Vizuális információkinyerés és tartalomalapú képkinyerési technikák képadatbázisokban

doktori (PhD) értekezés tézisei

Visual Information Retrieval and Content-based Image Retrieval in Image Databases

PhD thesis

Veréb Krisztián

Debreceni Egyetem

Debrecen, 2004

Tartalomjegyzék

Bevezetés

Jelen PhD tézisek a képadatbázisokkal és a tartalomalapú képkinyerési tecknikákkal foglalkozik. Új eredmények és újfajta megközelítési módok találhatók benne az illesztőalgoritmusok, a multimédiás indexelés, valamint a kérdésformalizáció és összetett illesztési stratégiák terén.

1. Általában a képadatabázisokról

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 nagymennyiségű 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 adatbáziskezelésben 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 nem csak 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.

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 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 képadatbázisok feladata első megközelítésben 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 a tulajdonságokat hasonlítani, 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 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 [23] [33] [16] [12] [29] és [20]-ban.

Meg kell viszont jegyezni, hogy gyakran csak a kép bizonyos részei tartalmaznak szignifikáns információt. Így biztosítani kell a részképeken alapuló illesztéseket, illetve biztosítani azt, hogy összetett kérdéseket lehessen ezáltal feltenni (ez gyakran hiányzik az elterjedt keresőrendszerekből). 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 [37] [38]-ban említett megközelítési módokat, 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 [42]-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 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 [15]-ben.

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.*

Amennyiben a kép objektumorientált absztrakciójáról, illetve tárolásáról van szó, az objektum az objektumorientált fogalmaknak megfelelő objektumot jelöli (lásd SIMULA, SMALLTALK programozási nyelvek, a [5] illetve a [8] objektumorientált adatbázisokra vonatkozó részeit).

Az alkalmazott programozási alapfogalmak tekintetében a [30], a képfeldolgozási alapok tekintetében a [32] és [14], az adatbázis-kezelő rendszerekkel kapcsolatosan a már említett [8], metamatikiai statisztikai és matematikai logika tekintetében pedig a [11] [31] és [34] [4] tárgyalt fejezeteit tekintem mérvadónak.

2. A keresésekről

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 ismert, hanem a hozzá kapcsolódó járulékos információkat. A második 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.

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 [38] és [39]-ban is ajánlom), majd végrehajtódik a vektorok egyezésének vizsgálata az adott keresési kritériumok alapján.

Természetesen ahhoz, hogy a keresést a rendszerben végrehajthassuk, szükséges valamilyen interfész, melyen keresztül a kérdéseinket feltehetjük. A magas szintű interfészek egyik fajtája a minta alapján történő lekérdezés [27]. 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 valamilyen rajzolási segítséget nyújt (megrajzolunk, felvázolunk egy képet) [7] [21]. 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 [20]. 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 [2]. Az ikon alapú kereséseknél is az a kellemes, hogy több különféle speciális algoritmus is alkalmazható.

Ezek után térjünk rá a keresés konkrétabb részleteire. 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

$$\delta : \mathrm{Obj} \times \mathrm{Obj} \to \mathbb{R}_0^+ \tag{1}$$

távolságfogalmat kell bevezetni (ld. (7)), 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 [1]:

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

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

$$\{\text{obj} \in \text{DB} | \delta(\text{obj}, \text{obj}_q) = 0\}, \tag{2}$$

vagy az ϵ keresés ahol

$$\{\text{obj} \in \text{DB} | \delta(\text{obj}, \text{obj}_q) < \epsilon\} \tag{3}$$

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

$$\{\text{obj} \in \text{DB} | \forall \text{obj}_p \in \text{DB}, \text{obj} \neq \text{obj}_p, \delta(\text{obj}, \text{obj}_q) \leq \delta(\text{obj}_p, \text{obj}_q)\}. \tag{4}$$

3. A tulajdonságokról

A képek illeszkedése a képek tulajdonságain alapszik. Ilyenkor tehát be kell vezetni egy

$$\mathcal{F} : \text{Obj} \to \mathbb{R}^d \tag{5}$$

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

$$\delta(\text{obj}_1, \text{obj}_2) = \delta_{vectors}(\mathcal{F}(\text{obj}_1), \mathcal{F}(\text{obj}_2)) \tag{6}$$

ahol $\delta_{vectors}$ a vektorokon értelmezett (akár Euklideszi) távolság (ld. (7)).

A kinyerhető tulajdonságvektorokat két nagyobb csoportba szokás sorolni [29]. 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.

Tehát 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. Használuk tehát a továbbiakban a

$$\delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_0^+ \tag{7}$$

alakú δ távolságot. Nézzük meg, miket kell kielégíteni egy δ távolságnak f_1, f_2 és f_3 képek esetén.

$$\begin{array}{ll} p_1 : \delta(\mathcal{F}(f_1), \mathcal{F}(f_1)) = \delta(\mathcal{F}(f_2), \mathcal{F}(f_2)) & \text{önhasonlóság} \\ p_2 : \delta(\mathcal{F}(f_1), \mathcal{F}(f_2)) \geq \delta(\mathcal{F}(f_1), \mathcal{F}(f_1)) & \text{minimalitás} \\ p_3 : \delta(\mathcal{F}(f_1), \mathcal{F}(f_2)) = \delta(\mathcal{F}(f_2), \mathcal{F}(f_1)) & \text{szimmetria} \\ p_4 : \delta(\mathcal{F}(f_1), \mathcal{F}(f_3)) + \delta(\mathcal{F}(f_3), \mathcal{F}(f_2)) \geq \delta(\mathcal{F}(f_1), \mathcal{F}(f_2)) & \text{háromszög egyenlőtlenség} \end{array} \tag{8}$$

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 [35] próbálkozott hasonlóval).

Tehát a hasonlósági mértékek a képekből már korábban kinyert tulajdonságvektorok összehasonlításán alapszanak. Ezek a vektorok főleg az alábbiakat jellemzik:

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

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.

4. A bináris illeszkedésről

Általában ez a „végső menedék” egy keresés során. 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 [13] [14] [25]. 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

$$g(x,y) = \begin{cases} 0, & \text{ha } f(x,y) \leq \frac{k}{2} \\ 1, & \text{egyébként} \end{cases} \tag{9}$$

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

$$g(x,y) = \begin{cases} 0, & \text{ha } f(x,y) \leq \overline{f(x,y)} \\ 1, & \text{egyébként} \end{cases} \tag{10}$$

ahol

$$\overline{f(x,y)} = \sum_{i=0}^{M} \sum_{j=0}^{N} \frac{f(i,j)}{M \cdot N}. \tag{11}$$

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

$$1 - \sum_{i=0}^{M} \sum_{j=0}^{N} \frac{|f(i,j) - g(i,j)|}{M \cdot N} \tag{12}$$

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

A módszer általánosításaival, kiterjesztésével szürkeskálás képekre, eltolás- és nagyítás invariáns változataival [36]-ben 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 [36]-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

$$D_{i,j,k}(x,y) = |f'(x_1 + i, y_1 + j) - g'(x,y)|, \tag{13}$$

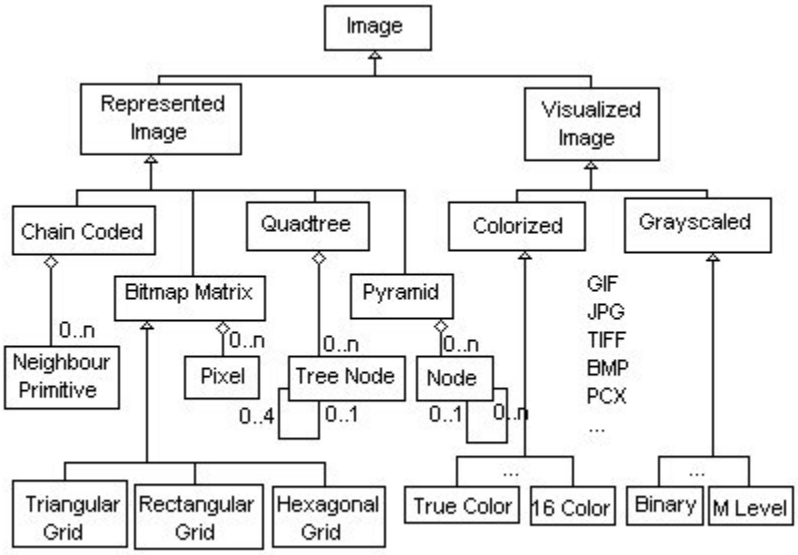
ahol $i = 0, \ldots, M-K$, $j = 0, \ldots, N-L$, $x = 0, \ldots, K$, $y = 0, \ldots, L$, $k = 0, \ldots, t_2$ és $x_1 = \frac{x(t_1+t_2)}{t_1+k}$ egész része, és $y_1 = \frac{y(t_1+t_2)}{t_1+k}$ egész része. Ha $x_1 > M$ vagy $y_1 > N$, akkor D nem definiált.

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

$$D_{i,j,k}(x,y) = |f(x_1 + i, y_1 + j) - g(x,y)|. \tag{14}$$

Vezessünk be egy r mérőszámot,

$$r_{i,j,k} = \sum_{m=0}^{K} \sum_{n=0}^{L} |D_{i,j,k}(m,n) - \overline{D_{i,j,k}}|, \tag{15}$$



1. ábra. Az objektummodell.

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:

$$D_{i,j,k}(x, y) = |\overline{f(x_1 + i, y_1 + j)} - g(x, y)|, \tag{16}$$

ahol

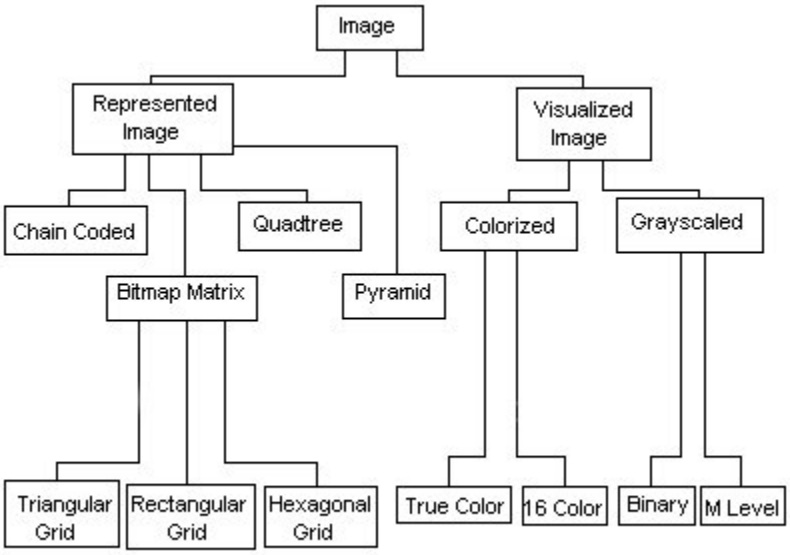
$$\overline{f(x_1 + i, y_1 + j)} = \frac{1}{2\Delta} \sum_{m=0}^{\Delta} \sum_{n=0}^{\Delta} f(x_1 + i - \frac{\Delta}{2} + m, y_1 + j - \frac{\Delta}{2} + n) \tag{17}$$

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

Így a fenti (12) 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.

5. A szemantikus képindexelésről

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. Másként kell illeszteni egy négyfa-reprezentált képet mint egy lánckódolt poligont, vagy egy bináris rendszámtáblát és egy truecolor tájképet. A megoldás az, ha a képeknek van valamilyen közös interfész felületük. Minden lánckódolt, négyfa-reprezentált stb. kép egy Kép. Azaz vannak közös műveleteik, amiket örökölnek és természetesen önspecifikusan megváltoztathatnak. Például az ős szintjén és a modellben mindenütt 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. Ugyanilyen módon, a modellben örökölt viselkedésmód az illesztés metódusa is. Itt is az alosztályok szintjén konkretizáljuk, milyen illesztési eljárást alkalmazunk. Tehát az ősosztály szerkezete adja számunkra az interfészt, az egységes megjelenést.



2. ábra. A típusfa.

A 1. ábra egy lehetséges objektummodellt mutat be [41]. 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.

5.1. 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. 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 említett objektumdiagramunknak egy egyszerűsített változatát (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. 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

$$\forall \text{obj}, \text{obj} \in \text{DB}, \exists C_j, j \in 1, \ldots, N, \text{obj} \in C_j. \tag{18}$$

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.

5.2. 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

$$\{\text{obj} \in \bigcup_{C_i < C_j} C_j | \text{obj}_q \in C_i, \delta(\text{obj}, \text{obj}_q) = 0\}, \tag{19}$$

- Az ϵ tipizált keresést, ahol a keresett objektumok halmaza

$$\{\text{obj} \in \bigcup_{C_i < C_j} C_j | \text{obj}_q \in C_i, \delta(\text{obj}, \text{obj}_q) < \epsilon\}, \tag{20}$$

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

$$\{\text{obj} \in \bigcup_{C_i<C_j} C_j| \\ \text{obj}_q \in C_i, \forall \text{obj}_p \in \text{DB}, \text{obj} \neq \text{obj}_p, \delta(\text{obj}, \text{obj}_q) \leq \delta(\text{obj}_p, \text{obj}_q)\}. \tag{21}$$

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. Így elméletileg van N darab indexünk $(I_1 \ldots, I_N)$. Fontos, hogy ezek az indexek nem csak 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

$$\forall \text{obj}, \text{obj} \in \bigcup_{C_i<C_j} C_j. \tag{22}$$

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.

6. 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.

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 [24]. 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.

6.1. Formalizmus

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*

$$\mathcal{C}_{x_1,y_1,x_2,y_2}(f)(x,y) := \begin{cases} f(x,y), & ha \begin{cases} (x,y) \in X, \ and \\ x_1 \le x \le x_2, \ and \\ y_1 \le y \le y_2 \end{cases} \\ nem\ definiált, & egyébként \end{cases} \tag{23}$$

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:

$$\underline{v} = (v_1, \ldots, v_l), \text{ahol } v_i \in \mathrm{Dom}_{F_i}. \tag{24}$$

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

$$\underline{x} = (x_1 \ldots, x_k), \tag{25}$$

alakúak. Ehhez tehát módosítva a korábbi tulajdonságvektor leképezésünket egy

$$\mathcal{F} : \mathrm{Dom}_{F_1} \times \cdots \times \mathrm{Dom}_{F_l} \to \mathbb{R}^k \tag{26}$$

tulajdonságvektor leképezést alkalmazhatunk. 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

$$\|\underline{x}\| = \sqrt{x_1^2 + \cdots + x_k^2}, \tag{27}$$

alakban. Amennyiben a normát belsőszorzattal definiáljuk, akkor használhatjuk a háromszög-egyenlőtlenséget (lásd távolság vs. hasonlóság, ill. (8))

$$\|\underline{x} - \underline{y}\| \le \|\underline{x} - \underline{z}\| + \|\underline{z} - \underline{y}\|, \tag{28}$$

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 [29]), 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.

$$N = \max_{\underline{x},\underline{y}\in\mathbb{R}^k} \{\|\underline{x} - \underline{y}\|\}. \tag{29}$$

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

$$\|\underline{x} - \underline{y}\| = \sum_{j=1}^{k} w_j |x_j - y_j|, \tag{30}$$

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 [19] [21] 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 definiáljuk, 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

$$Th(Q_N(f, g), t) := \begin{cases} 1, & \text{ha } Q_N(f, g) \leq t \\ 0 & \text{egyébként} \end{cases} \tag{31}$$

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 $Th(Q_N(f, g), t) = 0$.

6.2. 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. Én a Lukasiewicz logika alapjaiból merítettem [28]. 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.*

6.3. 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 nulladrendű logikai összekötőjeleket kell használnunk [4] [34]. 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.

Az első három lépés ugyanaz, mint a fuzzy esetben, az utolsó két lépés helyett vezessük be az alábbi hármat:

- 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.

7. Alakfelismerés

A lánckód az egyike a tradicionális képi adatszerkezeteknek. A láncokat az objektumok határvonalainak leírására használjuk a képfeldolgozás során [32] [22]. 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 [10] így gyakori eszközei az objektumhatárok leírásának.

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) [26]. 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 [14]. 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.

7.1. 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 lánckód a belső karakterisztikán alapul [14], a kontúr leírására szolgáló egyik leggyakoribb eszköz. Mielőtt rátérnénk a lánckódra, először vizsgáljuk meg, mit nevezünk kontúrnak.

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

Ez a definíció valójában egyenes következménye a [9] szomszédsági struktúrákon alapuló definíciójának.

A definíción túl még fontos megjegyeznünk, hogy a kontúrpontok bejárása mindig az óra járásával ellentétes irányban történik (ld. [14], [26]).

7.1.1. 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. [32]). 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. 3 ábra).

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

Ha előállítottuk a kontúr kódot, megkaphatjuk a kontúr hosszát. A kontúr hossza egy régió tulajdonság, mely könnyen számolható a lánckódból [32]. 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 két nem diagonális lépés együttese, melynek hossza 2. Ez is az oka annak, amiért a 8-irányítású lánckód sokkalta 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:

$$L = \sum_{i=1}^{n} U(l_i^f), \tag{32}$$

ahol 4-kapcsolódás esetén

$$U(l_i^f) = 1, \tag{33}$$

és 8-kapcsolódás esetén

$$U(l_i^f) = \begin{cases} 1, & \text{ha } l_i^f \text{ páros vagy 0,} \\ \sqrt{2}, & \text{ha } l_i^f \text{ páratlan.} \end{cases} \tag{34}$$

A lánckód nem csak 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 két 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. 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.

7.1.2. 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.

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.

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 ${l'_{i+1}}^f$ $({l'_i}^f + s^f_{i+1})$ 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.

7.2. 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.

7.2.1. 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 tarozó 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. [31]). 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

$$\chi^2 = m \cdot n \sum_{k=0}^{7} \mathcal{R}_k, \tag{35}$$

próbastatisztikát, ahol

$$\mathcal{R}_k = \begin{cases} \frac{(\frac{\nu_k}{m} - \frac{\mu_k}{n})^2}{\nu_k + \mu_k}, & \text{ha } \nu_k, \mu_k \neq 0. \\ 0, & \text{egyébként.} \end{cases} \tag{36}$$

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 [31]. 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.

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.

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.)

A módszer igazából poligonokra lett kifejlesztve, azaz nem tudja kezelni a görbéket. 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.

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.

7.2.2. Foltfelismerés sztochasztikus folyamatok segítségével

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

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

$$D(l')_t = \begin{cases} d_t & \text{ha } d_t \in \{0,1,2,3\} \\ -(8-d_t) & \text{egyébként} \end{cases} \tag{37}$$

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

$$P_{q,r}^{t,t+1} = P(D_{t+1} = r | D_t = q) \tag{38}$$

egy olyan kitétellel, hogy

$$\sum_{r=-3}^{3} P_{q,r}^{t,t+1} = 1, \text{ rögzített } t \text{ mellett}, \tag{39}$$

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:

$$P_{D_t,j}^{t,t+1} = \begin{cases} 1 & \text{ha } j = D_{t+1}, \\ 0 & \text{egyébként.} \end{cases} \tag{40}$$

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 (39)-ot.

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

$$D(l') \in \{-3,-2,-1,0,1,2,3\}, \tag{41}$$

és ismert a 3σ szabály, azaz 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:

$$P_{D_t,j}^{t,t+1} = \begin{cases} 0.6344, & \text{ha } j = 0, \\ 0.1587, & \text{ha } |j| = 1, \\ 0.0228, & \text{ha } |j| = 2, \\ 0.0013, & \text{ha } |j| = 3, \end{cases} \tag{42}$$

mely beláthatóan teljesíti (39)-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 $P^{t,t+1}_{D_t,j}$ 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

(43)

mely szintén teljesíti (39)-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 (39) kielégítésén túl a fenti konstrukció a

$$\sum_{j=-3}^{3} P^{t,t+1}_{D_t j} = 1, \text{ rögzített } t \text{ mellett}, \tag{44}$$

feltételt is kielégíti.

Meg kell említeni, hogy a fenti képletben használt értékek a standard normális eloszlás táblázatának oly módon módosított értékei, hogy a (39) 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_1l_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:

$$P^{t,t+1}_{q,r} = P(Y(t+1)=r|Y(t)=q) \tag{45}$$

a szokásos kitétellel, hogy

$$\sum_{i=0}^{7}\sum_{j=0}^{7} P^{t,t+1}_{i,j} = 1, \text{ rögzített } t \text{ mellett}, \tag{46}$$

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:

$$P^{t,t+1}_{Y(t),j} = \begin{cases} P^{t,t+1}_{D_t,i} & \text{ha létezik,} \\ 0 & \text{egyébként,} \end{cases} \tag{47}$$

ahol

$$i = \begin{cases} Y(t)-j & \text{ha } Y(t)-j>0 \text{ és } Y(t)-j\in\{0,1,2,3\} \\ Y(t)-j-8 & \text{ha } Y(t)-j>0 \text{ és } Y(t)-j\notin\{0,1,2,3\} \\ Y(t)-j+8 & \text{ha } Y(t)-j<0 \text{ és } Y(t)-j+8\in\{0,1,2,3\} \\ Y(t)-j & \text{ha } Y(t)-j<0 \text{ és } Y(t)-j+8\notin\{0,1,2,3\} \end{cases}, \tag{48}$$

ahol $j \in \{0,1,2,3,4,5,6,7\}$, tehát ezek nyolc elemű vektorok. Belátható, hogy ezek kielégítik (46)-ot, sőt a (44) 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.

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:

$$P_{q,r}^{t,t+1} = \sum_{i=1}^{N} \frac{C_{q,r}(l_t'^{(i)}, l_{t+1}'^{(i)})}{N}, \tag{49}$$

ahol

$$C_{q,r}(l_t'^{(i)}, l_{t+1}'^{(i)}) = \begin{cases} 1, & \text{ha } l_t'^{(i)} = q, \text{ és } l_{t+1}'^{(i)} = r, \\ 0, & \text{egyébként.} \end{cases} \tag{50}$$

Belátható, hogy ez szintén teljesíti (46)-ot, viszont (7.2.2)-t nem, így nem lehet (47)-et alkalmazni. Ez annak köszönhető, hogy az így kapott $P_{q,r}^{t,t+1}$ értékek nem vektort alkotnak, hanem egy 8×8 méretű mátrixot.

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

$$\frac{1}{T} \sum_{t=0}^{T} \left(1 + P_{k_t',k_{t+1}'}^{t,t+1} - \max_{i,j \in \{0,\ldots,7\}} P_{i,j}^{t,t+1} \right) \to 1, \tag{51}$$

$T \to \infty$ esetén.

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

$$0 \leq 1 + P_{k_t',k_{t+1}'}^{t,t+1} - \max_{i,j \in \{0,\ldots,7\}} P_{i,j}^{t,t+1} \leq 1, \tag{52}$$

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

$$0 \leq \sum_{t=0}^{n} \left(1 + P_{k_t',k_{t+1}'}^{t,t+1} - \max_{i,j \in \{0,\ldots,7\}} P_{i,j}^{t,t+1} \right) \leq n, \tag{53}$$

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:

$$P(k' \sim l') = \frac{1}{n} \sum_{t=0}^{n} \left(1 + P^{t,t+1}_{k'_t k'_{t+1}} - \max_{i,j \in \{0,\ldots,7\}} P^{t,t+1}_{i,j} \right), \tag{54}$$

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 (49) kielégíti (46)-t), míg a számláló ennek maximális értéke. Belátható, hogy nagyelemű lánckódok esetében, és viszonylag nagy kontúrzaj lehetősége esetén (54) jól alkalmazható, mert a zajt nem csak lokálisan kezeli, hanem globálisan is.

Ugyanilyen megfontolások alapján a másik esetben az alábbi képletet lehet alkalmazni

$$P(k' \sim l'^{(i)}) = \frac{1}{n} \sum_{t=0}^{n} \left(1 + P^{t,t+1}_{k'_t, k'_{t+1}} - \max_{j \in \{0,\ldots,7\}} P^{t,t+1}_{Y(t),j} \right). \tag{55}$$

8. Összefoglaló

A tézisekhez tartozó értekezés 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 helyzetét 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. Ezzel az OO indexelési technikával [38]-ben, és nagyobb mélységben [39]-ban illetve [40]-ben 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 nem csak *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 a *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. Ezekkel bővebben [38]-ben és [40]-ben foglalkozom.

Az illesztő algoritmusokat illetően [36]-ben 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. A módszereket külön leimplementáltam és teszteltem, Kormos Jánossal közösen bővebben foglalkoztunk velük a [17] és [18]-ben, illetve [42]-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.

Visual Information Retrieval and Content-based Image Retrieval in Image Databases

Imagine you are a designer working on the next Lord of the Rings movie. You have seen thousands of images, graphics, and photos. However, you can only recall a few characteristics of the images (perhaps it had a blue sky or sand dunes, etc.). How do you find the visual similarities? Perhaps you are a journalist who needs to make a comparation of the new year celebrations from around the world. How do you find the right video shots? Visual Information Retrieval is focused on finding visual imagery.

As databases are more and more popular, the claim for storing and querying images from databases has also appeared. Though special tools for solving these problems cannot be used in all cases. For example, when a given image database is only an extension of an existing large database containing text data (e.g., police registration). Storing images in legacy database is more cost-effective than procuring special new database engines only for storing images. Obviously, in each case (mainly in the latest), retrieval of images is based on a given matching strategy, which contains a given algorithm in most cases.

Storing of images and, mainly, their retrieval from databases differs from the storing and retrieving of other non-multimedia-like data. By the spread of the new Object-Relational or fully Object-Oriented databases, further possible solutions have occured. Nevertheless, matching algorithms and strategies used for retrieving images from the databases at present time do not really support the usage of complex matchings.

There are several retrieval paradigms used in Visual Information Retrieval. When text annotation is available, it can directly be used for keyword-based searchings. In many situations, text annotation does not exist or it is incomplete.

When text annotation is unavailable, we must turn to content-based retrieval methods. When using content-based retrieval methods, search is performed on features derived from the raw visual media such as the color or texture. The VIR paradigms include querying for similar images, sketch queries and iconic queries. In similar image queries the user selects a query image, and the system gives a set of similar images to the query image. In sketch-based queries, the user manually sketch a skeleton which will be the base of the query. When using iconic search, the user places symbolic icons where the visual features should be.

The ways of query methods in image databases have not changed significantly in the last 5-10 years. As the claims and Database Management Systems have developed, the approach of queries is often out of date. The question is how it is possible to keep up with the development in such a way that applied image enhancing and feature extracting methods become more efficient.

Databases suitable for storing and retrieving images i.e. image databases raise problems to solve. Such problem as the indexing of the images or the content-based image retrieval. New solutions in this last mentioned area could make radical changes in the earlier concepts.

This was the reason why I studied image databases. The document of my research 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 [38] and in more details in [39] and [40].

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 [38] and [40].

Corresponding to the matching algorithm in [36] 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 ($f_1(x, y) : Y \rightarrow \{0, 1\}$, $f_2(x, y) : Y \rightarrow \{0, 1\}$, $Y \subset X$), 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 $(\xi_1, \dots \xi_m)$ with elements m. Analogously assign the variable η to l'^{f_2} with the samlple $(\eta_1 \dots \eta_n)$. 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 $\{0 \dots 7\}$. Denote the multiset of these elemets by Ξ. Let us divide the values of ξ and η into eight sets, where $A_k = \{x | x \in \Xi, x = k\}$, $k = 0 \dots 7$, 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 ($Y(t) = l'_t$, $t = 1, \dots, n$), 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 $i = 1, \dots, N$. 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
$\frac{1}{T} \sum_{t=0}^{T} \left(1 + P_{k'_t,k'_{t+1}}^{t,t+1} - \max_{i,j \in \{0,\dots,7\}} P_{i,j}^{t,t+1}\right) \to 1$, as $T \to \infty$. In real $T \to n$, i.e., T tends to the length of the ethalon. Because $0 \leq 1 + P_{k'_t,k'_{t+1}}^{t,t+1} - \max_{i,j \in \{0,\dots,7\}} P_{i,j}^{t,t+1} \leq 1$ for each

$t \in \{0, \ldots, n\}$, so $0 \leq \sum_{t=0}^{n} \left(1 + P^{t,t+1}_{k'_t,k'_{t+1}} - \max_{i,j \in \{0,\ldots,7\}} P^{t,t+1}_{i,j}\right) \leq n$ 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 $P(k' \sim l') = \frac{1}{n} \sum_{t=0}^{n} \left(1 + P^{t,t+1}_{k'_t k'_{t+1}} - \max_{i,j \in \{0,\ldots,7\}} P^{t,t+1}_{i,j}\right)$, which is a special fortunate/all form. Analogously in the other case we can use the form $P(k' \sim l'^{(i)}) = \frac{1}{n} \sum_{t=0}^{n} \left(1 + P^{t,t+1}_{k'_t,k'_{t+1}} - \max_{j \in \{0,\ldots,7\}} P^{t,t+1}_{Y(t),j}\right)$. The methods were implemented and tested, we studied them in more details with János Kormos in [17] [18] and [42].

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.

Hivatkozások

[1] 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,

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

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

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

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

[6] 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),

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

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

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

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

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

[12] 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,

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

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

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

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

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

[18] 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

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

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

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

[22] 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,

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[40] 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

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

[42] 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

A. 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. 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*