Kis mélységélességű képek fókuszált területeinek kiválasztása gráf alapú eljárással

Keszler Anita, Szirányi Tamás

MTA SZTAKI
keszler@sztaki.hu,sziranyi@sztaki.hu

Absztrakt. Képek lényeges részleteinek kiválasztása a képelemzés egyik fontos részterülete, amely sok, jelenleg is megoldatlan, vagy részben megoldott problémát vet fel. Ezek közül az egyik jól ismert problémával, a kis mélységélességű képek fókuszált területeinek automatikus felismerésével foglalkozunk zajmentes, illetve pontszerű zajjal terhelt bemeneti képek esetén. A problémára egy gráf alapú sűrű részgráfkereső módszerre épülő megoldást javaslunk, amelyet szintetikus illetve valódi adatokon is teszteltünk.

1. Bevezetés

Képek lényeges részleteinek kiválasztása a képelemzés egyik fontos részterülete, amely sok, jelenleg is megoldatlan, vagy részben megoldott problémát vet fel. Ezek közül az egyik jól ismert problémával, a kis mélységélességű képek fókuszált területeinek automatikus felismerésével foglalkozunk zajmentes, illetve pontszerű zajjal terhelt bemeneti képekre. Ezen a területen már számos megoldást javasoltak, például dekonvolúciós technikákon alapuló algoritmusokat [6][8], de a jelenlegi cikk célja a képek szegmentálására alkalmas gráf alapú eljárások rövid áttekintése, illetve egy gráfok használatára épülő megoldás ismertetése a fenti problémára. Az általunk javasolt algoritmus újdonsága, hogy a képfeldolgozásban megszokott gráfos eljárásokhoz képest - például gráf vágások, gráfok spektrálfelbontása - nagy mértékben különböző módszert alkalmaz. Az algoritmus alapja egy kétlépcsős modell, amelynél az első lépésben egy módosított MSF-kereső eljárást használunk, a másodikban pedig egy ezt kiegészítő páros gráfos modellt. További előnye a kis számítási komplexitás és ebből következően a rövid futási idő.

A cikk a következő módon épül fel: A 2.1 fejezetben röviden ismertetjük a képek szegmentálására használt szokásos gráf alapú eljárásokat, ezek előnyeit és hátrányait. A 2.2 fejezetben bemutatjuk más felhasználási területek hatékony algoritmusait (szociális hálózatok elemzése), felsorolva a 2.1 fejezetben leírt módszerhez képesti előnyöket. A 2.3 fejezetben bemutatjuk az általunk javasolt algoritmust néhány zajmentes képeken végzett teszteredménnyel. A 3. fejezet a zajos bemeneti képekre vonatkozó algoritmust ismerteti, ami részben a 2.3 fejezet módszerén alapszik. Az eljárás mellett teszteredményeket is ismertetünk (3.3 fejezet).

2. Fókuszált képrészek kiemelése

2.1. Gráf vágáson alapuló eljárások

A képfeldolgozásban leginkább elterjedt gráf alapú eljárások a gráf vágásokat, illetve a gráfok spektrál felbontását használó szegmentáló algoritmusok [5][13][1]. Az ilyen eljárások során a cél, hogy a képet a leggyengébb pixel-kapcsolatok mentén partícionálják.

A gráf vágó algoritmusok többségénél negatívumként említhető, hogy inicializáláskor meg kell adni tanítómintákat, azaz olyan pixeleket, amiknél rögzítjük, hogy a vágás végén melyik szegmensbe kerüljenek. A minták automatikus felderítése általában nem megoldott, bár akad nem felügyelt tanuláson alapuló alkalmazás is [5]. További hátrány, hogy egy vágás során csak két szegmens keletkezik, így ha a keresett terület több különálló képrészletből áll, akkor többször kell lefuttatni az algoritmust. Az iterációk számának automatizált meghatározása így egy újabb nehezen megválaszolható kérdést vet fel.

A gráfok spektrálfelbontásán alapuló eljárások a vágásokkal szemben egyszerre több, de továbbra is fix számú komponensre osztják fel a gráfot. Ezek az algoritmusok a gráfok Laplace-mátrixával, illetve annak sajátértékeivel és sajátvektoraival dolgoznak. A Laplace mátrixot az alábbi módon definiáljuk:

1. Definíció. *Egy egyszerű $G = (V, E)$ gráf $L = L_{ij}$ Laplace-mátrixának elemeit az alábbi módon határozzuk meg:*

$$L_{ij} = \begin{cases} deg(v_i) & \textit{if } i{=}j \\ -1 & \textit{if } i \neq j \textit{ and } (v_i, v_j) \in E \\ 0 & \textit{egyébként} \end{cases}$$

A definícióból látható, hogy a mátrix $|V|^2$ elemet tartalmaz, így egy nagy méretű gráf spektrálfelbontása számításigényes feladat. A partíciók előre rögzített száma problémát jelent, hiszen így a felosztás részletessége független az adathalmaztól. Természetesen a számuk automatikus adaptálása az adathalmazhoz itt is megoldást jelentene, de ez jelenleg még nem megoldott probléma.

2.2. Sűrű részgráf-kereső eljárások

Az eddig bemutatott, képfeldolgozásban elterjedt gráf alapú módszerek mellett más alkalmazási területeken (például szociális hálók elemzése [2],[3][11]) sikeresen használnak sűrű részgráf-kereső eljárásokat is. Ezek közös jellemzője, hogy nem csoportosítják az összes objektumot az adathalmazból. Például közösségek keresésekor lehetnek olyan emberek, akik egyik csoportba sem tartoznak. Lényeges tulajdonságuk, hogy a kimenetként elvárt részgráfokat nem számuk alapján korlátozzák, hanem a részgráfok méretére, illetve a hozzájuk sorolt pontok kapcsolatának erősségére tesznek megkötéseket. Ennek a megközelítésnek az előnye, hogy csak az adathalmaztól függ, hogy eredményként hány sűrű részgráfot, azaz csoportot kapunk.

A következőkben bemutatunk egy olyan algoritmust, amely a sűrű részgráfkereső eljárások csoportjába tartozik, ugyanakkor alkalmas képfeldolgozási feladatok megoldására is. Ehhez először ismertetünk néhány szükséges definíciót.

A sűrű részgráfok a környezetükhöz képest több belső kapcsolattal rendelkeznek. Precízebben megfogalmazva a leggyakrabban alkalmazott sűrűség definíció az alábbi:

2. Definíció. *$G' = (V', E')$ sűrű részgráf a $G = (V, E)$ gráfban, ha:*

$$\mid E'(G') \mid \geq \gamma_{dense} \binom{\mid V'(G') \mid}{2} \tag{1}$$

,ahol γ_{dense} a sűrűség korlát nagyobb, mint a gráf átlagos sűrűsége, γ:

$$\gamma = \mid E(G) \mid / \binom{\mid V(G) \mid}{2} \tag{2}$$

Ha az élekhez súlyértéket is rendelünk, akkor ezt a sűrűség értéket az élsúlyok normált összegével még súlyozni kell. Páros gráfok esetén a sűrűség fogalom ettől eltér. Több definíció ismert, ezek közül a továbbiakban az ϵ-biklikk definícióját használjuk.

3. Definíció. *Legyen $G = (A, B, E)$ egy páros gráf. $0 \leq \epsilon \leq 1$.*
A $G' = (A', B', E')$ részgráf egy ϵ-biklikk, ha

$$\forall v_i \in A', |N'(v_i)'| \geq (1 - \epsilon) \cdot |B'|, ahol N'(v_i) \in B' \tag{3}$$

A sűrűség alapú gráf algoritmusok többségét klaszterezési problémákra alkalmazzák, ahol a sűrű részgráfok felelnek meg a klasztereknek. Ezek egyik legfontosabb eleme a klaszterek magjainak meghatározása. A magot a sűrű részgráf pontjainak egy részhalmaza alkotja. Nagy méretű gráfok esetén a magok felderítése különösen nagy kihívást jelent. Ugyanakkor, ha a magok rendelkezésre állnak, a klaszterek többi elemének meghatározása már egy lényegesen kisebb számításigényű feladat.

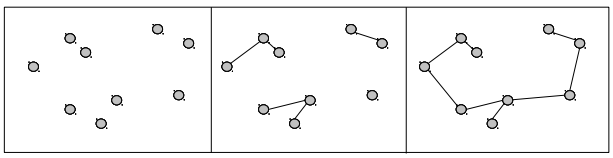
Az utóbbi években statisztikus módszerek kerültek előtérbe a magok felderítésére, mivel a legtöbbet rövid futási idő jellemzi. Például a [12] irodalomban a szerzők egy olyan véletlen mintavételezést használó algoritmust javasolnak, amely felsorolja a gráfban található ϵ-biklikkeket $O(|V|)$ futási idővel, ahol $|V|$ a gráf pontjainak számát jelöli. Bár a módszer futási ideje a klaszterezendő pontok számában lineáris, de az ϵ paraméterben exponenciális. Ez a paraméter jelzi, hogy milyen mértékben közelítjük meg a klaszter sűrűségével a teljes páros gráfot. További hátrány, hogy csak olyan esetekben alkalmazható, amikor a gráf pontjainak nagy része egy klaszterbe tartozik - körülbelül egy nagyságrendnyi eltérés lehet a pontok száma és a klaszter mérete között. Mivel véletlen mintavételezésen alapul, nem garantált, hogy ténylegesen megtalálja a klasztereket.

A gráfokon végzett véletlen bolyongást használó algoritmusok szintén ide sorolhatók [10][14]. Minél jobban hasonlít a gráf két pontja által reprezentált elem egymásra, annál nagyobb lesz a közöttük lévő él súlya. Egyik pontból a másikba

az élsúllyal arányos valószínűséggel lépünk a bolyongás során. Rögzített számú lépés után a bejárt összefüggő részgráf a kezdő pontot tartalmazó legszorosabban kapcsolódó, azaz legsűrűbb részgráf lesz. Vegyük észre, hogy egy adott pontot tartalmazó legsűrűbb részgráf még nem feltétlenül sűrű a gráf egészét tekintve. Ezt a problémát rendszerint véletlen inicializálással és az algoritmus sokszori futtatásával oldják meg.

2.3. Javasolt algoritmus a klasztermagok meghatározására

Az általunk javasolt algoritmus a klasztermagok felderítésére egy módosított minimális súlyú feszítőfa (MSF) kereső eljárás, melynek alapját a Kruskal-algoritmus [7] szolgáltatja. A Kruskal-algoritmus egy mohó eljárás, amely egy élsúlyozott gráfban megkeresi a legkisebb összsúlyú, még éppen összefüggő részgráfot, ahol az összsúly a részgráf élsúlyainak összege. A módszer attól mohó, hogy az MSF építéshez minden lépésben a legkisebb súlyú élt használja fel, amely nem alkot kört a korábban kiválasztott élekkel. Ez egy lokális optimalizálási eljárás, azonban bizonyítottan globális optimumhoz vezet. Ha ezt az algoritmust futtatnánk le egy összefüggő gráfon, akkor a kimenet egyetlen komponens lenne. Ugyanakkor egy köztes lépésben megállítva a futást, az adott iterációban legsűrűbb részgráfokat (összefüggő komponensek) kapjuk eredményül 1.



1. ábra: a) A gráf pontjai; két pont közelsége a hasonlóságuk mértékét jelzi; b) A Kruskal-algoritmus egy belső lépése; c) A gráf MSF-je

A klasztermag-kereső FINDCORES() eljárás lépései a következők:

Algoritmus [ClusterCores]=FINDCORES($G(V,E)$, d_{limit})

1 Meghatározzuk a távolságot a pontpárok között
2 Nagyság szerint rendezzük őket: $Dist_{order}$
3 Inicializálás: Legyen $G' = (V, E')$ egy olyan gráf, ahol $E' = \{\}$; $i = 1$;
4 $x = Dist_{order}(i)$;
5 ha $x \cup E$ kört tartalmaz, dobjuk el x-et;
6 ha D(Komponens(x)) $> d_{limit}$, dobjuk el x-et;
7 $E = E \cup x$; $i = i + 1$;
8 Ha még maradt él, vissza a 4. lépéshez.
9 ClusterCores = Összefüggő komponensek G'-ben

A Kruskal-algoritmusnak megfelelő sorrendben választjuk ki az éleket. Emellett minden iterációban ellenőrizzük az átmérőjét annak a komponensnek, amihez az éppen vizsgált él tartozik (Komponens(x)). Ha a komponens átmérője átlép egy határt (d_{limit}), akkor az élt (és a végpontját) nem vesszük hozzá a komponenshez (6. lépés). Ha az új él két korábbi komponenst kötne össze, akkor az új komponens átmérőjére vonatkozik a d_{limit} határ.

A FINDCORES() algoritmus futási ideje $O(|V|^2 \cdot log|V|^2)$ egy tetszőleges gráfra: az élsúlyok meghatározása bármely két pont között $O(|V|^2)$, az élsúlyok rendezése $O(|V|^2 \cdot log|V|^2)$. Ez ugyan lényegesen rosszabb, mint a [12] által bemutatott megoldás, de a mi módszerünk alkalmas kisméretű sűrű részgráfok keresésére is, ráadásul a véletlen mintavételezést is sikerül megkerülnie. Emellett abban az esetben, ha a bemeneti gráf ritka, azaz $|E| = O(|V|)$, a futási idő leszorítható $O(|V| \cdot log|V|)$ -re.

A kihívást nyilvánvalóan az algoritmus leállási feltételének, d_{limit} értékének beállítása jelenti. A komponensek (azaz klaszterek) átmérőjéből a sűrűség tényleges értéke nem számolható ki, de alsó becslés adható rá. Így a kimenetként kapott részgráfok elvárt sűrűségének γ_{dense} megfelelően kell beállítanunk a d_{limit}-t.

$$\gamma_{estimated} \geq d_{limit}/w_{max} \tag{4}$$

,ahol w_{max} a maximális élsúlyérték.

2.4. A klaszter-mag kereső eljárás alkalmazása képek fókuszban lévő területeinek kiválasztására

A korábbi gráf alapú képszegmentáló algoritmusokhoz hasonlóan a gráf itt is a kép pixeleiből épül fel. A ponthalmazt a képek pixelei alkotják, az élek a szomszédos pixeleket kötik össze (4-es szomszédságot figyelembe véve). A modell természetesen kiterjeszthető tetszőleges távolságra lévő pixelek összekötésével. Az élsúlyokat a pixelek szín illetve intenzitás értékeinek különbségéből számoljuk.

Mivel a gráf pontjainak fokszáma konstans, a FINDCORES() algoritmus ritka gráfokra vonatkozó futási idejére tett becslés az érvényes. Néhány futási eredményt a 2. ábrán ismertetünk.

(a) Az eredeti kép (b) Kimenet: a fókuszban lévő területek

(c) Az eredeti kép (d) Kimenet: a fókuszban lévő területek

(e) Az eredeti kép (f) Kimenet: a fókuszban lévő területek

2. ábra: Bemeneti képek és a klasztermag kereső FINDCORES() eljárás kimenete.

3. Hiányos és zajos adatok kezelése

A képszegmentálásra használt gráf alapú eljárások egyik legnagyobb hiányossága, hogy olyan esetekkel nem foglalkoznak, amikor a bemenetként kapott kép zajos információkat vagy hiányos adatokat tartalmaz. Ilyen esetekben a távolságszámítás a pixelek között nem végezhető el, ugyanis részben vagy teljesen hiányoznak az őket jellemző tulajdonságvektorok (intenzitás, szín, környezetben lévő pixelek tulajdonságai, stb).

A sűrű részgráfokat kereső módszerek többségére szintén jellemző, hogy feltételezik a bemeneti adatok megbízhatóságát és épségét, de itt már előfordulnak zajos adatokat kezelő módszerek is, például [9] vagy [4]. Ezeknek az algoritmusoknak közös vonása, hogy zajmentes esetben feltételezik, hogy a klaszter egy teljes részgráf vagy teljes páros részgráf. Bár ezek a modellek nem foglalkoznak élsúlyozott gráfokkal, súlyozás esetén ez annak felelne meg, mintha a klaszter bármely két pontja közötti kapcsolat kellően szoros lenne ahhoz, hogy megfeleljen a sűrűség követelményeknek. A zajtűrés ekkor úgy értelmezhető, mint tolerancia a hiányzó élekkel szemben a teljes részgráfhoz képest.

Mi egy ettől eltérő, élsúlyozott gráfokra vonatkozó zajmodellt használunk. A korábbi kétféle lehetséges állapot mellé (van él vagy nincs él két pont között) bevezetünk egy harmadikat is, amikor nincs információnk egy kapcsolatról. Ahogy azt korábban említettük, a pixeleket leíró tulajdonságvektorok nemcsak az intenzitás és szín értékekből állhatnak, hanem például a környezetükben lévő pixelekre vonatkozó értékekből álló vektorokat is tartalmazhatják: előfordulhat ugyanis, hogy részleges információkkal rendelkezünk a képpontokról.

Mivel ezt az eddig használt pixel-pixel gráffal nem tudjuk ábrázolni, áttérünk az ennél hatékonyabb páros gráf alapú modellezésre. A páros gráf egyik pontosztályában a pixelek szerepelnek, a másikban pedig az őket leíró tulajdonságvektorok egyes dimenziói.

A zajmodell alapján a bemeneti gráfot két részre osztjuk aszerint, hogy a pontok tulajdonságvektorai épek vagy hiányosak. A bemeneti páros gráfot egy M mátrixszal írhatjuk le, amely az előbb említett módon két részre osztható $M = [M_{comp}, M_{dam}]$. Az M mátrix $m(i)$ elemeit az alábbi módon definiáljuk:

$$m_{ij} = \begin{cases} \mu_{ij} & \text{Az } i. \text{ pixel } j. \text{ tulajdonságának értéke} \\ 0 & \text{ha hiányzik az érték} \end{cases}$$

, ahol $\mu_{ij} \in [-1, 1] \setminus \{0\}$ a tulajdonság normált értéke. Ha például egy pixel tulajdonságvektora a saját intenzitás és (színes kép esetén) a szín értékekből áll, valamint a négy szomszédos pixelhez tartozó értékekből, akkor a tulajdonságvektor 8 elemből áll. M_{comp}, V_{comp}, M_{dam}, V_{dam} rendre a teljes tulajdonságvektorokból álló mátrix, a hozzá tartozó pixelhalmazzal, valamint ugyanezek a hiányos tulajdonságvektorokra.

Ha csak a M_{comp}, V_{comp} adatokat tekintjük, akkor a FINDCORE() algoritmussal meg tudjuk határozni a klaszter magokat alkotó pixeleket, hiszen a páros gráfos modellből visszaállítható a hagyományos gráfos modell, ha minden adat rendelkezésre áll. A hiányos vektorú pixelek besorolásához pedig azt vizs-

gáljuk, hogy a meglévő tulajdonságaik mennyire egyeznek meg a klasztermagok tulajdonságaival.

A hiányos adatokat kezelő algoritmus a fent említett módszeren alapszik, és lépései az alábbiak:

Algoritmus [MV]=**CLUSTER**(M_{comp}, M_{dam}, ϵ)

1 [$ClusterCores$] =FINDCORES(M_{comp},d_{limit})
2 [C_{ClCore}] =CALC-CORECHAR(ClusterCores,ϵ)
3 [MV_{matrix}] =COMPUTE-MV(M_{comp},M_{dam})

Első lépésként az ép adatokból a FINDCORE() algoritmussal meghatározzuk a klasztermagokat. Ezután felépítjük a páros gráfot a teljes és a hiányos adatokból egyaránt. Karakterisztikus vektort számítunk az egyes klasztermagokhoz a CALC-CORECHAR() algoritmussal a hozzájuk besorolt pixelek tulajdonságvektorai alapján, amikkel ezután már összehasonlíthatóak a hiányos tulajdonságvektorú pixelek. Az összehasonlítás a COMPUTE-MV() algoritmussal történik.

3.1. A klasztermagok karakterisztikájának meghatározása a páros gráfban

A páros gráfos modellben a FINDCORE() algoritmus kimeneteként kapott klasztermagok sűrű páros részgráfot alkotnak, ahol az ϵ-biklikkek sűrűségdefinícióját használjuk. A gráfból kinyerhető legkisebb ϵ paraméternek megfelelő klasztermagok sűrűségértéke a d_{limit} értékből még pontosan nem határozható meg, de alulról becsülhető. Bár a klasztermagok a γ_{dense} sűrűségértéknek megfelelnek, a páros gráfban még külön meg kell határozni a releváns tulajdonságok részhalmazát. A teljes tulajdonságvektorban ugyanis lehetnek olyan tulajdonságok, amelyek csak az adott maghoz tartozó pixelek egy részére jellemzőek, és ezek elhagyásával egy sűrűbb páros részgráf nyerhető, lásd 3. ábra.

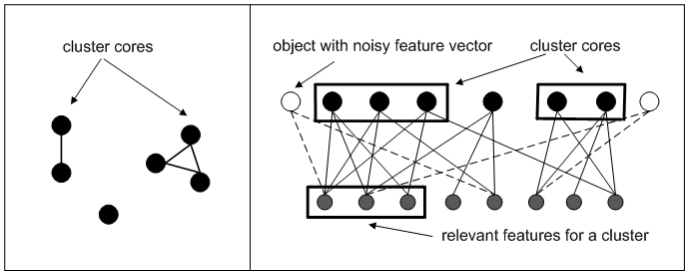
A klasztermag karakterisztikát a maghoz tartozó pixelek tulajdonságvektorainak átlagából számíthatjuk, súlyozva az egyes tulajdonságok fontosságával. A súlyozás egyik szempontja, hogy a magon belüli pontokra milyen arányban jellemző a tulajdonság, a másik szempontja, hogy a kép teljes pixelhalmazára mennyire jellemző. Ha a klasztermagon belül kicsi a szórása az adott tulajdonságnak, az azt jelzi, hogy ez a tulajdonság releváns a mag szempontjából. Ugyanakkor a kis szórás a teljes tulajdonságvektoron azt jelenti, hogy az adott tulajdonság *felesleges*, azaz nem hordoz fontos információt.

A klaszter karakterisztika így tulajdonságonként egy átlagértékből és egy súlytényezőből áll, meghatározásuk módját az alábbi algoritmusleírás tartalmazza: (Az $f()$ egy normalizáló függvény.)

Algoritmus $[c_{ClCore}, w_{ClCore}]$ =CALC-CORECHAR (ClusterCores)

1 $\forall c_i$ klasztermagra a ClusterCores-ban:
2 $w_{ClCore}(i,k) = \text{avg}(\mu_{jk})$ if $v_j \in c_i$
3 $\sigma_{cl}(i,k) = \text{disp}(\mu_{jk})$
4 $\forall F_k$ tulajdonságra:
5 $\sigma_{ClCores}(k) = disp(w_{ClCore}(i,k))$
6 $\forall c_i$ klasztermagra a ClusterCores-ban:
7 $c_{ClCore}(i,k) = f(\sigma_{ClCores}(k)/\sigma_{cl}(i,k))$

3.2. A hiányos tulajdonságvektorú adatok klaszterezése



3. ábra: A FINDCORES() kimeneteként kapott klasztermagok a hagyományos gráfban (balra). Páros gráfos modell: meghatározzuk a releváns paramétereket a klasztermagokhoz; A hiányos tulajdonságvektorú pixelek a meglévő tulajdonságok alapján kerülnek besorolásra.

A besoroláskor minden egyes v_i pixelhez kiszámoljuk, hogy milyen mértékben tartozik az egyes klasztermagokhoz ($w_{ClCore}(j)$)). Ehhez számolunk egy kapcsolódási erősséget a meglévő tulajdonságok alapján, illetve egy valószínűség értéket, ami azt tükrözi, hogy milyen arányban állnak rendelkezésre adatok a pixelről. Ezeket a továbbiakban tagsági értéknek és megbízhatóságnak fogjuk hívni. Ezek meghatározását a COMPUTE-MV() algoritmus végzi:

Algoritmus $[MV_{matrix}]$=COMPUTE-MV $(M_{comp}, M_{dam}, c_{ClCore}, w_{ClCore})$

1 $\forall v_i \in [M_{comp}, M_{dam}]$
2 $\forall c_j$ klaszter magra ha $m_{ij} \neq 0$
3 $diff(i,j) = \text{abs}(m(i) - w_{ClCore}(j))$
4 $d(i) = \text{norm}(diff(i))$
5 $MV(i,j,k) = d(i) \cdot c_{ClCore}(j)$
6 $MembValue(i,j) = sum_k[MV(i,j,k)]$
7 $Confidence(i,j) = sum(m_{ij})/F$ if $m_{ij} \neq 0$

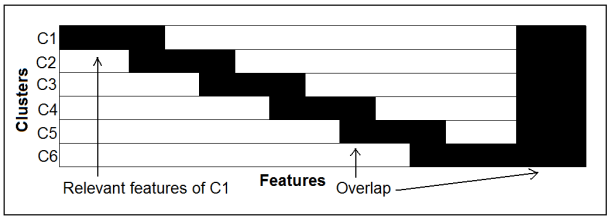
Vegyük észre, hogy az algoritmus az ép tulajdonságvektorú pontokra is kiszámolja a tagsági értékeket. Erre azért van szükség, mert a klasztermagok meghatározásakor a pixelek egy része kimaradhat, ha nem kapcsolódik szorosan egyik csoporthoz sem. Ebben a lépésben minden pixelt besorolunk valahova, de a klaszterek karakterisztikáit csak az egymáshoz szorosan kapcsolódó pixelek határozzák meg, így nem mindegy, hogy a kimaradó pixeleket a FINDCORES() algoritmuson belül vagy itt próbáljuk besorolni. A klasztermagok pixeleinek ismételt besorolása pedig lehetővé teszi, hogy a klasztermagok közötti átfedéseket is felismerjük.

3.3. Teszteredmények

Tesztelés szintetikus adathalmazon A hiányzó adatok besorolását végző algoritmust szintetikus adathalmazon is teszteltük. Ezzel a módszerrel tesztelhető az algoritmus néhány olyan tulajdonsága is, amely a képek fókuszban lévő területeinek kiválasztásakor nem szükséges, de más alkalmazások esetén fontos lehet. Ezért a szintetikus adatokon történő tesztelésnél klaszterezendő objektumokról és nem pixelekről beszélünk, ezzel is jelezve, hogy a módszer más területeken is alkalmazható.

A klaszterezendő objektumok tulajdonságvektorait valószínűségi modell segítségével állítottuk össze. A létrehozott klaszterek releváns tulajdonsághalmazai között átfedéseket hoztunk létre. Egyenletes eloszlású zajjal terheltük az adatokat, majd véletlenszerűen törültük a tulajdonság értékek egy részét a hiányzó adatokat modellezve, lásd 4. ábra.

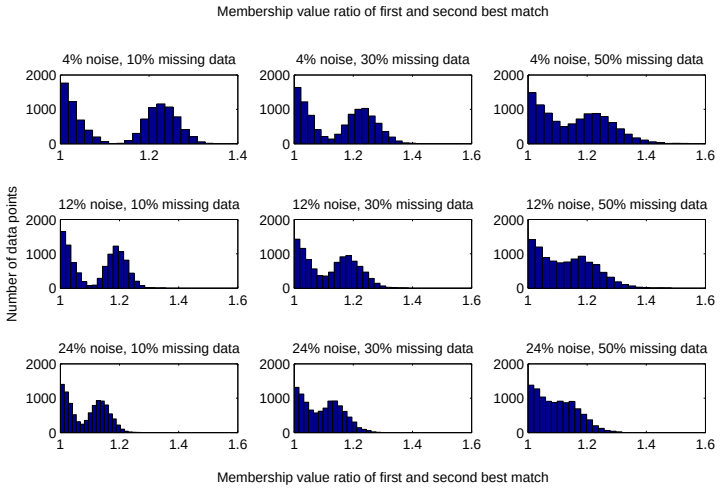
A 5. ábra a tagsági értékek megbízhatóságának teszteredményeit mutatja. A megbízhatóság alatt azt értjük, hogy milyen a legnagyobb, illetve a második legnagyobb tagsági érték aránya az egyes pixeleknél. Az eredményből látszik, hogy a zaj illetve a hiányzó adatok növelésével egyre hasonlóbb tagsági értéket kapunk a két legjobb klaszermagnál. Ennek oka, hogy a zavaró tényezők növelésével fokozatosan elmosódik a különbség a klaszterek között. Ugyanakkor a pixelek jelentős részénél még magas zajszint és hiányzó adat-arány esetén is egyértelmű a besorolás. Az 1. táblázat számos futtatási eredmény átlagát tartalmazza.



4. ábra: A szintetikus adathalmaz felépítése, klaszterek között átlapolódó releváns tulajdonságokkal.

1. táblázat: Teszteredmények. Átlagos teszteredmény különböző adathalmazokon: a klaszterezendő objektumok száma 10000 és 80000 között változott, a klaszterek száma 5 és 15 között.

	Zaj aránya	Hiányzó adatok aránya	Hiba aránya		Zaj aránya	Hiányzó adatok aránya	Hiba aránya
25% átfedés	0.2	0.1	0.0125	33% átfedés	0.2	0.1	0.0625
	0.3	0.1	0.2500		0.3	0.1	0.2250
	0.4	0.1	0.2750		0.4	0.1	0.0250
	0.5	0.1	9.1750		0.2	0.3	0.4125
	0.2	0.3	0.2125		0.3	0.3	3.9375
	0.3	0.3	0.5375		0.4	0.3	4.0500
	0.4	0.3	0.6000		0.2	0.4	0.2625
	0.5	0.3	0.7625		0.3	0.4	5.5500
	0.2	0.5	0.5125		0.4	0.4	18.7500
	0.3	0.5	0.8375		0.2	0.5	0.6750
	0.4	0.5	5.0875		0.3	0.5	2.7250



5. ábra: Teszteredmények különböző zaj és hiányzási arány mellett.

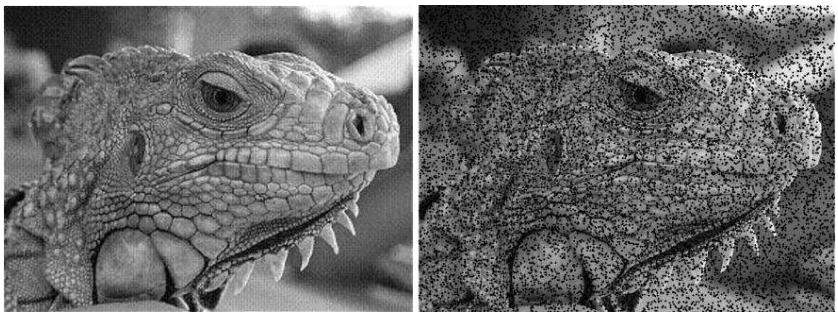
Teszteredmény képeken A klasztermagok meghatározásához a gráfot azokból a pixelekből építjük fel, amelyeknek a tulajdonságvektorai épek, az élek továbbra is a szomszédos pixeleket kötik össze, súlyaikat a környező pixelek intenzitáskülönbségéből számoltuk. Az eredeti és a zajjal terhelt tesztkép látható a 6. ábrán. A korábban bemutatott FINDCORES() algoritmus egy belső lépése illetve a kimenete a 7. kép bal illetve jobb oldalán láthatóak. A fehérrel jelzett pixeleket soroltuk be valamelyik klaszermaghoz, a feketék a kimaradó pixelek. A kimenet az egymáshoz legjobban hasonlító pixelek halmazaiból áll. Ez a fókuszált képek esetén azt jelenti, hogy az elmosott területeken lévő pixelek kerülnek először besorolásra, hiszen itt kisebb az intezitáskülönbség, mint például az elmosott területek és az éles területek határán, vagy az éles területeken belül. Simítási lépésként a kis méretű klasztermagokat töröljük.

Ezután következik a hiányzó intenzitásértékű pixelek klaszterezése. Ezt a korábbiakban ismertetett CALC-CORECHAR() és COMPUTE-MV() algoritmusok segítségével tesszük meg. Az eredmény a 8. képen látható. A klasztermagokhoz tartozó pixeleket leválasztottuk a képről, csak a fókuszban lévő területeket hagytuk meg.

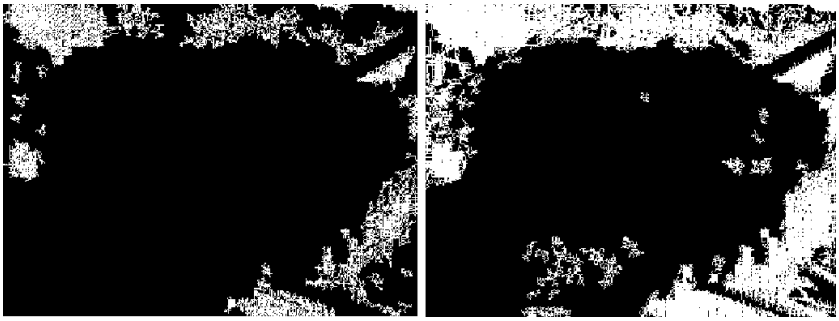
Az eredmény alapján látható, hogy az algoritmus helyesen ismeri fel a kép éles részeit. Ugyanakkor a fókusz-kereső eljárások egyik legnagyobb problémája, a homogén területek kezelése kis mértékű hibás besorolást itt is eredményezhet.

4. Összegzés

Bemutattunk egy gráf alapú eljárást kis mélyésgélességű képek éles területeinek detektálására. A módszer alapja egy kétlépcsős modell, amelynél az első lépésben egy módosított MSF-kereső eljárást alkalmazunk, a másodikban pedig egy ezt



6. ábra: (a)Eredeti szürkeskálás kép; (b)Zajos kép hiányzó fényességértékű pixelekkel (10% zaj).

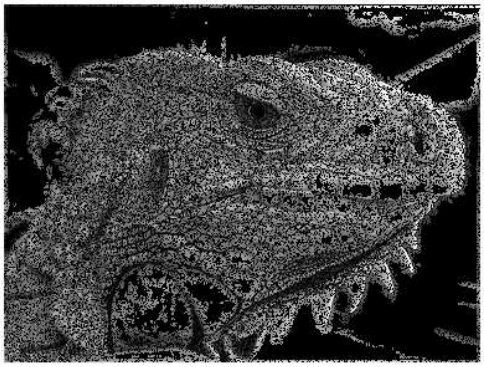


7. ábra: A klasztermagokat kereső eljárás két lépése. A fehér területek jelzik azokat a pixeleket, amelyeket besoroltunk valamelyik klasztermagba.

kiegészítő páros gráfos modellt. A fókuszált területek kiemelésének teszteredményein túl bemutattuk, hogy milyen eredmények érhetőek el zajos bemeneti adatok esetén.

Irodalom

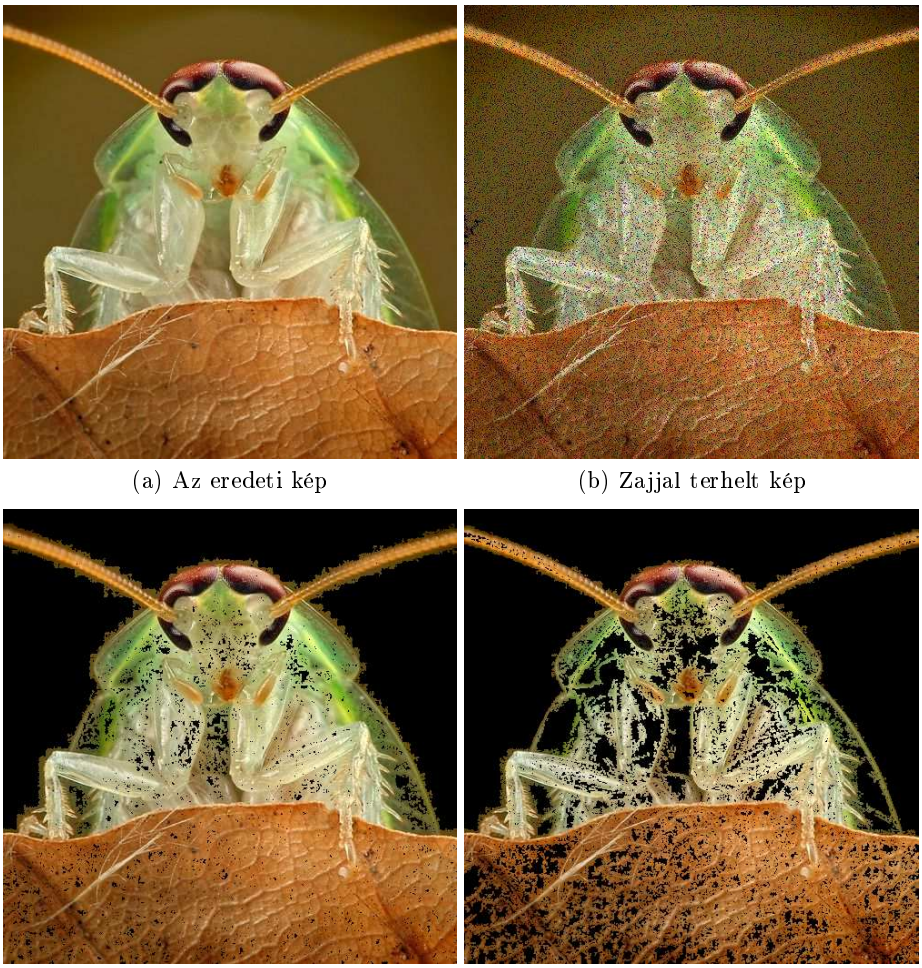
1. Jean Cousty, Gilles Bertrand, Laurent Najman, and Michel Couprie. Watershed cuts: Minimum spanning forests and the drop of water principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:1362–1374, 2009.
2. Nan Du, Bin Wu, Xin Pei, Bai Wang, and Liutong Xu. Community detection in large-scale social networks. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 16–25, New York, NY, USA, 2007. ACM.
3. C. Faloutsos, K. S. McCurley, and A. Tomkins. Connection subgraphs in social networks. In *Proceedings of the Workshop on Link Analysis, Counterterrorism, and Privacy (in conj. with SIAM International Conference on Data Mining)*, 2004.
4. P. Heggernes, D. Lokshtanov, J. Nederlof, C. Paul, and J.A. Telle. Generalized graph clustering: recognizing (p,q)-cluster graphs. In *Proceedings, 36th Interna-*



8. ábra: Az eredeti kép fókuszban lévő területének kiválasztása.

tional Workshop on Graph-Theoretic Concepts in Computer Science WG, volume LNCS, 2010.

5. Jong-Sung Kim and Ki-Sang Hong. Color-texture segmentation using unsupervised graph cuts. *Pattern Recogn.*, 42(5):735–750, 2009.
6. Levente Kovacs and Tamas Sziranyi. Focus area extraction by blind deconvolution for defining regions of interest. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1–6, 2007.
7. J.B. Kruskal. On the shortest spanning tree of a graph and the traveling salesman problem. *Proc. Amer. Math. Society*, 7:48–50, 1956.
8. D. Kundur and D. Hatzinakos. Blind image deconvolution. *IEEE Signal Processing Magazine*, (May):43–64, 1996.
9. J. Li, K. Sim, G. Liu, and L. Wong. Maximal quasi-bicliques with balanced noise tolerance: Concepts and co-clustering applications. In *SDM*, pages 72–83, 2008.
10. L. Lovasz. Random walks on graphs: A survey. 2(1):1–46, 1993.
11. N. Mishra, R. Schreiber, I. Stanton, and R. E. Tarjan. Clustering social networks. In *WAW'07: Proceedings of the 5th international conference on Algorithms and models for the web-graph*, pages 56–67, Berlin, Heidelberg, 2007. Springer-Verlag.
12. Nina Mishra, Dana Ron, and Ram Swaminathan. On finding large conjunctive clusters. In *In Computational Learning Theory*, volume 2777, pages 448–462, 2003.
13. Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.
14. Karsten Steinhauser and Nitesh V. Chawla. Identifying and evaluating community structure in complex networks. *Pattern Recogn. Lett.*, 31(5):413–421, 2010.



(a) Az eredeti kép

(b) Zajjal terhelt kép

(c) Fókuszban lévő területek kiemelése az eredeti képen

(d) Fókuszban lévő területek kiemelése a zajos képen

9. ábra: További teszteredmények zajmentes és 15% zajjal terhelt képen.