogy adott implementációk esetén nem biztos, hogy az invariáns illeszkedések vizsgálata megoldható, így ha például $Q_N^{T,S}$ nem értelmezhető, helyette mindig $Q_N^{\emptyset}$-et kell érteni. (Tehát az invariancia mentes illesztés helyettesítheti bármelyik invariáns illesztést.) Ha csak Q_N-et írunk, akkor mindegy, milyen illesztésről van szó.

Hogy jobban megértsük az invarianciákat, tekintsük át a következő néhány fogalmat és jelölést.

Legyen X egy digitális halmaz, $a, b \in X$, ahol $a = (a_x, a_y)$, $b = (b_x, b_y)$. Ekkor $a + b = r$, ahol $r = (a_x + b_x, a_y + b_y)$ és $a - b = s$, ahol $s = (a_x - b_x, a_y - b_y)$. Legyen $f : X \to \{0, \ldots, m\}$ egy digitális kép, és $T \in \mathbb{Z} \times \mathbb{Z}$ egy hely. Ekkor f_T egy eltolt változata f-nek T-vel eltolva, ha $f_T(x) = f(x + T)$ minden $x \in X$-re. Most pedig definiáljuk az eltolás invarianciát.

6. Definíció. *Legyen Q_N egy illesztés, f és g két digitális kép. Q_N eltolás invariáns, ha $Q_N(f, g) = Q_N(f_T, g) = Q_N(f, g_T) = Q_N(f_T, g_T)$ minden $T \in \mathbb{Z} \times \mathbb{Z}$-re.*

Legyen X egy digitális halmaz, $a \in X$, ahol $a = (a_x, a_y)$, és legyen l egy természetes érték, $l \in \mathbb{N}$. Ekkor $l * a = (la_x, la_y)$. Legyen $f : X \to \{0, \ldots, m\}$ egy digitális kép, és $l \in \mathbb{N}$ egy nagyítási mérték. Ekkor $l * f$ egy nagyított/kicsinyített (skálázott) változata f-nek l-szeresére skálázva, ha $l * f(x) = f(l * x)$ minden $x \in X$-re. Most pedig definiáljuk a nagyítás invarianciát.

7. Definíció. *Legyen $l \in \mathbb{N}$ egy nagyítási mérték, Q_N egy illesztés, f és g pedig két digitális kép. Ekkor Q_N nagyítás invariáns ha $Q_N(f, g) = Q_N(l*f, g) = Q_N(f, l*g) = Q_N(l * f, l * g)$.*

Legyen X egy digitális halmaz, $a \in X$, és $\varphi \in \mathbb{R}$ egy valós szám, $a = (a_x, a_y)$. Ekkor $a^{\varphi} = ([a_x \cos\varphi + a_y \sin\varphi], [-a_x \sin\varphi + a_y \cos\varphi])$, ahol $[x]$ jelöli x egész részét. Legyen $f : X \to \{0, \ldots, m\}$ egy digitális kép, $\varphi \in \mathbb{R}$ egy elforgatási szög. Ekkor f^{φ} egy elforgatott változata f-nek φ radiánnal elforgatva, ha $f^{\varphi}(x) = f(x^{\varphi})$ minden $x \in X$-re. Most pedig definiáljuk az elforgatás invarianciát.

8. Definíció. *Legyen $\varphi \in \mathbb{R}$ egy elforgatási szög, Q_N egy illesztés, f és g két digitális kép. Ekkor Q_N elforgatás invariáns, ha $Q_N(f,g) = Q_N(f,g^\varphi) = Q_N(f^\varphi,g) = Q_N(f^\varphi,g^\varphi)$.*

Ezek a legismertebb invarianciák. Természetesen bármilyen más invariancia is definiálható, illetve ezek is definiálhatók más módon is.

Most pedig adjuk meg, mi a küszöb. Erre azért van szükség, mert gyakran a kérdéseket feltevő végfelhasználók számára a távolsággal megadott illeszkedés nem érthető. Ilyen esetekben, az illeszkedést eldöntendő kérdésnek tekintve, ha a távolság egy adott küszöbszámtól kisebb, akkor a két adott bináris kép illeszkedik, ellenkező esetben nem. Ezt az alábbi küszöbölő függvénnyel fejezhetjük ki.

Legyen Th egy függvény, ahol

$$Th(Q_N(f,g),t) := \begin{cases} 1, & \text{ha } Q_N(f,g) \leq t \\ 0 & \text{egyébként} \end{cases} \tag{9.9}$$

A t érték a küszöbérték. Ha $Th(Q_N(f,g),t) = 1$, akkor g illeszkedik f-re, egyébként nem.

A Th függvényt gyakran predikátumként értelmezzük, azaz ha $Th(Q_N(f,g),t) = 1$, akkor a predikátum értéke igaz, és hamis ha $Th(Q_N(f,g),t) = 0$.

9.3. A fuzzy Cut-And-Or-Not megközelítési mód

Ahhoz, hogy definiáljuk a Cut-And-Or-Not megközelítést a fuzzy logika eszközeivel, definiáljuk a fuzzy logikai összekötőjeleket, és működésüket. Meg kell említenünk, hogy az alkalmazott fuzzy logika nem része a Cut-And-Or-Not megközelítésnek, az itt közölt helyett bármilyen más fuzzy logikát is lehet alkalmazni. Mi a Lukasiewicz logika alapjaiból merítünk [46]. Annyi átalakításra van szükségünk, hogy az illesztés által szolgáltatott eredményeket a Lukasiewitz összekötőjelek által feldolgozhatóvá kell tennünk.

Egy Q_N illesztés kiértékelés utáni értéke a Lukasiewicz logikának megfelelően $\frac{N-Q_N}{N} \in [0,1]$, ha Q_N egyenletes eloszlást követ. Amennyiben *a priori* ismereteket is felhasználtunk a Q_N értékek eloszlását illetően, akkor bármilyen más (például súlyozott) leképezést is alkalmazhatunk. A legjobb megoldás az, ha a Q_N értékeket egy egyenletes eloszlást követő $[0,1]$ fuzzy halmazba tudjuk leképeztetni. (Az irodalomból ismert, hogy a fuzzy halmazoknak összemérhetőknek kell lenniük. Amennyiben az összemérhetetlenségük minimális, azonos eloszlásúaknak tekinthetők.)

Ezek után az alábbi definíciókat adhatjuk meg:

9. Definíció. *Legyenek q_1 és q_2 kiértékelt illesztések, ekkor a $q_1 \wedge q_2 = \max\{0, q_1 + q_2 - 1\}$ a Lukasiewicz logikának megfelelő konjunkció.*

10. Definíció. *Legyenek q_1 és q_2 kiértékelt illesztések, ekkor a $q_1 \vee q_2 = \min\{1, q_1 + q_2\}$ a Luksiewicz logikának megfelelő diszjunkció.*

11. Definíció. *Legyen q kiértékelt illesztés, ekkor $\neg q = 1 - q$ Lukasiewicz negáció.*

Észrevehetjük, hogy a Lukasiewicz logikának megfelelő kiértékelés végül is nem más, mint a Q_N illesztés által értékként adható $[0, N]$ halmaz egyértelmű $[0, 1]$ halmazba történő konverziója, így ha a Q_N szerint a két kép identikus (tehát értéke 0), akkor az a Lukasiewicz kiértékelés után 1 lesz. Ha pedig a két kép távolsága nagyobb, mint N, azt a kiértékelés utáni 0 érték jelzi.

Így a fuzzy Cut-And-Or-Not algoritmus az alábbi módon definiálható:

- Adottak $f_1, \ldots, f_n$ és $g_1, \ldots, g_m$ digitális képek, valamint $Q_{1,N_1}, \ldots, Q_{k,N_k}$ illesztések.
- Képezzük ezekből a képekből a kívánt $\mathcal{C}_{x_{1i},y_{1i},x_{2i},y_{2i}} f_i$ és $\mathcal{C}_{x_{1j},y_{1j},x_{2j},y_{2j}} g_j$ részképeket, ahol $i = 0, \ldots, n$, $j = 0, \ldots, m$.
- Végezzük el a megfelelő részképekkel az illesztéseket, így előáll p számú illesztés végrehajtása esetén p darab $Q_{l,N_l}(\mathcal{C}_{x_{1i},y_{1i},x_{2i},y_{2i}} f_i, \mathcal{C}_{x_{1j},y_{1j},x_{2j},y_{2j}} g_j)$ illesztés, ahol $l \in \{1, \ldots, k\}$, $i \in \{1, \ldots, n\}$ és $j \in \{1, \ldots, m\}$.
- Értékeljük ki ezeket az illesztéseket a Lukasiewitz kiértékelés szerint, így előáll $q_1, \ldots q_p$ illesztési érték.
- Készítsünk ezekből a q_i, $i \in \{1, \ldots, p\}$ értékekből logikai formulát a Lukasiewicz logikának megfelelő összekötőjelek segítségével, és értékeljük ki.

Amennyiben a kiértékelt formulánk a Lukasiewicz logika szerint igaz, úgy a feltett és a Cut-And-Or-Not formalizmussal formalizált kérdésünkre is a válasz igaz, ellenkező esetben nem. Amennyiben ezt a fuzzy Cut-And-Or-Not megközelítést a naiv leíráshoz hasonlóan csak két képre (f-re és g-re) hajtjuk végre, úgy alkalmas az g-k cseréje mellett adatbázisban történő hasonlóságon alapuló keresésre is (mai divatos szóhasználat szerint tartalomalapú, content-based keresésre is).

Megjegyzés: Gyakori eset, hogy a q_i, $i \in \{1, \ldots, p\}$ illesztési értékeket súlyozzák valamilyen $w_1, \ldots, w_p$ súlyok segítségével, ahol $w_i \geq 0$, $i = 1, \ldots, n$ valós szám. Ha létezik i, hogy valamely $w_i > 1$, akkor lehetséges, hogy az adott, besúlyozott illesztési érték szintén egy egynél nagyobb értéket képvisel. Ekkor a fenti Lukasiewicz összekötőjelek nem alkalmazhatók. Mint említettem, a Cut-And-Or-Not megközelítésnek nem része az alkalmazott fuzzy logika, így azt tetszés szerint lecserélhetjük. Alkalmazhatjuk helyettük az alábbi általánosított fuzzy összekötőjeleket is, melyek már le tudják kezelni az ilyen eseteket is.

12. Definíció. *Legyenek w_1q_1 és w_2q_2 kiértékelt, súlyozott illesztések, ekkor a $w_1q_1 \wedge w_2q_2 = \min\{w_1q_1, w_2q_2\}$ általánosított fuzzy konjunkció.*

13. Definíció. *Legyenek w_1q_1 és w_2q_2 kiértékelt, súlyozott illesztések, ekkor a $w_1q_1 \vee w_2q_2 = \max\{w_1q_1, w_2q_2\}$ általánosított fuzzy diszjunkció.*

14. Definíció. *Legyen wq kiértékelt, súlyozott illesztés, ekkor $\neg wq = \max\{0, 1 - wq\}$ általánosított fuzzy negáció.*

9.4. A nulladrendű megközelítési mód

Természetesen a fuzzy logikához megfelelő erőforrások nem állhatnak rendelkezésre minden egyes implementáció esetében, viszont a nulladrendű (kvantorok nélküli) logika eszközei szinte minden egyes programozási nyelv esetében megtalálhatóak, így megmutatjuk, hogyan lehet a fuzzy Cut-And-Or-Not megközelítést nulladrendűvé alakítani. Ehhez a nulladrendű logikai összekötőjeleket kell használnunk [8] [54]. Így $\neg$ nulladrendű negáció, $\vee$ nulladrendű diszjunkció, $\wedge$ pedig nulladrendű konjunkció. Operandusaik igaz/hamis értékkel rendelkező predikátumok lehetnek.

Így tehát a nulladrendű Cut-And-Or-Not algoritmus az alábbi módon alakul:

- Adottak $f_1, \ldots, f_n$ és $g_1, \ldots, g_m$ digitális képek, valamint $Q_{1,N_1}, \ldots, Q_{k,N_k}$ illesztések.
- Képezzük ezekből a képekből a kívánt $\mathcal{C}_{x_{1i},y_{1i},x_{2i},y_{2i}} f_i$ és $\mathcal{C}_{x_{1j},y_{1j},x_{2j},y_{2j}} g_j$ részképeket ahol $i = 0, \ldots, n$, $j = 0, \ldots, m$.
- Végezzük el a megfelelő részképekkel az illesztéseket, így előáll p számú illesztés végrehajtása esetén p darab $Q_{l,N_l}(\mathcal{C}_{x_{1i},y_{1i},x_{2i},y_{2i}} f_i, \mathcal{C}_{x_{1j},y_{1j},x_{2j},y_{2j}} g_j)$ illesztés, ahol $l \in \{1, \ldots, k\}$, $i \in \{1, \ldots, n\}$ és $j \in \{1, \ldots, m\}$. A továbbiakban $Q_i(f_i, g_i)$ alakban jelöljük őket ahol $i = 1, \ldots, p$
- Alkossunk igényeink szerint $t_1, \ldots, t_p$ számú küszöbértéket úgy, hogy $0 < t_i < N_l$ ahol N_l az Q_i illesztéshez tartozó N_l érték, $i \in \{1, \ldots, p\}$, $l \in \{1, \ldots, k\}$.
- Képezzünk ebből a p darab illesztésből a Th predikátum segítségével p darab $Th_i(Q_i(f_i, g_i), t_i)$ predikátumot, ahol $i = 1, \ldots, p$.
- Készítsünk ezekből a $Th_i(Q_i(f_i, g_i), t_i)$, $i = 1, \ldots, p$ predikátumokból logikai formulát az elsőrendű logikai összekötőjeleink ($\vee, \wedge, \neg$) segítségével, és értékeljük ki.

Amennyiben a kiértékelt formulánk igaz, úgy a feltett és a Cut-And-Or-Not formalizmussal formalizált kérdésünkre is a válasz igaz, ellenkező esetben nem. Természetesen itt is igaz, hogy az illesztendő képek halmaza csak két képet tartalmaz, tehát f-hez illesztjük g-t, úgy a megközelítés jól alkalmazható adatbázisokban történő tartalomalapú képkinyerésre.

9.5. Példák

Ebben a szakaszban néhány példát mutatok be a Cut-And-Or-Not megközelítés alkalmazására a kérdés formalizációban. Legyenek adottak a következő illesztések

- Q^S_{1,N_1} lokális szín illesztés [40] [57], mely a pixelek értékeit és helyeit illeszti.
- $Q^{T,R,S}_{2,N_2}$ globális szín illesztés [40] [57], mely a színhisztogramok távolságait méri.
- Q^S_{3,N_3} struktúra illesztés [40], mely a képen lévő objektumok alakját és helyét illeszti.

- $Q^{T,S}_{4,N_4}$ textúra illesztés, [40], mely a képek textúráit vizsgálja.
- $Q^{\emptyset}_{5,N_5}$ általános bináris illesztés [57] [56], mely a szintrevágott képek pixeleinek értékeit vizsgálja.

Minden illesztés esetében legyen az N_i, $i = 1, 2, 3, 4, 5$ értéke 100, mint ahogy az [40]-ben is beállítható. Az eloszlások egyenletes eloszlást követnek. Legyen adott néhány segéd kép, mely a QBE-alapjait képezi. Ezek

- g_1 egy homogén kék kép. Méret: 50×50.
- g_2 egy fekete kitöltött kör fehér háttér előtt. Méret: 50×50.
- g_3 egy fehér háttér fekete pontokkal. Méret: 50×50.
- g_4 egy homogén piros kép. Méret: 50×50.

Most pedig tekintsünk néhány formalizált kérdést.

Szelektáljuk ki azokat a képeket, ahol egy kitöltött kör található a kép bal oldalán kék háttér előtt, mérete és színe ismeretlen, és a jobb felső 50×50 *méretű területen egy fekete kitöltött kör található fekete háttér előtt.*

A fuzzy Cut-And-Or-Not formalizáció segítségével a $C(f)$:

$$\left(Q^{T,R,S}_{2,N_2}(\mathcal{C}_{0,0\frac{m}{2},n}f, g_1) \wedge Q^{S}_{3,N_3}(\mathcal{C}_{0,0\frac{m}{2},n}f, g_2)\right) \wedge Q^{\emptyset}_{1,N_1}(\mathcal{C}_{m-50,0,m,50}f, g_2), \quad (9.10)$$

ahol f mérete $m \times n$. Így azon f-ek lesznek leválogatva, ahol egy t küszöbérték mellett $Th(C(f), t) = 1$.

Nézzünk most meg egy másik példát:

Szelektáljuk ki azokat a képeket, ahol a háttér piros, fekete pontokkal, és egy sötét kör található a bal felső 50×50*-es területen.*

A nulladrendű Cut-And-Or-Not formalizációval a $C(f)$:

$$\left(Th(Q^{T,R,S}_{2,N_2}(f, g_4), 60) \wedge Th(Q^{T,S}_{4,N_4}(f, g_3), 60)\right) \wedge Th(Q^{\emptyset}_{5,N_5}(\mathcal{C}_{0,0,50,50}f, g_2), 80). \quad (9.11)$$

Ahol $C(f)$ igaz, ott f a keresett kép. (A 60 és 80 értékek természetesen csak példák.)

10. fejezet

Alakfelismerés

A 4.5 alfejezetben említettem, hogy az alakok, a görbék illesztésében jól alkalmazhatóak a lánckód alapú megközelítések is. A lánckód a képfeldolgozásban gyakran használatos kontúrkódolási forma. A lánckód valójában iránykódok egy sorozata, mely leírja az objektum kontúrját egy adott kiindulási pontból. Én csak olyan objektumokkal foglalkozom, melyeknek csak külső kontúrja van. A hátrányai az ismert illesztési módszereknek az elforgatással illetve a skálázással szembeni hatékonyság illetve a zajérzékenység. Ha kevésbé zajérzékeny algoritmust akarunk, akkor statisztikai eszközöket kell használnunk. Ha bevezetjük az alak kódot a lánckódok helyet, akkor elérhetjük a függetlenséget az elforgatástól illetve a kiindulási pont megválasztásától.

10.1. Bevezetés

Az adatok és az algoritmusok a két legalapvetőbb részei a programoknak. Az adatszerkezetek megválasztása így tehát fundamentális kérdés a programozási feladatok folyamán. A lánckód az egyike a tradicionális képi adatszerkezeteknek. A láncokat az objektumok határvonalainak leírására használjuk a képfeldolgozás során [52] [36]. A lánckódok valójában olyan adatok, melyek szimbólumok sorozataként tekinthetők, ahol a szomszédos elemek a képen található primitívek (kontúrpixelek) szomszédságainak felelnek meg. A lánckódok illetve a Freeman kódok [16] így gyakori eszközei az objektumhatárok leírásának. Az objektumok lánckódokkal történő leírása pedig elengedhetetlen a szintaktikus mintaillesztési feladatokhoz, de ha invariáns tulajdonságokkal rendelkező mintaillesztési algoritmusokat szeretnénk készíteni, akkor új megközelítési módokat kell figyelembe vennünk.

Amennyiben a lánckódot mintaillesztéshez használjuk, akkor függetlenné kell tennünk a lánc előállításához használt kiindulási pont megválasztásától. Ezt a folyamatot nevezzük normalizálásnak. Egy lehetséges mód a lánckód normalizálására a differencia kód és az alak szám (shape number, alak kód) [44]. Az alak szám is egy határleírás, mely a 4-szomszédságon alapuló lánckódokon alapszik, és úgy definiálható, mint egy olyan differencia kód, mely számként értelmezve a legkisebb [21]. Mi egy kvázi elforgatás invariáns illesztést szeretnénk kapni, ezért a 4-szomszédságon alapuló kódok helyett a 8-szomszédságból indulunk ki.

A lánckódok, differencia kódok és alak kódok nagyon zajérzékenyek, és bármilyen jellegű skálázás problémákat okozhat, ha felismerésre használjuk őket. Módosított statisztikai módszerek alkalmazásával a felismerés zajérzékenysége és skálázással szembeni érzékenysége csökkenthető.

10.2. Kontúr leírások

Miután egy képet sikerült foltokra szegmentálnunk fontos az, hogy úgy tudjuk őket leírni és reprezentálni, hogy az a későbbi feldolgozást elősegítse. Két lehetőségünk van a foltok reprezentálására. Egyrészt jellemezhetjük őket belső (internális) jellegzetességeikkel (például kontúr), illetve külső (externális) jellegzetességeikkel (pl. a terület által tartalmazott pixelek). A lánckódok a belső karakterisztikán alapulnak [21]. Mielőtt rátérnénk a lánckódra, először vizsgáljuk meg, mit nevezünk kontúrnak.

15. Definíció. *Legyen adott egy $f(x, y)$ lyukat nem tartalmazó folt. Ekkor ennek a foltnak 4-kapcsolódás szerinti kontúrja alatt azon f-beli pontokat értjük, melyek a háttérrel 8-kapcsolódóak. 8-kontúr alatt pedig a háttérrel 4-kapcsolódó pontokat.*

Ez a definíció valójában egyenes következménye a [15] szomszédsági struktúrákon alapuló definíciójának.

A definíción túl még fontos megjegyeznünk, hogy a kontúrpontok bejárása mindig az óra járásával ellentétes irányban történik (ld. [21], [44]).

10.2.1. A lánckód

A lánckód használata gyakori a képfeldolgozásban. A lánckód objektumok leírására szolgál, mely egységnyi vonalszegmensek irányításának egy sorozata. Az ilyen láncok első elemét ki kell persze egészíteni egy olyan információval, hogy hol helyezkedik el, ha azt pontosan rekonstruálni akarjuk (ld. [52]). A lánckódok nemcsak az objektumok határainak leírására alkalmasak, hanem az egy pixel széles vonalas ábrák leírására is. Meg kell jegyeznünk, hogy a lánckód relatív ismérv.

A lánckódok tipikusan a 4- illetve a 8-kapcsolódásra épülnek, ahol a szomszédos szegmensek irányai az alábbi séma alapján kódolhatók (ld. 10.1 ábra).

10.1. ábra. A 4- és a 8-irányítású lánckódok irányai

Ha előállítjuk a 4- és 8-irányítású lánckódját a 10.2 ábrán látható objektumnak, akkor észrevehetjük, hogy a két kód különbözik. Észrevehető, hogy a 4-irányítású lánckód nem túl hatékony a 8-kapcsolódó kontúrok leírásához.

Példa: A 4-irányítású lánckódja a 10.2 ábrán levő objektumnak: 1 000 1 2 1 2 1 3 2 3 2 3 0 3 és a 8-irányítású lánckódja 1 00 2 33 55 6 7.

Ha előállítottuk a kontúr kódot, megkaphatjuk a kontúr hosszát. A kontúr hossza egy régió tulajdonság, mely könnyen számolható a lánckódból [52]. A vertikális és horizontális lépések egységnyi hosszúak, és a diagonális lépések 8-kapcsolódás esetén $\sqrt{2}$ hosszúságúak. Látható, hogy 4-kapcsolódás esetén a kontúr „hosszabb lesz", ugyanis a diagonális lépés kettő, nem diagonális lépés együttese, melynek hossza 2. Ez is az oka annak, amiért a 8-irányítású lánckód sokkal hatékonyabb. Az alábbi formula segít kiszámítani a kontúr hosszát. Legyen a reprezentált objektum lánckódja l^f, ahol az elemek $l_1^f \dots l_n^f$, és a kontúr hossza L az alábbi módon számolható:

$$L = \sum_{i=1}^{n} U(l_i^f), \tag{10.1}$$

ahol 4-kapcsolódás esetén

$$U(l_i^f) = 1, \tag{10.2}$$

és 8-kapcsolódás esetén

$$U(l_i^f) = \begin{cases} 1, & \text{ha } l_i^f \text{ páros vagy 0,} \\ \sqrt{2}, & \text{ha } l_i^f \text{ páratlan.} \end{cases} \tag{10.3}$$

A lánckód nemcsak a kontúr leírására alkalmas, hanem későbbi módszerekhez is kellemes, például vékonyításhoz illetve foltfelismeréshez.

Most csak olyan objektumokkal foglalkozom, melyeknek csak külső kontúrjuk létezik, azaz nem tartalmaznak lyukakat. Ezek az objektumok a foltok. Legyen tehát egy X digitális halmazunk ($X \subset \mathbb{Z} \times \mathbb{Z}$), amelyen értelmezünk egy lyukakat nem tartalmazó bináris objektumot, mely legyen $f(x, y) : Y \to \{0, 1\}$, ahol $Y \subset X$. Legyen ennek az $f(x, y)$ objektumnak a lánckódja $l_f = \mathfrak{L}(f(x, y))$, ahol $\mathfrak{L}$ a lánckódot képező transzformációnk.

Most megmutatom, hogy hogyan kell ezt a 8-irányítású lánckódot képezni. (A 4-irányítású lánckód hasonlóan számolható.) Tekintsük egy tetszőleges $p(x, y)$ pixelt $\mathbb{Z}^2$-beli koordinátákkal, és tekintsük ezen pixel nyolc-szomszédjainak egy rögzített bejárását, azaz képezzük $\mathcal{N}_8(p(x, y))$-t, és ennek az óramutató járásával ellenkező bejárását, azaz

$$\begin{array}{ccc} p_3 & p_2 & p_1 \\ p_4 & \boldsymbol{p} & p_0 \\ p_5 & p_6 & p_7 \end{array} \quad \text{, ahol a } p_i \text{ iránya } p\text{-től } i.$$

Az l^f lánckód nagyon könnyen előállítható. Rögzítsük tehát a kiindulási pontot, és képezzük a kontúrpontok megelőző kontúrpontjához képzett irányát a fenti bejárás szerint. Jelöljük az $i.$ elemét az l^f-nek l_i^f, $i = 1 \dots n$ alakkal, ahol n a láncelemek száma $l^f = (l_1^f \dots l_n^f)$.

A foltoknak természetesen a kontúrjuk kerületének hosszán kívül területük is fontos leíró tulajdonságuk. Felmerül persze a kérdés, hogy ez hogyan kapcsolódik szervesen a lánckódokhoz. A válasz nagyon egyszerű. A 4-irányítású lánckódok esetén belátható, hogy a 8-irányítású átlós lépések kettő, 4-irányítású lépésre bomlanak föl. Ekkor, (elméleti szinten) ha pl. 0-1 lépés helyett 1-0 lépést tekintünk, a lánckód hossza nem változik, csak annyi történik, hogy a lánckód által érintett kontúrpontok egyike „kilép" az objektumból (azaz a háttér egy pixele helyettesíti az érintett kontúrpontot).

Nagy elemszámú lánckódok esetében a zajok kezelésére adhat ez az elméleti megközelítés megoldást, méghozzá azokra az esetekre, amikor a folt területét is felhasználjuk az illeszkedésnél (ld.10.6. ábra). Ekkor ugyanis az ilyen lánckódelem-cserékkel két különböző területet (mondhatni egy alsó és egy felső közelítést) tudunk adni, melynek átlaga zajos képnél sokkal közelebb van az eredeti területhez, mint az, amelyik a lánckódból eleve adódik.

A következő szakaszokban 8-irányítású lánckódokkal foglalkozom. A lánckódról elmondható, hogy gyorsan számolható, belőle az objektum könnyen rekonstruálható és eltolás invariáns. Hátránya az, hogy nem elforgatás invariáns, nem skála invariáns, zajérzékeny és a kiindulóponttól nagyban függ.

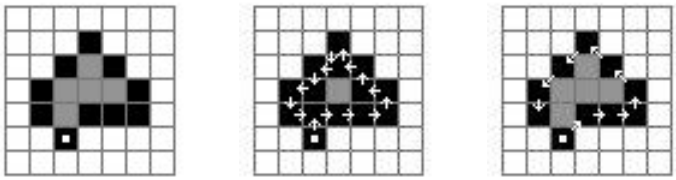
Esetünkben a kódot elforgatás invariánsnak nevezzük, ha egy foltnak, és annak elforgatottjának ugyanaz a kódja.

Ezek a tulajdonságok rontják a lánckódolt objektumokon történő foltkeresést, ezért az algoritmushoz ezt a kódot egy kicsit át kell alakítanunk.

10.2.2. A differencia kód és az alak szám

Vezessünk be egy módosított kódot, egy $d^f = \mathcal{D}(f(x,y))$ differencia kódot, ahol $\mathcal{D}$ a differencia kódot képező transzformáció. Egy differencia kódot a lánckódból úgy származtathatunk, hogy vesszük a lánckód elemei közötti különbséget (távolságot) az óra járásával ellenkező irányban. Azaz az 1 és a 3 értékek különbsége értelemszerűen 2, de a 7 és 1 értékek különbsége is 2, az 1 és 0 értékek különbsége pedig 7.

Fontos, hogy ha a $d_1^f \dots d_n^f$ differencia elemeket képezzük az $l_1^f \dots l_n^f$ lánckód elemekből, akkor a d_n^f elem az l_n^f és l_1^f elemek differenciája legyen, hiszen feltételezzük, hogy a lánckód zárt.



10.2. ábra. Az objektum megjelölt kiindulási ponttal, illetve az objektum kontúrjának 4- és 8-irányítású bejárása.

Példa: Legyen a 10.2 ábra objektumának lánckódja 1 00 2 33 55 6 7. A differencia kód 7 0 2 1 0 2 0 11 2.

Ez a kód kvázi elforgatás invariáns, ami azt jelenti, hogy invariáns azokra az esetekre, ha az objektum $k\frac{\pi}{4}$-vel van elforgatva, ahol k egy tetszőleges egész szám, azaz a differencia kód ugyanaz. (A 4-irányítású lánckód esetében az elforgatási szög $k\frac{\pi}{2}$.) Ezt a kódot ahhoz, hogy kezdőpont invariáns legyen tovább kell transzformálnunk, így jutunk el az alak kódhoz.

Az alak kódot úgy képezhetjük a differencia kódból, hogy képezzük a folt összes kontúrpontjából a differencia kódot, és lexikografikus rendezéssel a legkisebbet választjuk ki. Ugyanezt az eredményt szolgáltatja, ha bármely differencia kódból — azt periodikusnak feltételezve, ahol ő egy periódus — kiindulva megkeressük a leghosszabb nullás futamot, és azt tekintjük a kód elejének. Több leghosszabb nullás futam esetén az ilyen futamokkal indított differencia kódok közül kell a lexikografikusan legkisebbet kiválasztani.

Példa: (ld. 10.2 ábra). A differencia kódja a 1 00 2 33 55 6 7 lánckódnak 7 0 2 1 0 2 0 11 2, és ennek alak kódja 0 11 2 7 0 2 1 0 2 és nem 0 2 0 11 2 7 0 2 1.

Ez már kvázi elforgatás invariáns, kezdőpont invariáns kód, és egyértelmű. Azaz ugyanazon objektum különböző kiindulási pontból vett lánckódjaiból képzett alak kódok megegyeznek, azaz legyen az alak kódot előállító transzformáció $\mathcal{S}$, ekkor $\mathcal{S}({l^f}_{(1)}) = \mathcal{S}({l^f}_{(2)})$, ahol ${l^f}_{(1)}$ és ${l^f}_{(2)}$ ugyanazon $f(x, y)$ bináris objektum különböző kiindulási pontból képzett lánckódjai.

Sajnos az alak kód sem skála invariáns, és szintén probléma az is, hogy míg a lánckódra igaz volt, hogy ha az elemeinek a számát szakaszonként n-szeresére növeljük, akkor az objektum — kis hibával — egy n-től függő k-szoros méretűvé válik, addig ez az alak kódról nem mondható el. Viszont ha belőle visszatranszformáljuk a lánckódot, akkor különböző kiindulópontból vett lánckódok esetén is, az alak kóddá és vissza történő transzformáció ugyanazon eredményt adja, azaz ha $f(x, y)$ két különböző kiindulópontból vett lánckódjai ${l^f}_{(1)}$ és ${l^f}_{(2)}$, akkor $\mathcal{S}^{-1}(\mathcal{S}({l^f}_{(1)})) = \mathcal{S}^{-1}(\mathcal{S}({l^f}_{(2)}))$. Fontos, hogy az alak kódból történő visszaalakítás esetén magunknak kell választani egy kezdőirányt, ami bármi lehet, de az algoritmus alkalmazásai esetén konzisztensen ugyanannak kell lennie (a normalizáció esetén ez leginkább a 0 irány, mint elfogadott standard).

Természetesen az így visszaalakított lánckód nem az eredeti objektum lánckódja, hanem annak valamilyen elforgatottjának kódja, de ez számunkra nem fontos, hiszen mi csak az objektum alakjával foglalkozunk, hiszen azt akarjuk azonosítani.

Jelöljük tehát ezt az újonnan képzett lánckódot ${l'}^f = \mathcal{S}^{-1}(\mathcal{S}(l^f))$-vel, ahol l^f az $f(x, y)$ egy tetszőleges lánckódja.

Legyen s^f alak kód $s^f = (s_1^f \dots s_n^f)$, ahol s_i^f a i. eleme a s^f-nak, és legyen ${l'}^f = ({l'_1}^f \dots {l'_n}^f)$ egy normalizált lánckód, ahol $\mathcal{S}^{-1}(s^f) = {l'}^f$. A normalizált lánckód elemei az alábbi módon számíthatók:

- Legyen ${l'_1}^f \; s_1^f$.
- Legyen ${l'_{i+1}}^f \; ({l'_i}^f + s_{i+1}^f) \mod 8$, ahol $i = 1 \dots n-1$.
- Ezután mozgassuk az utolsó ${l'_n}^f$ elemet az első ${l'_1}^f$ elem elé.

Első lépésben a kiindulási irányunk 0. Az utolsó lépés azért szükséges, mert enélkül az első futam első eleme — ha létezik — a normalizált lánckód végére kerülne. Ez nem probléma, hiszen a lánckód gyűrű, de ezzel a lépéssel a további műveleteket nagyban megkönnyíthetjük.

Amennyiben a lánckódok nem azonos hosszúságúak, úgy az alábbi algoritmussal egyenlő hosszúságúakra hozhatjuk őket. Legyen l' hossza $n_{l'}$ és k' hossza $n_{k'}$.

Legyen p az $n_{l'}$ és $n_{k'}$ legkisebb közös többszöröse. Növeljük meg mindkét mintát p hosszúságúra úgy, hogy az l'-ben lévő futamokat $\frac{n_{l'}}{p}$-szereseikre növeljük, a k'-ban lévő futamokat pedig $\frac{n_{k'}}{p}$-szereseikre. Ezek után (ha az l' lánckódot tekintjük etalonnak) képezzünk klasztereket az l' lánckódban a futamhatárok mentén, majd a k' lánckódban úgy, hogy annak i. klaszterébe azok az elemek kerülnek, melyek k' lánckódban található helyük szerinti sorszámuk megegyezik azon elemek sorszámával, melyek az l' lánckódban az i. klaszterbe esnek. Ezek után a k' lánckódban található futamok elemeit helyettesítsük azokból az elemekből képzett, az eredeti futamokkal megegyező méretű futamokkal, mely elemek a futamokban a legtöbbször szerepeltek. Ha több ilyen is van egy futamon belül, akkor a mediánból (azaz helyük szerinti sorrendjük szerint a középső elemből) képzett, az eredetivel megegyező méretű futammal. Ezek után a két (azonos futamszámmal rendelkező) lánckódok futamainak hosszát csökkentsük le $\frac{p}{n_{l'}}$-szereseikre. Ezek után visszakaptuk az l' lánckódot, és egy olyan módosított k'' lánckódot, hogy $n_{k''} = n_{l'}$, és futamaik száma is megegyezik. A továbbiakban az egyszerűség kedvéért k'' helyett k'-t írunk.

10.3. Foltfelismerés lánckódolt objektumokon

Adott egy $X \subset \mathbb{Z} \times \mathbb{Z}$ digitális halmaz, valamint két lánckódolt lyukakat nem tartalmazó bináris objektum, folt ($f_1(x,y) : Y \to \{0,1\}$, $f_2(x,y) : Y \to \{0,1\}$, $Y \subset X$), és l^{f_1} és l^{f_2} lánckódok. A kérdés az, hogy az $f_1(x,y)$ objektum nagyítás-, eltolás- és $k\frac{\pi}{4}$-szerint elforgatás invariánsan illeszkedik-e a $f_2(x,y)$ objektumra. Hogy megadjuk a választ, statisztikai módszereket fogunk alkalmazni.

10.3.1. A χ^2-próba alapú módszer

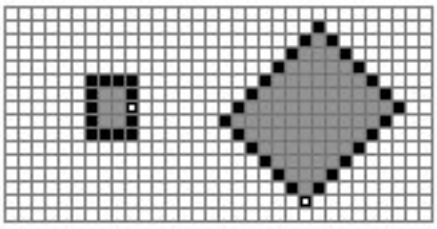
Képezzük az $l'^{f_1} = \mathcal{S}^{-1}(\mathcal{S}(l^{f_1}))$ és $l'^{f_2} = \mathcal{S}^{-1}(\mathcal{S}(l^{f_2}))$ normalizált lánckódokat. Tekintsük ezt a két lánckódot úgy, mint két valószínűségi változót, és a lánckód elemeit pedig a hozzájuk tartozó mintáknak, azaz l'^{f_1}-hez rendeljük hozzá a ξ valószínűségi változót, melyhez tartozó m elemű minta a $(\xi_1, \dots \xi_m)$, l'^{f_2}-hez pedig a η változót, amelyhez a $(\eta_1 \dots \eta_n)$ minta tartozik, majd vizsgáljuk meg e két valószínűségi változó homogenitását χ^2-próbával, ugyanis ha a két lánckód csak nagyításban, elforgatásban, illetve eltolásban különbözik, akkor tekinthetjük őket úgy, mint egy populációból vett két eltérő elemszámú mintát (ld. [51]). Mivel ezek a minták lánckódok, így a ξ_i és η_j értékek a $\{0 \dots 7\}$ halmaz elemei. Jelölje az elemek multihalmazát Ξ. Osszuk ezt föl a ξ és a η értékeit nyolc halmazra úgy, hogy $A_k = \{x | x \in \Xi, x = k\}$, $k = 0 \dots 7$, és legyen ν_k azon ξ_i mintaelemek száma, melyre $\xi_i \in A_k$, és μ_k azon η_i mintaelemek száma, melyre $\eta_i \in A_k$. Ezek után vezessük be a

$$\chi^2 = m \cdot n \sum_{k=0}^{7} \mathcal{R}_k, \tag{10.4}$$

próbastatisztikát, ahol

$$\mathcal{R}_k = \begin{cases} \frac{(\frac{\nu_k}{m} - \frac{\mu_k}{n})^2}{\nu_k + \mu_k}, & \text{ha } \nu_k, \mu_k \neq 0. \\ 0, & \text{egyébként.} \end{cases} \tag{10.5}$$

Erről a statisztikáról bebizonyítható, hogy megfelelően nagy m és n esetén, ha a minta lánckódja illeszkedik valamilyen módon az objektumunk lánckódjára, akkor a χ^2-próbastatisztika eloszlása megközelítőleg 7 szabadságfokú chi-négyzet eloszlás lesz [51]. Rögzített szignifikanciaszinttel p-re a p értékek egy χ^2 táblából könnyen meghatározhatók. Ezen információk birtokában azt mondhatjuk, hogy f_1 illeszkedése f_2-re nem zárható ki, ha 0.01 szignifikanciaszinten a próbastatisztikánk értéke kisebb, mint 18.48, 0.05 szignifikancia mellett kisebb, mint 14.07, illetve 0.1 szignifikancia mellett kisebb, mint 12.02. Ezek az értékek természetesen egy χ^2 eloszlástáblából lettek meghatározva.



10.3. ábra. A két folt jelölt kiindulási ponttal

Példa: (ld. 10.3 ábra). Legyen l^1*=22 444 6666 000 22 egy négyszög lánckódja, és legyen egy nagyított,* $\frac{\pi}{4}$*-el elforgatott változatának lánckódja* l^2*=1111111 333333 5555555 777777, ekkor az alak számok* $\mathcal{S}(l^1)$*=000 2 00 2 000 2 00 2 és* $\mathcal{S}(l^2)$ *=00000 2 0000 2 00000 2 0000 2. Legyen a kezdő irányítás 0, ekkor* l'^1*=0000 222 4444 666, és* l'^2*=000000 22222 444444 66666, kiszámolva a* χ^2 *értékeket az eredmény* $m = 14$, $n = 22$, $\nu_0 = 4$, $\nu_1 = 0$, $\nu_2 = 3$, $\nu_3 = 0$, $\nu_4 = 4$, $\nu_5 = 0$, $\nu_6 = 3$, $\nu_7 = 0$, *és* $\mu_0 = 6$, $\mu_1 = 0$, $\mu_2 = 5$, $\mu_3 = 0$, $\mu_4 = 6$, $\mu_5 = 0$, $\mu_6 = 5$, $\mu_7 = 0$, *és* χ^2*=0.0234. Ez az érték elég kicsi ahhoz, hogy kijelenthessük, a két objektum illeszkedése nem vethető el.*

Kis elemszámú minták esetén viszont ez az eljárás nem mindig szolgáltat megfelelő értékeket. Amennyiben kis elemszámú mintáink vannak, azaz a foltjaink kontúrjai, lánckódjai rövidek, úgy módosításokat kell végrehajtanunk.

Legyen p az m és az n legkisebb közös többszöröse. Növeljük meg mindkét mintánkat úgy, hogy a benne található futamokat (azonos számsorokat) növeljük $\frac{p}{m}$, illetve $\frac{p}{n}$-szereseikre, ezáltal elértük, hogy kaptunk két olyan mintát, melyek elemszámai megegyeznek (p) és megfelelően nagyok.

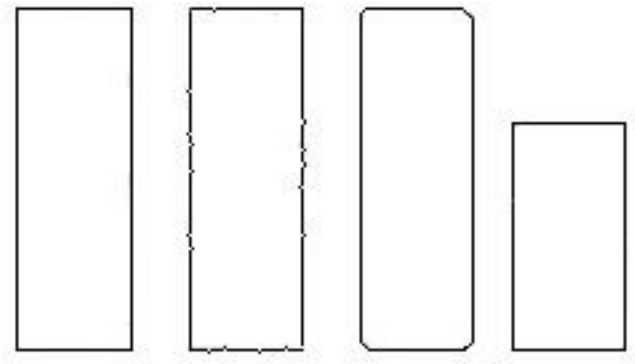
Jelölje ezeket a módosított mintákat $(\xi'_1 \dots \xi'_p)$ és $(\eta'_1 \dots \eta'_p)$. Ezek után válasszuk ki az eredeti két változónk közül azt, melynek eredeti elemszáma kevesebb volt, azaz $m < n$ esetén $(\xi'_1 \dots \xi'_p)$-t, $n < m$ esetén $(\eta'_1 \dots \eta'_p)$-t, egyenlőség esetén pedig válasszuk azt, melyben a futamok száma kevesebb. Ha l futam volt, l klasztert fogunk alkotni. Vágjuk tehát l klaszterre a futamhatárok mentén a mintát. Vágjuk szét a másik mintát is úgy, hogy a j-edik klaszterbe a minta következő k_j eleme tartozzon, ahol k_j a másik minta j-dik futamának hossza. Ezek után alakítsuk át a mintákat úgy, hogy mind a két mintában az i-edik klaszterbe tartozó mintaértékekhez adjunk hozzá $i \cdot 10$-et. Jelölje ezt a két új mintát $(\xi''_1 \dots \xi''_p)$ és $(\eta''_1 \dots \eta''_p)$.

Vezessünk be két új változót, ζ_1 és ζ_2 változókat, és legyen ζ_1 mintája $(\xi_1'' \dots \xi_p'')$ és ζ_2 mintája $(\eta_1'' \dots \eta_p'')$. Ezek után már alkalmazhatjuk a χ^2 tesztünket, vagy más próbát.

10.3.2. χ^2 eredmények

Vizsgáljuk meg a 10.4 ábrát. A χ^2 teszt hatékonysága nagyobb, ha a lánckódok elemeinek száma nagy. Fontos megjegyezni, hogy e lánckódok jó tulajdonságai csak poligonok esetében teljesülnek. Amennyiben a lánckódok elemeinek száma megközelítőleg 400-500 körül mozog, úgy a tesztek jó eredményt szolgáltatnak. Amennyiben megzajoljuk az objektumokat kb. 15-25% zajjal, úgy a módszer már nagymértékben romlik. (Zaj jelen esetben a lánckódelemek megváltozása, mely leggyakrabban úgy fordul elő, hogy egy adott ξ_l érték $(\xi_i \pm 1) \mod 8$ alakra változik.) Legyen az etalon g_1. Ekkor a χ^2 próba a következő eredményeket szolgáltatja:

Képek	χ^2 **értékek**
g_2	2.13615
g_3	7.84314
g_4	20.1681



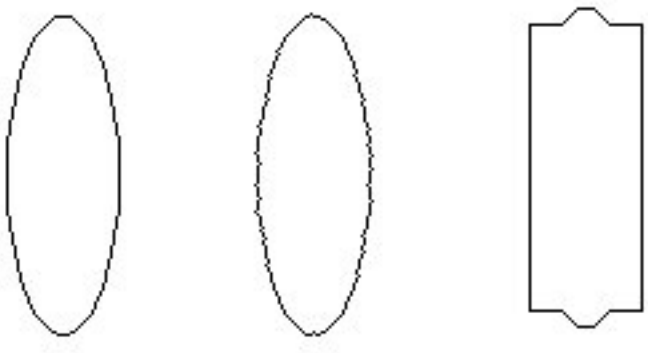
10.4. ábra. Teszt képek: g_1, g_2, g_3, g_4.

Realizálható, hogy nem zárhatjuk ki, hogy g_2 és g_3 illeszkedik g_1-re. A g_4 nem illeszkedik az etalonra. Az g_2 kb. 10% zajt tartalmaz, ez az oka annak, hogy a χ^2 függvény értéke nagyobb, mint zaj nélkül. (Azonos képek esetén természetesen az érték pontosan 0.)

Az is észrevehető, hogy a módszer elkülönítette az ellipsziseket, amennyiben az etalon g_1 (ld 10.5 ábra). Az g_5 egy egyszerű ellipszis, g_6 pedig egy 15% kontúrzajt tartalmazó változata.

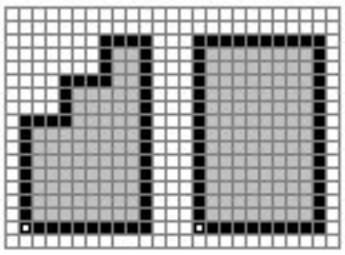
Képek	χ^2 **értékek**
g_5	67.1363
g_6	68.3796
g_7	2.35779

A módszer igazából poligonokra lett kifejlesztve. Ha az etalon g_5, és az illesztett folt g_6, a χ^2 próba eredménye 116.457, azaz a próba elveti a foltok illeszkedését, azaz nem tudja kezelni a görbéket.



10.5. ábra. Teszt képek: g_5, g_6, g_7.

Van néhány speciális eset, mikor a módszer olyan képet is elfogad, mely különbözik az etalontól. Ha a lánckódok közel azonos elemeket tartalmaznak sorrendfüggetlenül, akkor a módszer rossz eredményt szolgáltathat. Ebben az esetben érdemes a klaszterező változatot alkalmazni, vagy a foltok területének összehasonlítását is figyelembe venni.



10.6. ábra. A h_1 és h_2 képek rögzített kiindulási ponttal.

A 10.6 ábrán láthatjuk, hogy h_1 lánckódja 000000000 22222222222222 4444 66 5 44 66 5 44 66666666 és h_2 lánckódja 000000000 22222222222222 444444444 66666666666 666. Ebben az esetben a χ^2 próba 3.19771-t ad eredményül, pedig a foltok különbözőek. A klaszterező változat használatával természetesen azt kapjuk eredményül, hogy kizárható a h_1 és a h_2 illeszkedése. Összehasonlítva a foltok területeit szintén ugyanezt az eredményt kapjuk, hiszen azonos skálázás mellett tekintve azokat $T_{h_2} > T_{h_1}$ jelentősen ($T_{h_2} >> T_{h_1}$) (T_X jelenti X területét).

A χ^2 próba jó eredményeket produkált, de a kontúrzaj nagyban leronthatja a módszer hatásfokát. A kérdés az, hogyan növelhetjük meg a hatékonyságot, és hogy hogyan viselkednek a lánckódok zaj esetén. Természetesen más statisztikai módszereket is megvizsgáltam, hasonló eredményeket kapva. Független minták esetén a Fisher egzakt tesztet, a Mann Whitney U tesztet, illetve nem független minták esetében a Cochran féle Q teszt vizsgálatát, valamint a Wilcoxon féle előjeles rang próbákat ejtettem meg.

10.4. Foltfelismerés sztochasztikus folyamatok segítségével

Egy másik megközelítési mód, ha a kódokat nem valószínűségi változóknak, hanem sztochasztikus folyamatok realizációjának tekintjük.

10.4.1. Az egyszerű modell

Most bemutatom, hogyan lehet a lánckódok illeszkedését megvizsgálni, ha modellként a sztochasztikus folyamatok leírását használjuk zajmentes, illetve zajos esetben.

Képezzünk a lánckódokból egy-egy irányított differencia kódot.

Az irányított differencia kódokat, vagy más néven az előjeles differencia kódokat az alábbi módon képezhetjük az eredeti differencia kódokból, (elemeit most d_t-vel jelöljük). Képezzük most ebből az irányított differencia kódot az alábbi képlettel

$$D(l')_t = \begin{cases} d_t & \text{ha } d_t \in \{0,1,2,3\} \\ -(8-d_t) & \text{egyébként} \end{cases} \tag{10.6}$$

A $D(k')$ analóg módon előállítható.

Vegyük észre, hogy a lánckódok természetéből adódóan $D(l') \neq 4$ és $D(l') \neq -4$, azaz $D(l') \in \{-3,-2,-1,0,1,2,3\}$.

Tekintsük a $D(l')$ kódot, mint egy sztochasztikus folyamat realizációját (továbbiakban a rövidség kedvéért D_t-t írunk $D(l')_t$ helyett). Az egylépéses valószínűség-átmenetek pedig

$$P_{q,r}^{t,t+1} = P(D_{t+1} = r | D_t = q) \tag{10.7}$$

egy olyan kitétellel, hogy

$$\sum_{r=-3}^{3} P_{q,r}^{t,t+1} = 1, \text{ rögzített } t \text{ mellett}, \tag{10.8}$$

azaz minden egyes t időpillanatból 1 valószínűséggel lépünk át a $t+1$ időpillanatba.

Az adott $D(l')$ realizációból természetesen csak egy olyan valószínűség-átmenet halmazt adhatunk meg, ahol rögzített t mellett, ha a $P_{ij}^{t,t+1}$ értékek közül ($i = -3,\ldots,3$, $j = -3,\ldots,3$) valamely értéke 1, a többi pedig 0. Hogy melyik 1 az a $D(l')$-ből megadható. Megfigyelhetjük, hogy nem történik más, mint minden egyes t időpillanathoz megadunk egy 7×7 méretű mátrixot, mely tartalmazza, a következő lépésben mely értéket veszi föl D_{t+1} és milyen valószínűséggel. Belátható, hogy mivel D_t-ben vagyunk, ezt az állapotot ismerjük, így nekünk egy adott t időpillanatban csak a mátrix $P_{D_t,j}^{t,t+1}$, $j = -3,\ldots,3$ vektorára van szükségünk.

Ez a vektor pedig az alábbi alakú:

$$P_{D_t,j}^{t,t+1} = \begin{cases} 1 & \text{ha } j = D_{t+1}, \\ 0 & \text{egyébként.} \end{cases} \tag{10.9}$$

Természetesen — mivel $D(l')$ egy etalon lánc — a valóságban előforduló láncok ettől eltérhetnek a zaj miatt, azaz minden egyes t időpillanatban a következőbe történő lépés esetén az átmenetvektor értékeitől eltérhetünk. Ennek az eltérésnek a megadása,

a zaj jellegzetességének megadása, egy fontos dolog. Feltételezzük, hogy a zaj normális eloszlású (egyenes vonal esetén, azaz $l'_t = l'_{t+1}$), minden egyes lépésben $\mathcal{N}(0, \sigma_t^2)$-t követ, úgy, hogy kielégíti a (10.8)-ot.

Mivel itt diszkrét esetben vagyunk, csak közelíteni tudjuk a $\mathcal{N}(0, \sigma_t^2)$ eloszlást. Fontos viszont észrevennünk, hogy

$$D(l') \in \{-3, -2, -1, 0, 1, 2, 3\}, \tag{10.10}$$

és ismert a 3σ szabály, azaz hogy ha valamilyen ξ véletlen változóra igaz, hogy $\xi \sim \mathcal{N}(m, \sigma^2)$, akkor $P(m - 3\sigma < \xi < m + 3\sigma) \simeq 0.9972$, azaz annak valószínűsége hogy ξ az m $\pm 3\sigma$ környezetén kívülre esik elenyésző, így érdemes a $\mathcal{N}(0, 1)$-et approximálni az alábbi módon:

$$P_{D_t,j}^{t,t+1} = \begin{cases} 0.6344, & \text{ha } j = 0, \\ 0.1587, & \text{ha } |j| = 1, \\ 0.0228, & \text{ha } |j| = 2, \\ 0.0013, & \text{ha } |j| = 3, \end{cases} \tag{10.11}$$

mely beláthatóan teljesíti (10.8)-ot és jól közelíti a $\mathcal{N}(0, 1)$-et.

Ha $l'_t \neq l'_{t+1}$, akkor egy olyan haranggörbét tekintünk melynek D_{t+1}-nél van maximum helye. Fontos megemlíteni, hogy a diszkrét értelmezési tartományt ciklikusnak tekintjük, így az eloszlás nem más mint a már eddig is felhasznált $\mathcal{N}(0, 1)$ standard normális eloszlás, csak éppen nem j szerint. A $P_{D_t,j}^{t,t+1}$ ekkor az alábbi táblázattal definiálható:

	$j=-3$	$j=-2$	$j=-1$	$j=0$	$j=1$	$j=2$	$j=3$
$D_t=-3$	0.6344	0.1587	0.0228	0.0013	0.0013	0.0228	0.1587
$D_t=-2$	0.1587	0.6344	0.1587	0.0228	0.0013	0.0013	0.0228
$D_t=-1$	0.0228	0.1587	0.6344	0.1587	0.0228	0.0013	0.0013
$D_t=0$	0.0013	0.0228	0.1587	0.6344	0.1587	0.0228	0.0013
$D_t=1$	0.0013	0.0013	0.0228	0.1587	0.6344	0.1587	0.0228
$D_t=2$	0.0228	0.0013	0.0013	0.0228	0.1587	0.6344	0.1587
$D_t=3$	0.1587	0.0228	0.0013	0.0013	0.0228	0.1587	0.6344

(10.12)

mely szintén teljesíti (10.8)-ot és jól közelíti a $\mathcal{N}(0, 1)$-et. Sőt, ez tekinthető a fentebbi eset általánosításának, hiszen egyenes mentén, azaz $D_{t+1} = 0$ esetén is jól működik.

A (10.8) kielégítésén túl a fenti konstrukció a

$$\sum_{j=-3}^{3} P_{D_t j}^{t,t+1} = 1, \text{ rögzített } t \text{ mellett}, \tag{10.13}$$

feltételt is kielégíti.

Meg kell említeni, hogy a fenti képletben használt értékek a standard normális eloszlás táblázatának oly módon módosított értékei, hogy a (10.8) teljesítése érdekében az $1 - P(m - 3\sigma < \xi < m + 3\sigma)$ érték a $P(\xi = m)$ értékhez lett hozzáadva, hiszen a valóságban annak a valószínűsége, hogy egy egyenes vonal lánckódja 2π fordulatot tegyen 0, ugyanis az lehetetlen esemény. (Az olyan sorozatok, amelyek tartalmazzák az $l_1 l_2$ szekvenciát, ahol $l_1, l_2 \in \{0, 1, 2, 3, 4, 5, 6, 7\}$, és $|l_2 - l_1| = 4$ beláthatóan nem lánckódok.) Ezért nem lehet $|D_t| = 4$, azaz a fenti módosításunk megengedhető.

Most pedig — a felismerés kedvéért — képezzünk ebből a folyamatból egy másik folyamatot (jelöljük $Y(t)$-vel) oly módon, hogy az $Y(t)$ elemei az eredeti l' lánckód elemei ($Y(t) = l'_t$, $t = 1, \ldots, n$), melynek átmenetvalószínűség-mátrixa a szokásos:

$$P_{q,r}^{t,t+1} = P(Y(t+1) = r | Y(t) = q) \tag{10.14}$$

a szokásos kitétellel, hogy

$$\sum_{i=0}^{7} \sum_{j=0}^{7} P_{i,j}^{t,t+1} = 1, \text{ rögzített } t \text{ mellett}, \tag{10.15}$$

azaz itt is minden egyes t időpillanatból 1 valószínűséggel lépünk át a $t+1$ időpillanatba.

Mivel az l' lánckód és a $D(l')$ előjeles differencia kód kezdeti irányítástól eltekintve kölcsönösen egyértelműen megfeleltethetők egymásnak, így az egylépéses átmenetvalószínűségek is megfeleltethetők egymásnak. Így a számunkra fontos átmenetvektorok legyenek az alábbiak:

$$P_{Y(t),j}^{t,t+1} = \begin{cases} P_{D_t,i}^{t,t+1} & \text{ha létezik,} \\ 0 & \text{egyébként,} \end{cases} \tag{10.16}$$

ahol

$$i = \begin{cases} Y(t) - j & \text{ha } Y(t) - j > 0 \text{ és } Y(t) - j \in \{0,1,2,3\} \\ Y(t) - j - 8 & \text{ha } Y(t) - j > 0 \text{ és } Y(t) - j \notin \{0,1,2,3\} \\ Y(t) - j + 8 & \text{ha } Y(t) - j < 0 \text{ és } Y(t) - j + 8 \in \{0,1,2,3\} \\ Y(t) - j & \text{ha } Y(t) - j < 0 \text{ és } Y(t) - j + 8 \notin \{0,1,2,3\} \end{cases}, \tag{10.17}$$

ahol $j \in \{0,1,2,3,4,5,6,7\}$, tehát ezek nyolc elemű vektorok. Belátható, hogy ezek kielégítik (10.15)-ot, sőt a (10.13) miatt az alábbiakat is:

$$\sum_{j=0}^{7} P_{Y(t),j}^{t,t+1} = 1, \text{ rögzített } t \text{ mellett}.$$

Innentől a feladat úgy határozható meg, milyen valószínűséggel lehet a k' kód az $Y(t)$ folyamat egy realizációja (jelöljük ezt $k' \sim l'$ alakban). Erre a későbbiekben válaszolunk.

10.4.2. A tanítható modell

Természetesen az átmenetvalószínűség-mátrix adta megközelítési mód nagyon kellemes abból a szempontból, hogy az algoritmus kvázi „tanítható". Ez alatt azt kell érteni, hogy nem egy etalon l' lánckódhoz illesztünk egy k' mintát, hanem egy l' sorozat segítségével meghatározzuk az átmenetvalószínűségeket, azaz betanítjuk a rendszert, és az így előálló (az elfogadható esetekre felkészített) etalon átmenetvalószínűségek szerint vizsgáljuk meg annak a valószínűségét, a k' lehet-e számunkra elfogadható realizáció.

Az ilyen betanított modell esetében, ha megfelelően nagy elemszámú mintánk van a potenciális lánckódokat illetően, akkor akár a zaj külön kezelésétől el is tekinthetünk, hiszen a potenciális lánckódrealizációk már magukban hordozzák a zajt. Tekintsünk

tehát egy $l'^{(i)}$ lánckód sorozatot, ahol $i = 1, \ldots, N$. Képezzünk ezekből a normalizált lánckódokból azonos hosszúságú lánckódokat a már korábban említett módon, azaz jelöljük ki etalonnak $l'^{(1)}$-et, és igazítsuk hozzájuk a többi hosszát. Ezek után, most kivételesen ne készítsük el az előjeles differencia kódokat, hanem az $Y(t)$ folyamat átmenetvalószínűségeit az alábbi módon adjuk meg:

$$P_{q,r}^{t,t+1} = \sum_{i=1}^{N} \frac{C_{q,r}(l_t'^{(i)}, l_{t+1}'^{(i)})}{N}, \tag{10.18}$$

ahol

$$C_{q,r}(l_t'^{(i)}, l_{t+1}'^{(i)}) = \begin{cases} 1, & \text{ha } l_t'^{(i)} = q, \text{ és } l_{t+1}'^{(i)} = r, \\ 0, & \text{egyébként.} \end{cases} \tag{10.19}$$

Belátható, hogy ez szintén teljesíti (10.15)-ot, viszont (10.4.1)-t nem, így nem lehet (10.16)-ot alkalmazni. Ez annak köszönhető, hogy az így kapott $P_{q,r}^{t,t+1}$ értékek nem vektort alkotnak, hanem egy 8×8 méretű mátrixot.

10.4.3. A modellek kiértékelései

Tegyük fel, hogy az $l'^{(i)}$ sorozat minden egyes tagjának hossza végtelen. Ekkor a k' végtelen hosszúságú lánckód konvergál az $l'^{(i)}$ lánckód sorozathoz, ha

$$\frac{1}{T} \sum_{t=0}^{T} \left(1 + P_{k_t', k_{t+1}'}^{t,t+1} - \max_{i,j \in \{0,\ldots,7\}} P_{i,j}^{t,t+1} \right) \to 1, \tag{10.20}$$

$T \to \infty$ esetén.

Belátható viszont, hogy a valóságban $T \to n$ azaz T az etalon hosszához tart. Mivel

$$0 \leq 1 + P_{k_t', k_{t+1}'}^{t,t+1} - \max_{i,j \in \{0,\ldots,7\}} P_{i,j}^{t,t+1} \leq 1, \tag{10.21}$$

minden $t \in \{0, \ldots, n\}$-re így azt mondhatjuk, hogy

$$0 \leq \sum_{t=0}^{n} \left(1 + P_{k_t', k_{t+1}'}^{t,t+1} - \max_{i,j \in \{0,\ldots,7\}} P_{i,j}^{t,t+1} \right) \leq n, \tag{10.22}$$

ahol ha ez a fenti érték 0, akkor a k' teljes mértékben különbözik az $l'^{(i)}$ sorozattól, és n ha a lehető legnagyobb mértékben megegyezik velük. Így tehát alkalmazhatjuk az alábbi gyakoriságot:

$$P(k' \sim l') = \frac{1}{n} \sum_{t=0}^{n} \left(1 + P_{k_t' k_{t+1}'}^{t,t+1} - \max_{i,j \in \{0,\ldots,7\}} P_{i,j}^{t,t+1} \right), \tag{10.23}$$

mely egy speciális kedvező/összes alak, ugyanis a számláló a k'-ben előforduló átmenetek $l'^{(i)}$ sorozat által meghatározott átmenetvalószínűségeinek, illetve a legkedvezőbb átmenetvalószínűségek megegyezésének bekövetkezési valószínűségeinek összege (mivel (10.18) kielégíti (10.15)-t), míg a számláló ennek maximális értéke. Belátható, hogy nagyelemű lánckódok esetében, és viszonylag nagy kontúrzaj lehetősége esetén (10.23) jól alkalmazható, mert a zajt nemcsak lokálisan kezeli, hanem globálisan is.

Ugyanilyen megfontolások alapján a másik esetben az alábbi képletet lehet alkalmazni

$$P(k' \sim l'^{(i)}) = \frac{1}{n} \sum_{t=0}^{n} \left(1 + P^{t,t+1}_{k'_t,k'_{t+1}} - \max_{j \in \{0,\ldots,7\}} P^{t,t+1}_{Y(t),j} \right). \tag{10.24}$$

10.4.4. Tapasztalati eredmények

A fenti eredményeket a már jól ismert 10.4 ábrán látható objektumokra alkalmazva a következő értékeket kaptam.

Amennyiben a g_1 objektumot tekintjük etalonnak, úgy az egyszerű modell az alábbi eredményeket szolgáltatja:

Kép	P
g_2	0.945182
g_3	0.469663
g_4	0.318229

ahol P a (10.23) értéke. Realizálható, hogy a g_2 illeszkedik az etalonra. Azt is kijelenthetjük, hogy a g_4 illeszkedése kizárható.

A tanítható modell esetében számos kontúrzajjal megzajolt négyzet lánckódját tanítottam be a rendszerrel. Az eredeti négyzet és minden megzajolt négyzet illeszkedett a modell szerint, de az egyéb zajolt négyzetek csak akkor kerültek elfogadásra, ha azok zajai a betanított zajok keverékei voltak (azaz nem jelent meg új helyen zaj).

11. fejezet

Konklúzió

A dolgozat eddigi részeiben bemutattam a legfontosabb VIR és CBIR paradigmákat, és a hozzájuk kapcsolódó kutatási eredményeimet. Elsőként az illesztő algoritmusok tökéletesítésére próbáltam törekedni, így foglalkoztam [56]-ban a bináris illesztés általánosítási lehetőségeivel. Mivel a bináris illesztés általában csak egy végső, ritkán alkalmazott algoritmus a CBIR rendszerekben, így más algoritmusok felé terelődött a figyelem. Kormos Jánossal a lánckódokon alapuló statisztikai mintaillesztésekkel foglalkoztunk [27]-ben és [28]-ban. A megközelítés továbbfejlesztéseként alkalmaztuk a sztochasztikus folyamatokat is (ld. [62]), mely előrelépést jelent a visszacsatolást is alkalmazó rendszerek esetében is (lásd tanítható modell, 10.4.2, 2. fejezetek). Az alkalmazott algoritmusokon, azaz a képadatbázisok képfeldolgozói oldalán túl megvizsgáltam az adatbázis-kezelő rendszerek oldaláról adódó problémákat is. Ha nemcsak BLOB-okban akarunk képeket tárolni, különféle egyéb technikákra is szükségünk van. Ezekre ad megoldást az Oracle *inter*Media és Visual Information Retrieval rendszerei [40], [41]. Erre alapulva vizsgáltam meg a képek objektumorientált tárolását adatbázisokban a [61]-ben. Az objektumorientált technológia persze egyéb lehetőségeket nyitott, melyeket [58]-ban vizsgáltam meg. A tapasztalatokat felhasználva kezdtem el objektummodellen alapuló szemantikus indexelési technikákkal foglalkozni [59] és [60]-ban. Az alkalmazott illesztő algoritmusok implementációját, illetve a hasonlóság megadásához való felhasználásukat [57]-ben jártam körbe, és így jutottam el ahhoz, hogy adatbázis-kezelői szinten kell a kérdéskörrel foglalkozni adatbázis közeli nyelveken (lásd [38]). Ilyen megoldásokat alkalmaztunk például Jónás Richárddal és Kollár Lajossal a [24]-ben leírt rendszer esetén is (mely tekinthető úgy, mint egy visszakereső algoritmus nélküli képadatbázis). A különféle tapasztalatokat összegyűjtve alakult ki a hasonlóság megadására a fuzzy technikát alkalmazó Cut-And-Or-Not megközelítési mód, mely lehetővé teszi a részképeken alapuló illesztéseket és az összetett mintaillesztési kérdések alkalmazhatóságát is. A Cut-And-Or-Not megközelítési móddal bővebben a [58]-ban és [60]-ban foglalkoztam.

Alkalmazás

12. fejezet

Bevezetés

Ebben a részben egy létező alkalmazást mutatok be, melyet az eredmények tesztelése érdekében hoztam létre. Az alakfelismerési szakaszban leírt statisztikai lánckód illesztő módszerek aránylag könnyen tesztelhetők, ugyanis – mint képfeldolgozási módszerek – a képadatbázisoktól függetlenül is implementálhatók, sőt, a tesztelésükhöz sincs szükség képadatbázisokra, csak megfelelő mennyiségű lánckódra. Az ismertetett módszereket magam is külön, C++ nyelven implementáltam, a teszteléseket pedig egy Celeron 366 típusú számítógéppel végeztem, 224 MByte RAM-mal. A tesztek eredményeit már leírtam az előző részben, de megtalálhatók [27]-ben és [28]-ban is beigazolva a lánckódon alapuló mintaillesztés működését.

Belátható viszont, hogy mind a szemantikus filterezés, mind a Cut-And-Or-Not megközelítési mód nem tesztelhető egy komplett, létező képadatbázis nélkül. Ezen két módszer eredményeinek bemutatása érdekében kellett létrehoznom a már említett képadatbázist, így ebben a részben annak részleteiről lesz szó.

13. fejezet

A CD-adatbázisról

Maga az adatbázis zenei CD-borítók képeit, illetve a hozzájuk tartozó CD-k adatait tartalmazza, úgymint előadók, számcímek, stílus, kiadás éve, stb. Az adatbázis lehetővé teszi a CD-k közötti szöveges kereséseket, illetve a képek alapján történő kereséseket. A Cut-And-Or-Not technika lehetővé teszi, hogy részképeken alapuló bonyolult kérdéseket is feltehessünk. Egy előre elvégzett hierarchikus osztályozásnak köszönhetően a szemantikus indexelés nagyságrendekkel felgyorsítja a keresést az elvégzendő illesztések számának csökkentésével.

13.1. Motiváció

Az adatbázis ötlete egy megtörtént eseten alapszik. Egyik ismerősöm mutatott egy számot nekem, amely nagyon megtetszett. Láttam, ahogy fogja a CD-t, kiveszi a tokjából, majd beteszi a meghajtóba. Nem igazán figyeltem, mit mondott, ki volt az előadó, a dallamot jegyeztem meg csak, és a borító külsejét. Nos ezek után, pár héttel később pont egy CD-boltban jártam, és eszembe ötlött, meg kellene a CD-t venni. Na igen, de az eladóknak nem tudtam semmilyen lényeges információval szolgálni az előadót vagy albumcímet illetően. Mindössze a zenei stílust tudtam behatárolni, és a borítót ismertem volna fel, ha látom. El is kezdtem keresni, de lehetetlen feladatnak tűnt a több ezer CD átnézése. Ha akkor, egy olyan adatbázissal rendelkeztem volna, mint amit itt bemutatok, perceken belül megtaláltam volna a keresett CD-t.

Természetesen ne legyenek illúzióink, és ne várjunk csodákat. A tartalomalapú képkeresési módszerek még korántsem olyan kifinomultak, mint azt mi szeretnénk. Az alkalmazott, elterjedt illesztési technikák (lásd első rész) leginkább általános, paraméterezhető technikák, azok megfelelő alkalmazásához jól fel kell paraméterezni őket, hiszen az algoritmusok nem tudhatják, mit keresek, milyen ismérvek és elvek alapján, ha nem adjuk meg azokat nekik paramétereikben. Az itt alkalmazott keresések is paraméterezhető keresések, bár a felhasználó feladatát nagyságrendekkel megkönnyíti az a tény, hogy leginkább csak fontossági sorrendet kell beállítani a keresések között különféle súlyok segítségével.

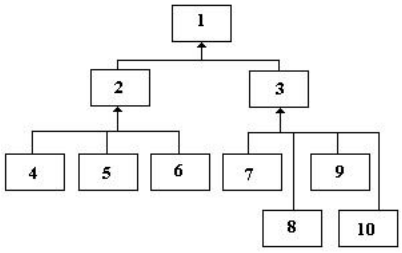
13.2. A keresés menete és a felhasználói interfész

Az adatbázis implementációjához az Oracle adatbázis-kezelő rendszer 9.0.1-es verzióját használtam fel a megfelelő *inter*Media modullal [41]. Az implementáció nyelve a PL/SQL volt [38]. A rendszer teljes mértékben vastag szervert és vékony klienst követel meg. A kliens oldaláról csak egy web böngészőre van szükség a keresés végrehajtásához és az eredmények megtekintéséhez. A szerver egy Celeron 633 típusú számítógép volt, Windows 2000 Professional operációs rendszerrel, 192 MByte RAM-mal. Itt kell megemlítenem, hogy ez a konfiguráció egy nagyon lassú konfiguráció a feladatok elvégzéséhez.

Mint említettem, a kliens oldalon csak egy böngészőre van szükség (minimum IE 5.0), ugyanis az adatbázis a PL/SQL Server Pages technológia segítségével egy Oracle HTTP szerveren (Apache) keresztül vezérelhető.

13.2.1. A szemantikus szűrésről

A dolgozat korábbi fejezeteiben kifejtettem az [59]-ben is leírt szemantikus képindexelési lehetőséget. Itt szükség van egy hierarchikus osztályozásra a keresés érdekében. Ez az osztályozás esetünkben a 13.1. ábrán látható osztályhierarchián alapszik, ahol



13.1. ábra. Az osztálydiagram.

a számmal jelölt osztályok az alábbiakat jelentik:

- 1: 'General Images', általános kép, azaz ő a gyökér. Minden, az adatbázisban letárolt borító egy általános kép. Amely képek sem nem fényképek, sem nem grafikák, azok is itt találhatók.

- 2: 'Graphics', grafika. Az általános kép közvetlen leszármazottja, nem fénykép, hanem rajz, festmény, generált objektum. Amely kép nem illik bele egyik grafikai alosztályba sem, az is itt található.

- 3: 'Photo', fénykép. Szintén az általános kép közvetlen leszármazottja. Nem grafika, hanem fényképtechnológiával készült montázs, kép. Amely kép nem illik bele egyik fénykép alosztályba sem, az is itt található.

- 4: 'Graphics: Persons', rajzolt személy(ek). Grafikai eszközökkel történő emberábrázolás.

- 5: 'Graphics: Animals or other creatures', grafikai eszközökkel ábrázolt állatok, „élő" (meseszerű, vagy épp félelmetes szörnyszerű) teremtmények.
- 6: 'Graphics: Normal or nonfigurative objects', grafikai eszközökkel elkészített tárgyak, nonfiguratív rajzok.
- 7: 'Photo: Persons'. Fényképtechnikával elkészített portrék, személyek.
- 8: 'Photo: Animals', Lefényképezett állatok.
- 9: 'Photo: Objects', Fényképtechnikával készített tárgyak fotói, illetve tárgymontázsok, épületek.
- 10: 'Photo: Landscapes and natural pictures'. Természetfotók, tájképek, növények, fák képei.

Az itt felvázolt osztálydiagram tökéletesen kielégíti a 8.2 fejezet végén megadott feltételeket, így tekinthető úgy, mint típusfa.

Az interfészként funkcionáló, nyitó HTML lapon rádiógombok segítségével állíthatjuk be, mely osztályba tartozó képekre szeretnénk az adott keresést végrehajtani (ld. 13.2. ábra).

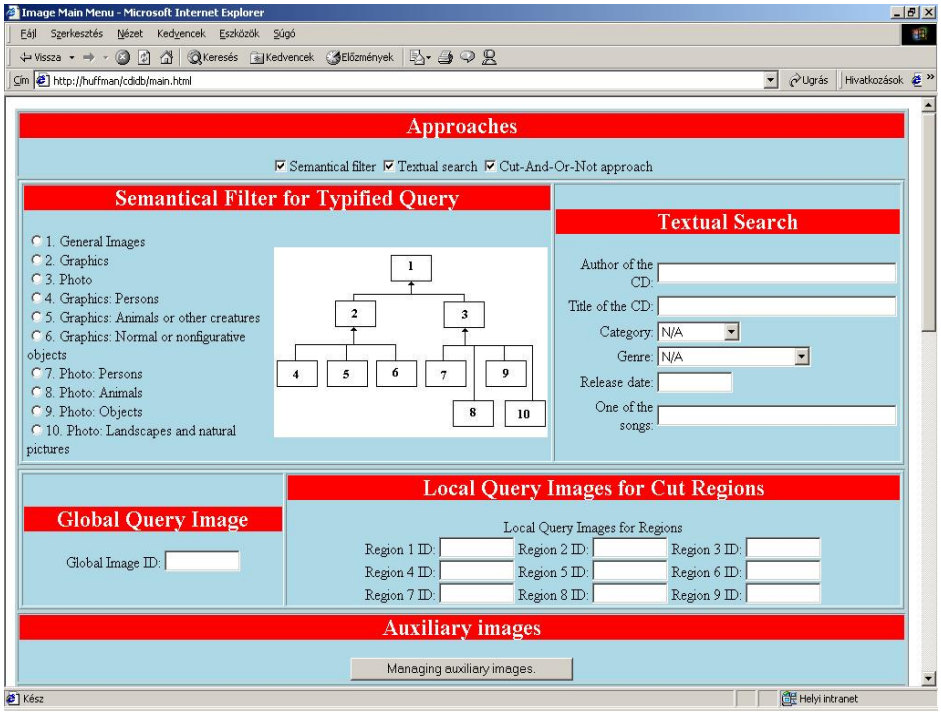
13.2.2. A szöveges keresésről

Mivel elsődleges szempont a képek kinyerése, nem felejtkezhetünk el a szöveges információkon alapuló képkeresésekről sem. Főleg ha meggondoljuk, hogy mint előszűrő feunkció, ez is nagyságrendekkel csökkentheti a végrehajtandó képillesztések számát (például ha tudjuk, az előadó neve D-vel kezdődött, vagy csak a klasszikus CD-k között keresünk). Az rendszer lehetővé teszi, hogy az alábbi információkra keressünk rá:

- Author of the CD: az előadó, illetve szerző neve.
- Title of the CD: az album címe.
- Category: a CD kategóriája (soundtrack, misc, stb.).
- Genre: a CD hanganyagának zenei stílusa (rock, swing, stb.).
- Release date: a CD kiadásának éve.
- One of the songs: megadhatjuk egy szám címét, vagy annak részletét.

Természetesen az SQL LIKE keresésének köszönhetően részsztringeket is megadhatunk a % és ? joker karakterek segítségével, ahol a % jelenti a tetszőleges számú karaktert, míg a ? a pontosan egy karaktert.

Az interfészen ezek a mezők kitöltendő szövegmezőkként jelennek meg (ld. 13.2. ábra). Amely mezőket nem töltjük ki, azok nem vesznek részt a keresésben. A kitöltött mezők között AND kapcsolat realizálódik.



13.2. ábra. Az interfész teteje.

13.2.3. A Cut-And-Or-Not keresésről

Nos, a képadatbázisból történő kinyerésnél azért a legfontosabb lépés a tartalomalapú képkinyerés. Ekkor az első részben leírtak szerint meg kell határozni valamilyen kereső képet, valamilyen kritériumot, majd a kinyert tulajdonságvektorokon elvégezni az illesztést. A rendszerben az Oracle *inter*Media illesztő algoritmusait használtam fel. Ezek az alábbiak:

- Q_{1,N_1} Color: a színek eloszlását vizsgálja a képen. Azt vizsgálja, milyen színek találhatók, de azt nem, azok hol találhatók.
- Q_{2,N_2} Shape: a képen található azonos színű foltok, objektumok alakjait vizsgálja, tekintet nélkül azok elhelyezkedésére.
- Q_{3,N_3} Texture: a képen lévő textúrák illeszkedését vizsgálja, itt sincs figyelemmel azok elhelyezkedésére.
- Q_{4,N_4} Location: ez egy önmagában nem használható illesztés. Más illesztéssel használva az ott vizsgált tulajdonságok (szín, alak, textúra) elhelyezkedését is figyelembe veszi.

A tulajdonságvektorok kinyerése előtt egy szegmentáció hajtódik végre a képen. Sajnos a szegmentált kép nem elérhető (ez valószínűleg a bonyolult szegmentációs algoritmus védelme érdekében van így).

Egy adott illesztés esetén az Oracle kiszámolja minden egyes tulajdonságvektor távolságát a fenti négy illesztés szerint. A távolság az Oracle-ben egy 0 és 100 közötti valós szám. (Tehát nincs másról szó, mint arról, hogy az Oracle által használt tulajdonságvektorok tere is rendelkezik határral, azaz a (9.7)-nak megfelelően a maximális norma esetünkben $N_1 = N_2 = N_3 = N_4 = 100$). Két kép esetén a képek totális távolsága a négy tulajdonság szerinti illesztési távolság súlyozott összegéből áll elő, azaz meg kell adnunk négy, 0 és 1 közé eső valós súlyt (w_1, w_2, w_3, w_4), melyeket az Oracle benormál úgy, hogy azok összege 1 legyen (legyenek a normált súlyok w'_1, w'_2, w'_3, w'_4). Ezután a teljes távolság a teljes_távolság$=w'_1 Q_{1,N_1} + w'_2 Q_{2,N_2} + w'_3 Q_{3,N_3} + w'_4 Q_{4,N_4}$ képlettel adódik. Megjegyzem, hogy ebből hasonlósági mértéket (pontszámot) (azaz 0 ha nincs illeszkedés, 1 ha identikusak a képek) a

$$(100 - \text{teljes\_távolság})/100 \tag{13.1}$$

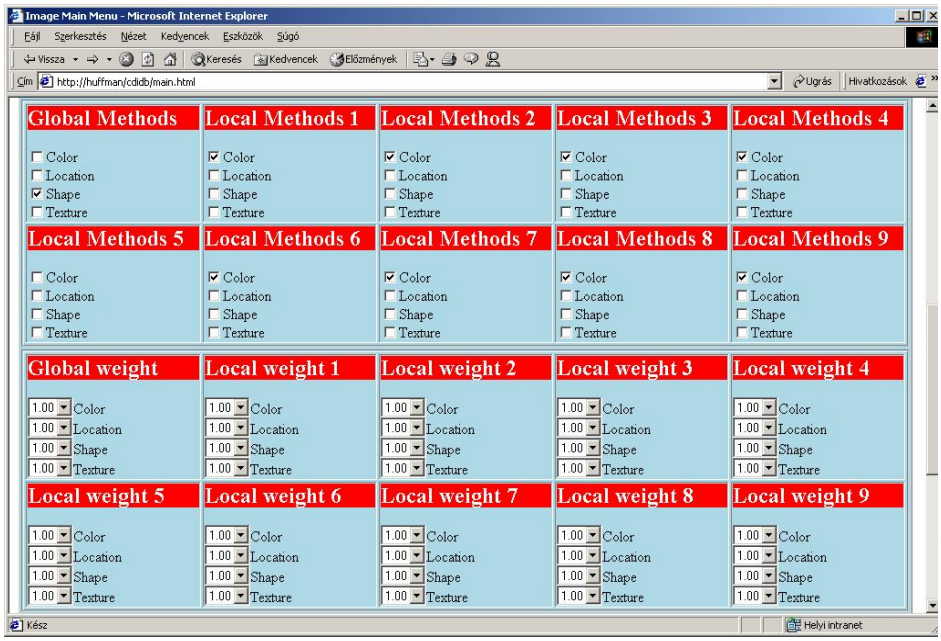
képlettel kapunk.

A Cut-And-Or-Not módszer egyik kulcsfontosságú lépése a vágás. Viszont azt be kell látni, hogy például 160 kép esetén, ha a kereső képet négy részre vágom, akkor az $4 + 160 \cdot 4 = 644$ vágást jelent, mely a keresés sebességét nagyon lelassítja. Ezért érdemes a képeket a konkrét alkalmazás igényei esetén előre feldarabolni (például rendszámtáblák esetén a betűk mentén, stb.). A gyakorlatban egy CD-borító esetén megfigyelhetjük, hogy gyakori az, hogy a fő motívum, vagy a teljes borítón helyezkedik el, vagy annak közepén. Ez utóbbi eset implikálja, hogy a Cut-And-Or-Not technika esetén érdemes a képet kilenc azonos méretű részképre vágni. Ilyenkor tehát egy kép illesztése esetén illeszthetünk a teljes képre, illetve annak kilenc részképére. Tehát képenként $10 \cdot 4$ illesztés lehetséges az Oracle szerint, ami 40 darab súly megadását jelenti.

Az interfészen jelölőnégyzetek segítségével beállíthatjuk, mely illesztésre tartunk igényt, és azt milyen súllyal vegye figyelembe a rendszer (ld. 13.3. ábra). Az interfészen megadható súlyok 0 és 1 közé eső két tizedes pontossággal rendelkező valós számok. A 0 a nem szignifikáns, míg az 1 az extrém fontos illesztési kritériumot jelöli az adott tulajdonságokon.

Az illesztendő részképeket, illetve magát a globális, teljes képre illesztendő kereső képet, azaz a tíz képet az azonosítóikkal adhatjuk meg. Az azonosítókat leolvashatjuk egy külön ablakban, ahol a segédképeket meg is tekinthetjük (ld. 13.4. ábra). Az ablakban a segédképek 50×50 méretű indexképei láthatók, így azok letöltése nem időigényes. Az indexképekre klikkelve, szükség esetén, az adatbázisból letölthetők az adott képek teljes nagyságukban is. Az azonosítók beírására külön szövegdobozokban van lehetőség az interfész HTML lapon (ld. 13.2. ábra).

Ha minden részképre, és a globális képre is illesztettünk, akkor a végeredmény 10, 0 és 100 közé eső távolság lesz. Ezekből illesztési eredményt a Cut-And-Or-Not formalizációhoz, a már korábban említettek szerint, a (13.1) képlettel kapunk. Ezeket az interfészen még tovább súlyozhatjuk, azaz megadhatjuk, a 10 illesztési eredmény közül melyiket mennyire tartjuk fontosnak. Ez az interfészen a *Total Weight* címszó alatt adható meg. A Cut-And-Or-Not formalizáció tehát már ezeket a súlyozott értékeket használja fel. Előfordulhat olyan eset, hogy valamely részkép sokkal fontosabb, mint a többi. Az interfészen ezért a totális súlyok 0 és 2 közé eső, két tizedes pontosságú valós számok. Mivel ilyenkor, a 9.3. fejezetben leírtak szerint előfordulhat, hogy valamely besúlyozott illesztési érték nagyobb, mint 1, a fuzzy konjunkció, diszjunkció



13.3. ábra. Az interfész közepe.

és negáció a 12, 13 és 14 definíciók szerint adódik.

A fuzzy logikai formula, ha a tíz besúlyozott illesztési eredmény rendre $Q0, \ldots, Q9$, akkor a

$$\begin{aligned} &\Delta_0 Q0 \nabla_1 \Delta_1 Q1 \nabla_2 \Delta_2 Q2 \nabla_3 \Delta_3 Q3 \nabla_4 \Delta_4 Q4 \\ &\nabla_5 \Delta_5 Q5 \nabla_6 \Delta_6 Q6 \nabla_7 \Delta_7 Q7 \nabla_8 \Delta_8 Q8 \nabla_9 \Delta_9 Q9 \end{aligned} \tag{13.2}$$

képlettel adódik, ahol a Δ_i, $i = 1, \ldots, 9$ vagy $\neg$, vagy üres, és a ∇_j, $j = 1, \ldots, 9$ pedig $\wedge$ vagy $\vee$.

Ha valamely részképre nem történt illesztés, akkor azt 0 súllyal és az őt megelőző pozíción fuzzy VAGY ($\vee$) összekötő jellel kell a szekvenciába befűzni.

Az interfész tetjén beállíthatjuk (ld. 13.2. ábra), milyen megközelítési elvet akarunk használni (szemantikus szűrés, szöveg alapú szűrés/keresés, Cut-And-Or-Not megközelítés). Ha a Cut-And-Or-Not megközelítés mellett bármely másikat is kiválasztjuk, az (vagy azok) mindig hamarabb futnak le, mint maga a képillesztés, ezáltal csökkentve a teljes keresés sebességét.

Az interfész utolsó pontjában beállíthatjuk (ld. 13.5. ábra), mely eredmények kerüljenek az outputra. Cut-And-Or-Not megközelítés esetén beállíthatunk egy küszöbértéket a (9.9)-hez hasonlóan. A különbség itt annyi, hogy (mivel a Cut-And-Or-Not eredménye egy hasonlósági pontszám, nem egy távolság) esetünkben a küszöbérték feletti pontszámmal rendelkező képek kerülnek az outputra. Ha nem alkalmazunk képillesztést, vagy nem a pontszámok szerint akarjuk az outputot megadni, megadhatjuk, maximálisan hány eredmény jelenjen meg (ha van pontszám, akkor azok csökkenő pontszámértékkel fognak megjelenni).

13.4. ábra. Az segédképek listája.

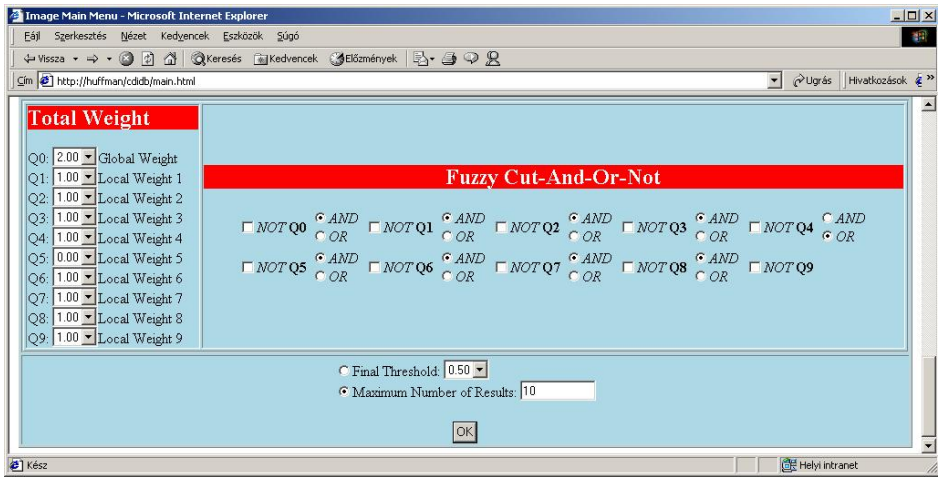
13.3. Implementációs technikák

Először tekintsük át a tárolási szerkezeteket és az alkalmazott táblákat. A háttérben megnyugvó adatbázis szerkezete a 13.6 ábrán bemutatott Bachman diagramon látható. A CDINFO tábla szerkezete az alábbi:

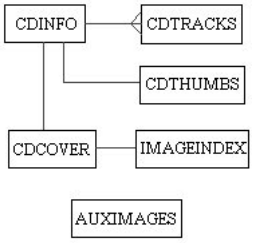
```
CREATE TABLE cdinfo (
cdid NUMBER PRIMARY KEY NOT NULL,
author VARCHAR2(90),
title VARCHAR2(90),
category VARCHAR2(30),
genre VARCHAR2(30),
releasedate VARCHAR2(4) );
```

Észrevehető, hogy ez tartalmazza az általános szöveges információkat, úgymint a CD előadója (szerzője), az album címe, a CD kategóriája (filmzene, egyéb, stb.), illetve a zenei stílus valamint a kiadás éve. Ehhez a táblához kapcsolódik az elsődleges kulcson keresztül 1:N kapcsolattal a számokat tartalmazó CDTRACKS tábla:

```
CREATE TABLE CDTRACKS (
trackid NUMBER,
```



13.5. ábra. Az interfész alja.



13.6. ábra. Az adatbázis séma.

```
cdid NUMBER REFERENCES cdinfo(cdid),
tracktitle VARCHAR2(90),
tracktime VARCHAR2(6),
PRIMARY KEY ( trackid, cdid ) );
```

Ez a tábla tartalmazza a számok címeit, illetve azok hosszát (perc:másodperc alakban). A borítók képei egy külön táblában, a CDCOVER táblában vannak letárolva, mely 1:1 kapcsolattal kapcsolódik a CDINFO táblához (a későbbi logikai kapcsolatok megtartása érdekében a CDCOVER tábla saját kulccsal rendelkezik, hogy a tulajdonságvektorok már őhozzá kapcsolódhassanak, ne a CDINFO táblához.) Ez a tábla tartalmazza, a borító képén túl, a Cut-And-Or-Not megközelítéshez szükséges előre elkészített részképeket. Ezek technikai okokból kerültek tárolásra, ugyanis a részképek kivágása sokkal időigényesebb feladat, minthogy minden illesztéskor végrehajthatók legyenek, így célszerűbbnek látszott a vágást csak egyszer végrehajtani, és a részképeket letárolni.

```
CREATE TABLE cdcover (
```

```
coverid NUMBER PRIMARY KEY,
cdid NUMBER REFERENCES cdinfo(cdid),
cdcoverimg ORDSYS.ORDImage,
cdcoverimg1 ORDSYS.ORDImage,
cdcoverimg2 ORDSYS.ORDImage,
cdcoverimg3 ORDSYS.ORDImage,
cdcoverimg4 ORDSYS.ORDImage,
cdcoverimg5 ORDSYS.ORDImage,
cdcoverimg6 ORDSYS.ORDImage,
cdcoverimg7 ORDSYS.ORDImage,
cdcoverimg8 ORDSYS.ORDImage,
cdcoverimg9 ORDSYS.ORDImage );
```

Mivel az illesztés megköveteli, hogy mind a kereső kép, mind az illesztett kép adatbázisban legyen tárolva, létezik egy AUXIMAGES tábla, mely a kereső képeket, azok indexképeit és a szükséges tulajdonságvektorokat tartalmazza. Létezik még egy CDTHUMBS tábla, mely a CD-borítók kicsinyített változatait, indexképeit tárolja, ugyanis maguk a borítók olyan nagy méretűek, hogy azokat csak indokolt esetben érdemes a hálózaton keresztül letölteni.

```
CREATE TABLE auximages (
auximgid NUMBER PRIMARY KEY,
auximg ORDSYS.ORDImage,
auxthumb ORDSYS.ORDImage,
auximgsig ORDSYS.ORDImageSignature );

CREATE TABLE cdthumbs (
cdid NUMBER PRIMARY KEY REFERENCES cdinfo(cdid),
cdthumb ORDSYS.ORDImage );
```

A képek szemantikus indexelése és a tulajdonságvektorok tárolása az IMAGEINDEX táblában történik. Ez a tábla a tíz tulajdonságvektoron kívül tartalmaz még tíz igaz/hamis jelentéssel funkcionáló oszlopot, mely az osztályhierarchiában betöltött szerepet jelenti. Ezek karakteres oszlopok, és 'Y' érték szerepel ott, ahol az adott kép szerepelhet az oszloppal azonosított típusú objektumok helyén, és 'N', ahol nem. Az oszlopok az alábbi osztályoknak felelnek meg:

- IMG: General Images.
- GRAPH: Graphics.
- PHOTO: Photo.
- GRAPHPERSON: Graphics: Persons.
- GRAPHANIMAL: Graphics: Animals or other creatures.
- GRAPHOBJECT: Graphics: Normal or nonfigurative objects.
- PHOTOPERSON: Photo: Persons.
- PHOTOANIMAL: Photo: Animals.

- PHOTOOBJECT: Photo: Objects.
- PHOTOLANDSCAPE: Photo: Landscapes and natural pictures.

Az 'Y' és 'N' értékekkel történő kitöltés az osztályok közti öröklődést is figyelembe veszi, azaz például az IMG oszlop (azaz az általános kép osztály, a gyökér) az minden sorban 'Y' értékkel rendelkezik.

```
CREATE TABLE imageindex (
coverid NUMBER REFERENCES cdcover(coverid),
cdcoverimgsig ORDSYS.ORDImageSignature,
cdcoverimgsig1 ORDSYS.ORDImageSignature,
cdcoverimgsig2 ORDSYS.ORDImageSignature,
cdcoverimgsig3 ORDSYS.ORDImageSignature,
cdcoverimgsig4 ORDSYS.ORDImageSignature,
cdcoverimgsig5 ORDSYS.ORDImageSignature,
cdcoverimgsig6 ORDSYS.ORDImageSignature,
cdcoverimgsig7 ORDSYS.ORDImageSignature,
cdcoverimgsig8 ORDSYS.ORDImageSignature,
cdcoverimgsig9 ORDSYS.ORDImageSignature,
img CHAR,
graph CHAR,
photo CHAR,
graphperson CHAR,
graphanimal CHAR,
graphobject CHAR,
photoperson CHAR,
photoanimal CHAR,
photoobject CHAR,
photolandscape CHAR );
```

Minden egyes osztályt reprezentáló oszlopra fel van építve egy bitmap index (lásd [39]), mely az ilyen típusú adatok indexeléséhez tökéletes. A szignatúrák, tulajdonságvektorok indexelése, egy az Oracle által védett adatpartíciós technikával történik (lásd 5.2. fejezet). Az indexek az alábbi SQL utasításokhoz hasonló utasítások segítségével hozhatók létre (helytakarékossági okokból nem mutatom be az összes index létrehozását célzó utasítássort, csak példákat hozok bitmap- és adatpartíciós indexelésre):

```
CREATE bitmap INDEX cdimgindex ON imageindex (coverid,img);
CREATE INDEX imgindexglobal ON imageindex(cdcoverimgsig)
  INDEXTYPE IS ORDSYS.ORDImageIndex;
```

Az ORDSYS tulajdonában lévő ORDImage és ORDImageSignature objektumok az *inter*Media részét képezik, és a képek valamint azok tulajdonságvektorainak tárolását hivatottak ellátni. Bővebben a [41]-ben olvashatunk róluk.

Az Oracle jól szervezett *inter*Media moduljának, valamint a PL/SQL nyelv lehetőségeinek köszönhetően a rendszer aránylag kevés kódolással létrehozható volt, összességében mintegy 5000 sornyi PL/SQL, PL/SQL Server Pages és HTML kódot tartalmaz. Ebből a legfontosabb az illesztéseket levezénylő tárolt eljárás, mely PL/SQL-ben íródott, és mintegy 1000 soros.

14. fejezet

Példák

Mielőtt a részletes példákra rátérnék, szeretnék egy-két számszerű adatot közölni az elkészült rendszerről. A rendszer 163 CD-ről tartalmaz információkat. A CD-borítók nyers, feldolgozatlan képei megközelítőleg összességében 1 GByte méretű tárterületet igényeltek. A képek átlagosan 1420 × 1420 pixel méretűek. A képeken méretváltoztatás nem történt, mindössze egy tömörítés lett rajtuk végrehajtva. Mivel a rendszer a képek felhasználását tekintve nem kritikus (orvosi, rendészeti, stb.) rendszer, így egy 90%-os minőség-megőrzésű JPG konverzióval oldottam meg a tömörítést. A CD-borítók jellegéből adódóan (gyakoriak a nagy kiterjedésű szegmensek, azonos színű foltok), drasztikus méretcsökkenést sikerült elérni, a 163 CD-borító összmérete megközelítőleg 200 MByte.

A Cut-And-Or-Not megközelítés érdekében a képek beszúrásakor azok kilenc azonos méretű részképre történő vágása is végrehajtódik. A kilenc részkép a tulajdonságvektorok kinyerése érdekében szintén tárolásra került, így az adatbázis megközelítőleg 400 MByte képet tárol a CD-kről. Ehhez hozzájönnek még a CD-k kicsinyített indexképei, ugyanis a találatok kiíratásakor a nagy hálózati forgalom elkerülése érdekében csak azok jelennek meg. A valódi CD-borítók csak külön kérésre töltődnek le. Az indexképek 200 × 200 pixel méretűek, átlag 100 KByte fizikai mérettel.

A CD-ről tárolt szöveges információk mérete Byte-okban mérve nem jelentős, viszont az nem elhanyagolható, hogy egy 5000 soros szöveges állományból kinyerve a szükséges információkat, az adatbázis képekkel és szöveges információkkal történő feltöltése csak egy 26000 soros INSERT SQL utasításokat tároló állománnyal volt lehetséges.

Segédképekből a dolgozat írása idején 354 db volt tárolva, melyek nyers mérete megközelítőleg 400 MByte volt. Természetesen itt is egy konverziót hajtottam végre a tömörítés érdekében, így a tömörített méretük megközelítőleg 100 MByte lett. A képek mérete változó, a 100×100 pixel mérettől az 1600×1600 pixel méretig terjednek. Ezekről a képekről is készültek indexképek 50 × 50-es, 1-2 KByte-os mérettel.

Minden képből kinyerésre kerültek a tulajdonságvektorok (szignatúrák), melyek az Oracle szerint 3-4 KByte méretűek. Tehát a 163 kép, a $163 \cdot 9$ részkép, és a 354 segédkép tulajdonságvektorainak mérete megközelítőleg 7-8 MByte (kb. 2000 vektor). Ez már elég nagy ahhoz, hogy indexet építsünk rájuk. Az adatbázis így tartalmaz húsz indexet is, melyekből 10 indexeli a képeket, és 10 a szemantikus indexelés bejegyzéseit.

Ha mindent összevetünk, akkor láthatjuk, hogy az adatbázis teljes mérete megközelítőleg 700 MByte, és összesen 1984 képet tárol valamilyen formában. Ezek mellett, ha hozzávesszük, hogy a rendszer – mint ahogy azt már fentebb említettem – egy elég szerény képességű Celeron 633-as gépen futott 192 MByte RAM-mal, a kapott eredmények minden tekintetben kiválónak tekinthetők.

Meg kell jegyeznem, hogy a felhasznált segédképek mind az Internetről lettek letöltve. Minden kép vagy teljes mértékben szabadon felhasználható, vagy a forrás megjelölésével szabadon felhasználható. Ezen utóbbi képek forrásai az alábbiak voltak:

- Űrkutatási képek, NSSDC Image Catalog, NASA, `http://nssdc.gsfc.nasa.gov/imgcat`
- Római vonatkozású képek, VRoma Image Archive, `http://www.vroma.org/images/image_search.html`
- Textúrák, Mayang's Free Textures v8.1, `http://www.mayang.com/textures`

14.1. Szöveges keresések

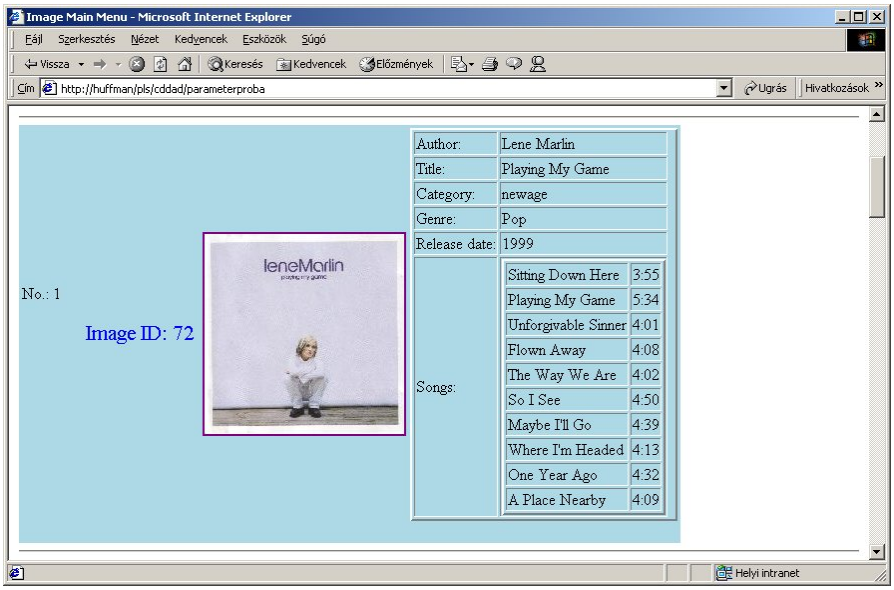
Elsőként nézzünk két rövid példát a szöveges keresésekre.

1. példa

Tegyük fel, hogy *Lene Marlin* CD-it keressük. Mivel tudjuk az előadót, egyszerűen kitöltjük a megfelelő mezőket. Az interfész tetején a megközelítési módokat tartalmazó jelölőnégyzeteknél csak a *Textual search* mezőt jelöljük be, majd az *Author of the CD* mezőbe beírjuk, hogy *Lene Marlin*. Más dolgunk már nincs, csak megadni, hány találatot jelezzen ki a kereső, azaz kiválasztani a *Maximum number of results* rádiógombot az interfész alján, és mondjuk megadni, hogy az első 10 találatot szeretnénk látni. Az eredményablak például a 14.1. ábrán látható lesz. Jól látható az első oszlopban, hogy hanyadik találat van kijelezve, a második oszlopban kékkel szedve az eredmény azonosítója, majd a CD borító kicsinyített képe. Ha erre a képre kattintunk, akkor letöltődik a valódi borító teljes méretében. Az utolsó oszlopokban a CD szöveges adatai láthatók. A keresés sebessége 0.05 másodperc volt.

2. példa

A második példában azon 2000-es kiadású CD-ket keressük, melyek valamelyik dalcíme tartalmazza a *Love* szót. Itt is csak a *Textual search* jelölőnégyzetet kell beixelnünk, majd megadni a *Release date* mezőt, illetve a *One of the songs* mezőbe a *%Love%* sztringet beírni. Mivel itt már egy tábla-összekapcsolás is lezajlik, hiszen a dalszövegek egy másik táblában vannak, illetve az SQL LIKE keresési módszere a % esetében egy kicsit időigényesebb (2387 dalcím van tárolva), a keresés 0.8 másodpercet vett igénybe. A visszaadott CD-k a *Bad Religion New America* albuma az *I Love My Computer* című dala miatt, a *The Corrs In Blue* albuma az *All The Love In The World* című szám miatt, és az *Iron Maiden* zenekar *Brave New World* CD-je a *The Thin Line Between Love And Hate* című számmal.



14.1. ábra. Az 1. példa keresésének eredménye.

14.2. A szemantikus index használata

A szemantikus index használata a szöveges kereséshez hasonlóan használható önmagában is, de leginkább a keresendő képek csoportjának szűkítésére használható.

3. példa

Tegyük fel, hogy keressük az *R.E.M.* zenekar összes tárolt lemezét. Ekkor a szöveges keresés esetén az *R.E.M.* sztringet szerzőként megadva, a *Maximum number of results* mezőbe a tárolt CD-k számát írva eredményként 12 CD adatait kapjuk meg, a keresés pedig 0.12 másodpercig tartott. Ha tudjuk, hogy a CD borítóján egy stilizált tigris-szerű szörny látszik, akkor beixeljük a *Semantical filter* opciót is, és a hierarchiában megadjuk, hogy az ábrázolt motívum az *5. Graphics: Animals or other creatures* osztályba sorolandó. Ekkor már nem kapjuk meg mind a 12 *R.E.M.* CD-t eredményként csak a *Monster* címűt, mert csak az az egy ábrázol grafikus eszközökkel megrajzolt állatot vagy egyéb teremtményt. A keresés sebessége esetünkben nem csökkent jelentősen, szintén a 0.1 másodperc körül mozog.

14.3. Cut-And-Or-Not keresések

A Cut-And-Or-Not megközelítésről már többször volt szó, ebben és a következő fejezetben néhány példán keresztül bemutatom, hogyan is lehet kamatoztatni a megközelítésben rejlő képességeket.

4. példa

Tegyük fel, hogy a *Bad Religion* zenekar *The Gray Race* című albumát keressük. És tegyük fel azt is, hogy pontosan ezeket az információkat nem ismerjük. Viszont azt tudjuk, hogy kilenc fekete-fehér (pontosabban szürkeskálás) portré volt az elején. Mi sem egyszerűbb ennél, mondhatni, hiszen nincs más dolgunk, mint a segédképekből kiválasztani egy szürkeskálás portrét, és azt keresni a CD-borító mind a kilenc régiójában (ld. 14.2. ábra). Az emberi fejen a színek, és azok helyei nagyon jellemzőek

14.2. ábra. A 4. példa kereső képe.

(sötétebb általában a haj, a szemgödrök, a száj, világosabbak az arcok és az orrok), tehát érdemes minden régióban a *Color* és a *Location* keresési módszereket alkalmazni. Ha ezeket egyforma súlyozással vesszük igénybe, akkor már csak annyi dolgunk van, hogy a globális súlyt 0-ra vesszük (hiszen csak a kilenc régióra keresünk, nem az egész képre), és a

$$Q0 \vee Q1 \wedge Q2 \wedge Q3 \wedge Q4 \wedge Q5 \wedge Q6 \wedge Q7 \wedge Q8 \wedge Q9$$

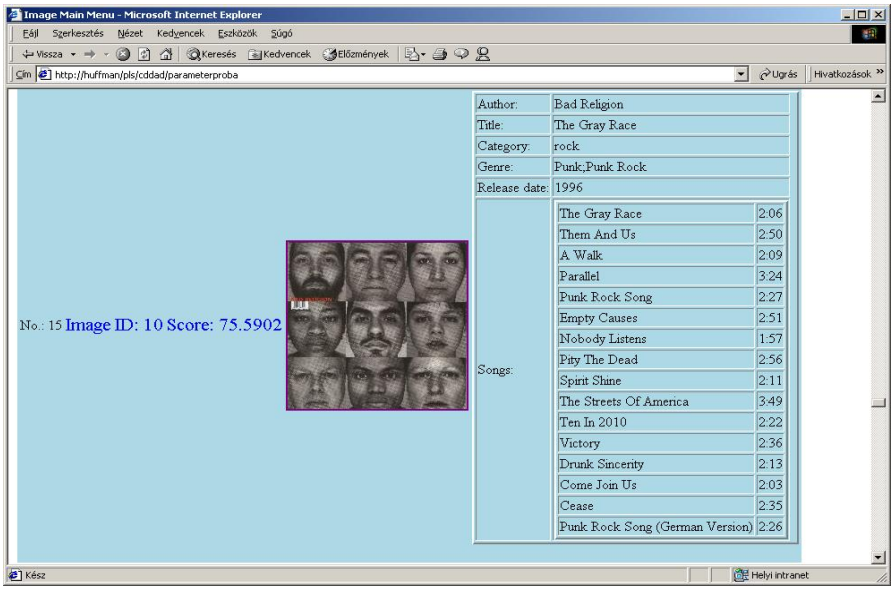
szekvenciát beállítani a fuzzy Cut-And-Or-Not keresőnek, hiszen minden régió fontos nekünk. Tegyük fel, hogy csak a 0.60 feletti eredménnyel rendelkező képekre vagyunk kíváncsiak. A keresés eredménye a 14.3. ábrán látható. Sajnálatos tény viszont az, hogy csak a 15.-ként találta meg a kereső a képet 75.5902-es pontszámmal, azaz 0.755902-es hasonlóság értékkel, amely bőven a 0.60 felett van. Az őt megelőző 14 kép 0.75 és 0.80 közötti hasonlóság értékkel rendelkezik, és mind szürke színű grafikákat tartalmaznak. A keresés 163 képet érintett, és 13.88 másodpercig tartott. Bár látjuk, jól dolgozik a kereső, érdemes leszűkíteni a keresendő képek halmazát, ezáltal pontosítva az eredményeket is, és gyorsítva magát a keresést is.

A következő fejezetben folytatom a Cut-And-Or-Not megközelítés lehetőségeinek bemutatását, és mellette megmutatom, hogyan gyorsítja, illetve pontosítja a szemantikus és a szöveges előszűrés a kereséseket.

14.4. Vegyes keresések, a keresések gyorsítása

5. példa

Maradjunk még egy picit ennél a *Bad Religion* albumnál. Mivel tudjuk, hogy emberfejeket, portrékat ábrázol, miért ne használnánk ki ezt a fontos információt. Jelöljük be az interfész tetején, hogy a szemantikus indexelést is szeretnénk használni, majd



14.3. ábra. A 4. példa keresésének eredménye.

adjuk meg, hogy az ábrázolt motívum a *7. Photo: Persons* osztályba tarozik. A többi beállított értéket viszont hagyjuk változatlanul. Nos, nem meglepő, hogy ebben az esetben a keresés már nem érinti az összes, 163 tárolt képet, csak 60-at. Ennek köszönhetően már sorrendben a 7. a keresett CD-borító, és a keresés sebessége is nagyságrendekkel csökkent, ugyanis csak 6.2 másodpercig tartott. Ez 50%-os sebességnövekedést jelent esetünkben, és mindez a szemantikus indexelésnek köszönhető. Ha emellett még azt is tudjuk, hogy az előadó neve B betűvel kezdődik (azaz a szöveges kereső *Author of the CD* mezőjének értéke *B%*), akkor a keresés már csak 5 CD-t érint, és azok közül is az első a keresett CD. A keresés sebessége pedig 0.91 másodperc.

6. példa

Az előző példában láthattuk, hogy hogyan lehet a részképekre keresési feltételeket megadni. Természetesen a megközelítési mód megengedi, hogy globálisan, a teljes képre kiterjedően adjunk meg keresési feltételt. Például tegyük fel, hogy láttunk egy CD-t az ismerősöknél, de nem tudjuk ki az előadó. Mindössze arra emlékszünk, hogy egy vörös háttérben álló szarvas volt az elején. Ez egy eléggé tipikus kép. Naplemente, bőgő szarvas, nem is látszik az állat, csak a sziluettje. Ilyet könnyen találhatunk. A segédképeink között is akad ilyen. A szarvas sziluettje szinte adja az ötletet, hogy próbáljunk a színek mellett az alakra is illeszteni. Mivel csak a globális képre illesztünk, csak annak súlya legyen 1, a többié nulla, és a fuzzy formulánk a következőképpen alakul:

$$Q0 \vee Q1 \vee Q2 \vee Q3 \vee Q4 \vee Q5 \vee Q6 \vee Q7 \vee Q8 \vee Q9$$

hiszen ami 0 súllyal szerepel, az biztos hamis érték, és az fuzzy VAGY összekötővel beillesztve a formulába, annak igazságértékén nem változtat, így a kifejezés értéke csak a $Q0$-tól függ.

Nos, itt szépen belefutunk egy csapdába, ugyanis a kereső rendszer nem tudja, hogy szarvas van a képen. Csak azt, hogy valami ágas-bogas dolog, ami nagyrészt piros és fekete színeket tartalmaz. És hát megesik, hogy valami az adott kritériumok alapján jobban hasonlít a kereső képre, mint maga a keresett kép. A 14.4. képen látható a kereső kép, a keresett kép, és a megtalált kép. A megtalált *Metallica* al-

14.4. ábra. A 6. példa kereső képe, keresett képe és elsőként megtalált képe.

bum 0.828808-es hasonlósági pontszámot kapott, míg a keresett *Kosheen* album csak 0.701177-et. A keresés 3.76 másodpercig tartott. Ilyen esetekben is segíthet a szemantikus indexelés, hiszen pontosan tudjuk, mit ábrázol a borító. Jelöljük be hát a szemantikus keresést is, és adjuk meg, hogy a borítón ábrázolt kép a *8. Photo: Animals* osztályba tartozik. Ekkor a keresés elsőként visszaadott eredménye a keresett *Kosheen* zenekar *Resist* CD-je. A keresés sebessége itt is jelentősen lecsökkent, mindössze 0.31 másodperc. Ez 12-szeres gyorsulást jelent.

7. példa

Nézzük például azt az esetet, amikor a *Nirvana* zenekar *Nevermind* CD-jét keressük kép alapján. A képen egy gyermek úszik a vízben. Ilyen képet sem nehéz találni a segédképek között. Mivel a víz és az úszó gyermek színe elég fontos, másrészt a gyermek középen helyezkedik el, tehát a színek lokációja is fontos, a *Color* és a *Location* illesztéseket választjuk a globális képre. A 14.5. képen láthatjuk a kereső képet és a keresett *Nirvana* CD borítóját. A két kép akár a színeken alapuló illesztések mintapéldája is lehet, hiszen elsőre megtalálta a 163 kép közül 0.805217 hasonlósági pontszámmal. A keresés 4.85 másodpercet vett igénybe. Itt is, ha felhasználjuk a szemantikus indexelés előnyeit, és megadjuk, hogy a képen látható motívum a *7. Photo: Persons* osztályba tartozik, akkor a keresés már csak 2.01 másodpercig tart, tehát a keresés ideje a felére csökkent.

8. példa

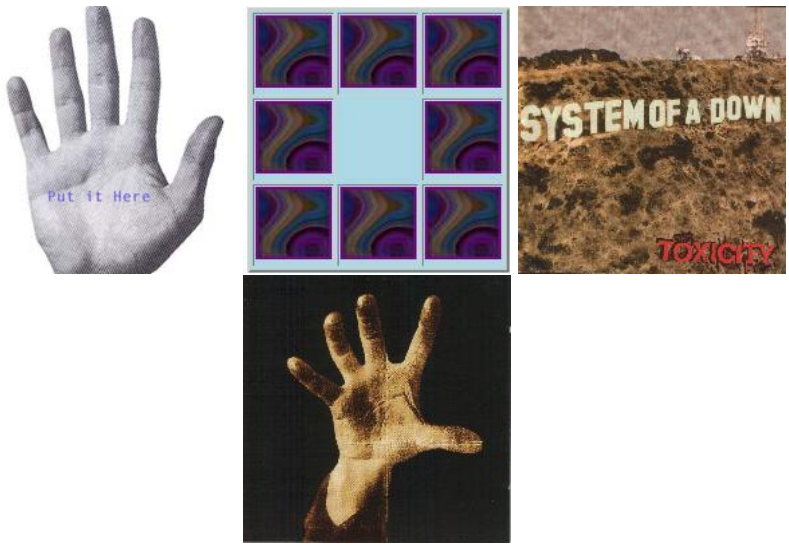
Az eddigiekben nem használtuk ki a Cut-And-Or-Not megközelítés tagadásban rejlő lehetőségeit. Hát most nézzünk akkor erre egy példát. Tegyük fel, hogy a *System of a Down* zenekar egyik CD-jét keressük, melyen sötét háttér előtt egy tenyér látszik.

14.5. ábra. A 7. példa kereső képe és keresett képe.

Tenyeret ábrázoló képünk van, de sajnos nem sötét a háttér. Sebaj, a tenyeret globálisan, alak illesztéssel keressük, hiszen a tenyérnek az alakja a fontos, lokálisan pedig szín alapján valamilyen sötét képet kell illesztenünk. Ezen túlmenően még azt is tudjuk, hogy szemantikusan a kép egy fotó, és hogy szövegesen a zenekar neve S betűvel kezdődik. Mivel középen a tenyér látható, ezért ott nem illesztünk, tehát a totális súlyoknál az 5. régió súlya 0. A színeket a totális súlyozásban 1-es súllyal vesszük figyelembe, míg a tenyér alakja nagyon fontos, azt 2-es súllyal érdemes vizsgálni. Maga a fuzzy formula az alábbi:

$$Q0 \wedge Q1 \wedge Q2 \wedge Q3 \wedge Q4 \vee Q5 \wedge Q6 \wedge Q7 \wedge Q8 \wedge Q9$$

A 14.6. ábrán látható a globális kereső kép, a lokális kereső kép, az elsőként megtalált kép és a keresett kép. Látható, hogy nem jártunk igazán nagy sikerrel (bár megem-



14.6. ábra. A 8. példa globális- és lokális kereső képe, az elsőként megtalált, illetve a keresett kép.

lítem, hogy másodikként a keresett CD-t is megtalálta a kereső.) Mit lehet ilyenkor csinálni? Hát, láthatjuk, hogy a megtalált CD jobb alsó sarkában piros felirat található. A keresett CD-n ilyen nincs. Tehát a segédképek közül berakunk a 9. régióba egy piros képet, marad a szín illesztés, csak a fuzzy formulát kell az alábbi módon megváltoztatni:

$$Q0 \wedge Q1 \wedge Q2 \wedge Q3 \wedge Q4 \vee Q5 \wedge Q6 \wedge Q7 \wedge Q8 \wedge \neg Q9$$

Ennek hatására a kereső elsőként adja vissza az általunk keresett képet. Maga a keresés 0.64 másodpercig tartott.

9. példa

Az előző példákban főként a színek illeszkedését vizsgáltuk. Most nézzünk egy olyan esetet, ahol a textúra és az alak kap nagy hangsúlyt. Biztos mindenki ismeri a *Pink Floyd The Wall* albumát. Nem meglepő, hogy egy téglafal látható rajta. Téglafalat ábrázoló textúrát könnyű beszerezni. Tehát a *Shape* és a *Texture* illesztéseket felhasználva, illetve azt, hogy az album egy rockzenei album (*Genre: Rock*), a 14.7. ábrán látható kereső kép és keresett kép 0.81449-es hasonlósági pontszámmal másodikként már megtalálásra kerül 2.76 másodperc alatt. Ez nem egy nagy idő, de még ez is csökkenthető, ha megadjuk, az ábrázolt motívum grafika. Ekkor a keresés már csak 1.82 másodperc. Ez körülbelül 35%-os sebességnövekedést jelent.

14.7. ábra. A 9. példa kereső képe és keresett képe.

15. fejezet

Prológus

15.1. A rendszer összefoglalása

Az elkészült rendszer egy QBE alapú rendszer (ld. 6.2. fejezet), mely ötvözi a minta képeken alapuló vizuális információkinyerést a szöveges keresésekkel. A keresések gyorsítása érdekében a tárolt képek az általuk ábrázolt motívumok szerint egy hierarchikus osztályozással osztályozva vannak, és az így kialakult osztályok fel vannak indexelve (ld. 8. fejezet). Ez a szemantikus, tartalom alapú indexelés nagyban meggyorsítja a kereséseket. A rendszer a tartalomalapú képkinyerési technikák közül a szín, elhelyezkedés, alak és textúra illeszkedéseket vizsgálja (ld. első rész). A keresések egy százfokú skálán paraméterezhetők, az eredmények egy kétszázfokú skálán súlyozhatók, a végső eredmény a fuzzy logikai összekötők segítségével egy logikai formulával adható meg. A rendszer a Cut-And-Or-Not megközelítésnek köszönhetően támogatja a részképeken alapuló illesztést és az összetett kérdések alkalmazását (ld. 9. fejezet).

A következő táblázatban az előző fejezetben bemutatott példákról adok egy összefoglalást.

Példa	Color	Location	Shape	Texture	Szöveges	Idő	Gyors idő
1.	-	-	-	-	+	0.05	-
2.	-	-	-	-	+	0.8	-
3.	-	-	-	-	+	0.12	0.1
4.	+	+	-	-	-	13.82	-
5./a	+	+	-	-	-	13.82	6.2
5./b	+	+	-	-	-	-	0.91
6.	+	-	+	-	-	3.76	0.31
7.	+	+	-	-	-	4.85	2.01
8.	+	-	+	-	+	-	0.64
9.	-	-	+	+	+	2.76	1.82

Az első oszlop a példa sorszáma. Utána az látható, hogy mely illesztés volt a példában alkalmazva. Az utolsó két oszlop közül az első a szemantikus indexelés

nélküli, míg a második a szemantikus indexeléssel együtt mért keresési időket mutatja másodpercekben.

Megfigyelhető, hogy a szöveges keresések átlagos ideje 0.32 másodperc. A képeken alapuló keresések átlagos ideje 6.29 másodperc. Ha használjuk a szemantikus indexelést, akkor a gyorsított keresések átlagos ideje 1.98 másodperc. Láthatóan a két érték között nagy a különbség, az átlagos lekérdezési idő a harmadára csökkenthető. Ha lebontjuk külön-külön a sebesség növekedéseket, az 5. példában a keresés ideje a felére, a 6. példában az 1/12-edére, a 7. és 9. példákban pedig szintén megközelítőleg a felére csökkent a keresés ideje.

Mindent összevetve a rendszer jó igazolása a [57] [58] [59] [60] és [61]-ben leírtaknak. Meg kell jegyeznem, hogy a szemantikus indexelést a rendszer csak relációs szinten alkalmazza, mivel az adatbázis-kezelő rendszer által alkalmazott illesztések miatt az [59]-ben kifejtett OO megközelítés illesztési- és vektor polimorfizmusa nem vált szükségessé, így azok nem képezik részét a rendszernek.

15.2. Összehasonlítás más rendszerekkel

Természetesen – bár az elkészült rendszer beváltotta a hozzá fűzött reményeket – összehasonlítottam más létező, a weben elérhető rendszerrel. Ez utóbbi szempont – mármint az Interneten történő elérhetőség – egy nagyon fontos szempont, hiszen a lokális gépeken futó, C/C++ nyelven implementált célszoftverek nem tartoznak szűkebb értelemben a bárki által elérhető, nyílt képadatbázisok közé, míg az általam implementált rendszer egyértelműen az ilyen rendszerek csoportjába tartozik. Az itt közölt összehasonlítás a [60]-ban megjelent összehasonlítás egy bővebb változata.

Az összehasonlítás során az alábbi rendszereket vizsgáltam:

- 1: Amore (NEC), Advanced Multimedia Oriented Retrieve Engine,
  `http://www.ccrl.com/amore/`
- 2: Blobworld,
  `http://elib.cs.berkeley.edu/photos/blobworld/start.html`
- 3: CIRES, Content-based Image REtrieval System,
  `http://amazon.ece.utexas.edu/~qasim/cires.htm`
- 4: NETRA,
  `http://maya.ece.ucsb.edu/Netra/netra.html`
- 5: SIMPLIcity, PennState University, Multimedia Information Technology Research Group,
  `http://jzw.stanford.edu/IMAGE/simp_java/`
- 6: PicToSeek (Zomax),
  `http://zomax.wins.uva.nl:5345/ret_user/`

Most nézzük sorra ezeket a rendszereket!

1: A rendszer általános rendszer weboldalakon található képek keresésére. A képeket szöveges információval kell ellátni. Ilyenek például az oldal címe, ahol a kép található, az oldal fejléce, a képet körülölelő bekezdés tartalma, stb. és ezen lehet

szöveges keresést végrehajtani. A rendszer lehetővé teszi, hogy a szöveges keresés mellett színeken, illetve alakokon alapuló keresést is végrehajtsunk. A keresőnek beállítható egy négyfokú skálán, mennyire fontos a szín és/vagy az alak illeszkedése. Az illesztéshez ki kell választani egy kategóriát, hogy milyen csoportba tartozó képeket keresünk (sport, utazás, művészet stb.), majd kiválasztani egy képet a tárolt képek közül (vagy megadni egy URL-t). A rendszer nem támogatja a részképeken alapuló keresést, ezáltal az összetett kérdéseket sem. Nem támogatja a textúra illesztést sem.

2: A tárolt képeket a rendszer itt is csoportokra osztja (állatok, növények, épületek, stb.). Miután kiválasztottunk egy csoportot, kiválaszthatunk egy kereső képet. Opcionálisan egy kulcsszót is megadhatunk. A kereső képen a rendszer egy nagymértékű szegmentációt hajt végre. Ezután kiválaszthatunk egy foltot, egy szegmenst, melyen megadhatjuk, hogy a kijelölt szegmens egy háromfokú skálán mennyire fontos a négy ismert illesztés szemszögéből (Color, Texture, Location, Shape). A szegmensen kívüli területről csak annyit adhatunk meg, fontos-e avagy sem. Ez már több illesztést biztosít, mint az előző rendszer, támogatja a részképen alapuló keresést, de nem támogatja az összetett kérdéseket.

3: Ebben a rendszerben szintén először egy csoportosításból kell választanunk, hogy milyen képeket akarunk illeszteni (állatok, objektumok, stb.). Ezután kiválaszthatunk egy kereső képet, majd megadhatjuk, mennyire fontos maga a csoportosítás, amiből választottunk (ha kicsi értéket adunk meg, akkor más csoportokat is figyelembe vesz az illesztéskor), illetve mennyire fontos a szín és a textúra az illesztéskor. A rendszer nem támogatja sem a részképeken alapuló illesztést, sem a szöveges keresést.

4: A rendszer szintén egy csoportválasztással kezdődik, melyben megadhatjuk milyen jellegű képeket illesztünk (tulipánok, jég, óceáni élet, stb.). Ezután kiválasztunk az adott csoportból egy kereső képet. A rendszer egy szegmentációt hajt végre a képen, és megadhatjuk, az adott szegmensek az ismert Color, Location, Texture és Shape illesztések szerint milyen fontosságúak. A rendszer biztosítja tehát a részképeken alapuló illesztést, de az összetett kérdéseket nem. Alkalmazható a kulcsszó alapú keresés is.

5: Ebben a rendszerben vagy fotókat vagy grafikákat kereshetünk. A rendszer biztosít egy rajzoló felületet, ahol felskiccelhetjük az ábránkat, illetve megadhatunk egy URL-t, ahol a kereső képünk található (vagy választhatunk a tárolt képekből). A vázlat alapú keresésnek köszönhetően a rendszer a színekre, az alakokra és azok elhelyezkedésére illeszt. Nem támogatja a részképeken alapuló illesztést, nincs kulcsszavas keresés, és a kereső algoritmusok sem paraméterezhetőek.

6: A rendszernek beállíthatjuk, hogy fotókat, grafikákat vagy arcokat akarunk illeszteni, majd megadhatunk egy kereső képet, és kiválaszthatjuk a használt algoritmust. Nem a szokványos értelemben vett algoritmusokat használja a rendszer, hanem a színek, a szín hisztogramok illesztésére ad több lehetőséget (binek illeszkedése, fuzzy illeszkedés, stb.). Nem támogatja a részképeken való illesztést, a szöveges illesztést, az eredmények nem súlyozhatók.

Mindent egybevetve azt láthatjuk, hogy a legtöbb rendszer nem biztosítja a legelterjedtebb négy fő tartalomalapú képkinyerési megközelítést, azaz a színek, az alakok, a textúrák és ezek elhelyezkedésének vizsgálatát, hanem ebből csak néhányat, leggyakrabban a szín illeszkedést nyújtja. A kulcsszó alapú szöveges keresés sem támogatott mindenütt. A szemantikus csoportosítás a legtöbb rendszernél megjelenik, bár kevés

az, ahol hiearchiába rendezhetők a csoportok, így ezek a rendszerek igazából nem tükrözik az ábrázolt motívumok közötti szemantikus alá-, illetve fölérendeltségi kapcsolatokat. A részképeken alapuló illesztések sem gyakoriak. Az összetett illesztések formalizációja (például logikai formulák segítségével) pedig egyáltalán nem elterjedt.

Az általam implementált rendszer főleg ez utóbbi területen ad előrelépést azzal, hogy lehetővé teszi a fuzzy logika segítségével az összetett kérdések alkalmazásának lehetőségét (kiemelve például azt a tényt, hogy negatív illeszkedést egyik rendszerben sem lehet vizsgálni, míg a Cut-And-Or-Not megközelítéssel ez triviális). Ezen kívül kiemelném még azt a tényt, hogy bár a szemantikus osztályozás megjelenik, nincs igazán kihasználva az, hogy a csoportok hierarchiába rendezhetők. Ez alól csak a vizsgált 3. rendszer a kivétel. Az általam implementált rendszer a képek tárolásához és visszakereséséhez egy objektumrelációs adatbázis-kezelő rendszert használ (Oracle9*i* ORDBMS), azaz más, adatbázist használó alkalmazásokkal szükségszerűen együtt tud működni, nagyobb adatbiztonságot tesz lehetővé, és biztosítja az alkalmazott illesztési algoritmusok bezárását, elrejtését is (a felhasználónak az egész adatbázist fel kellene törnie ahhoz, hogy az alkalmazott módszerekben hibát rejthessen el, vagy azokat megszerezze).

A következő táblázat az említett rendszerek vizsgált pontjairól ad tájékoztatást. A táblázatban 7. sorszámmal az általam implementált rendszer látható.

Rendsz.	Color	Location	Shape	Texture	Szöveg	Részkép	Param.
1.	+	+	-	-	+	-	+
2.	+	+	+	+	+	+	+
3.	+	+	-	-	-	-	+
4.	+	+	+	+	+	+	+
5.	+	+	+	-	-	-	-
6.	+	-	-	-	-	-	-
7.	+	+	+	+	+	+	+

Az első oszlopban a vizsgált rendszer sorszáma található, majd a négy ismert tartalomalapú képkinyerési paradigma. A hatodik oszlop a kulcsszó alapú keresést mutatja, a hetedik a részképeken alapuló keresést, míg az utolsó az algoritmusok paraméterezhetőségét, illetve az eredmények súlyozását jelenti.

Összefoglaló

A dolgozat három részből áll.

Az első részben a vizuális információkinyerés technikáiról adok irodalmi áttekintést, melyek három nagyobb csoportba oszthatók. Az első csoportba a tartalomalapú képkinyerési technikák sorolhatók, melyek a képek különféle tulajdonságain alapuló mintaillesztések matematikai modelljei. Ezek a modellek általában a képeken található színeket, alakokat, mintákat és azok egymáshoz viszonyított helyzetüket veszik alapul. A második, jóval kisebb csoportban az alkalmazott indexelési technikák találhatók, melyek a képek gyorsabb visszakereshetőségét szolgálják. A harmadik csoportba a keresőrendszerek interfészei kerültek. Ezek az interfészek a legtöbb esetben nemcsak a kereső kép megalkotására, és ezáltal a keresés elvégzésére alkalmasak, hanem segítségükkel a kérdések valamilyen formában formalizálhatók is.

A dolgozat második részében az első részben megemlített technikák három nagyobb csoportjához fűződő saját eredményeim szerepelnek. A képek indexelését illetően az információkinyerő rendszerek hierarchikus osztályozásának ötletét használtam fel az objektumorientáltság előnyeinek kihasználása mellett. Az adatbázisban tárolandó $\mathrm{obj}_i \in \mathrm{Obj}$ képeket egy $\boldsymbol{C}$ osztályhierarchiába kell szervezni, melyeknek C_i, $i = 1, \ldots, N$ osztályai tartalmazzák a képeket. Hogy valós osztályhierarchiánk legyen, a képeket mint objektumokat objektumorientált módon tartalmuk alapján modellezni kell. A modellezésnél viszont (a hierarchia megőrzése érdekében) előírjuk, hogy $\boldsymbol{C}$-nek csak egy gyökere lehet, azaz $\exists C_k \forall C_i$, $i, k \in \{1, \ldots, N\}$, $i \neq k$, $C_k < C_i$, és $\nexists C_j$, $C_j < C_k$, $j \in \{1, \ldots, N\}$ (jele C_0), és minden osztálynak csak egy közvetlen őse lehet, kivéve a gyökeret, azaz $\forall C_i \exists C_k$, $i, k \in \{1, \ldots, N\}$, $i \neq k$, $C_i \neq C_0$, $C_k \rightarrow C_i$, és $\forall C_l$, $C_l \rightarrow C_i$, $C_l = C_k$. Így $\boldsymbol{C}$ egy általános fa. Ezután a hierarchiában szereplő osztályok mindegyikéhez hozzárendelünk egy már ismert képindexelési technikát, és a hierarchiát e felé az indexsorozat felé tesszük, mint másodlagos indexet. Ezzel az OO indexelési technikával [58]-ban, és nagyobb mélységben [59]-ben, illetve [60]-ban foglalkozom.

A kérdésformalizációt illetően egy fuzzy alapú megközelítési módot ismertetek. A megközelítésnek egyik legnagyobb előnye az, hogy lehetővé teszi a részképeken alapuló illesztéseket, melyeket a legtöbb modell nem támogat. Másik nagy előnye az, hogy a részképeken történő illesztések eredményeit és a fuzzy logikák logikai összekötőjeleit felhasználva lehetővé teszi a komplex illesztési kérdések feltevését is. Ezzel a módszerrel tehát már olyan kérdést is feltehetünk, melyben az illesztési eredmények nemcsak *ÉS* szóval köthetők össze (teljesüljön A *ÉS* B illesztési kritérium), hanem azok vagylagos viszonyát is megvizsgálhatjuk (*VAGY* összekötő), vagy éppen az illeszkedés hiányát is kérhetjük (*NEM* összekötő). A megközelítési módnak ezért — mivel

támogatja a képek részképekre vágását, valamint az *ÉS*, *VAGY* és *NEM* összekötők használatát — a neve Cut-And-Or-Not megközelítési mód. A megközelítés jól kezeli azt, ha több különféle Q_{i,N_i} illesztésünk van. Mivel az illesztések a tulajdonságvektorok valamilyen módon vett normái, és a tulajdonságvektorok tere az adatbázisban pedig véges, így létezik maximális norma minden egyes Q_i illesztéshez (N_i).

A technika alapja, hogy adottak az adatbázisban $f_1, \ldots, f_n$ és $g_1, \ldots, g_m$ digitális képek, valamint $Q_{1,N_1}, \ldots, Q_{k,N_k}$ illesztések. Képezzük ezekből a képekből a kívánt $\mathcal{C}_{x_{1i},y_{1i},x_{2i},y_{2i}} f_i$ és $\mathcal{C}_{x_{1j},y_{1j},x_{2j},y_{2j}} g_j$ részképeket ahol $i = 0, \ldots, n$, $j = 0, \ldots, m$. Ezután végezzük el a megfelelő részképekkel az illesztéseket, így előáll p számú illesztés végrehajtása esetén p darab $Q_{l,N_l}(\mathcal{C}_{x_{1i},y_{1i},x_{2i},y_{2i}} f_i, \mathcal{C}_{x_{1j},y_{1j},x_{2j},y_{2j}} g_j)$ illesztés, ahol $l \in \{1, \ldots, k\}$, $i \in \{1, \ldots, n\}$ és $j \in \{1, \ldots, m\}$. Értékeljük ki ezeket az illesztéseket valamilyen fuzzy kiértékelés szerint, így előáll $q_1, \ldots q_p$ illesztési érték. Készítsünk ezekből a q_i, $i \in \{1, \ldots, p\}$ értékekből logikai formulát a fuzzy logikának megfelelő összekötőjelek segítségével, és értékeljük ki. Az illesztések kiértékelése például a $q = \frac{N - Q_N}{N} \in [0, 1]$ alakban is történhet. Az összekötőjelek súlyozatlan esetben a $q_1 \wedge q_2 = \max\{0, q_1 + q_2 - 1\}$, a $q_1 \vee q_2 = \min\{1, q_1 + q_2\}$ illetve a $\neg q = 1 - q$ alakban történhetnek, ahol q, q_1, q_2 kiértékelt illesztések. Ha valamilyen súlyozást is alkalmazunk, akkor a $w_1 q_1 \wedge w_2 q_2 = \min\{w_1 q_1, w_2 q_2\}$, a $w_1 q_1 \vee w_2 q_2 = \max\{w_1 q_1, w_2 q_2\}$ illetve a $\neg wq = \max\{0, 1 - wq\}$ általánosított alakokat érdemes használni, ahol w, w_1, w_2 valós súlyok. Ezekkel bővebben [58]-ban és [60]-ban foglalkozom.

Az illesztő algoritmusokat illetően [56]-ban a bináris illesztés általánosításának kérdéskörét jártam körbe. A bináris illesztés általában egy keresőrendszer esetén csak a legutolsó esetben alkalmazott módszer, hiszen a legtöbb esetben zajérzékeny és pontatlan. Jóval kifinomultabb illesztési módszert mutatok be az alakfelismerés témakörében a második részben. Az illesztés a képen található foltok (azonos színű területek, pacák) külső kontúrjainak alakját hasonlítja össze egy előre megadott (minta) folt kontúrjával. A kontúrok a 8-irányítású lánckódokkal kódolhatók. Két különböző technikát is ismertetek, az egyik a χ^2 teszten alapul, a másik pedig a sztochasztikus folyamatok elméletén.

Az első esetben adott egy $X \subset \mathbb{Z} \times \mathbb{Z}$ digitális halmaz, valamint két lánckódolt lyukakat nem tartalmazó bináris objektum, folt ($f_1(x, y) : Y \to \{0, 1\}$, $f_2(x, y) : Y \to \{0, 1\}$, $Y \subset X$), és l^{f_1} és l^{f_2} lánckódok. A kérdés az, hogy az $f_1(x, y)$ objektum nagyítás-, eltolás- és $k\frac{\pi}{4}$-szerint elforgatás invariánsan illeszkedik-e a $f_2(x, y)$ objektumra. A kódokból a differencia kódok és az alak kódok segítségével képezhetjük az $l'^{f_1} = \mathcal{S}^{-1}(\mathcal{S}(l^{f_1}))$ és $l'^{f_2} = \mathcal{S}^{-1}(\mathcal{S}(l^{f_2}))$ normalizált lánckódokat. Ezek után tekinthetjük ezt a két lánckódot úgy, mint két valószínűségi változót, és a lánckód elemeit pedig a hozzájuk tartozó mintáknak, azaz l'^{f_1}-hez rendeljük hozzá a ξ valószínűségi változót, melyhez tarozó m elemű minta a $(\xi_1, \ldots \xi_m)$, l'^{f_2}-hez pedig a η változót, amelyhez a $(\eta_1 \ldots \eta_n)$ minta tartozik, majd vizsgáljuk meg e két valószínűségi változó homogenitását χ^2-próbával, ugyanis ha a két lánckód csak nagyításban, elforgatásban, illetve eltolásban különbözik, akkor tekinthetjük őket úgy, mint egy populációból vett két eltérő elemszámú mintát. Mivel ezek a minták lánckódok, így a ξ_i és η_j értékek a $\{0 \ldots 7\}$ halmaz elemei. Jelölje az elemek multihalmazát Ξ. Osszuk ezt föl a ξ és a η értékeit nyolc halmazra úgy, hogy $A_k = \{x | x \in \Xi, x = k\}$, $k = 0 \ldots 7$, és legyen ν_k azon ξ_i mintaelemek száma, melyre $\xi_i \in A_k$, és μ_k azon η_i mintaelemek száma, melyre $\eta_i \in A_k$. Ezek után vezessük be a $\chi^2 = m \cdot n \sum_{k=0}^{7} \mathcal{R}_k$, próbastatiszti-

kát, ahol $\mathcal{R}_k = \begin{cases} \frac{(\frac{\nu_k}{m} - \frac{\mu_k}{n})^2}{\nu_k + \mu_k}, & \text{ha } \nu_k \neq \mu_k \neq 0. \\ 0, & \text{egyébként.} \end{cases}$ Erről a statisztikáról bebizonyítható, hogy megfelelően nagy m és n esetén, ha a minta lánckódja illeszkedik valamilyen módon az objektumunk lánckódjára, akkor a χ^2-próbastatisztika eloszlása megközelítőleg 7 szabadságfokú chi-négyzet eloszlás lesz. Rögzített szignifikanciaszinttel p-re a p értékek egy χ^2 táblából könnyen meghatározhatók.

A második eset, amikor sztochasztikus folyamatok realizációinak tekintjük a lánckódokat, két alesetre bontható. Az egyikben a modell nem tanítható, viszont jól kezeli a zajos eseteket, a másik esetben a modell tanítható, így a zajok lehetőségeit egy lánckód sorozattal taníthatjuk be a rendszernek. A nem tanítható esetben a lánckódot D előjeles differencia kóddá alakítva megadhatjuk a $P^{t,t+1}_{D_t,j}$ egylépéses átmenetvalószínűségeket. Ezután képezhetünk egy olyan $Y(t)$ folyamatot, melynek elemei az eredeti l' lánckód elemei ($Y(t) = l'_t$, $t = 1, \ldots, n$), melynek átmenetvalószínűség-mátrixának elemei $P^{t,t+1}_{q,r} = P(Y(t+1) = r | Y(t) = q)$ alakúak, ahol $\sum_{i=0}^{7} \sum_{j=0}^{7} P^{t,t+1}_{i,j} = 1$, rögzített t mellett, azaz minden egyes t időpillanatból 1 valószínűséggel lépünk át a $t+1$ időpillanatba. Mivel az l' lánckód és a $D(l')$ előjeles differencia kód kezdeti irányítástól eltekintve kölcsönösen egyértelműen megfeleltethetők egymásnak, így az egylépéses átmenetvalószínűségek is megfeleltethetők egymásnak. Így a számunkra fontos átmenetvektorok legyenek az alábbiak (a $P^{t,t+1}_{D_t,j}$ átmenetvalószínűségekből adódnak)

$$P^{t,t+1}_{Y(t),j} = \begin{cases} P^{t,t+1}_{D_t,i} & \text{ha létezik,} \\ 0 & \text{egyébként,} \end{cases}$$

ahol $i = \begin{cases} Y(t) - j & \text{ha } Y(t) - j > 0 \text{ és } Y(t) - j \in \{0,1,2,3\} \\ Y(t) - j - 8 & \text{ha } Y(t) - j > 0 \text{ és } Y(t) - j \notin \{0,1,2,3\} \\ Y(t) - j + 8 & \text{ha } Y(t) - j < 0 \text{ és } Y(t) - j + 8 \in \{0,1,2,3\} \\ Y(t) - j & \text{ha } Y(t) - j < 0 \text{ és } Y(t) - j + 8 \notin \{0,1,2,3\} \end{cases}$,
ahol $j \in \{0,1,2,3,4,5,6,7\}$, tehát ezek nyolc elemű vektorok. Belátható, hogy ezek kielégítik a $\sum_{j=0}^{7} P^{t,t+1}_{Y(t),j} = 1$ feltételt rögzített t mellett.

A tanítható esetben tekintsünk egy $l'^{(i)}$ normalizált lánckód sorozatot, ahol $i = 1, \ldots, N$. Jelöljük ki etalonnak $l'^{(1)}$-et, és igazítsuk hozzájuk a többi hosszát. Az $Y(t)$ folyamat átmenetvalószínűségeit az alábbi módon adjuk meg:

$$P^{t,t+1}_{q,r} = \sum_{i=1}^{N} \frac{C_{q,r}(l'^{(i)}_t, l'^{(i)}_{t+1})}{N}$$

ahol $C_{q,r}(l'^{(i)}_t, l'^{(i)}_{t+1}) = \begin{cases} 1, & \text{ha } l'^{(i)}_t = q, \text{ és } l'^{(i)}_{t+1} = r, \\ 0, & \text{egyébként.} \end{cases}$

A végeredményhez tegyük fel, hogy az $l'^{(i)}$ sorozat minden egyes tagjának hossza végtelen. Ekkor a k' végtelen hosszúságú lánckód konvergál az $l'^{(i)}$ lánckód sorozathoz, ha $\frac{1}{T} \sum_{t=0}^{T} \left(1 + P^{t,t+1}_{k'_t,k'_{t+1}} - \max_{i,j \in \{0,\ldots,7\}} P^{t,t+1}_{i,j}\right) \to 1$, ahogy $T \to \infty$. Belátható viszont, hogy a valóságban $T \to n$ azaz T az etalon hosszához tart. Mivel $0 \leq 1 + P^{t,t+1}_{k'_t,k'_{t+1}} - \max_{i,j \in \{0,\ldots,7\}} P^{t,t+1}_{i,j} \leq 1$ minden $t \in \{0, \ldots, n\}$-re, így azt mondhatjuk, hogy $0 \leq \sum_{t=0}^{n} \left(1 + P^{t,t+1}_{k'_t,k'_{t+1}} - \max_{i,j \in \{0,\ldots,7\}} P^{t,t+1}_{i,j}\right) \leq n$, ahol ha ez a fenti érték 0, akkor a k' teljes mértékben különbözik az $l'^{(i)}$ sorozattól, és n, ha a lehető

legnagyobb mértékben megegyezik velük. Így tehát alkalmazhatjuk a $P(k' \sim l') = \frac{1}{n}\sum_{t=0}^{n}\left(1 + P^{t,t+1}_{k'_t k'_{t+1}} - \max_{i,j\in\{0,\ldots,7\}} P^{t,t+1}_{i,j}\right)$, gyakoriságot, mely egy speciális kedvező/összes alak, ugyanis a számláló a k'-ben előforduló átmenetek $l'^{(i)}$ sorozat által meghatározott átmenetvalószínűségeinek, illetve a legkedvezőbb átmenetvalószínűségek megegyezésének bekövetkezési valószínűségeinek összege, míg a számláló ennek maximális értéke. Ugyanilyen megfontolások alapján a másik esetben az alábbi képletet lehet alkalmazni $P(k' \sim l'^{(i)}) = \frac{1}{n}\sum_{t=0}^{n}\left(1 + P^{t,t+1}_{k'_t,k'_{t+1}} - \max_{j\in\{0,\ldots,7\}} P^{t,t+1}_{Y(t),j}\right)$. A módszereket külön leimplementáltam és teszteltem, Kormos Jánossal közösen bővebben foglalkoztunk velük a [27] és [28]-ban, illetve [62]-ben.

A Cut-And-Or-Not technika és a szemantikus indexelés eredményeinek teszteléséhez létrehoztam egy teljes, működő képadatbázist. A harmadik részben a létrehozott Oracle alapú rendszer technikai bemutatása található, illetve különféle példákon keresztül annak működése is bemutatásra kerül. Természetesen a rész végén megtalálható az általam implementált rendszer más működő rendszerekkel történő összehasonlítása is.

Summary

The document has three parts.

In the first part a short overview can be read on the literature of the visual information retrieval, which can be classified into three groups. The first one is the class of content-based image retrieval, namely the class of the mathematical models of pattern matching based on various features of images. These models often use the colours, shapes, textures of the images and their locations and spatial or geometrical characteristics as well. The second group — which is a lot smaller than the previously mentioned — includes the indexing techniques used for a faster image retrieval and querying. The rest is the third, namely the group of interfaces. In most cases these are not only used for the creature of the query image, but using these interfaces the queries can be formalized as well.

The second part contains my own research work and results corresponding to the three groups of techniques mentioned in the previous part. For the indexing of images I used the idea of the information retrieval systems, i.e., the hierarchical classification combined with the benefits of the object-oriented systems. The images $\mathrm{obj}_i \in \mathrm{Obj}$ (storeable in the database) can be organised into a class hierarchy $\boldsymbol{C}$, where its classes C_i, $i = 1, \ldots, N$ contain the images. For a valid class hierarchy the images has to be modelled by an object-oriented way based on their contents. In this step it is recommended to guarantee that $\boldsymbol{C}$ has only one root (for keeping the hierarcy features), i.e., $\exists C_k \forall C_i, i, k \in \{1, \ldots, N\}$, $i \neq k$, $C_k < C_i$, and $\nexists C_j, C_j < C_k$, $j \in \{1, \ldots, N\}$ (denoted by C_0), and for each classes there is only one direct parent (except the root), i.e., $\forall C_i \exists C_k$, $i, k \in \{1, \ldots, N\}$, $i \neq k$, $C_i \neq C_0$, $C_k \rightarrow C_i$, and $\forall C_l$, $C_l \rightarrow C_i$, $C_l = C_k$. Thus $\boldsymbol{C}$ is a general tree. Then legacy indexing techniques have to be assigned for each class in the hierarchy, and the hierarchy itself is now a secondary index built up on the series of indexes. One can read about this OO indexing technique in [58] and in more details in [59] and [60].

In the scope of the query formalization the work contains a fuzzy approach. It has a big advantage, i.e., it ensures the queries based on image parts. It is not a very well supported feature of the other models. Another advantage is that the image part matching results can be combined by fuzzy logical connectives, thus complex image queries can be formalized. So the matching results can be combined not only with connectives *AND* (criteria A *AND* B have to be fulfilled), but their alternativity can be examined (*OR* connective), or the lack of the satisfiction of a criterion (connective *NOT*). So, the approach supoorts the cut of images (image part matchings), and the connectives *AND*, *OR* and *NOT*, so its name is Cut-And-Or-Not approach. The approach can be used well in case of more than one matching algorithm Q_{i,N_i} too.

There exixsts a maximal norm (N_i) for each matching Q_i in the space of the feature vectors in the database. It comes from its finite feature.

The base of the technique is the given digital images $f_1, \ldots, f_n$ and $g_1, \ldots, g_m$ and matchings $Q_{1,N_1}, \ldots, Q_{k,N_k}$ in the database. Let us compose from these images the needed image parts $\mathcal{C}_{x_{1i},y_{1i},x_{2i},y_{2i}} f_i$ and $\mathcal{C}_{x_{1j},y_{1j},x_{2j},y_{2j}} g_j$, where $i = 0, \ldots, n$, $j = 0, \ldots, m$. Then execute the appropriate matchings, thus one can get p matching results $Q_{l,N_l}(\mathcal{C}_{x_{1i},y_{1i},x_{2i},y_{2i}} f_i, \mathcal{C}_{x_{1j},y_{1j},x_{2j},y_{2j}} g_j)$ (in case of p matchings), where $l \in \{1, \ldots, k\}$, $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. After the evaluation of these results we can get $q_1, \ldots q_p$ evaluated matching results. Compose from these values q_i, $i \in \{1, \ldots, p\}$ a logical formulae by fuzzy logical connectives and evaluate it (for example by using the formulae $q = \frac{N - Q_N}{N} \in [0, 1]$). Without weights the connectives can be evaluated by $q_1 \wedge q_2 = \max\{0, q_1 + q_2 - 1\}$, $q_1 \vee q_2 = \min\{1, q_1 + q_2\}$ and $\neg q = 1 - q$, where q, q_1, q_2 evaluated matching values. In case of weights it is recommended to use the generalised forms, i.e., $w_1 q_1 \wedge w_2 q_2 = \min\{w_1 q_1, w_2 q_2\}$, $w_1 q_1 \vee w_2 q_2 = \max\{w_1 q_1, w_2 q_2\}$ and $\neg w q = \max\{0, 1 - w q\}$, where w, w_1, w_2 are real weights. It can be examined in more details in [58] and [60].

Corresponding to the matching algorithm in [56] I examined the binary matching. In the retrieval systems this matching is often only the last used method, because it is very noise sensitive and imprecise. A more sophisticated technique is introduced corresponding to the shape matching in the second part. The matching uses the outer boundary of the patches (homogene coloured areas). The boundaries can be encoded by 8-directioned chain codes. The second part contains two different techniques as well. One of them is based on the χ^2 test, and the other one is based on the theory of stochastic processes.

In the first case there is given a digital set $X \subset \mathbb{Z} \times \mathbb{Z}$, and two chain coded binary objects, patches without holes $(f_1(x, y) : Y \to \{0, 1\}$, $f_2(x, y) : Y \to \{0, 1\}$, $Y \subset X)$, and its chain codes l^{f_1} and l^{f_2}. The question is whether the $f_1(x, y)$ can be matched in a scale- translation- and a rotation (by $k\frac{\pi}{4}$) invariant way to the $f_2(x, y)$. From the chains one can get normalized chains $l'^{f_1} = \mathcal{S}^{-1}(\mathcal{S}(l^{f_1}))$ and $l'^{f_2} = \mathcal{S}^{-1}(\mathcal{S}(l^{f_2}))$ by using the shape and difference codes. Then the two codes can be considered as two stochastical variables, and the elements of the chains are the samples. So let us assign to the l'^{f_1} the variable ξ, which has a sample $(\xi_1, \ldots \xi_m)$ with elements m. Analogously assign the variable η to l'^{f_2} with the samlple $(\eta_1 \ldots \eta_n)$. Then examine the homogenity of the two variables with a χ^2 test. If the chains only differ in rotation, translation or scale, then they can be considered as two samples from the same population. Because these samples are chains, the elements are in the set $\{0 \ldots 7\}$. Denote the multiset of these elemets by Ξ. Let us divide the values of ξ and η into eight sets, where $A_k = \{x | x \in \Xi, x = k\}$, $k = 0 \ldots 7$, and let ν_k be the number of elements ξ_i where $\xi_i \in A_k$, and analogously μ_k for η_i where $\eta_i \in A_k$. Then introduce a $\chi^2 = m \cdot n \sum_{k=0}^{7} \mathcal{R}_k$, test statistic where $\mathcal{R}_k = \begin{cases} \frac{(\frac{\nu_k}{m} - \frac{\mu_k}{n})^2}{\nu_k + \mu_k}, & \text{ha } \nu_k \neq \mu_k \neq 0. \\ 0, & \text{otherwise.} \end{cases}$ It can be proven, that it has a chi-square distribution with number of 7 degree of freedom, if the chains are matched in case of large number of n and m. With a fixed significance the values of p can be defined well from a χ^2 table.

The second case is when the chains are considered as realizations of stochastic processes. This case has two subcases. One of them is a non-learning model which can

handle the noise well. The other is a learning one, which can learn the possible noises and their locations from a series of chains. In the non-learning case the chain can be converted into a signed difference code D to determine the one step transition possibilities $P_{D_t,j}^{t,t+1}$. Then one can compose a process $Y(t)$ which elements are the elements of the original chain ($Y(t) = l'_t$, $t = 1, \ldots, n$), and its matrix of transition possibilities has elements $P_{q,r}^{t,t+1} = P(Y(t+1) = r | Y(t) = q)$, where $\sum_{i=0}^{7} \sum_{j=0}^{7} P_{i,j}^{t,t+1} = 1$ with a fixed t, so in every moment t the possibility of the next step to the moment $t+1$ is 1. From the one-to-one correspondence between l' and $D(l')$ the one-step transition possibilities have also a analogue transformation. So let the transition vectors in question be (from the possibilities $P_{D_t,j}^{t,t+1}$)

$$P_{Y(t),j}^{t,t+1} = \begin{cases} P_{D_t,i}^{t,t+1} & \text{if exists,} \\ 0 & \text{otherwise,} \end{cases}$$

where $i = \begin{cases} Y(t) - j & \text{if } Y(t) - j > 0 \text{ and } Y(t) - j \in \{0,1,2,3\} \\ Y(t) - j - 8 & \text{if } Y(t) - j > 0 \text{ and } Y(t) - j \notin \{0,1,2,3\} \\ Y(t) - j + 8 & \text{if } Y(t) - j < 0 \text{ and } Y(t) - j + 8 \in \{0,1,2,3\} \\ Y(t) - j & \text{if } Y(t) - j < 0 \text{ and } Y(t) - j + 8 \notin \{0,1,2,3\} \end{cases}$,

where $j \in \{0,1,2,3,4,5,6,7\}$, so they are vectors with eight elements. They satisfy the condition $\sum_{j=0}^{7} P_{Y(t),j}^{t,t+1} = 1$ with a fixed t.

In the learning case consider a series of normalized chains $l'^{(i)}$ where $i = 1, \ldots, N$. Mark $l'^{(1)}$ as an ethalon determining the length of the other chains. The transition possibilities for $Y(t)$ are

$$P_{q,r}^{t,t+1} = \sum_{i=1}^{N} \frac{C_{q,r}(l_t'^{(i)}, l_{t+1}'^{(i)})}{N}$$

where $C_{q,r}(l_t'^{(i)}, l_{t+1}'^{(i)}) = \begin{cases} 1, & \text{if } l_t'^{(i)} = q, \text{ és } l_{t+1}'^{(i)} = r, \\ 0, & \text{otherwise.} \end{cases}$

For the final result let us suppose that all of the elemets of $l'^{(i)}$ have infinite length. An inifinite k' converges to the series of chains $l'^{(i)}$ if
$\frac{1}{T} \sum_{t=0}^{T} \left(1 + P_{k'_t,k'_{t+1}}^{t,t+1} - \max_{i,j \in \{0,\ldots,7\}} P_{i,j}^{t,t+1}\right) \to 1$, as $T \to \infty$. In real $T \to n$, i.e., T tends to the length of the ethalon. Because $0 \leq 1 + P_{k'_t,k'_{t+1}}^{t,t+1} - \max_{i,j \in \{0,\ldots,7\}} P_{i,j}^{t,t+1} \leq 1$ for each $t \in \{0, \ldots, n\}$, so $0 \leq \sum_{t=0}^{n} \left(1 + P_{k'_t,k'_{t+1}}^{t,t+1} - \max_{i,j \in \{0,\ldots,7\}} P_{i,j}^{t,t+1}\right) \leq n$ where its value is 0 if k' totally differs from the series $l'^{(i)}$, and the value is n if they equal in the largest extent. So we can use the frequency $P(k' \sim l') = \frac{1}{n} \sum_{t=0}^{n} \left(1 + P_{k'_t k'_{t+1}}^{t,t+1} - \max_{i,j \in \{0,\ldots,7\}} P_{i,j}^{t,t+1}\right)$, which is a special fortunate/all form. Analogously in the other case we can use the form
$P(k' \sim l'^{(i)}) = \frac{1}{n} \sum_{t=0}^{n} \left(1 + P_{k'_t,k'_{t+1}}^{t,t+1} - \max_{j \in \{0,\ldots,7\}} P_{Y(t),j}^{t,t+1}\right)$. The methods were implemented and tested, we studied them in more details with János Kormos in [27] [28] and [62].

For testing the Cut-And-Or-Not approach and the semantical indexing I developed a fully functioning image database based on Oracle. In the third part I study the system's technical parameters in more details, and there are some examples as well. Finally the comparison with other systems can be read.

Irodalomjegyzék

[1] Amsaleg, L., Gros, P., *Content-based Retrieval Using Local Descriptors: Problems and Issues from a Database Perspective*, Pattern Analysis and Applications, 4, (2001), 108-124,

[2] Böhm, C., Berchtold, S., Keim, D. A.,, *Searching in High-Dimensional Spaces – Index Structures for improveing the Performance of Multimedia Databases*, ACM Computing Surveys, Vol. 33, No. 3, (2001), 322–373,

[3] Carson, C., Thomas, M., Belongie, S., Hellerstein, J., Malik, J., *Blobword: A system for Region-Based Image Indexing and Retrieval*, Conf. on Visual Information and Information Systems, (1999), 509–516

[4] Chang, C. C., Lee, S. Y., *Retrieval of similar pictures on pictorial databases*, Pattern Recogn. 24, 7, (1991), 675-681,

[5] Chang, C. C., Wu, T. C., *An exact match retrieval scheme based upon principal component analysis*, Pattern Recognition Letters 16, (1995), 465-470,

[6] Chang, S. F., *Content based indexing and retrieval of visual information*, IEEE Signal Processing Magazine 14, (4), (1997), 45-48,

[7] Chang, E., Zhu, R., OL/G – A Query Language for Geometric Databases, 1st Int. Conf. on GIS in Urban Regional and Environment Planning, Samos, Greece, (1996), 271-286,

[8] Church A., *Introduction to Mathematical Logic*, v. 1. Princeton, (1956)

[9] Date, C. J., Darwen, H., *Foundation for object/relational databases — The third manifesto* Addison-Wesley, (1998)

[10] Eakins, J. P., *Automatic image content retrieval: Are we going anywhere?* In Proceedings of the 3rd International Conference on Electronic Library and Visual Information Research, (1996),

[11] Egenhofer, M., *Spatial-Query-by-Sketch*, VL'96, IEEE Symposium on Visual Languages, (1996), 60–67

[12] Egenhofer, M., *Spatial SQL: A Query and Presentation Language*, IEEE Trans. Knowledge Data Eng. 5 (2), (1991), 161-174

[13] Elmasri, R., Navathe, S. B., *Fundamentals of Database Systems*, The Benjamin/Cummings Publishing Company, Inc. (1994)

[14] El-Kwae, E. A., Kabuka, M. R., *Efficient Content-Based Indexing of Large Image Databases*, ACM Transactions on Information Systems, Vol. 18, No. 2, (2000),

[15] Fazekas, A., *Bevezetés a digitális képfeldolgozásba*, Egyetemi jegyzet, KLTE, (1998)

[16] Freeman, H. *On the encoding of arbitrary geometric configuration*, IRE Transactions on Electric Computers, EC-10(2) p: 260–268, (1961)

[17] Fried, E *Klasszikus és lineáris algebra* Budapest, Tankönyvkiadó, (1985)

[18] Fu, K. S., *Picture Syntax*, *Pictorial Information Systems*, LNCS 80, Spreinger, (1980), 104–127

[19] Fuertes, J. M., Lucena, M., de la Blanca N. P., Chamorro-Martínez J., *A scheme of colour image retrieval from databases* Pattern Recognition Letters 22, (2001), 323-337,

[20] Glasbey, C. A., *An analysis of histogram-based thresholding algorithms*, CVGIP-Graphical Models and Image Processing 55 (1993) 532–537

[21] Gonzales, R. C. , Woods, R. E., *Digital image processing*, Addison-Wesley, Reading, Massachusetts, (1992)

[22] Grosky, W. I., Mehrotra, R., *Index-based object recognition in pictorial data management* Comput. Vision Graph. Image Process. 52, 3, (1990), 416-436,

[23] Guttman, A., *R-Trees: A dynamic index structure for spatial searching.* Proc ACM SIGMOD, Boston, MA, (1984), 47-57,

[24] Jónás, R., Kollár, L., Veréb, K., *Interaktív internetes képfeldolgozás oktató és képarchiváló rendszer*, Informatika a felsőoktatásban, (2002), 766–772

[25] Kaygin, S., Bulut, M. M., *A new one-pass algorithm to detect region boundaries*, Pattern Recognition Letters 22, (2001), 169–1178

[26] Kazman, R., Kominek, J., *Supporting the Retrieval Process in Multimedia Information Systems*, Proceedings of HICSS '97, VI, (1997), 229–238

[27] Kormos, J., Veréb, K., *Recognition of Chain-Coded Patches with Statistical Methods*, Mathematical and Computer Modelling, Vol.: 38, 7-9, 2003, 903–907

[28] Kormos, J., Veréb, K., *Recognition of chain-coded patches* COMCON 8, Proceedings of 8^{th} International Conference on Advances in Communication and Control (Telecommunications/Signal Processing), (2001), 37–45

[29] Kosch, H., Döller, M., Böszörményi, L., *Content-Based Indexing and Retrieval Supported by Mobile Agent Technology*, MDIC 2001, LNCS 2184, (2001), 1525–166,

[30] Lew, M. S., Denteneer, D.,*Fisher keys for content based retrieval*, Image and Vision Computing 19, (2001), 561–566

[31] Lew, M. S., *Principles of Visual Information Retrieval*, (ed.), Springer, (2001),

[32] Li, J. Z., Ozsu, M. T., Szafron, D., Oria, V., MOQL: A Multimedia Object Query Language, 3rd Int. Workshop on Multimedia Information Systems, Como, Italy, (1997), 19-28,

[33] Manjunath, B. S., Chapella, R., *Unsupervised Texture Segmentation Using Markov Random Field Models*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 13(5), (1991), 478–482

[34] Markkula, M., Sormounen, E., *Searching for photo-journalists practices in pictorial IR*, Workshop on Image Retrieval, University of Northumbria, Newcastle, (1998),

[35] Matusiak, S., Daoundi, M., Blu, T., Avaro, O., *Scketch-Based Images Database Retrieval*, MIS'98, LNCS 1508, (1998), 185–191

[36] Meer, P., Sher, C. A., Rosenfeld, A. *The chain pyramid: Hierarchical contour processing*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 12, p: (1990), 363-375,

[37] Meghini, C., Sebastiani, F., Straccia, U., *A Model of Multimedia Information Retrieval*, Journal of the ACM, 48(5), (2001), 909–970

[38] *Oracle PL/SQL 101*, McGraw-Hill Professional Publishing, ISBN: 007212606X (2000)

[39] *Oracle9i SQL Reference*, Release 1 (9.0.1), Part Number A90125-01, Oracle Corporation, (2002)

[40] *Oracle Visual Information Retrieval User's Guide and Reference*, Release 8.1.7, Part No. A85335-01, Oracle Corporation, (2000)

[41] *Oracle interMedia User's Guide and Reference*, Release 9.0.1 Part Number A88786-01, Oracle Corporation (2002)

[42] *Ornager, S., Image retrieval: theoretical and empirical user studies on accessing information in images*, ASIS'97, (1997), 202–211

[43] Otsu, N., *A threshold selection method from grey-level histograms*, IEEE Transactions on Systems, Man and Cybernetics 9(1) (1979) 62–66

[44] Palágyi, K. *Shape representation/description*, Proceedings of the 8^{th} SSIP, Zagreb, Croatia, (2000),

[45] Papadias, D., Sellis, T., *A Pictorial Query-By-Example Language*, Journal of Visual Languages and Computing, 6(1), (1995), 53–72

[46] Rodriguez, A. J., Torrens, A., Verdu, V., *Lukasiewicz logic and Wajsberg algebras*, Bulletin of the Polish Academy of Sciences, Secr. Logic, (1990), 51–55

[47] Roussopoulos, N., Faloutsos, C., Sellis, T., *An Efficient Pictorial Database System for PSQL*, IEEE Trans. Soft. Eng. 14 (5), (1988), 639-650,

[48] Sánchez, G., Lladós, J., Tombre, K., *A mean string algorithm to compute the average among a set of 2D shapes*, Pattern Recognition Letters 23, (2002), 203–213

[49] Santini, S., *Exploratory Image Databases, Content-Based Image Retrieval*, Academic Press, (2001),

[50] Sebesta, R. W., *Concepts of Programming Languages*, The Benjamin/Cummings Publishing Company, Inc. (1992)

[51] Sheskin, D. J. *Parametric and nonparametric statistical procedures*, CRC Press, Boca Raton, New York, (1999)

[52] Sonka, M., Hlavac, V., Boyle, R., *Image Processing Analysis and Machine Vision*, PWS Publishing, 2nd. ed. (1998)

[53] Stanchev, P. L., *General Image Retrieval Model*, Proceeding of the Conference of the Union of Bulgarian Math, (1998), 63–71

[54] Tarski, A., *Logic, Semantic and Mathematics*, Clarendon Press, Oxford, (1956)

[55] Tversky, A., *Similarity, separability, and the triangle inequality*, Psychological Review, 89, (1982), 123–154

[56] Veréb, K., *Generalization of the Binary Pattern Matching in Image Processing*, Mathematical and Computer Modelling, Vol.: 38, 7-9, (2003), 969–974

[57] Veréb, K., *Content-based Image Retrieval Algorithms at Database Server Level*, KEPAF, Képfeldolgozók és alakfelismerők 3. Konferenciája, (2002), 52–60

[58] Veréb, K., *Kutatási irányzatok az objektumorientált képadatbázisok terén*, Informatika a felsőoktatásban, (2002), 975–981

[59] Veréb, K., *Objektum alapú keresési és indexelési technológia képadatbázisokhoz*, V. Országos Objektumorientált Konferencia, (2002), http://zenith.sch.bme.hu/~ooffk/oookea/Vereb_Krisztian.rtf

[60] Veréb, K., *On a Hierarchical Indexing Fuzzy Content-Based Image Retrieval Approach*, CEUR Vol.: 76, Proc. of the VLDB 2003 PhD Workshop, Berlin, Germany, (2003), http://CEUR-ws.org/Vol-76/vereb.pdf

[61] Veréb, K., *Image Objects in Object Relational Database Management Systems*, European Conference in Object Oriented Programming, Poster, (2001),

[62] Veréb, K., *Concepts for Image Databases*, Computer Science Reports, Report 14/03, BTU Cottbus, Proc. of Emerging Database Research in East Europe Workshop, (2003), 134–137

[63] White, D. A., Jain, R., *Similarity indexing with SS-tree.* Proc 12th ICDE, New Orleans, LA, (1996), 516-523

[64] Yang, L., *A Hypertext Query Language for Images.* SIGMOD Record 23(1), (1994), 16–20

A. Függelék

A szerző publikációi

- Referált publikációk:

  [1] János Kormos, Krisztián Veréb, *Recognition of Chain-Coded Patches with Statistical Methods*, Mathematical and Computer Modelling, Vol.: 38, 7-9, 2003, 903–907

  [2] Krisztián Veréb, *Generalization of the Binary Pattern Matching in Image Processing*, Mathematical and Computer Modelling, Vol.: 38, 7-9, 2003, 969–974

- Lektorált nemzetközi publikációk:

  [3] Richárd Jónás, Lajos Kollár, Krisztián Veréb, *Recipe Representation by Robot Kitchen*, Second Aizu International Student Forum-Contest on Multimedia, Japán, 1999. július 28-30, 70–78

  [4] János Kormos, Krisztián Veréb, *Recognition of chain-coded patches* COMCON 8, Proceedings of 8^{th} International Conference on Advances in Communication and Control (Telecommunications/Signal Processing), Athén, 2001, 37–45

  [5] Krisztián Veréb, *On a Hierarchical Indexing Fuzzy Content-Based Image Retrieval Approach*, CEUR Vol.: 76, Proc. of the VLDB 2003 PhD Workshop, Berlin, Germany, 2003, http://CEUR-ws.org/Vol-76/vereb.pdf (online)

  [6] Krisztián Veréb, *Concepts for Image Databases*, Computer Science Reports, Report 14/03, BTU Cottbus, Proc. of Emerging Database Research in East Europe Workshop, 2003, 134–137

- Lektorált hazai publikációk:

  [7] Krisztián Veréb, *Content-based Image Retrieval Algorithms at Database Server Level*, KEPAF, Képfeldolgozók és alakfelismerők 3. Konferenciája, 2002, 52–60 (angol nyelvű)

  [8] Jónás Richárd, Kollár Lajos, Veréb Krisztián, *Interaktív internetes képfeldolgozás oktató és képarchiváló rendszer*, Informatika a felsőoktatásban, 2002, 766–772

[9] Veréb Krisztián, *Kutatási irányzatok az objektumorientált képadatbázisok terén*, Informatika a felsőoktatásban, 2002, 975–981

[10] Veréb Krisztián, *Objektum alapú keresési és indexelési technológia képadatbázisokhoz*, V. Országos Objektumorientált Konferencia, 2002, http://zenith.sch.bme.hu/~ooffk/oookea/Vereb_Krisztian.rtf (online)

[11] Veréb Krisztián, *Tartalomalapú képkinyerés képarchívumokból – van ilyen?*, NWS 2003, http://nws.iif.hu/ncd2003/docs/ehu/EHU-35.htm (online)

[12] Veréb Krisztián, *Tartalomalapú képkinyerés képarchívumokból – egy lehetséges megoldás*, NWS 2003, http://nws.iif.hu/ncd2004/docs/ehu/037.pdf (online)

- Közlésre benyújtott publikációk:

[13] Krisztián Veréb, *On the visualization of image databases*, 6^{th} ICAI, 2004.

[14] András Hajdu, János Kormos, Krisztián Veréb, Attila Fazekas, Lajos Kollár, Zoltán Zörgő, *Intelligent urban traffic development support system—statistical approach*, 6^{th} ICAI, 2004.

[15] Attila Fazekas, Lajos Kollár, Zoltán Zörgő, András Hajdu, János Kormos, Krisztián Veréb, *Intelligent urban traffic development support system—the simulation software and the database*, 6^{th} ICAI, 2004.

[16] Kormos János, Veréb Krisztián, *Statisztikus alakfelismerés lánckódolt objektumokon*, Alkalmazott Matematikai Lapok, 2004.

B. Függelék

A szerző konferencia előadásai

- 1999, *Japán*, *Aizui Egyetem*, *Distributed Multimedia Systems'99* konferencia, *Richard Jónás, Lajos Kollár, Krisztián Veréb: Recipe Representation by Robot Kitchen*
- 1999, *Automata and Formal Languages'99* konferencia, Vasszécsény, *Richard Jónás, Lajos Kollár, Krisztián Veréb: Recipe Representation by Robot Kitchen* és annak XML-es, formális-nyelvi vonatkozásai
- 2001, 5^{th} *International Conference on Applied Informatics*, *Veréb Krisztián: An approach to generalize the binary pattern matching in image processing*
- 2001, *European Conference of Object Oriented Programming*, Budapest, *Krisztián Veréb: Image Objects in Object-Relational Databases Management Systems, Poster*
- 2002, *Képfeldolgozók és Alakfelismerők 3. Konferenciája*, Domaszék, *Krisztián Veréb: Content-based Image Retrieval Algorithms at Database Server Level*
- 2002, *CSCS*, The Third Conference of PhD Students in Computer Science, Szeged, *Gergely Kovásznai, Krisztián Veréb: Mathematical Morphology in Image Processing by SLD Resolution*
- 2002, *CSCS*, The Third Conference of PhD Students in Computer Science, Szeged, *Tibor Csaki, Krisztian Vereb: The Jodie(+) Programming Language*
- 2002, *CSCS*, The Third Conference of PhD Students in Computer Science, Szeged, *Richard Jonas, Lajos Kollar, Krisztian Vereb: A Web-based Solution for Education of Image Processing*
- 2002, *CSCS*, The Third Conference of PhD Students in Computer Science, Szeged, *Krisztian Vereb: Complex Pattern Matching Strategies in Image Database: The Cut-And-Or-Not Approach*
- 2002, *AFL'02*, 10^{th} International Conference on Automata and Formal Languages, Debrecen, *János Kormos, Krisztián Veréb: Probabilistic Approaches in object boundary matchings*

- 2002, *AFL'02*, 10^{th} International Conference on Automata and Formal Languages, Debrecen, *Tibor Csáki, Krisztián Veréb: Automata implementations in the programming languages Jodie and Jodie+*
- 2002, *IF2002*, Informatika a felsőoktatásban 2002, Debrecen, *Jónás Richárd, Kollár Lajos, Veréb Krisztián: Interaktív internetes képfeldolgozás oktató és képarchiváló rendszer*
- 2002, *IF2002*, Informatika a felsőoktatásban 2002, Debrecen, *Veréb Krisztián: Kutatási irányzatok az objektumorientált képadatbázisok terén*
- 2002, *VOOOK*, V. Országos Objektumorientált Konferencia 2002, Dobogókő, *Veréb Krisztián: Objektum alapú keresési és indexelési technológia képadatbázisokhoz*
- 2003, *NWS*, Networksop 2003, Pécs, *Veréb Krisztián: Tartalomalapú képkinyerés képarchívumokból – van ilyen?*
- 2003, *VLDB Emerging Database Research in Eastern Europe*, Berlin, Germany, *Krisztián Veréb: Concepts for Image Databases*
- 2003, *VLDB 2003 PhD Workshop*, Very Large Databases 2003 PhD Workshop, Berlin, Germany, *Krisztián Veréb: On a Hierarchical Indexing Fuzzy Content-Based Image Retrieval Approach*
- 2004, 6^{th} *ICAI*, International Conference on Applied Informatics, *Krisztián Veréb: On the Visualization of Image Databases*,
- 2004, 6^{th} *ICAI*, International Conference on Applied Informatics, *Attila Fazekas, András Hajdu, Lajos Kollár, János Kormos, Krisztián Veréb, Zoltán Zörgő: Intelligent urban traffic development support system* (plenary)
- 2004, *NWS*, Networkshop 2004, Győr, *Veréb Krisztián: Tartalomalapú képkinyerés képarchívumokból – egy lehetséges megoldás*

Vizuális információkinyerés és tartalomalapú képkinyerési technikák képadatbázisokban

Értekezés a doktori (PhD) fokozat megszerzése érdekében az informatika tudományában.

Írta: Veréb Krisztián okleveles programtervező matematikus

Készült a Debreceni Egyetem Matematika és Számítástudományok doktori iskola Informatika programja keretében

Témavezető: Dr. Kormos János

A doktori szigorlati bizottság:

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

A doktori szigorlat időpontja 200... ....

Az értekezés bírálói:

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

A bírálóbizottság :

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

Az értekezés védésének időpontja: 200... ....