Digitális képelemzés alapvető algoritmusai

Csetverikov Dmitrij

Eötvös Lóránd Tudománygyetem
Informatikai Kar

Tartalomjegyzék

Ábrák jegyzéke

Táblázatok jegyzéke

1. fejezet

Bevezetés

1.1. A jegyzet tematikája

Jelen jegyzetfüzet röviden összefoglalja a "Digitális képelemzés alapvető algoritmusai" c. kurzus tartalmát. A kurzus fő feladata azon alapvető képfeldolgozási és -elemzési módszerek és algoritmusok bemutatása, amelyekkel egy kezdő felhasználó konkrét alkalmazásokban nagy valószínűséggel találkozik. Ezekre az alapvető módszerekre épülnek bonyolultabb algoritmusok is, amelyekről a jegyzet ugyan nem szól, de megértésüket nagy mértékben megkönnyíti.

Egy rövid kurzusban lehetetlen áttekinteni a módszerek mögött álló elméleteket. A főbb elméleti alapokat legtöbbször bizonyítások nélkül mutatjuk be, viszont arra törekszünk, hogy az algoritmusokat olyan részletességgel ismertessük, hogy a hallgató ezeket reprodukálni, beprogramozni tudja. Egyes összetettebb esetekben viszont csak az algoritmusok ötletét, vázlatát írjuk le, de ilyenkor is matematikailag korrekt módon járunk el.

Kurzusunk tehát gyakorlatias, de egyben igényes is, olyan értelemben, hogy sokéves kutató és fejlesztő tapasztalatunk birtokában igyekszünk hiteles képet nyújtani arról, hogy mit és mikor érdemes használni, mi az, ami tényleg működik és bekerült a képelemzés megbízható eszköztárába. Ezt a működést mindig numerikus példákkal és eredményképekkel igyekszünk illusztrálni, ami elősegíti az algoritmusok felfogását és leendő alkalmazását.

Tematikailag, a kurzus célja elvezetni a hallgatót az elemi képszűrési algoritmusoktól a képszegmentáláson át egészen azon módszerekig, amelyek leírják a képen levő objektumok alakját és lehetővé teszik megkülönböztetésüket, felismerésüket. A jegyzetben az alábbi főbb témákat és területeket érintjük:

- Képelemzés feladatai és alkalmazásai
- Képszűrés
- Megfeleltetés és mintaillesztés
- Élek és sarkok detektálása

- Képküszöbölés és régió alapú szegmentálás
- Vázszerű reprezentációk és távolság-transzformáció
- Morfológiai képfeldolgozás
- Alakelemzés

A jegyzet *bevezető részében* megtárgyaljuk az alapfogalmakat és áttekintjük a képelemzés legfontosabb feladatait és gyakorlati alkalmazásait. Több konkrét alkalmazást valós képekkel szemléltetünk. A *képszűrés* egy kiterjedt szakterület, amelyet csak alapjaiban ismertetünk olyan szinten, amely a zajszűrés, képjavítás és a további fejezetek megértéséhez szükséges. Ennek ellenére itt is egy sor igen hasznos, esetenként viszonylag összetett algoritmust mutatunk be. A *megfeleltetés* a számítógépes látás egyik fundamentális problémája, amikor lokális felismerés révén több képben azonosítunk pontokat, illetve pontok környezetét. Ennek az egyik alapvető eszköze a mintaillesztés.

Ahogy a szavak betűkből állnak, a képeket is szokás lokális sajátságok (jellemzők) – élek, sarkok, foltok, vonalak – sokaságára bontani. A képi jellemzők támpontot adnak a képek elemzéséhez és összehasonlításához. Az *éldetektálás* egy gyakori képfeldolgozási feladat, amely segítségével be tudjuk határolni a képen látható régiókat, tárgyakat. A sarokpontok kinyerése elősegíti az alakzatok leírását és a mozgás észlelését.

A homogén képrégiókat egy másik módszer a *küszöbölés* révén is meghatározhatjuk. A küszöbölés a képszegmentáció nagyon hasznos, elterjedt eszköze. Mivel azonban nem vesz igénybe geometriai információt, a kialakult régiók összefüggősége nem garantált, és a régiók alakjáról szóló, esetleges előzetes tudást sem tudjuk beépíteni a szegmentációs folyamatba. Ezt a régió alapú szegmentálási eljárások teszik lehetővé.

Az utolsó három témakör a képszegmentálással kapott régiók alakleírásáról szól. Megismerünk két *vázszerű reprezentációt*, a középvonalat és a vázat. Az előbbi a távolságtranszformáció segítségével nyerhető ki a digitális képből, az utóbbira egy hatékony algoritmust mutatunk be. A *morfológiai képfeldolgozás* révén is kaphatunk vázszerű reprezentációt, de a morfológia igazi alkalmazási területe nem a nagy alakzatok precíz leírása, hanem a képen szétszórt, sok kisebb alakzat közelítő, statisztikai leírása. Végül, az utolsó fejezetben ismertetjük a kétdimenziós alakzatokat és az orientációjukat számszerűen jellemző nyomatékokat, valamint a kontúr alapú görbület-becslési és sarokdetektálási eljárásokat.

1.1.1. Alapfogalmak

A *képelemzés* kifejezést a szakirodalomban többféleképpen használják, ezért célszerű rögtön az elején tisztázni ezt és a többi alapfogalmat, különös tekintettel a képfeldolgozás és a számítógépes grafika kapcsolatára. Szóhasználatunkat az 1.1. táblázat foglalja össze. A táblázatban a "képek" alatt egy vagy több képet, valamint videót is értünk.

E szerint a *képfeldolgozás* bemenete kép, képhalmaz, vagy video, kimenete pedig egy feldolgozott, azonos adatstruktúrájú vizuális információ. *Képelemzés* esetén az adatstruktúra

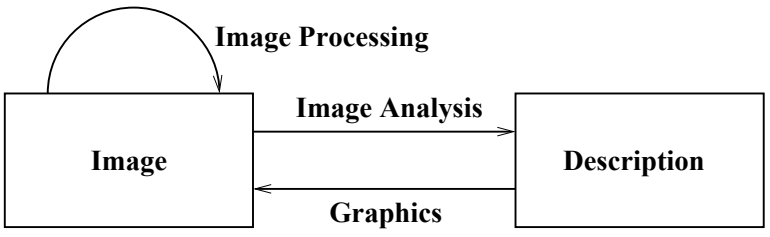
1.1. táblázat. A képelemzés fogalma.

terület	bemenet	kimenet
képfeldolgozás	képek	feldolgozott képek
képelemzés	képek	képleírások
alakfelismerés	képleírások	objektum osztályok
számítógépes látás	képek	3D-s modellek

változik: képekből képleírások lesznek. Az *alakfelismerés* képleírásokkal operál és objektum osztályokat hoz létre. Végül, a *számítógépes látás* célja pedig háromdimenziós modellek megalkotása képek vagy videók alapján. Ehhez szükséges van feldolgozásra, elemzésre és felismerésre egyaránt, ezért a számítógépes látás magába foglalja az előző három kategóriát.

A *számítógépes grafika* és a képelemzés viszonyát az 1.1. ábra illusztrálja. A klasszikus számítógépes grafika matematikai modellekből indul ki és képeket, látványt hoz létre. Matematikailag ez direkt probléma, szintézis. Képelemzés ezzel szemben egy nehezebb, inverz probléma, azaz analízis, amely során képekből matematikai modell készül. Számítógépes grafika 3D-s valóságot képez le képsíkra, számítógépes látás pedig a képvetületek alapján próbálja a világot térben rekonstruálni. A nyilvánvaló különbségek ellenére meg kell jegyezni, hogy a modern számítógépes grafika számítógépes látási eszközöket is alkalmaz, és ez fordítva is igaz, tehát a két terület egyre jobban közeledik.

Ezek után, tisztázzuk a *digitális kép* fogalmát. Ez alatt egy két- vagy többdimenziós mátrixot értünk, amely egy tárgy, színtér vagy egy másik kép sík- vagy térbeli reprezentációja. A mátrix értéke nem feltétlenül a felület által visszavert, és a kamera által érzékelt elektromágneses energiát tükröző világosság-, színkód, vagy hőmérséklet (hőkamerák). Az érték lehet a felület és az érzékelő közötti térbeli távolság (távolság kép, *range image*), vagy valamilyen szimbólum vagy címke: képpontokhoz hozzárendelt osztálycímke, index, például, talajtípus, mezőgazdasági felhasználás típusa, stb. Kurzusunkban kizárólag nem színes, szürke árnyalatú képekkel foglalkozunk, amikor egy képpont (pixel) értéke világosságkód, intenzitás, amely tipikusan 0 (fekete) és 255 (fehér) közé esik. Speciális esetekben a kép kétértékű, vagyis *bináris* lesz, ahol az egyik érték az objektum, a másik a háttér.



1.1. ábra. A számítógépes grafika és a képelemzés viszonya.

A számítógépes látás *főbb céljai* a következők: ismert objektumok detektálása és felismerése; ismeretlen objektumok modellezése; pozíció és orientáció meghatározása; geometriai tulajdonságok mérése (távolságok, méretek); mozgáselemzés; szín- és textúraelemzés. Ebben a kurzusban szó lesz, többek között, olyan képelemzési módszerekről, amelyek szükségesek detektáláshoz és felismeréshez, pozíció- és orientáció-meghatározáshoz, valamint geometriai tulajdonságok méréséhez. Modellezéssel, mozgáselemzéssel és szín- és textúraelemzéssel a kurzus rövidsége miatt nem tudunk foglalkozni, ami persze nem jelenti azt, hogy ezek a témák nem fontosak.

1.1.2. A képelemzés alkalmazásai

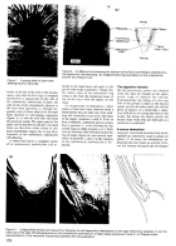
Ebben az alfejezetben áttekintjük a képelemzés több aktuális alkalmazását, esetenként példaképekkel illusztrálva. Az alkalmazások sokaságára való tekintettel legtöbbször nem tudunk részletekbe menni, de maga a felsorolás is mutatja, hogy számtalan olyan gyakorlati feladatot érint a modern képelemzés, amelyet az ember látása segítségével old meg. Mivel ez a kör egyre tágul, valószínűsíthető, hogy egyre több mérnökre és programozóra lesz szükség, aki ért a képelemzéshez, ezért is érdemes az alapjait megismerni.

1.2. táblázat. Dokumentum-feldolgozási, orvosi, ipari és robotikai alkalmazások.

Alkalmazások	Területek
Levél szortírozás, cimkeolvasás, banki papírok feldolgozása, szövegolvasás, műszaki rajzok értelmezése	Dokumentum-feldolgozás
Tumordetektálás, belső szervek méret- és alakmérése, kromoszóma-elemzés, vérsejtek számlálása	Orvosi képelemzés
Alkatrész-felismerés szereléshez, hibadetektálás, minőségellenőrzés	Ipari automatizálás
Tárgy- és környezet-felismerés, vizuális alapú mozgásirányítás	Robotika

Az 1.2. táblázat felsorol néhány fontosabb dokumentum-feldolgozási, orvosi, ipari és robotikai alkalmazást. Egyes dokumentum-feldolgozási alkalmazásokat az 1.2. ábra szemléltet. Ezekben a tartalom automatikus szegmentálása (szöveg, ábra, képlet, stb.), a térképek, bankszámlák és műszaki rajzok olvasása, valamint az aláírások ellenőrzése a tipikus cél.

Az 1.3. ábra bemutat több jellegzetes orvosi és orvosbiológiai alkalmazást, amellyel munkánk során találkoztunk. Az MRI segítségével készült térdfelvétel-sorozatok alapján fel lehet építeni a térd 3D-s modelljét, ami fontos lépés egy új, modern térdprotézis létrehozása felé. A radiológiai sejtvizsgálat egyik feladata a sugárbetegség mértékének a mérése, ami azért

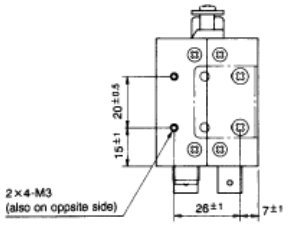


folyóirat oldal

térkép

kézírás

bankszámla



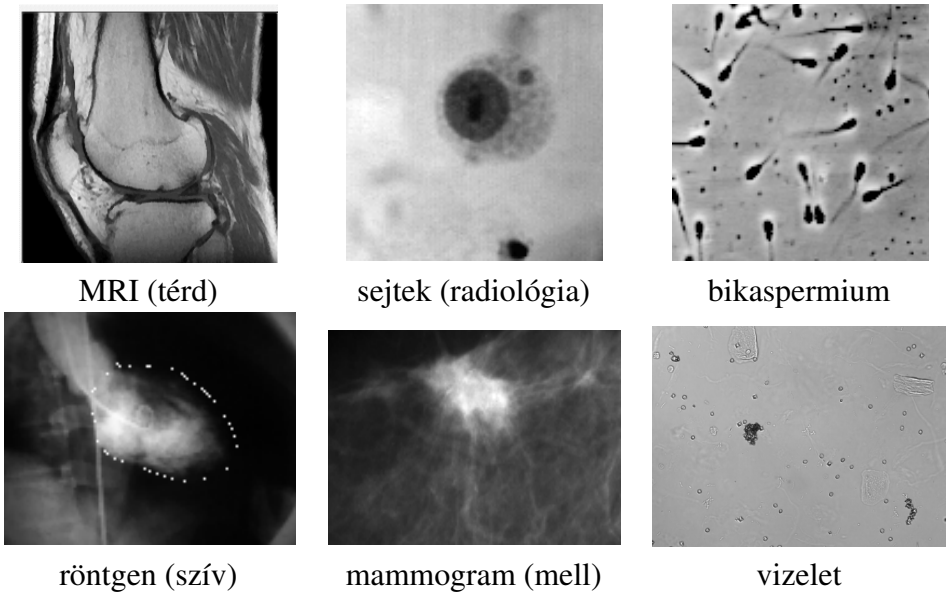
műszaki rajz

1.2. ábra. Dokumentum-feldolgozási alkalmazásokat illusztráló példaképek.

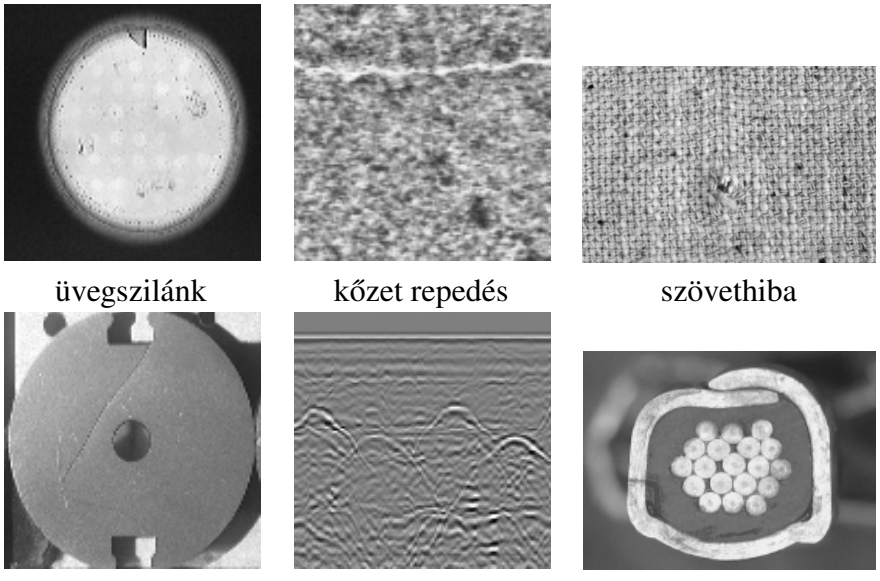
fontos, mert a könnyen mérhető sugárzás-dózis nem feltétlenül tükrözi a betegség mértékét. A bikaspermium mozgáselemzése révén megállapítható, hogy a minta mennyire életképes. A röntgenfelvételen látható szívkamra határát korrigálni kell, mert a valóságban nem ott van, ahol látjuk.

Az egyes ipari alkalmazásokat illusztráló példaképeket az 1.4. ábra tartalmazza. Itt látható több, minőség-ellenőrzéssel kapcsolatos kép: szilánk az üveg alján, kőzet repedés, szövethiba és ferritmag repedés. Az idegen testet (szilánkot) és a hibákat megbízhatóan észlelni kell, mégpedig minimális számú hamis jelzés mellett, mert azok ellehetetlenítik az ellenőrzési folyamatot. Az üvegszilánk esete rávilágít arra, hogy az "idegen test" egy alkalmazásspecifikus fogalom, amelyet nem könnyű definiálni és formalizálni. Ultrahangos talajképeket építés előtt szoktak készíteni, a földben levő nagyobb tárgyak jellegzetes hullámmintákat produkálnak. A kábel keresztmetszet egy mikroszkóp kép, itt a szálak közötti üregek (zárványok) mérése az egyik kulcsfeladat.

Az 1.3. táblázat tömör áttekintést ad térinformatikai, biztonsági, ember-gép interakciós (*human-computer interaction*, HCI), térmegfigyelési, film- és játékipari, multimédia és távérzékelési és más alkalmazásokról. Ezekkel ennél részletesebben nem foglalkozunk. De már a felsorolásból is világos, hogy mennyire gazdag a képelemzés világa, és hány egészen különböző feladattal kell megbirkóznia Gyakorlatilag lehetetlen, hogy az összes feladat megoldása mögött egy egységes matematikai apparátus álljon. Az viszont lehetséges, hogy olyan hatékony, bevált módszerkészletünk legyen, amellyel sok esetben elindulhatunk a megoldás felé.



1.3. ábra. Orvosi alkalmazásokat illusztráló példaképek.



1.4. ábra. Ipari alkalmazásokat illusztráló példaképek.

1.3. táblázat. További alkalmazások összefoglalója.

Alkalmazások	Területek
Térképek készítése fényképekből, időjárás-térképek készítése, épületek és utcák 3D-s modellezése	Térképészet, térinformatika
Ujjlenyomat illesztés, arcfelismerés, járáselemzés, más biometrikus mérések, például, fül, írisz	Bűnüldözés, biztonság
Arckifejezés-elemzés, szemmozgás-követés, gesztus-felismerés	Ember-gép interakció
Autók és emberek követése, események és tevékenységek elemzése	Térmegfigyelés
Kép- és video alapú színtér-rekonstrukció, fotórealisztikus modellezés	Film- és játékipar, kulturális örökség
Kép és video alapú keresés, indexelés; alakzat, textúra és mozgás reprezentációja és kódolása	Multimédia adatbázisok
Célkeresés és azonosítás, légi járművek és rakéták irányítása	Radarképek feldolgozása
Multispektrális képelemzés, időjárás előrejelzés, városi, mezőgazdasági és vízi területek megfigyelése és osztályozása	Távérzékelés

1.1.3. Képelemzés és felismerés fázisai

Közismert, hogy nincs más, valós világot műszaki eszközökkel mérő és leíró szakterület, ahol akkora a "távolság", olyan hosszú a lánc az eredeti mérések (pixelértékek) és a végeredmény (színtér értelmezés) között, mint a számítógépes látásban. Minden más területen a mérések sokkal "közelebb" állnak a végeredményhez. A képelemzésre jellemző "távolságot" *szemantikai résnek* (semantic gap) szokták hívni, amelyet csak fokozatosan, több lépésben lehet áthidalni. Az alábbiak összefoglaljuk a szakirodalomban hagyományosan kiemelt lépéseket, a képelemzés és -felismerés fázisait, valamint a rájuk jellemző eszközöket, esetenként néhány konkrét példával szemléltetve. A szakkifejezések angol megfelelőit zárójelben, dőlt betűvel adjuk meg.

- **Képalkotás** (*imaging*): kamerák és más érzékelők, világítás, fényvisszaverődési modellek.

- **Képjavítás** (*enhancement*): képminőség javítása, képkorrekció, zavaró vagy fölösleges információ eltüntetése.
  - zajszűrés, kontrasztemelés
- **Sajátság kiemelés** (*feature extraction*): jellemzőpontok meghatározása, lokális képleírások hozzárendelése képelemekhez.
  - él- és sarokdetektálás, küszöbölés
- **Régió alapú szegmentálás** (*region-based segmentation, grouping*): hasonló tulajdonságokkal rendelkező, összefüggő képrészek kiemelése.
  - összefüggő komponensek, élláncok
- **Régió leírás**: régiók geometriai, szín- és textúraleírása, régiók közötti relációk meghatározása.
  - terület, súlypont, orientáció, méretek, görbület, szín, textúra
- **Megfeleltetés, illesztés** (*correspondence, matching*): a modell és a kapott képleírás megfeleltetése, képértelmezés.
  - betűfelismerés betűrészek megfeleltetése alapján

A jegyzet további fejezeteiben bemutatjuk az egyes fázisokra jellemző, alapvető algoritmusokat.

1.2. Irodalom és köszönet

A kurzus megírása során az következő szakkönyveket használtuk fel:

- E. Trucco, A. Verri, "Introductory Techniques for 3-D Computer Vision", Prentice Hall.
- R. Klette, P. Zamperoni, "Handbook of Image Processing Operators", J.Wiley and Sons.
- I. Pitas, "Digital Image Processing Algorithms", Prentice-Hall.
- R.C. Gonzales, R.E. Woods, "Digital Image Processing", Addison-Wesley.
- R.M. Haralick, L.G. Shapiro, "Computer and Robot Vision", Addison-Wesley.
- A.K. Jain, "Fundamentals of Digital Image Processing", Prentice-Hall.

- M. Sonka, V. Hlavac, R. Boyle, "Image Processing, Analysis and Machine Vision", Thomson.
- B. Jähne, "Digital Image Processing", Springer.
- W.K. Pratt, "Digital Image Processing", J.Wiley.
- A. Rosenfeld, A.C. Kak, "Digital Picture Processing", Academic Press.

A kurzus megírásában az alábbi volt tanítványaim segítettek:

- Verestóy Judit
- Lerch Attila
- Szabó Zsolt

2. fejezet

Képszűrés

Képszűrés a digitális képfeldolgozás központi fogalma és legfontosabb művelete. Jegyzetünkben kizárólag olyan lokális szűrőkkel foglalkozunk, amelyek a képtérben, azaz közvetlenül a képérékekkel operálnak, szemben olyan operátorokkal, amelyek más, például frekvencia térben dolgoznak, amihez először megfelelő (pl. Fourier) transzformációt hajtanak végre.

A fejezet elején megadjuk az általános lokális operátor definícióját, amelyet fokozatosan leszűkitünk úgy, hogy eljutunk a konvolúciós szűrő fogalmáig, ami a leggyakrabban használt szűrőtípus. Nem csak lineáris, hanem nemlineáris, valamint adaptív szűrésről is lesz szó, amikor a szűrő alkalmazkodik az aktuális képkörnyezethez, a lokális kontextushoz, elkerülve ezzel több nemkívánatos mellékhatást.

2.1. Konvolúciós szűrés

2.1.1. Lokális operátorok

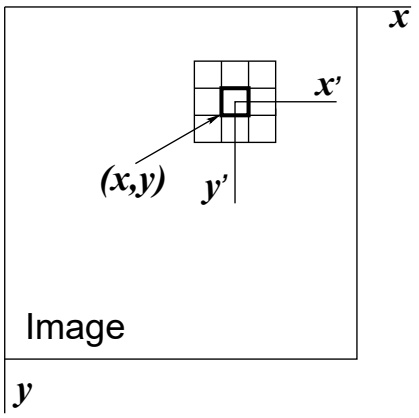
Legyen $f(x, y)$ a bemeneti (input) kép, $g(x, y)$ a kimeneti (output) kép, $N(x, y)$ az (x, y) pont valamely környezete, például egy négyzetes ablak. Ebben a környezetben bevezetjük az x', y' a lokális koordinátákat úgy, hogy a lokális koordinátarendszer origója az (x, y) pontban van.

Az *általános lokális operátor* definícióját az 2.1. ábra szemlélteti. Az (x, y) pontban az eredmény csak a pont környezetétől függ:

$$g(x, y) = T[f(x, y)],$$

ahol T a környezeten definiált operátor.

Ez a nagyon általános definíció burkoltan feltételezi, hogy csak a közeli képelemek összefüggnek, korrelálnak egymással, azaz a korreláció csökken a távolsággal. Ez legtöbbször igaz is, de nem mindig: periodikus képekben az értékek nagy távolságon is összefüggnek, hiszen a periodus ismeretében sok távoli értéket is meg tudunk jósolni.



2.1. ábra. Egy 3×3-as ablak az (x, y) pontban. x', y' a lokális koordináták.

A definíció lényege, hogy egy kis mozgó ablakon keresztül szemléljük, mintavételezzük a képet és kiszámítjuk az eredményt. Ez bizony komoly korlátozást jelent, mintha egy kis mozgó lyukon keresztül megfigyelnénk a világot és ez alapján cselekednénk. A lokálitás jellemző a képelemzési műveletek többségére. Mint később látni fogjuk, ez releváns következményekkel jár.

Az általános lokális operátor hatása azonban nem feltétlenül korlátozódik a környezetre. A *rekurzív operátorok* esetén az aktuális eredmény a bemenettől és az előző eredményektől is függhet. A kimenet nincs elválasztva a bemenettől, és az operátor működése során a bemenet módosul, mert a bemeneti képmátrixba írjuk be az eredményt. Ennek a hatása annál jelentősebb, minél nagyobb a környezet, az ablak.

A rekurzív operátorok hasznosak, de sokkal bonyolultabbak, mint a *nemrekurzív* lokális operátorok, ezért kurzusunkban csak ez utóbbiakkal foglalkozunk. A nemrekurzív operátoroknál az eredmény csak a bemenet aktuális környezetétől függ. A kimenet el van választva a bemenettől, és a működés során a bemenet nem módosul, így a művelet hatása korlátozódik a környezetre.

Egy kicsit részletezve, az általános nemrekurzív operátor definíciója a következő:

$$g(x, y) = \phi[x, y, f(x', y') : (x', y') \in N(x, y)],$$

ahol $f(x', y') : (x', y') \in N(x, y)$ a környezetbeli értékek listája. Adaptív operátorokban a ϕ művelet függhet az x, y-től: az $N(x, y)$ környezet változhat, az eredmény kiszámítási módja szintén változhat. A ϕ operátor természetesen *nemlineáris* is lehet. Egy A operátor akkor lineáris, ha minden α és β konstansra

$$A(\alpha p + \beta q) = \alpha Ap + \beta Aq.$$

2.1.2. Korreláció és konvolúció

A lineáris eltolás-invariáns operátor eredménye a bemeneti értékek lineáris kombinációja, más szóval, az f képnek a w maszkkal való *kereszt-korrelációja*

$$g(x,y) = (f \otimes w)(x,y) \doteq \sum_{\substack{(x',y')\in W \\ (x+x',y+y')\in F}} f(x+x', y+y') \cdot w(x',y'). \tag{2.1}$$

Itt W az ablakon (*window*) belüli, F a képen belüli pozíciók halmaza. A W ablak és a $w(x',y')$ súlymátrix nem függ az x,y-től. A w súlymátrix gyakran használt angol neve *kernel*.

Az f kép és a w maszk *konvolúciója*

$$g(x,y) = (f * w)(x,y) \doteq \sum_{\substack{(x',y')\in W \\ (x-x',y-y')\in F}} f(x-x', y-y') \cdot w(x',y'). \tag{2.2}$$

Szemben az előző definícióval, itt a W ablak értékeit az ellenkező sorrendben olvassuk be. Azonban a kurzusban csak szimmetrikus maszkokkal fogunk dolgozni, ezért nem fogunk különbséget tenni a korreláció és a konvolúció között, és a konvolúciós szűrésről fogunk beszélni.

Az alábbiakban bizonyítás nélkül felsoroljuk a konvolúció legfontosabb tulajdonságait. A képletekben f, g tetszőleges képek, w, v tetszőleges maszkok.

1. Korreláció a tükrözött maszkkal való konvolúció:

$$f \otimes w = f * w^{\sim}, \tag{2.3}$$

   ahol $w^{\sim}(x,y) \doteq w(-x,-y)$

2. Kommutativitás (felcserélhetőség):

$$w * v = v * w. \tag{2.4}$$

3. Asszociativitás:

$$(f * w) * v = f * (w * v), \tag{2.5}$$

   ahol a $w * v$ kifejezésben a w maszkot nullákkal körülvett képnek tekintjük és a v maszkkal konvolváljuk; az eredmény egy nagyobb maszk lesz.

4. Disztributivitás:

$$(f + g) * w = f * w + g * w. \tag{2.6}$$

5. Homogénitás: tetszőleges α konstansra

$$(\alpha f) * w = \alpha(f * w). \tag{2.7}$$

2.1.3. Példák szűrőkre és szűrésre

Ebben az alfejezetben bemutatunk néhány konkrét szűrőmaszkot és numerikus példákkal illusztráljuk a szűrők működését.

A 2.2. ábrán négy különböző 3×3-as *átlagszűrő* látható, közülük az első egy *dobozszűrő*, ami egy olyan átlagszűrő, ahol az összes súly egyenlő. (Az átlagszűrő definícióját később adjuk meg.) A többi átlagszűrőben a súlyok csökkennek a középponttól való távolsággal, ami azt jelenti, hogy a középponttól távolabb levő képelemek kisebb hatással vannak az eredményre.

Az ábrán szereplő normálótényező a maszkelemek összege. A normálás garantálja, hogy az eredmény az eredeti intenzitástartományon belül marad. A szűrők mérete nem véletlenül páratlan, mert csak így lehet egyértelműen meghatározni a középpixelt. Léteznek páros méretű szűrők is, de azokat ritkábban használják.

$$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \qquad \frac{1}{6}\begin{bmatrix}0&1&0\\1&2&1\\0&1&0\end{bmatrix} \qquad \frac{1}{8}\begin{bmatrix}0&1&0\\1&4&1\\0&1&0\end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}$$

2.2. ábra. 3×3-as átlagszűrők. A bal szélső szűrő egy dobozszűrő.

A 2.3. ábra bemutat két 5×5-ös átlagszűrőt. Az egyiket két darab 3×3-as dobozszűrő konvolúciójaként fejezhetjük ki. Ilyenkor gyorsabb megvalósításra van kilátás, mert a műveletek száma négyzetesen nő a szűrőmérettel: 5×5-ös szűrő esetén pontonként $5 \times 5 = 25$ szorzás és 24 összeadás; két 3×3-as szűrő esetén pontonként $2 \times 3 \times 3 = 18$ szorzás és $2 \times 8 = 16$ összeadás. Ráadásul, ebben a konkrét esetben a dobozszűrőkkel elkerüljük a szorzást.

A másik, szintén körszimmetrikus szűrő az alábbi képlet diszkrét változata:

$$w(r) = 8 - r^2,$$

ahol $r = \sqrt{x^2 + y^2}$ a középponttól való távolság.

$$\frac{1}{81}\begin{bmatrix}1&2&3&2&1\\2&4&6&4&2\\3&6&9&6&3\\2&4&6&4&2\\1&2&3&2&1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} * \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} \qquad \frac{1}{100}\begin{bmatrix}0&3&4&3&0\\3&6&7&6&3\\4&7&\mathbf{8}&7&4\\3&6&7&6&3\\0&3&4&3&0\end{bmatrix}$$

2.3. ábra. 5×5-ös átlagszűrők. A baloldali szűrő két 3×3-as dobozszűrő konvolúciója.

A konvolúciós szűrő működését a 2.4. ábrával szemléltetjük, ahol egy egyszerű numerikus példát mutatunk. Baloldalon a 6×6 pixeles bemeneti képmátrix, jobboldalon a kimeneti

képmátrix az első és a második kiszámított értékkel, középen pedig a szűrő látható. A bemeneti képen a szűrő aktuális pozíciója ki van emelve. A kimeneti képen a megfelelő pozícióba beírjuk az eredményt, de a széle üres marad, mert ott az eredményt nem tudjuk meghatározni. Az első kiszámított érték a következőképpen alakul ki:

$$\frac{3\cdot 1+2\cdot 2+8\cdot 1+2\cdot 2+2\cdot 4+7\cdot 2+2\cdot 1+3\cdot 2+9\cdot 1}{16}=\frac{58}{16}\approx 4.$$

3	**2**	**8**	7	8	8
2	**2**	**7**	8	7	7
2	**3**	**9**	9	8	8
1	2	9	9	7	8
2	2	8	8	8	8
2	3	7	7	9	7

$$* \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} =$$

–	–	–	–	–	–
–	4	..	..	..	–
–	..	..	..	..	–
–	..	..	..	..	–
–	..	..	..	..	–
–	–	–	–	–	–

3	**2**	**8**	**7**	8	8
2	**2**	**7**	**8**	7	7
2	**3**	**9**	**9**	8	8
1	2	9	9	7	8
2	2	8	8	8	8
2	3	7	7	9	7

$$* \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} =$$

–	–	–	–	–	–
–	4	6	..	..	–
–	..	..	..	..	–
–	..	..	..	..	–
–	..	..	..	..	–
–	–	–	–	–	–

2.4. ábra. Numerikus példa konvolúciós szűrő alkalmazására.

2.1.4. A képszél probléma kezelése

Az előbbi numerikus példában találkoztunk a képszél problémával. Egy $D_W \times D_W$ méretű szűrő esetén a kieső képszél szélessége $\lfloor D_W/2 \rfloor$, tehát minél nagyobb a szűrő, annál nagyobb a kitöltetlen sáv. Ha több szűrőt alkalmazunk egymás után és kihagyjuk a kép szélét, a kieső sáv tovább nőhet.

A képszél probléma kezelésére, a kieső sáv kitöltésére nincs elméletileg korrekt megoldás, csak heurisztikus megoldások vannak. Ezek közül az alábbiakban felsorolunk néhányat.

- *Töltsük ki nullákkal*! Ez a legegyszerűbb megoldás, amely azonban nemkívánatos, erős mesterséges éleket eredményezhet és megzavarhatja a kapott képértékek újranormálását (pl. a [0,255] tartományra).
- *Töltsük ki az eredménykép átlagértékével*! Ezt az egyszerű megoldást szoktuk javasolni, mert így kevésbé erős mesterséges éleket kaphatunk és nem változtatjuk meg az eredménykép értéktartományát.
- *Töltsük ki a legközelebbi kiszámított pixelértékkel*! Ez kissé bonyolultabb, és nem biztos, hogy megéri. Akkor célszerű alkalmazni, ha minden áron el akarjuk kerülni a mesterséges élek megjelenését.

- *Tekintsük a képet periodikusnak (hengernek)*! Ez a régen elterjedt, de mára kevésbé népszerű megoldás elfogadható, ha amúgy is feltételezzük a periodicitást, például a Fourier transzformáció alkalmazásához. Egyébként, miért is tennénk?

2.2. A zajszűrés alapjai

A klasszikus digitális képfeldolgozás elméletében fontos szerepet játszanak a különböző zajtípusok explicit matematikai modelljei. A modellek alapján le lehet vezetni a zajtípusok kezelésére leginkább alkalmas, matematikai értelemben optimális szűrőket.

A gyakorlati képelemzésben ezzel szemben legtöbbször olyan heurisztikus megoldásokat használnak, amelyek mögött nem áll egy teljes matematikai modell és elmélet. A mi jegyzetünkre is ez lesz a jellemző. Ennek ellenére érdemes megismerni néhány fontosabb zajtípust, amelyet az alábbiakban tekintünk át.

- **Additív képfüggetlen**, azaz fehér zaj:

$$g(x, y) = f(x, y) + v_{add}(x, y), \tag{2.8}$$

  ahol $f(x, y)$ az inputkép, $g(x, y)$ az outputkép, $v(x, y)$ a zaj. Ez a tipikus csatornazaj (jeltovábbítási zaj, *transmission noise*).

- **Nemkorrelált multiplikatív** zaj:

$$g(x, y) = f(x, y) \cdot v_{mult}(x, y). \tag{2.9}$$

  Ez a televíziós rasztersorokra jellemző amplitudó-moduláció (változás).

- **Kvantálási** zaj (hiba):

$$v_{kvant}(x, y) = g_{kvant}(x, y) - f_{eredeti}(x, y) \tag{2.10}$$

  Az eredeti jelérték folytonos, a kvantált jelérték diszkrét, a különbség véletlen zajként jelenik meg.

- **Só-és-bors** zaj (*salt-and-pepper, or peak noise*): Ez a pontszerű, a képpel nem korreláló, véletlen zaj legtöbbször szélsőértékű (fekete és fehér). Jellemző egyes fajta űrfelvételekre.

Bár a kurzunkban szereplő heurisztikus zajszűrési eljárásokban nem használunk explicit matematikai zajmodellt, szem előtt kell tartanunk, hogy különböző szűrők különböző zajfajták csökkentésére alkalmasak. Például, az átlagszűrő a nulla átlagú véletlen zaj, a mediánszűrő pedig a só-és-bors zaj csökkentésére. Ezért a megfelelő szűrő kiválasztásához kívánatos az előzetes zajelemzés akkor is, ha nem tudunk felállítani és felhasználni egy komplett zajmodellt.

Egyszerű általános megjegyzésként elmondhatjuk, hogy a kis csoportokban jelentkező zajos pixeleket könnyebb detektálni és kiszűrni, mint a nagyobb csoportokat. Ha az ablakban a zajmentes értékek vannak többségben, egyszerűbb a zajmentes érték becslése. Ha azonban a torz, zajos értékek alkotják a többséget, a becslés nagy valószínűséggel hibás lesz.

2.3. Lineáris simítószűrők

Ebben a fejezetben megismerünk több alapvető konvolúciós szűrőt, amelyet zajszűrés mellett más célokra is gyakran használnak. A már említett **átlagszűrő** (*mean filter*) alatt olyan, képtérben működő lineáris simítószűrőt (*spatial linear smoothing filter*) értenek, amelyben a súlyok nemnegatívak, nem nőnek a középpontól való távolsággal, és 1 az összegük:

$$\begin{aligned} &0 \leq w_{mean}(x,y) \leq 1, \\ &\quad w_{mean}(x_1,y_1) \leq w_{mean}(x_2,y_2), \;\; \text{ha } x_1^2+y_1^2 > x_2^2+y_2^2, \\ &\sum_{x,y} w_{mean}(x,y) = 1. \end{aligned} \tag{2.11}$$

A gyakorlatban a súlyok gyakran egész számok, és a maszk alkalmazása után a súlyok összegével normálják az eredményt.

A **dobozszűrő** (*box filter*) a legegyszerűbb és a leggyorsabb, azonos súlyokkal rendelkező átlagszűrő. Egy $(2M+1) \times (2N+1)$-es méretű ablakban az eredmény a képértékek egyszerű, nem súlyozott átlaga:

$$g(x,y) = \frac{1}{(2M+1)\times(2N+1)} \sum_{x'=-M}^{M} \sum_{y'=-N}^{N} f(x+x', y+y') \tag{2.12}$$

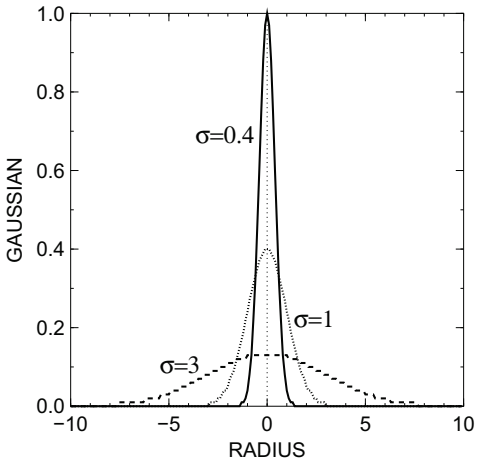
2.3.1. Gauss-szűrő

A Gauss-szűrő a legelterjedtebb álagszűrő, amelyben a súlyokat a normáleloszlás (Gauss-eloszlás) adja:

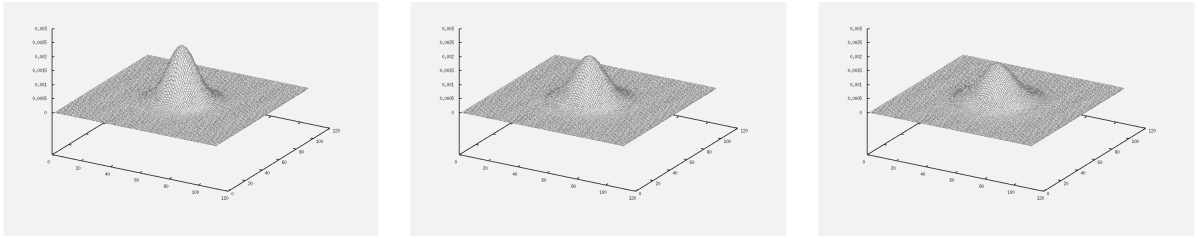
$$w_G(x,y) = \frac{1}{\sum\limits_{(x,y)\in W} e^{-\frac{r^2(x,y)}{2\sigma^2}}} e^{-\frac{r^2(x,y)}{2\sigma^2}}, \tag{2.13}$$

ahol $r^2(x,y) = x^2 + y^2$ a maszk középpontjától való távolság. A Gauss-szűrő maszkja körszimmetrikus, mert csak az r-től függ. Az exponens miatt a maszk harangalakú, a σ paraméter szabályozza a szűrő méretét. Nagyobb σ nagyobb szűrőt és erősebb simítást eredményez.

A paraméternek a szűrő alakjára való hatását a 2.5. ábra szemlélteti. A σ növelésével a függvény ellaposodik és egyre szélesebb lesz, mert a normálás miatt állandó az integrálja, a



2.5. ábra. A Gauss-szűrő alakja növekvő σ-ra 2D-ben.



$\sigma = 9$ $\sigma = 10$ $\sigma = 11$

2.6. ábra. A Gauss-szűrő alakja növekvő σ-ra 3D-ben.

súlyok összege. Ezt a jelenséget a 2.6. ábrán is megfigyelhetjük, ahol a függvény felületét mutatjuk.

Amikor σ nullához tart, folytonos esetben $w_G(x, y)$ a Dirak-féle δ-függvényhez tart. Diszkrét esetben azonban ez csak annyit jelent, hogy egy ponton túl a szűrő csak a közepső pixelt veszi figyelembe, mert ilyen kicsi lesz a mérete. Diszkretizáláskor ugyanis $w_G(r)$-t elvágjuk $r_{max} = k\sigma$-nál, ahol tipikusan $k = 2, 5$, mert ebben benne van a szűrőtérfogat (az "energia") döntő része.

Megjegyzés: Az ellaposodás miatt $w_G(r)$-t nem szabad úgy elvágni, hogy a függvény értékére szabunk egy rögzített alsó korlátot, amely alatt a függvényt lenulázzuk.

A Gauss-szűrő szeparálható (*separable*):

$$w_G(x, y) = w_G(x) \cdot w_G(y), \tag{2.14}$$

mert $\exp(a + b) = \exp(a) \cdot \exp(b)$. Ez biztosítja a gyors implementáció lehetőségét, hiszen egy 2D-s szűrő helyett két 1D-s szűrőt tudunk alkalmazni. A műveletek száma ezzel $O((2r_{max})^2)$-ről $O(2 \cdot 2r_{max})$-ra csökken.

2.3.2. Simítószűrés felhasználásai és tulajdonságai

Simítószűrővel zajszűrést elsősorban a nulla átlagú fehér zaj esetén végezhetünk, mert az átlagban az ellenkező előjelű zajok véletlenszerűen semlegesítik egymást. Minél nagyobb a szűrő, annal nagyobb a semlegesítés valószínűsége és a zajcsökkentés mértéke. Átlagszűrővel történő zajszűrésnek azonban vannak negatív mellékhatásai: a kontrasztcsökkenés és az élelmosódás (*edge blurring*).

Amikor az egyre nagyobb méretű Gauss-szűrőt alkalmazzuk egy képre, a finom részletek egyre jobban tűnnek el. Ezt felhasználják az alulmintavételezésre *subsampling*) és a felbontás csökkentésre úgy, hogy a szűrés után alkalmazzák a decimációt (minden második sor és oszlop elhagyását), utána iterálják a folyamatot. A létrejövő adatstruktúrát *képpiramisnak* hívják.

Ha nem célunk ez a rögzitett mértékű, drasztikus felbontás csökkentés, felépíthetjük az un. **mértékteret** (*scale-space*-t), amely egy képből Gauss-szűréssel nyert képsorozat növekvő σ mellett. Ez az adatstruktúra hatékony, változó részletességű képelemzést tesz lehetővé.

Az alábbiakban összefoglaljuk a simítószűrés főbb tulajdonságait:

- A simítószűrés elmossa az éleket, csökkenti a maximumokat és növeli a minimumokat.
- A kimenet intenzitás tartománya benne van a bemenet intenzitás tartományában. A képdinamika és a kontraszt legtöbbször csökken.
- A simítószűrés új intenzitásértéket hozhat létre, amely a bemeneti képen nem volt. Például, bináris kép simítása több szürkeségi szintű képet eredményez.

- Hibás bemeneti értékek (*outlier*-ek) nagy mértékben befolyásolhatják az eredményt. Az outlier-ek a normális zajszinten felüli, teljesen hibás adatok. (Pl. a só-és-bors zaj véletlenszerű szélső értékekből, outlier-ekből áll.)
- Az átlag tehát *nem robusztus* mennyiség, így az átlagszűrés kevésbé alkalmas a só-és-bors zaj eltávolítására.

2.4. Mediánszűrő

A mediánszűrő eredménye az ablakban levő értékek *mediánja*. A medián meghatározásához szortirozzuk az értékeket növekvő sorrendben és kiválasztjuk a szortirozott sorozat közepén levő értéket. Például, ha egy 3×3-as ablakban az értékek

$$(1, 1, 3, 2, 5, 4, 4, 12, 11),$$

akkor a szortirozott sorozat

$$(1, 1, 2, 3, \mathbf{4}, 4, 5, 11, 12),$$

és a medián **4**.

A medián meghatározását *voksolásnak* lehet tekinteni, amelyben szortirozáskor minden pixel voksol egy helyre. A mediánt mindig a többség, a "közép" választja, a szélső értékek pedig kiesnek, mert a szortirozott sorozat szélére szorulnak.

A medián főbb tulajdonságai a következők:

- A medián meghatározása *nemlineáris* művelet, mert P és Q számsorozatra ugyan $Med(\alpha P) = \alpha Med(P)$, de

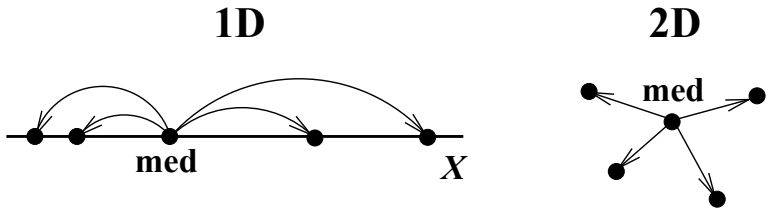
$$Med(P + Q) \neq Med(P) + Med(Q).$$

- Az átlaggal ellentétben a medián *robusztus* statisztikai mennyiség (*robust statistics*). Ha a hibás adatok aránya kevesebb mint 50%, nem befolyásolják az eredményt. A töréspont (*breakdown point*) 50%, azaz az ennél nagyobb adatszennyezettségre már összedől a medián.

Egyes feladatokban a képpontok értéke nem skalár, hanem vektor. Például, amikor képsorozatok alapján elmozdulásvektorokat, sebességeket számítunk, akkor vektormezőket kapunk, ahol hibás vektorok is lehetnek. A medián kiterjesztése több dimenzióra nem triviális feladat, hiszen vektorokat csak hosszúság szerint tudunk szortirozni, ami nem elég.

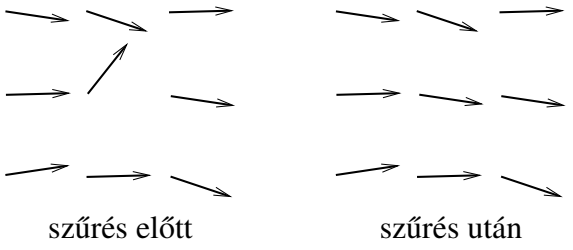
Ennek ellenére létezik egy matematikailag korrekt kiterjesztés, amely az 1D-s esetben megegyezik a fentiekben bevezetett mediánnal. Az ötletet a 2.7. ábra szemlélteti. Egy sorozat számait tekintsük pontoknak az X tengelyen. A medián össztávolsága a többi ponttól mindig a legkisebb. Más szóval, a medián mindig a *legbelsőbb pont*. Ez a tulajdonság bizonyíthatóan ekvivalens a mi medián-definíciónkkal. A medián fogalmát így ki lehet terjeszteni többdimenziós terekre, vektorokra is.

A mediánszűrő az alábbi tulajdonságokkal rendelkezik:



2.7. ábra. Mediánszűrő többdimenziós kiterjesztése.

- A mediánszűrő eltávolítja a só-és-bors zajt úgy, hogy nem mossa el az éleket és nem csökkenti a kontrasztot.
- A mediánszűrő törli a vékony vonalakat, ha a vonalvastagság kevesebb mint a szűrőméret fele. Ilyenkor a háttérpixelek többségben vannak az ablakban és ők adják a mediánt.
- A mediánszűrő lekerekíti a sarkokat.
- A vektoros mediánszűrőt vektormezők javítására és simítására használják. A szűrő módosítja a hibás vektorokat, amelyek elütnek a környezettől. Iteratív alkalmazásával elsimítjuk a vektormezőt. A folyamatot a 2.8. ábra illusztrálja.

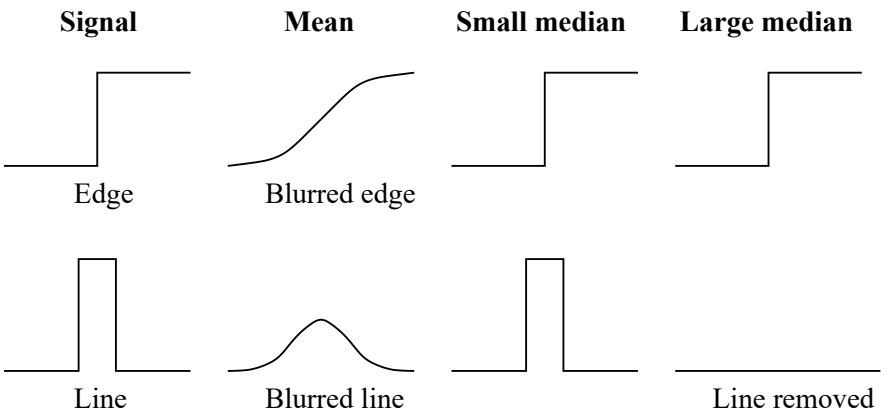


2.8. ábra. Vektoros mediánszűrő alkalmazása.

2.5. Átlag- és mediánszűrő összehasonlítása

Ahhoz, hogy jobban megértsük az átlag- és mediánszűrő közötti különbségeket, az alábbiakban ábrákkal és szűrési eredményekkel szemléltejük az algoritmusok működését. A 2.9. ábra demonstrálja, hogy mi történik egy ideális egydimenziós éllel (lépcsőfok alakú jellel) és egy 1D-s vonallal (vonalmetszettel). Az ábrán a felső sor az él, az alsó sor a vonal feldolgozását mutatja.

Méretétől függetlenül az átlagszűrő mindig elmossa az élet, a mediánszűrő viszont érintetlenül hagyja. A vonal is elmosodik átlagszűrés alatt, miközben a mediánszűrés eredménye



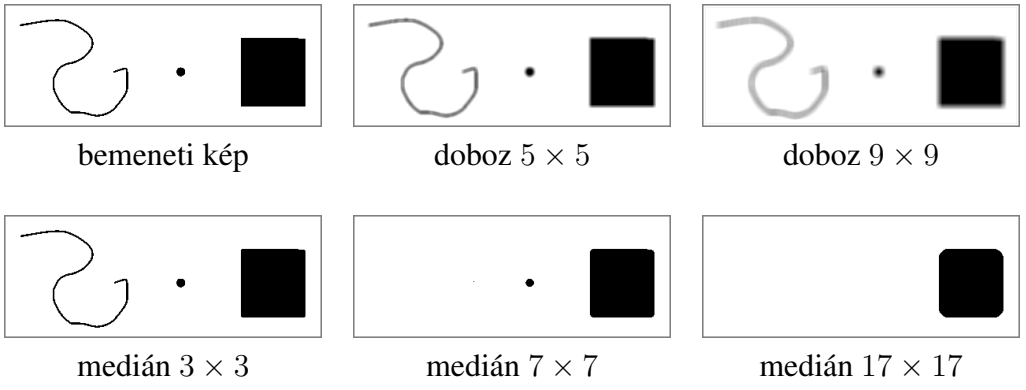
2.9. ábra. Ideális él és vonal szűrése 1D-ben.

drasztikusan függ a szűrő méretétől. Kisebb szűrőre a vonal nem változik, de teljes egészében eltűnik, amikor a szűrőméret meghaladja a vonalvastagság kétszeresét.

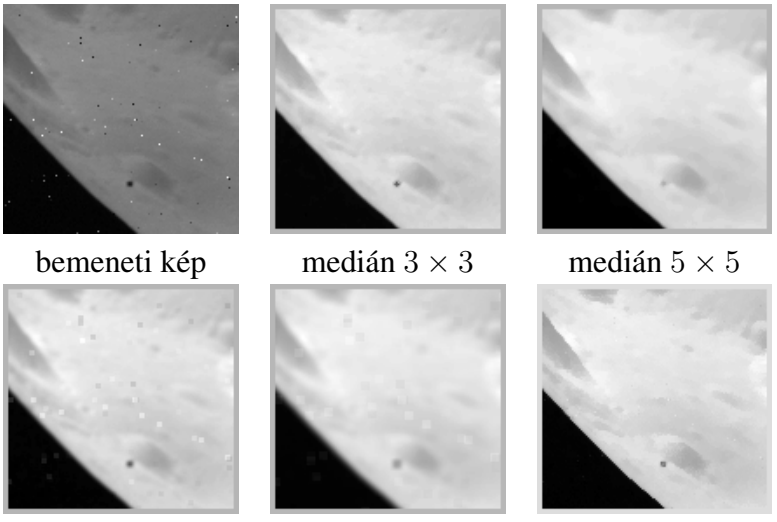
A 2.10. ábrán bináris képek szűrésére mutatunk példákat növekvő szűrőméret mellett. Az előző ábrával összhangban, a dobozszűrő csak elmossa az alakzatokat, de nem tünteti el. A négyzet sarkai egy kicsit lekerekednek. A mediánszűrő hatása látványosabb: amint a szűrőméret eléri a kritikus határt, a megfelelő kis vastagságú alakzat megszűnik. A négyzet olyan nagy, hogy még a nagyméretű mediánszűrő sem tudja eltüntetni, de szemmel láthatóan lekerekíti a sarkait.

Végül a 2.11. ábra olyan szürke képeket mutat, amelyekkel a só-és-bors zaj szűrését illusztráljuk. Az eredményképeket az intenzitás szerint újraskáláztuk, ezért világosabbak, mint ez eredeti kép. Az "adap.dob." rövidítés egy adapív szimmetrikus dobozszűrőt takar, amelyről később lesz szó.

Láthatjuk, hogy a dobozszűrő nem tünteti el a só-és-bors zajt, csak elmossa. Ráadásul, az élek és a részletek egyre halványabbak lesznek. Ezzel szemben, a mediánszűrő hatékonyan csökkenti a zajt, szemmel látható mellékhatások nélkül. Az utolsó kép egy adaptív szűrő



2.10. ábra. A doboz és a medián szűrő összehasonlítása bináris képekre.



2.11. ábra. A doboz és a medián szűrő összehasonlítása só-és-bors zajra.

eredményét mutatja, amelyben ugyan van átlagolás, de okos módon történik és nem mossa el az élet. A probléma tehát itt nem az átlaggal van, hanem azzal, hogy minek az átlagát vesszük. Ehhez a kérdéshez az adaptív szűrés ismertetése során még visszatérünk.

2.6. Laplace-szűrő

A folytonos Laplace-operátor definíciója a következő:

$$\Delta f(x,y) \doteq \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \Big(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\Big) f \tag{2.15}$$

Ahhoz, hogy a diszkrét esetben egyszerű 3×3-as maszkot kapjunk, a deriváltakat különbségekkel közelítjük:

$$\frac{\partial f}{\partial x} \longrightarrow \begin{bmatrix} -1 & 1 & 0 \end{bmatrix}$$

$$\frac{\partial^2 f}{\partial x^2} \longrightarrow \begin{bmatrix} -1 & 1 & 0 \end{bmatrix} - \begin{bmatrix} 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$$

$$\Delta f \longrightarrow \begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Normalizálás után az alábbi közelítő Laplace-szűrőt, w_L-t, kapjuk:

$$\Delta f(x,y) \approx f(x,y) - Av(x,y) = f(x,y) * w_L, \tag{2.16}$$

$$\frac{1}{4}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

4 szomszéd 8 szomszéd

2.12. ábra. Egyszerű maszkok Laplace-szűrésre.

ahol Av a szomszédos képelemek átlaga:

$$Av(x,y) \doteq \frac{1}{4}\Big[f(x-1,y) + f(x,y-1) + f(x+1,y) + f(x,y+1)\Big]$$

Ez a megoldás a 2.12. ábrán látható, 4-szomszédos Laplace-szűrőt jelenti. Amennyiben az összes 8 szomszédot vesszük figyelembe, az ábrán bemutatott másik maszkot kapjuk.

Az alábbiakban felsoroljuk a Laplace-szűrő főbb tulajdonságait:

- Az eredmény közel áll az eredeti és a simított kép különbségéhez. A lassú képváltozásokat levonjuk, a gyors változások megmaradnak. Ha nincs változás, nulla az eredmény (válasz, *response*).
- A kimeneti kép értéktartománya elvileg $[-255, 255]$. Egy pixel és a szomszédjai különbsége azonban gyakran kicsi, ezért a gyakorlatban az értéktartomány lényegesen szűkebb.
- A Laplace-szűrő kiemeli az intenzitás-változásokat és a finom részleteket: kontúrokat, foltokat, vékony vonalakat.
- A szűrő zaj-érzékeny, mert magasrendű deriváltakat tartalmaz. Egy simítószűrőt alkalmazhatunk elötte, hogy a képfüggvény deriválható legyen.
- *Laplacian-of-Gaussian* (LoG) szűrő a Laplace- és a Gauss-szűrő kombinációja:

$$w_{LoG} = w_G * w_L \tag{2.17}$$

  A Gauss-szűrő alkalmazása után a képfüggvény simább, deriválhatóbb lesz, ezért w_{LoG} kevésbé zajérzékeny, mint w_L. Mint később látni fogjuk, a LoG szűrő nulla-átmenetei, előjel-váltásai *élpontok*.

Amikor megjelenítünk egy Laplace-szűrt képet, számolnunk kell azzal, hogy – szemben a simítószűrőkkel – az eredmény negatív is lehet. Két lehetőségünk van: az abszolútérték leképezés, vagy a normalizált érték leképezés. Az első esetben elveszítjük az információ egy részét, mert elvész a változás előjele. Viszont jól láthatjuk azokat a képrészleteket, ahol finom változások vannak. (Ezek tipikusan jól texturált rágiók.) A másik esetben az értékeket leképezzük a $[0, 255]$ tartományra és így ábrázoljuk az eredményképet. Ez nem jár információ-veszteséggel, de gyakran kevésbé szemléletes képet eredményez.

bemenet Laplace abszolút Laplace normalizált

2.13. ábra. Példa Laplace-szűrő alkalmazására.

A fentieket illusztrálja a 2.13. ábra, ahol mind a két megoldásra adunk példát. A leképezéstől függően más és más részletek látszanak. Megfigyelhető, hogy a kontúrok és más jól texturált képrészek ki vannak emelve, a fokozatos képváltozások pedig el vannak nyomva.

2.7. Gyors szűrők

Egy szűrő számításigénye több tényezőtől függ, ezek közül a szűrő mérete a leginkább kézenfekvő paraméter. Nagy felbontású képek esetén gyakori, hogy nagy méretű szűrőket kell alkalmazni, amelyek hatékony megvalósítása kulcskérdés, hiszen a direkt, definíció szerinti megvalósítás túl lassú lesz. Ebben a fejezetben a probléma két hatékony megoldását ismertetjük, nevezetesen, a *szeparálható szűrőket* és a *futószűrést*.

2.7.1. Szeparálható szűrők

Definíció szerint egy 2D-s szeparálható szűrő akkor szeparálható, ha két 1D-s szűrőre bontható:

$$w(y,x) = v(y) \cdot u^T(x)$$

ahol $u^T(x)$ a transzponált (horizontális) vektor, $\cdot$ a diadikus szorzat. Ez azt jelenti, hogy a szűrőmátrix (maszk) minden eleme a két 1D-s szűrő megfelelő elemeinek a szorzata. A definíciót a 2.14. ábra szemlélteti, amelyen a baloldalon egy 3×3-as szeparálható szűrő látható, a jobboldalon pedig elmagyarázzuk, hogy mit is jelent egy oszlopvektor és egy sorvektor szorzata.

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

	1	2	1
1	1	2	1
2	2	4	2
1	1	2	1

2.14. ábra. Szeparálható szűrő példája.

Egy $D_W \times D_W$-s ablakra a műveletigény minden képpontban eredeti szűrő esetén $O(D_W^2)$, a szeparálható szűrő esetén pedig $2 \cdot O(D_W)$. Minél nagyobb a szűrő, annál nagyobb a nyereség.

De hogyan bontsunk egy 2D-s szűrőmátrixot több 1D-s szűrő lineáris kombinációjára? Használhatjuk erre a Szinguláris Érték Dekompozíciót (*Singular Value Decomposition*). Az SVD mindig ad eredményt, de nem biztos, hogy az gyorsabb lesz, mint az eredeti 2D-s változat. Ugyanis a sebesség függ az 1D-s szűrők számától, amely nagy is lehet.

Mint már tudjuk, a Gauss-szűrő szeparálható:

$$w_G(x, y) = w_G(x) \cdot w_G(y)$$
$$w_G(x) = C \cdot e^{-\frac{x^2}{2\sigma^2}},$$

ahol C a normalizáló tényező. A dobozszűrő is szeparálható, mert egy dobozszűrő mátrix két 1D-s egységvektor szorzata. De mint rövidesen látjuk fogjuk, ez nem a legjobb megoldás: a dobozszűrő futó implementációja még gyorsabb.

2.7.2. Futószűrők

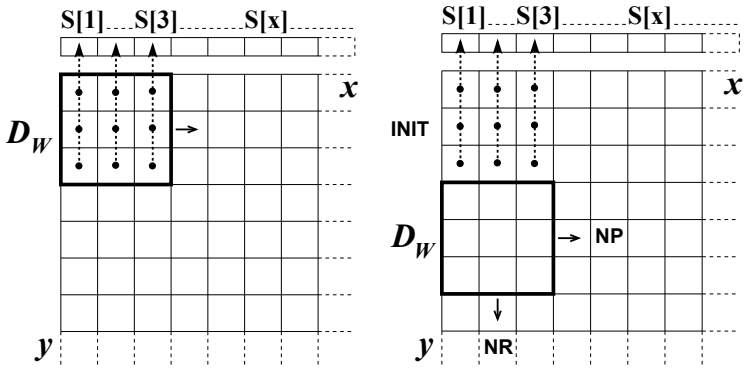
A futószűrés ötlete nagyon egyszerű. Amikor az ablak a következő pozícióba lép, ne számítsuk ki az új értéket az eredeti definíció szerint, hanem használjuk fel az előző pozícióban kapott értéket és módosítsuk azt! A felfrissítés (*update*) művelet azért lehet hatékony, mert az ablak tartalma csak kis mértékben változik: egy oszlop kilép, egy oszlop belép.

Amilyen egyszerű maga az ötlet, olyan bonyolult lehet annak a megvalósítása, hiszen futószűrő megoldások különböző szűrőkre léteznek, így a dobozszűrőre és a mediánszűrőre. A futószűrés kiterjeszthető tetszőleges alakú ablakra is, ahol még nehezebb a megoldás. A módszer hatékonysága az eredmény kiszámítási módjától függ. Egy additív mennyiség, például az átlag könnyen módosítható, egy nemlineáris mennyiség, például a medián nehezebben.

Az ötletet más, bonyolultabb matematikai műveletekre is ki lehet terjeszteni egy futó ablakban (*data window*). Többek között léteznek ilyen algoritmusok a Gyors Fourier Transzformációra (FFT) és a Szinguláris Érték Dekompozícióra. A mi célunk azonban a lényeg bemutatása, ezért a legegyszerűbb, bár nem triviális esetre, a futó (*running*) dobozszűrőre szorítkozunk.

A futó dobozszűrő működését a 2.15. ábra szemlélteti. A kép mérete M sor és N oszlop, az ablakméret $D_W \times D_W$. Az algoritmus adatstruktúrája az $S[x]$ tömb, hossza N. Az adatstruktúrát úgy inicializáljuk, hogy a kezdő sorra kiszámítjuk az $S[x]$ oszlopösszegeket. Ez az egyetlen művelet, amely az ablak méretétől függ.

Minden sor elején a kezdő pozícióra kiszámítjuk az S_W ablakösszeget. Amikor egy soron belül a következő helyre lépünk (Next Position, NP), felfrissítjük az aktuális S_W értéket, ami csak abból áll, hogy levonjuk a kilépő és hozzáadjuk a belépő oszlopösszeget. Minden sor végén a következő sor elejére ugrunk (Next Row, NR), ilyenkor felfrissítjük az összes $S[x]$-t,



2.15. ábra. A futó dobozszűrő inicializálása és működési elve.

ami egy kilépő pixelérték levonásából és egy belépő pixelérték hozzáadásából áll. Ezek a műveletek már függetlenek az ablakmérettől.

Amikor a kép sokkal nagyobb mint a szűrő, $M \gg D_W$ – ami az esetek döntő többségében teljesül – a futó dobozszűrő műveletigénye az $S[x]$ inicializálása erejéig nem függ az ablakmérettől. Így a gyakorlatban a 25×25-ös futó dobozszűrő csaknem ugyanolyan gyors, mint az 5×5-ös. Ha a kép nem négyzetes, és $N \gg M$, transzponáljuk a képmátrixot, a szűrés után pedig állítsuk vissza!

2.8. Képpiramis

Futólag már említettük a képpiramist, mint változó felbontású képstruktúrát. Ebben a fejezetben részletezzük a fogalmat, bevezetjük a Gauss- és a Laplace-képpiramist, algoritmusokat adunk a két képpiramis hatékony felépítésére, valamint példát mutatunk a Laplace-képpiramis alkalmazására.

2.8.1. Gauss-képpiramis

A *Gauss-piramis* a csökkenő felbontású képmásolatok sorozata, amely az alábbi műveletek segítségével jön létre:

1. **Képszűrés** kisméretű Gauss-szűrővel.
2. **Alulmintavételezés**, tipikusan, decimálás révén.
3. **Iteráció**, azaz a két művelet megismétlése.

A Gauss-piramis létrehozására használt szeparálható 5×5-ös Gauss-szűrő standard alakja

$$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}^{\mathrm{T}}, \tag{2.18}$$

2.16. ábra. Gauss-képpiramis példája.

a decimálás pedig nem más, mint minden második sor és oszlop törlése. Ez rögzitett arányú felbontás-csökkenést eredményez: a következő szinten a felbontás mindig a felére csökken.

Az eljárást a 2.16. ábra illusztrálja, ahol háromszintű Gauss-piramist láthatunk, ebből az alsó szint, a piramis alja maga az eredeti kép. A képet elsimítjuk, a felbontás a felére csökken. A finom részletek fokozatosan eltűnnek, ami lehetőséget teremt változó részletességű képelemzésre.

2.8.2. Laplace-képpiramis

Laplace-piramis annyiban különbözik a Gauss-piramistól, hogy a Gauss-szűrő helyett a Laplace-szűrőt használjuk. Pontosabban, a Gauss-piramis szintjeit szűrjük meg Laplace-szűrővel, mert az előbbi biztosítja a szükséges képsimítást. A Laplace-piramis alja az eredeti felbontású Laplace-szűrt kép.

A gyakorlatban a Laplace-szűrő helyett a szeparálható Gauss-szűrőt (2.18) alkalmazzák és a piramis alja az eredeti, és a Gauss-szűrt kép különbsége lesz:

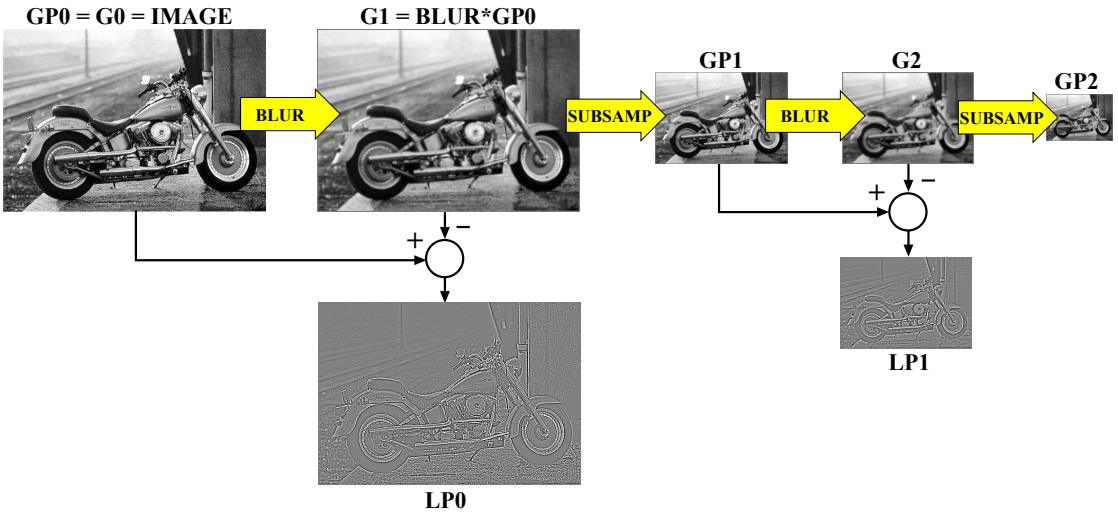
$$LapPyrBottom = Image - GaussImage.$$

Ez azért lehetséges, mert, mint tudjuk (2.16 alapján)

$$LaplaceImage \approx Image - BlurredImage$$

A Laplace-piramis építési folyamatát a 2.17. ábra illusztrálja és részletezi. Az ábrán a Blur az alábbi szeparálható 5×5-ös Gauss-szűrővel történő simítást (elmosást) jelenti:

$$\tfrac{1}{16}\begin{bmatrix}1 & 4 & 6 & 4 & 1\end{bmatrix} \cdot \tfrac{1}{16}\begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix} = \tfrac{1}{256}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\4 & 16 & 24 & 16 & 4\\6 & 24 & 36 & 24 & 6\\4 & 16 & 24 & 16 & 4\\1 & 4 & 6 & 4 & 1\end{bmatrix}$$

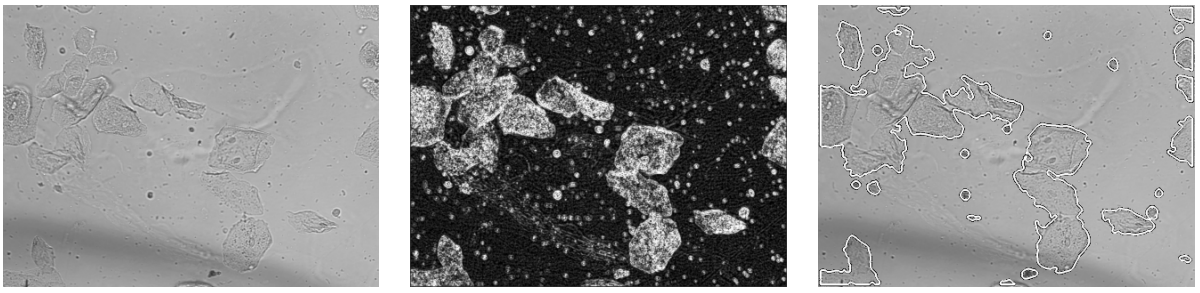


2.17. ábra. Laplace-képpiramis építése Gauss-szűrővel.

2.18. ábra. Laplace-képpiramis példája.

A 2.17. ábrán egy példát láthatunk, amelyet érdemes összevetni a 2.16. ábrával. A Gauss-piramissal ellentétben a Laplace-piramis megőrzi a finom képrészleteket, miközben a lassú képváltozások eltűnnek. Ez utóbbi tulajdonság lehetőség nyújt a lassan változó háttér eltüntetésére, ahogy ezt egy konkrét alkalmazás kapcsán mindjárt látni is fogjuk.

A 2.19. ábrán különböző méretű, alakú és textúrájú sejtek láthatók. Egyes sejtek kontrasztja igen alacsony. Az alkalmazásban a cél a sejtrégiók meghatározása volt. A piramis kiemeli a sűrű képváltozású régiókat. Az objektumok jól láthatók, pedig a kontraszt alacsony és a háttér változó. Minden sejtrégiót sikerült kiemelni, és a zajos képen nincs hamis detektálás (*false positive*). Az alacsony kontraszt ellenére a határok elég pontosak.



bemeneti sejtkép | piramis 2.szintje, kinagyítva | detektált sejtek

2.19. ábra. Sejtrégiók kiemelése Laplace-piramis segítségével.

2.9. Adaptív zajszűrés

Miért is van szükség az adaptivitásra? Eddig kizárólag a nemadaptív szűrőkkel foglalkoztunk, amikor rögzített volt a környezet-kiválasztás (pl. fix méretű ablak) és rögzített volt a környezeten definiált operátor is (pl. a medián). Az adaptivitás a *lokális kontextus* felhasználása, amitől az eredmény javulását várjuk: elkerülhetjük az átlagszűrőkre jellemző élelmosódást, valamint a mediánszűrőkre jellemző sarok-lekerekítést.

Ezen nemkívánatos hatások elsődleges oka az, hogy nem vesszük észre, hogy az ablak az objektum és a háttér határán van, emiatt az eredmény kiszámításakor összekeverjük a két különböző intenzitási osztályhoz tartozó értékeket.

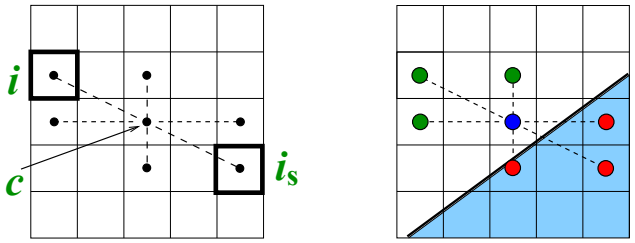
Az adaptív környezet-kiválasztással megpróbáljuk elválasztani az objektum képelemeket a háttér képelemektől, a releváns értékeket pedig a zajtól. Csak a releváns pixeleket fogjuk felhasználni. Eddig a környezet a teljes ablak volt, most az ablakban csak bizonyos képelemeket fogjuk figyelembe venni.

A kiválasztott környezeten definiált operátor viszont fix marad. Eddig is fix függvényt használtunk, most is ez lesz.

A képelem-kiválasztásra egy $n \times n$-es ablakban több lehetőség van, ezek közül az alábbiakban felsorolunk néhányat:

- **Standard** környezet: az összes n^2 pixel.
- k **legközelebbi szomszéd** (k*-nearest neighbours,* k*-NN*): az a k pixel, amely intenzitás szerint legközelebb van a c középpixelhez. A k egyik lehetséges beállítása $k = n\left[\frac{n}{2}\right] + (n-1)$. Például, ha $n = 3$, akkor $k = 5$.
- **Szigma-legközelebbi szomszédok**: az i pixelt akkor választjuk, ha $|I(i) - I(c)| < k\sigma_{zaj}$. A σ_{zaj} zajszórás becslésére felhasználhatjuk a kép háttér részeit, ahol a változás elsősorban a zajnak tulajdonítható. Elterjedt beállítás $k = 2$.

A fenti receptek közös gyenge pontja, hogy nem vesznek figyelembe geometriai relációkat az ablakon belül. A **szimmetrikus legközelebbi szomszédok** módszere, amelyet



szimmetrikus pixelpár választás kontúron

2.20. ábra. Szimmetrikus legközelebbi szomszédok.

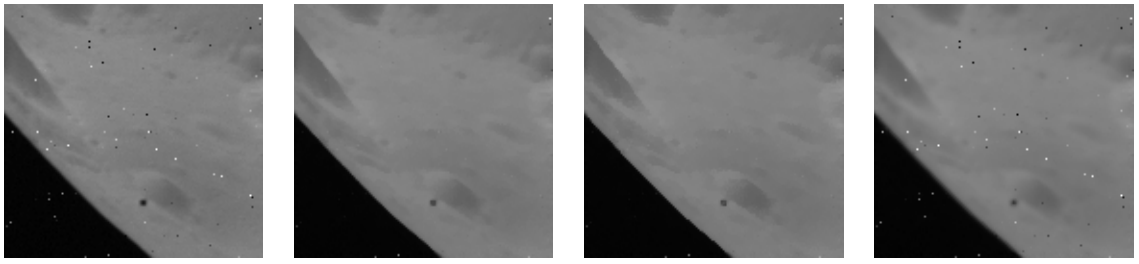
a 2.20. ábra szemléltet, tartalmaz ilyen relációkat. Itt az i pixelt akkor választjuk, ha

$$|I(i) - I(c)| < |I(i_s) - I(c)|\,,$$

ahol $\{i, i_s\}$ a közép-szimmetrikus képelemek egyik párja; az összes ilyen párt kell megvizsgálni.

A lokális kontextus ebben az esetben a pixelek intenzitása és elrendezése. Az eljárás különösen az élekre van jótékony hatással: az él ugyanazon oldalán levő képelemeket választja, ezzel elkerüli "az élen keresztül történő átlagolást" és az élelmosódást. A módszerrel el lehet kerülni a sarkok lekerekítését is, ha nem párosával, hanem négyesével, keresztszerűen vizsgáljuk a szimmetrikus pixeleket és csak a legközelebbit választjuk. (Mivel a pixelek háromnegyede elvész, csak nagyobb ablakkal alkalmazható az eljárás.)

A 2.21. ábra illusztrálja, hogy mind a k-NN szűrő, mind a szimmetrikus legközelebbi szomszéd szűrő képes eltávolítani a só-és-bors zajt akkor is, ha a kiválaszott értékek átlagát, nem mediánját vesszük. Ezzel szemben láthatjuk, hogy a szigma-szűrő nem tünteti el a só-és-bors zajt. Ennek az oka, hogy egy zajos pixelre az $I_{zajos} \pm 2\sigma_{zaj}$ intervallum nem tartalmaz zajmentes pixeleket, mert $|I_{zajos} - I_{zajmentes}| > 2\sigma_{zaj}$. Emiatt a szűrő a I_{zajos} zajos értéket választja és a zajt nem távolítja el. Ezen az sem segít, ha az átlag helyett a mediánt számítjuk.



input kép k-NN átlag szimm. átlag szigma átlag

2.21. ábra. Adaptív 5×5-ös szűrők összehasonlítása a só-és-bors zajra.

3. fejezet

Megfeleltetés és mintaillesztés

3.1. Megfeleltetés és illesztés a számítógépes latásban

Ebben a fejezetben elsősorban a *mintaillesztés* kérdéseivel foglalkozunk. A mintaillesztés egy alacsony szintű, elemi, de kritikus felismerési probléma, amikor el kell dönteni, hogy egy képminta hasonlít-e egy képrészre. Az egyszerűnek tűnő feladatot nagymértékben bonyolítja, hogy a mintavétel mindig más körülmények között történik, mint annak a képnek a felvétele, amelyben a hasonló részt keressük. Ezért csak olyan módszerek jöhetnek számításba, amelyek robusztusak a változásokkal szemben.

Mielőtt rátérnénk a mintaillesztési algoritmusokra, célszerű megismerni azokat a számítógépes látási feladatcsoportokra, ahol megjelenik a *megfeleltetés*, azaz a képpontok és környezetek azonosítása két vagy több olyan képen, amely ugyanazt a színteret mutatja, csak másképpen. A feladatok és alkalmazások puszta felsorolása jelzi az illesztés jelentőségét.

3.1.1. Megfeleltetést igénylő feladatcsoportok

Az **adatregisztráció és fúzió** problémája akkor merül fel, amikor **különböző érzékelőkkel** adatokat, tipikusan képeket veszünk fel ugyanarról a tárgyról. Orvosi alkalmazásokban, például az emberi testről MRI, PET és röntgen képeket készítenek. A különböző fizikai eredetű képeket össze kell illeszteni, megfeleltetni. Az orvosi képalkotásban az ilyen képeket *modalitásoknak* hívják és multimodális képregisztrációról beszélnek. (Az illesztés szó angol megfelelője a *matching*, a megfeleltetésé pedig a *correspondence*.)

Ha nem képi, hanem más mérési adatokról, például 3D-s ponthalmazokról van szó, akkor az adatregistráció szót használják, így például az angol *3D data registration* kifejezés legtöbbször a mért felületek vagy pontfelhők illesztését jelenti. Ha azonban különböző adatstruktúrájú adatokat illesztenek össze, akkor inkább adatfúzióról beszélnek. Ez lehet többek között a video és a hang vagy a kép és a felület adatfúziója, regisztrációja.

A **mozgáselemzés** problémaköréhez is több megfeleltetési feladat tartozik. Ebben az esetben **különböző időpontokban** képeket készítünk változó, mozgó színtérről, például, ha

arcmozgást, kifejezést vizsgálunk, vagy térmegfigyelést végzünk. Ilyenkor kereshetjük az egymásnak megfelelő pontokat, az elmozdulásokat (*displacements*) és a változásokat, ami a mozgáselemzés lényegét alkotja. A jellegzetes példák a mozgáskövetés (*motion tracking*) és az optikai áramlás becslése (*optical flow estimation*).

A **sztereó látás** egy harmadik fontos terület, ahol a megfeleltetésnek kiemelt szerepe van. Itt **különböző szemszögből** készítünk képeket egy színtérről és keresünk egymásnak megfelelő képpontokat. Az illesztés biztosítja a diszparitásokat, vagyis a két felvétel közötti pontelmozdulásokat. A diszparitás és a bázistávolság alapján triangulációval meghatározzuk a mélységet. (A bázistávolság (*baseline*) a kamerák közötti távolság, a mélység (*depth*) pedig a kamera és a 3D-s pont közötti távolság.) A klasszikus sztereó esetén kalibrált kamerapárral (*stereo rig*) dolgozunk, általános esetben a több felvétel alapján történő 3D-s rekonstrukció a feladat.

3.1.2. A megfeleltetés kritikus problémái

A megfeleltetési probléma sikeres megoldása kritikus kérdés a számítógépes látásban, mert megnyitja az utat a további problémák megoldása felé. Ehhez azonban több elvi nehézséget kell leküzdeni. Az egyik kulcskérdés a megfeleltetési módszereknek a képalkotási változásokkal szembeni robusztussága, beleértve a térbeli (látószög, távolság, perspektíva) és a fotometrikus (megvilágítás, fényvisszaverődés) változásokat.

A más tényezőkkel szembeni robusztusság is fontos. Ezek közé tartoznak a zaj, a képtorzítás, és különösen a takarás (*occlusion*), mert nem minden pontnak van megfelelője, és nem tudjuk, hogy melyiknek nincs. Tehát, felmerül a láthatóság (*visibility*) problémája.

Jegyzetünkben a 3D-s térbeli tényezőket nem vizsgáljuk és lényegesen leszűkítjük a szóba jöhető változások körét. A következő transzformációkat fogjuk vizsgálni:

- **Geometriai**: 2D-s eltolás és elforgatás.
- **Fotometrikus**: intenzitás-skálázás és eltolás $I' = aI + b$, ahol I az eredeti, I' a módosított intenzitás.

A lineáris intenzitás-transzformáció praktikus jelentőséggel bír. Némileg egyszerűsítve a valós helyzetet, az a szorzó a *direkt megvilágítás* erőssége, amely az objektumra irányított, közvetlen fényt jellemzi. A b paraméter pedig a *szórt fény* (*ambient light*) erőssége, ami a színtér globális fényességét tükrözi, azaz a minden irányból érkező fényt.

3.2. Mintaillesztés

Mint már említettük, mintaillesztéssel olyan kisebb képrészeket keresünk, amelyek egy adott mintára hasonlítanak. A mintát magából a képből, vagy egy másik képből vehetünk. Így sztereó illesztésnél is lokális mintát keresünk, amikor pontokat (pontkörnyezeteket) megfeleltetünk, korrelálunk. Mozgásbecslés is gyakran mintaillesztéssel, *block matching*-gel történik, de kereshetünk minta szerint egy képi adatbázisban is.

Formalizálva a mintaillesztés fogalmát, minden lehetséges (x, y) pozícióban összehasonlítjuk a $w(x', y')$ képmintát (részképet) az $f(x', y')$ képpel. Más szóval minden (x, y) pontban *illesztjük* a $w(x', y')$-t az $f(x + x', y + y')$-hez. Olyan (x, y) helyeket keresünk, ahol vagy kicsi az *eltérés* a minta és a kép között, vagy nagy a *hasonlóság* a minta és a kép között (angolul: *low dissimilarity*, *high similarity*).

Az alábbiakban megadunk és elemezünk néhány eltérési és hasonlósági mértéket, amely egyre összetettebb lesz. A bonyolultság növelése nem öncélú, hanem azt a célt szolgálja, hogy a végén olyan mértéket kapjunk, amely invariáns lesz az intenzitás tetszőleges lineáris transzformációjára, vagyis a direkt megvilágítás és a szórt fény változására.

3.2.1. Eltérési mértékek

A legegyszerűbb gyakran alkalmazott eltérési mérték a **négyzetes különbségek** összege:

$$SSD(x, y) = \sum \Big(f(x + x', y + y') - w(x', y') \Big)^2, \tag{3.1}$$

ahol az egyszerűség kedvéért

$$\sum = \sum_{\substack{(x',y')\in W \\ (x+x',y+y')\in F}},$$

és a szűréshez hasonlóan W a lokális pozíciók halmaza a w mintán belül, F pedig a globális pozíciók halmaza az f képen belül.

Az SSD a *Sum of Squared Differences* rövidítése. A mérték *nem* invariáns, mert nem találja meg az elforgatott mintát, és egyáltalán nem kezeli a megcélzott lineáris intenzitásváltozást.

A lineáris változás eltolásparaméterének a kezeléséhez bevezetjük az **intenzitás-eltolásra korrigált** (*shift-corrected*) SSD-t:

$$SSD_{SC}(x, y) = \sum \Big(\big[f(x + x', y + y') - \overline{f}(x, y) \big] - \big[w(x', y') - \overline{w} \big] \Big)^2, \tag{3.2}$$

ahol $\overline{f}(x, y)$ az intenzitás-átlag az aktuális képrészen, amelyet minden (x, y) pozícióban ki kell számítani, amihez felhasználhatjuk a futó dobozszűrőt. $\overline{w}$ a minta átlaga, ezt csak egyszer kell meghatározni.

$SSD_{SC}(x, y)$ segítségével kompenzálhatjuk az intenzitás-eltolást, és ezzel a mérték nem lesz érzékeny a szórt fény változására. Viszont továbbra sem tudjuk kezelni a direkt megvilágítás változását. Hogy ezt a problémát is megoldjuk, olyan hasonlósági mértéket vezetünk be, amely megfelelő normálással elvégzi a feladatot.

3.2.2. Hasonlósági mértékek

A **nemnormalizált kereszt-korreláció** definícióját már ismerjük, hiszen a konvolúciós szűrés kapcsán már találkoztunk vele. A könnyebb áttekintés érdekében azonban most megis-

ételjük:

$$CC(x,y) = \sum f(x+x', y+y') \cdot w(x', y').$$

A $CC(x,y)$ függvény formailag ugyanaz, mint az f képnek a w maszkkal való szűrése. A kereszt-korreláció és a konvolúció tulajdonságait szintén ismerjük, ezért amit a szűrésről tudunk, itt is alkalmazható, beleértve a normalizálást, a szeparálhatóságot és a futószűrést. Vigyázni kell azonban arra, hogy a szűrőinkkel ellentétben a $w(x', y')$ nem feltétlenül szimmetrikus, sőt csak kivételes esetekben az. Ez korlátot szab ezen tulajdonságok felhasználására.

A $CC(x,y)$ mérték *nem* invariáns sem az intenzitás-eltolásra, sem a skálázásra. A másik gond vele az, hogy amikor $w > 0$ és f nagy, $CC(x,y)$ is nagy, függetlenül attól, hogy w és f hasonlítanak-e vagy sem. Ezekel a problémákat normalizálással fogjuk kiküszöbölni. A **normalizált kereszt-korreláció**

$$NCC(x,y) = \frac{1}{N_1} \sum \big[f(x+x', y+y') - \overline{f}(x,y)\big] \cdot \big[w(x', y') - \overline{w}\big], \tag{3.3}$$

ahol

$$\begin{aligned} N_1 &= \sqrt{S_f(x,y) \cdot S_w}, \\ S_f(x,y) &= \sum \big[f(x+x', y+y') - \overline{f}(x,y)\big]^2, \\ S_w &= \sum_{(x',y') \in W} \big[w(x', y') - \overline{w}\big]^2. \end{aligned}$$

$S_f(x,y)$-t sokszor kell kiszámítani, S_w-t csak egyszer.

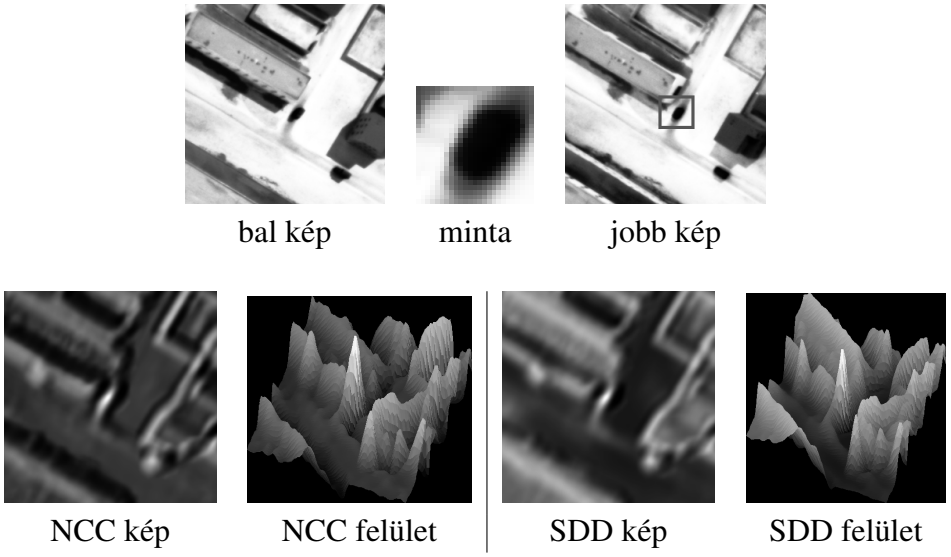
A normalizált kereszt-korreláció *invariáns minden lineáris intenzitás-változásra*, ezzel el is értük a kitűzött célunkat. Azonban a gyakorlatban gond lehet vele ott, ahol a kép alig változik, és $S_f(x,y)$ nagyon kicsi. Az NCC ilyenkor numerikusan instabillá válik, amin úgy lehet valamelyest segíteni, hogy egy kis pozitív ϵ-t adunk hozzá az $S_f(x,y)$-hez. A másik ad hoc megoldás a **módosított NCC**:

$$MNCC(x,y) = \frac{1}{N_2} \sum \big[f(x+x', y+y') - \overline{f}(x,y)\big] \cdot \big[w(x', y') - \overline{w}\big], \tag{3.4}$$

ahol $N_2 = S_f(x,y) + S_w$.

Az MNCC és az NCC között csak a normalizálásban van különbség. Az MNCC-vel elkerülhető a numerikus instabilitás, de elméletileg a mérték csak eltolás-korrigált. A gyakorlaban mégis alkalmazzák, mert a skálázásra sem nagyon érzékeny.

A 3.1. ábrán példát mutatunk valós mintaillesztésre. Egy sztereó képpárt látunk. A jobb képen levő mintát a bal képen keressük. A minta a felső sor középén kinagyítva is látható. Az NCC a normalizált kereszt-korreláció, az SSD a négyzetes különbségek összege, de hasonlóságként átértelmezve. Ehhez a mérték megfelelően normálizált reciprokát vesszük. Mivel ezt mindig megtehetjük, nincs elvi különbség a hasonlósági és az eltérési mértékek között.



3.1. ábra. Mintaillesztés példája. A felső sorban a minta ki van nagyítva.

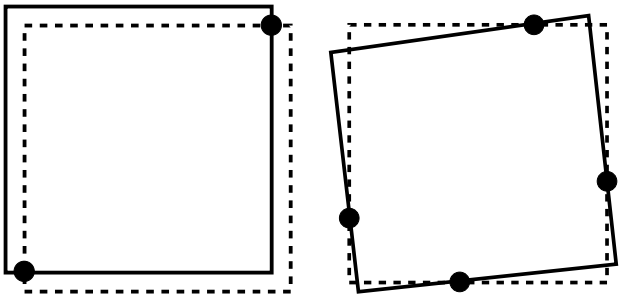
Az eredményt kétféleképpen jelenítjük meg, képként és felületként. Az első esetben a képpont értéke arányos a pontban számított hasonlósági mértékkel. A második esetben a jobb szemléltethetőség érdekében a hasonlósági képet intenzitás-felületként mutatjuk, amelyre rátesszük magát az intenzitást is. Megfigyelhetjük, hogy az NCC eredmény meggyőzőbb, mert a keresett helyen a függvény maximuma jobban emelkedik ki. De azt is látjuk, hogy hamis helyeken is vannak nagy értékek. Ezt a problémát a következő fejezetben tárgyaljuk.

3.3. Robusztusság és lokalizációs pontosság

A mintát és a képet minden képpontban összehasonlítjuk. Csúsztatott mintával dolgozunk, ami ahhoz vezet, hogy nem csak a pontos helyen korrelál a minta és a kép, hanem a szomszédos pontokban is. Nincs garáncia arra, hogy pont ott találunk maximumot, ahol szeretnénk. Zajos és/vagy kissé torzított képre ez a jelenség még erősebb lesz, és a maximum még távolabb lehet az igazi helytől. Előfordulhat, hogy a korrelációs függvény annyira ellaposodik, hogy el is veszítjük az illesztést.

Az, hogy több szomszédos pontban is nagy lehet a keresési művelet eredménye, nem illesztés-specifikus, hanem minden olyan detektálási feladatra jellemző, amely csúsztatott ablakkal történik. Ezzel később az él- és a sarokdetektálás során is szembesülünk. Egy másik probléma, amely inkább a mintaillesztésre jellemző, a hasonlósági függvények korlátozott mivolta. Nem feltétlenül pontosan azt fejezik ki, amit szeretnénk. Emiatt számtalan numerikus és valós példa mutatja, hogy az illesztés nem mindig elég erős, illetve ott is találunk "zajos" hasonlóságot, ahol szemmel nem látjuk.

Kérdés, hogy mivel tudnánk javítani az illesztés élességén? Egyes feladatokban választhatjuk, hogy az egész mintát illesszük-e (*területillesztés*), vagy csak a kontúrján levő ponto-



3.2. ábra. Területillesztés kontra kontúrillesztés: ideális és torz minta.

kat (*kontúrillesztés*). A 3.2. ábra illusztrálja a két lehetőség közötti különbségeket, az esetleges előnyöket és hátrányokat. Az ábrán a mintát szaggatott, a keresett objektumot folytonos vonallal mutatjuk.

Az *ideális* objektum esetén a minta kis eltolására a kontúrátfedés drasztikusan csökken, a területátfedés pedig alig csökken. Ez azt eredményezi, hogy a kontúrillesztés élesebb lesz, mint a területillesztés.

Az *eltorzított* vagy elforgatott objektum esetén a kontúrátfedés kicsi, ezért az objektumot elveszíthetjük. A területátfedés viszont nagy, és nagy valószínűséggel megtaláljuk az objektumot. A kontúrillesztés így kevésbé robusztus.

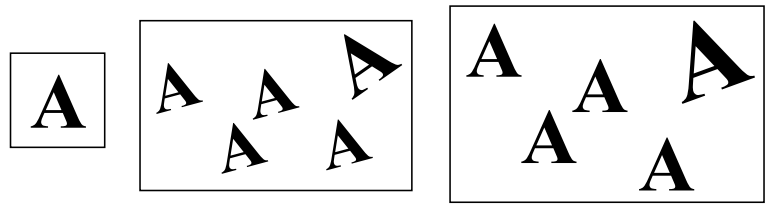
Ez a megfigyelés elvezet a robusztusság és a lokalizációs pontosság összefüggéséig, ellentétéig. A kontúrillesztés élesebb, tehát precízebben lokalizál. Viszont kevésbé robusztus, mert elveszíthetünk objektumokat. Ezzel szemben a területillesztés kevésbé éles és kevésbé precízen lokalizál, viszont robusztusabb. (Teljesen más szempont a végrehajtási sebesség, amely a kontúrillesztésnél legtöbbször nagyobb.)

A robusztusság és a lokalizációs pontosság ellentéte más detektálási feladatoknál is megjelenik, és a kérdés mélyebb, mint amilyennek az első látásra tűnik. Az ellentétet feloldani és a két követelménynek egyszerre eleget tenni nem egyszerű, ez elvi probléma.

3.4. Invariancia, robusztusság, sebesség

Ebben a fejezetben a mintaillesztés néhány fontos problémáját tárgyaljuk, amelyet az alábbiakban foglaljuk össze:

- A *méretváltozásra és az elforgatásra* való invariancia, például, amikor közelebbről és/vagy elforgatva látjuk a mintát.
- A *képtorzítással* szembeni robusztusság, például, kisebb perspektív torzítás esetén.
- A *zajos illeszkedésekkel* szembeni védettség, amikor váratlanul jól illeszkedő, nem keresett képrészek vannak.



3.3. ábra. Képnormalizálás méretre és orientációra.

- A *számításigény*, amely nagy minták esetén jelentősen nő.

3.4.1. Invariancia és robusztusság

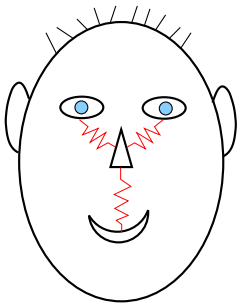
A **méretváltozás és elforgatás** kezelésére több módszert dolgoztak ki. *Képnormalizálással* a képet standard méretre és orientációra transzformáljuk. Az eljárás feltételezi, hogy a képen belül nincs méret- vagy orientáció-változás, és a képorientációt egyértelműen lehet definiálni. Az ötletet a 3.3. ábra szemlélteti, ahol a jobb felső sarokban levő **A** betű mérete és elforgatása eltér a többiétől. A normalizálás után ez a betű nem fog illeszkedni, a többit viszont mintával megtaláljuk. Nyilvánvaló, hogy a nyomtatott szöveg esetén a képorientáció egyértelmű, hiszen a sorok ehhez kellő anizotrópiát biztosítanak. Kérdés, hogy más esetben mi legyen az orientáció?

A másik lehetőséget az *adaptív megoldások* nyújtják, például úgy, hogy minden pozícióban megváltoztatjuk a minta méretét és orientációját és kiválasztjuk a legjobban illeszkedő méretet és orientációt. Ez a megoldás nagyon lassú, amikor a lehetséges méretek és szögek száma nagy, tehát csak kisszámú méret és elforgatás esetén használható.

Ennél praktikusabb és a mai képelemzésben elterjedtebb megoldás az *invariáns leírások* alkalmazása. Nem képeket, hanem olyan jellemzőpontokat és hozzájuk rendelt, lokális képleírásokat hasonlítunk össze, amelyek nem érzékenyek méretváltozásra és elforgatásra.

A **torzítástűrő illesztésre** is lehet ezt a módszert alkalmazni, feltéve, hogy a jellemzőpontok és a lokális képleírások robusztusak a torzítással szemben. Egy másik megközelítés a rugalmasan összekötött alminták használata, amit a 3.3. ábrával illusztrálunk. Az almináakat illesztjük, miközben a "rugók" lehetővé teszik a minta korlátozott változtatását. Ehhez bevezetünk egy célfüggvényt, amely bünteti a nagyobb változtatásokat úgy, hogy ezeknek nagyobb lesz a költsége. A módszer akkor működik jól, amikor az alminták elég jellegzetesek a megbízható illesztéshez.

A **hamis illeszkedések kiszűrésére** nincs sok lehetőség, ha nem rendelkezünk további geometriai jellegű információval és csak a hasonlósági mértékre támaszkodunk. A sztereó illesztésben gyakran alkalmazott módszer az illesztési konzisztencia ellenőrzése, az oda-vissza illesztés. Az egyik képen levő minta az illesztés révén egy párt, társmintát választ a másik képen. Csak akkor fogadjuk el a párosítást, ha az ellenkező irányban is érvényes. Ha az



3.4. ábra. Arcminta mint rugalmasan összekötött alminták rendszere.

ellenkező irányú illesztésnél a kiválasztott társminta nem a kiinduló mintát választja, hanem egy tőle távol levő másik mintát, akkor az illesztést hamisnak nyilványítjuk és eldobjuk.

E miatt a kiinduló pontnak nem lesz párja, nagy valószínűséggel azért, mert a másik képen takarásba került. A 3.5. ábra példát ad az oda-vissza illesztésnek a takart pontok kiszűrésére való felhasználására. Az ábra jobb oldalán levő képek az illesztési hibát mutatják úgy, hogy a világosabb pixelek nagyobb hibával rendelkeznek. A konzisztencia-ellenőrzés kiszűri a takart pontokat, és sok olyan nagy hibájú pont eltűnik, amely az előtérben levő tárgyak szélén van, ahol a takarás leginkább jelentkezik.

3.4.2. Mintaillesztés felgyorsítása

A mintaillesztés felgyorsítására sok trükk létezik, ezeket csak vázlatosan ismertetjük. Az egyik lehetőség, hogy nem képekkel és mintákkal, hanem *lokális jellemzőpontjaikkal* dolgozzunk, például, élekkel vagy sarkokkal. Ez akkor hasznos, amikor a képi sajátságok viszonylag ritkák, de megbízhatóak. Sajnos, ez a megoldás torzításérzékeny lehet, amint ezt a kontúrillesztés kapcsán megtapasztaltuk.

Ha nagy a minta, amelyet sok helyen kereszt-korrelálni kell a képpel, a *Gyors Fourier Transzformációval* (FFT-vel) lényeges sebesség-növekedést tudunk eszközölni. Ezt a követ-

bal kép jobb kép eredet ill.hiba konzisztens ill.hiba

3.5. ábra. Az oda-vissza illesztés hatása.

kező alapvető tétel teszi lehetővé:

$$f \otimes w = IFFT\Big[FFT\big[f(x,y)\big]^* \cdot FFT\big[w(x,y)\big]\Big], \tag{3.5}$$

ahol az $IFFT$ az inverz FFT, X^* az X komplex konjugáltja. Egy $N \times N$-es mintára az FFT műveletigénye $O(N^2 \log N)$, a kereszt-korreláció direkt megvalósítása ezzel szemben $O(N^4)$ műveletet igényel, ami nagyon nagy különbség. Természetesen kisebb minták esetén nem éri meg az FFT alkalmazása. Egyes vélemények szerint a határ valahol a 13×13-es mérétnél húzódik, ami fölött már érdemes megfontolni a megoldást. Minden esetre saját tapasztalatunk alapján tudjuk, hogy 30×3-as ablaknál már nagy a nyereség.

Vannak nagyon egyszerű, de mégis hatékony trükkök is. A mintára egyáltalán nem hasonló képrészekből gyakran lényegesen több van, mint a hasonlóból. A *nem illeszkedő képrészek gyors kiszűrésének* az alapötlete a következő. Gyorsan kiszűrjük a nyilvánvalóan nem illeszkedő régiókat, és csak a megmaradt, válogatott régiókat, jelölteket vizsgáljuk meg alaposan. Az ilyen megoldás alkalmazásakor óvatosnak kell lenni, mert ha ügyetlenül szűrűnk, elveszíthetünk egyes igazi, keresett régiókat.

A gyors kiszűrés többféle módon történhet. A *ritkított mintavételezésnél* először nem minden pontban illesztünk, hanem nagyobb lépéssel mozgatjuk a mintát. Kiszűrjük a nyilvánvalóan nem illeszkedő helyeket, aztán csak a megmaradt helyekkel foglalkozunk. Ezt egy képpiramissal is meg lehet csinálni. Gyakorlatilag, a kereszt-korrelációs függvény egyre finomabb mintavételezéséről van szó, aminek a potenciális veszélye, hogy durvább felbontásnál elveszíthetjük a nagyon keskeny csúcsokat, maximumokat.

Alternatív megoldásként kiszámíthatjuk a minta és a vizsgált képrész *egyszerű tuljadonságait* és átugorhatjuk a régiót, ha a tulajdonságok nagyon eltérnek. Vagy kisebb *almintákat* használva kiszűrjük a régiót, ha az alminták nem illeszkednek. Végül *küszöböt* szabhatunk a kumulatív (összegző) eltérési mértékre és kiszűrhetjük a régiót, amikor a mérték eléri a küszöböt. Ily módon nem kell végigszámolni az eltérést, elég összegyűjteni az evidenciát, hogy nincs illeszkedés. A jó küszöb beállítása azonban nem egyszerű dolog, ezért legyen inkább óvatos, viszonylag magas az értéke!

4. fejezet

Éldetektálás

Az élek kétségkívül a leggyakrabban használt lokális képi sajátságok. Jelentőségük abban rejlik, hogy az élek alkotják a képen látható objektumok kontúrjait, ezért az élkiemelés az első lépés az objektumok behatárolása és meghatározása felé.

Ebben a fejezetben először röviden áttekintjük a fundamentális képi jellemzőket és tisztázzuk az él fogalmát, különös tekintettel a képi élek és a fizikai objektumhatárok kapcsolatára és különbségére. Ezek után ismertetjük az éldetektálás elveit, fázisait és minőségi kritériumait. A fejezet legnagyobb részét a gradiens alapú élszűrőknek szenteljük, amelyeknek a jellegzetes utófeldolgozási lépése az éllokalizáció. Megismerjük a zero-crossing éldetektort is, ahol a Laplace-szűrő játszik főszerepet. A fejezet végén összehasonlítjuk a megtárgyalt éldetektorokat és összefoglaljuk a tulajdonságaikat és alkalmazási területeiket.

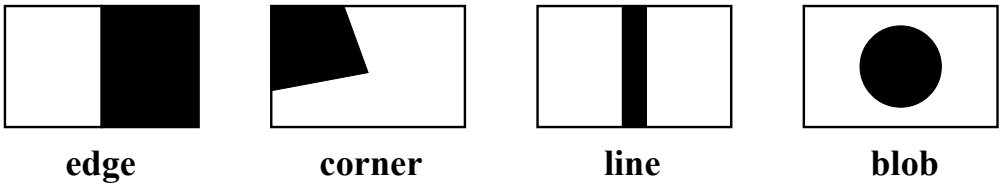
4.1. Az éldetektálás elvei

A képfeldolgozásban leginkább az alábbi lokális képi sajátságokat, jellemzőket használják:

- **Él**: egy nagyobb, a kontúrra merőleges intenzitás-változás, a jelen fejezet tárgya.
- **Sarok**: egy hirtelen forduló a kontúron, a következő fejezet tárgya.
- **Vonal**: egy keskeny, hosszú régió.
- **Folt**: egy kompakt régió.

A fenti, közismert fogalmakat a 4.1. ábra szemlélteti. Bár a vonalak és a foltok is fontos strukturális elemek, kiemelésük bonyolultabb, mint az első kettőé, ezért jegyzetünkben róluk nem lesz szó.

Az élek esetén két összefüggő, de mégis világosan elkülönülő fogalomról kell beszélnünk. A *képi élek* ugyanis nem mindig esnek egybe a *fizikai élekkel*, amelyek a 3D-s tárgyak élei, fizikai határai. A képi élek hirtelen intenzitás-változások (*intensity discontinuities*), a fizikai élek ezzel szemben hirtelen felület-változások (*surface discontinuities*). Például, az

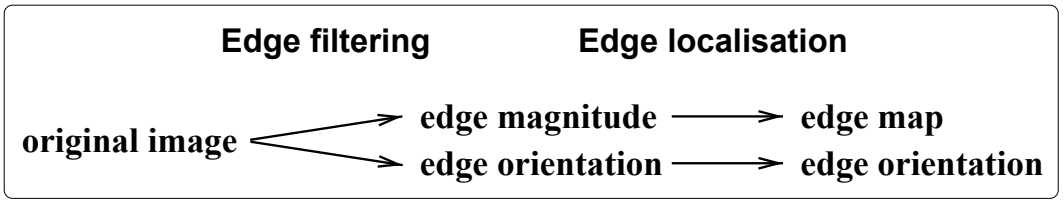


4.1. ábra. Alapvető képi sajátságok.

arnyékok élei nem esnek egybe felület-változásokkal, bár fizikai élekből erednek, azok vetületei.

A képfeldolgozásban keresett élek jelentőségét az adja, hogy gyakran mégiscsak egybeesnek a fizikai élekkel, az objektumok határaival, alapot biztosítva a kontúralapú képszegmentáláshoz. Az emberi szem is a látás alacsony szintjén, "hardverben" detektálja az éleket. Nem intenzitásokkal, hanem intenzitás-különbségekkel operál, így tud hatékonyan alkalmazkodni megváltozó fényviszonyokhoz.

A 4.2. ábrán bemutatjuk az éldetektálás tipikus, alapvető fázisait. Az *élszűrés* előtt simító zajszűrőt alkalmazhatunk, ezzel megalapozva a képfüggvény deriválhatóságának feltételeit. Egy élszűrő élekre reagál, vagyis felerősíti az éleket és elnyomja a kisváltozású régiókat. Egy tipikus élszűrő nem csak az *élerősséget* (magnitúdót) eredményezi, ami a lokális kontrasztot jellemzi, hanem az *élorientációt*, a kontúr lokális orientációját is megadja.



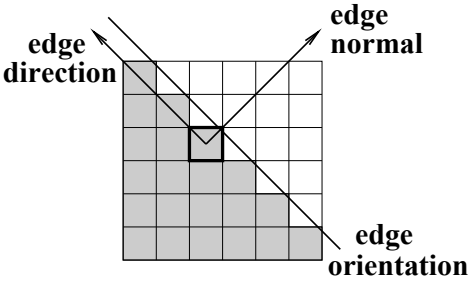
4.2. ábra. Az éldetektálás alapvető fázisai.

Az *éllokalizálás* egy vagy több utófeldolgozási lépést tartalmaz, amellyel eltünteti a zajos és az úgynevezett "fantom" éleket. Ezzel vékony, folytonos kontúrokat biztosít és olyan bináris *élképet* eredményez, amely jelzi, hogy egy adott képelem élpont-e. Az élpontokhoz hozzárendelhetjük az élorientációt is.

A bevezetett fogalmakat a 4.3. ábra illusztrálja, ahol a később ismertésre kerülő 3×3-mas Prewitt éldetektort alkalmazva példát mutatunk finom élek kiemelésére. Az eredeti képen két vízszintes vonalat húztunk. A vonalak képmetszeteket generálnak, ezek az ábra alsó sorában láthatók. A képmetszeteken jól kivehetők a nagyobb és kisebb vonalmenti intenzitás-változások, azaz globális és finom élek. Az élerősséget és -orientációt szürkeségi értékkel ábrázoljuk úgy, hogy az erősebb élpixelek világosabbak. Az élorientáció *cirkuláris adat*, mert ugrik a 0-nál és a π-nél. Szürkeségi értékkel való megjelenítése így nem igazán szemléletes.

4.3. ábra. Éldetektálási példa a Prewitt operátorral.

A 4.4. ábra szemlélteti az élnormálvektor, az élirányvektor és az élorientáció közötti különbségeket. Az *élnormálvektor* az élre merőleges, a leggyorsabb intenzitás-növekedés irányába mutató egységvektor. Az *élirányvektor* az élnormálvektorra merőleges, vele pozitív szöget bezáró egységvektor, amely párhuzamos az éllel. Végül, az *élorientáció* egy $\mod \pi$ cirkuláris adat, azaz nem igazi vektor.



4.4. ábra. Élnormálvektor, élirányvektor és élorientáció.

Mint már emítettük, az élszűrők alkalmazzák a képfüggvény deriváltjait, hogy felerősítsék az élre merőleges intenzitás-változásokat és elnyomják az ilyen változásokat nem tartalmazó régiókat. Az élszűrésre a leggyakrabban a következő operátorokat alkalmazzák, amelyeket folytonos alakban írunk fel.

- A vektor értékű **gradiens operátor**:

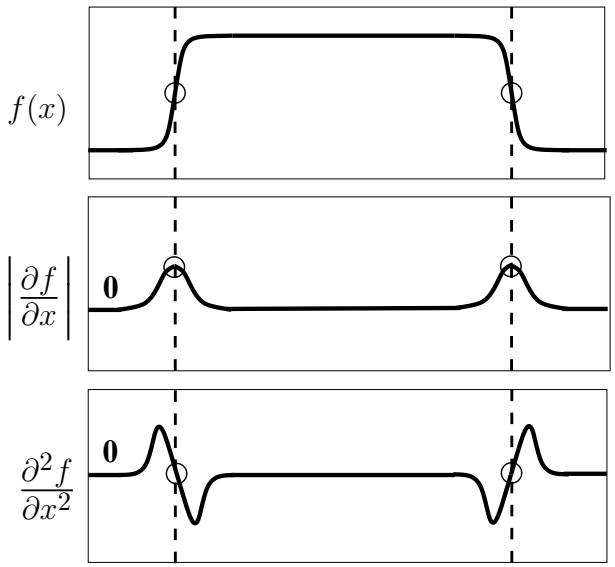
$$\nabla f(x,y) \doteq \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \tag{4.1}$$

- A skalár értékű **Laplace operator**:

$$\Delta f(x,y) \doteq \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2},$$

amelyet már ismerünk, de a jobb áttekinthetőség kedvéért itt megismételünk.

Az élek és a deriváltak kapcsolatát a 4.5. ábra magyarázza. Az élek az első derivált abszolút értékének maximumai, vagy a második derivált nulla átmenetei, előjel-váltásai. E szerint az élkeresésben a gradiens vektor nagysága, vagy a Laplace-szűrt kép nulla átmenetei a mérvadók. A továbbiakban először az első, aztán a második lehetőséggel foglalkozunk.



4.5. ábra. Az élek és a deriváltak kapcsolata.

Mielőtt azonban rátérnénk a konkrét éldetektáló módszerek ismertetésére, el kell gondolkodnunk azon, hogy mit is várunk tőlük, melyek a **jó lineáris élszűrő kritériumai**. Ezeket az elvárásokat a következőképpen lehet összefoglalni:

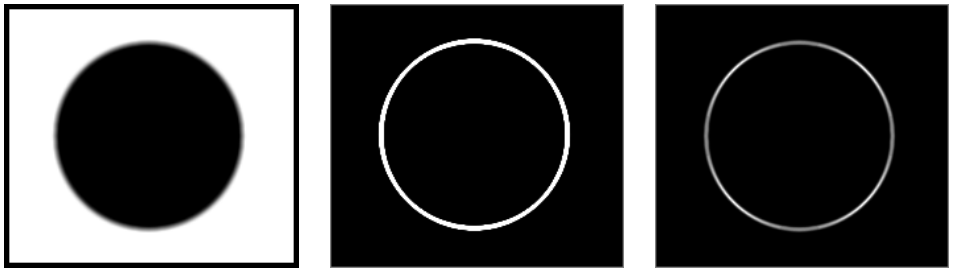
1. Legyen *nulla az eredmény* ott, ahol nincs képváltozás. Egy lineáris élszűrőre nézve ez azt jelenti, hogy a maszkelemek összege nulla kell hogy legyen: $\sum_{r,c} w(r,c) = 0$.

2. Legyen *jó a detektálás*, azaz legyen minimális az alábbi események előfordulása:

   - hamis, zajos élek detektálása (*false positives*)
   - valós élek elvesztése (*false negatives*)

3. Legyen *jó a lokalizálás*: a detektált él a lehető legközelebb legyen a tényleges élhez.

4. A szűrő legyen *izotróp*: az eredmény ne függjön az él orientációjától.



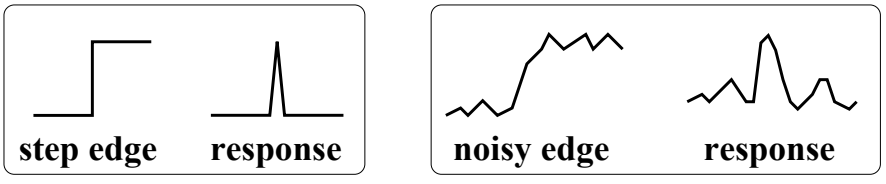
eredeti kép izotróp élszűrő anizotróp élszűrő

4.6. ábra. Az izotrópia-kritérium illusztrációja.

5. A szűrő *egy élet csak egyszer* jelezzen (*single response*): legyen minimális a valós él körüli hamis lokális maximumok száma.

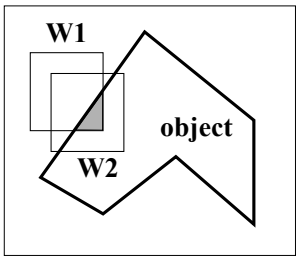
A fenti kritériumok többsége magától értetődő, kivéve talán az utolsó kettőt, ezért ezeket ábrákkal szemléltetjük. Amint a 4.6. ábrán látjuk, egy köralakú objektum esetén egy izotróp élszűrő minden irányra azonos magnitúdót produkál. Ezzel szemben egy anizotróp élszűrő irányfüggő magnitúdót eredményez, mégpedig ebben a konkrét példában úgy, hogy a $45° \cdot k$ irányokban erősebb ($k = 1, 2, \ldots$), a $90° \cdot k$ irányokban gyengébb az élerősség. Természetesen másféle irányfüggőség is lehet.

Az utolsó kritérium ennél egy kicsit bonyolultabb, hiszen nem nem világos, hogy egy szűrő mitől is jelezne többször egy élet. A helyzetet a 4.7. ábra illusztrálja, ahol az ideális él esetén az elvárás, hogy a válaszfüggvénynek csak egy maximuma legyen. Ezzel szemben a valóságban egy zajos, elmosott él több szomszédos maximumot produkál.



4.7. ábra. Egy valós él körüli hamis lokális maximumok.

Ennek az elsődleges oka azonban *nem* a zaj vagy az elmosodás, hanem az, hogy csúsztatott ablakkal dolgozunk. Erről a jelenségről a mintaillesztés kapcsán már szó esett. A 4.8. ábra mutatja, hogy az ablak csúsztatása közben egy kontúrszakasz többször is megjelenik az ablakban, és amíg benne van, addig nemnulla, esetleg nagy magnitúdót kapunk attól függően, hogy az ablakon belül a kontúrszakasz éppen hol van. Így jönnek létre az igazi él szomszédságában megjelenő fantom élek.



4.8. ábra. A "single response" kritérium illusztrációja.

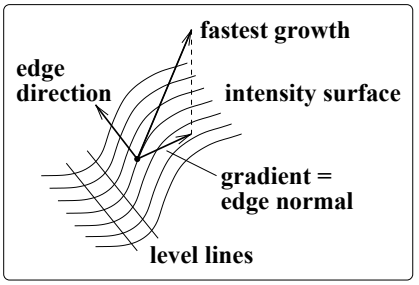
4.2. Gradiens élszűrők

Feltételezve, hogy a képfüggvény deriválható, minden pontban meghatározzuk a (4.1) képlettel definiált gradiensvektort. A gradiensvektor *magnitúdója* és *szöge*

$$M(x,y) = \sqrt{f_x^2 + f_y^2}, \tag{4.2}$$

$$\Theta(x,y) = \arctan\frac{f_x}{f_y}. \tag{4.3}$$

A két komponens jelentését a 4.9. ábra szemlélteti, ahol egy kontúrszakaszt mutatunk intenzitás felületként, szintvonalakkal és a gradiensvektorral, amely az X, Y síkban fekszik. $\Theta(x,y)$ a leggyorsabb intenzitás-növekedés iránya, $M(x,y)$ pedig a növekedés nagysága.



4.9. ábra. Az él intenzitás-felülete, a szintvonalak és a gradiens.

4.2.1. Egyszerű gradiensszűrők és a Canny-éldetektor

A legkisebb méretű 3×3-as gradiensszűrők esetén a parciális deriváltakat különbségekkel közelítjük, ezzel az X és Y irányú, G_x és G_y deriváltmaszkokat kapjuk:

$$f * G_x = f_x, \quad f * G_y = f_y,$$

ahol G_y a G_x 90-fokos elforgatottja.

$$\frac{1}{3}\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix} \quad \frac{1}{4}\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} \quad \frac{1}{2+\sqrt{2}}\begin{bmatrix}-1 & 0 & 1\\ -\sqrt{2} & 0 & \sqrt{2}\\ -1 & 0 & 1\end{bmatrix}$$

Prewitt Sobel izotróp

4.10. ábra. Egyszerű 3×3-as gradiensszűrők G_x komponense.

Gyakran a 4.10. ábrán látható Prewitt- és a Sobel-szűrőt használják. Az ábrán szintén szereplő izotróp szűrő kevésbé érzékeny az él orientációjára, mert a súlyok jobban tükrözik a középponttól való távolságot. Emiatt azonban nem egész, hanem valós számokkal kell operálni, így a gyakorlatban ezt a megoldást csak akkor alkalmazzák, amikor kifejezetten fontos az izotrópia.

A három szűrő mindegyike bizonyos tulajdonságokkal rendelkezik, amelyek a lineáris élszűrőkkel szemben támasztott, általános követelményekből erednek. Az első kritériumunknak megfelelően a maszkelemek összege mindig nulla. A maszkok a pozitív elemek összegével vannak normálva, hogy az ideális, egységnyi erősségű élre az eredmény 1 legyen. A G_x maszkok tükörszimmetrikusak a maszk középpontján áthaladó, vízszintes tengelyre, a függőleges tengelyre pedig antiszimmetrikusak. A szimmetria azt jelenti, hogy nincs okunk különbséget tenni a jobb és a bal között, amikor az élre merőlegesen haladunk. Ez a tulajdonság nem csak a fenti három, hanem az összes élszűrőt jellemzi. Az antiszimmetria pedig az él S alakú felfutását tételezi föl, ami nem mindig teljesül.

Ha ezeket a tulajdonságokat kötelező elvárásnak, megkötésnek tekintjük, akkor egy általános, 9 szabadságfokú 3×3 mátrixból kiindulva könnyen levezethető az egyparaméteres *maszkcsalád*, amelynek a 4.10. ábrán bemutatott maszkok is tagjai.

A kisméretű élszűrők zajérzékenyek és csak a finom élek detektálásra alkalmasak. Az alábbiakban egy olyan élszűrőt vezetünk be, amelynek a mérete, zajérzékenysége és finomsága szabályozható.

A lineáris élszűrőkkel dolgozó éldetektorok között a **Canny-éldetektor** optimális a zajosított ideális élre, ha a zaj additív, nemkorrelált és normáleloszlású. Az optimalitási kritérium két alkritériumot egyesít, a jó detektálást és a jó lokalizálást. A "single response" kritérium teljesítéséhez az éldetektor két utófeldolgozási eljárást alkalmaz, a *nem-maximumok törlését* (*non-maxima suppression, NMS*) és a *hiszterézis-küszöbölést* (*hysteresis thresholding*).

Az eredeti, matematikailag szigorúan levezetett Canny-szűrő elég bonyolult, de van egyszerűsített, közelítő változata, amely a következő két lépesből áll:

1. Gauss-szűrővel elsimítjuk a képet

$$g(x, y) = f(x, y) * w_G(x, y; \sigma)$$

2. Utána alkalmazzuk a gradiensszűrőt és megkapjuk az élerősséget és orientációt.

A σ paraméter meghatározza a szűrő méretét. A paraméter beállítása a kívánt részletességtől és a zajszinttól függ.

A nagyméretű Gauss-szűrő jelentősen elsimítja a képet, ezért a gradiensszűrőnek is alkalmazkodni kell hozzá, hiszen egy kisméretű élszűrő alig talál rajta élet. A gyakorlatban nem a fenti két műveletet végzik, hanem egy egyesített, szeparálható szűrőt alkalmaznak.

Felhasználva a lineáris szűrők tulajdonságait

$$\big(f(x,y) * w_G(x,y)\big)w_\nabla = f(x,y) * \big(w_G(x,y) * w_\nabla\big) = f(x,y) * \big(\nabla w_G(x,y)\big)$$

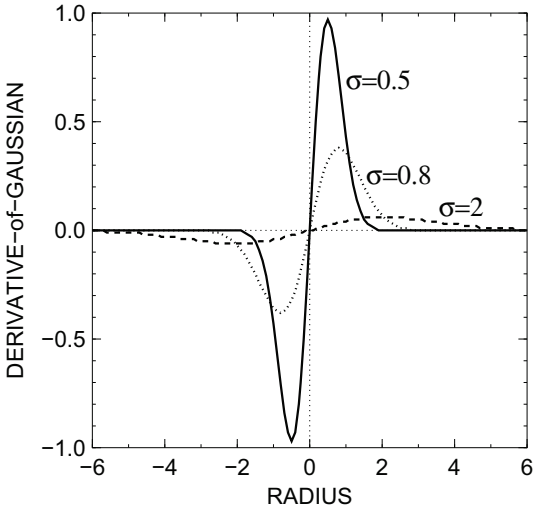
és a Gauss-szűrő szeparálhatóságát

$$w_G(x,y) = w_G(x) \cdot w_G(y),$$

megkapjuk a gradiensszűrő vektort:

$$\nabla w_G(x,y) = \big(w_G(y) \cdot w'_G(x), w_G(x) \cdot w'_G(y)\big), \quad \text{ahol} \tag{4.4}$$
$$w'_G(x) \doteq \frac{\partial w_G(x)}{\partial x} = C \cdot x \exp\left\{-\frac{x^2}{2\sigma^2}\right\}$$

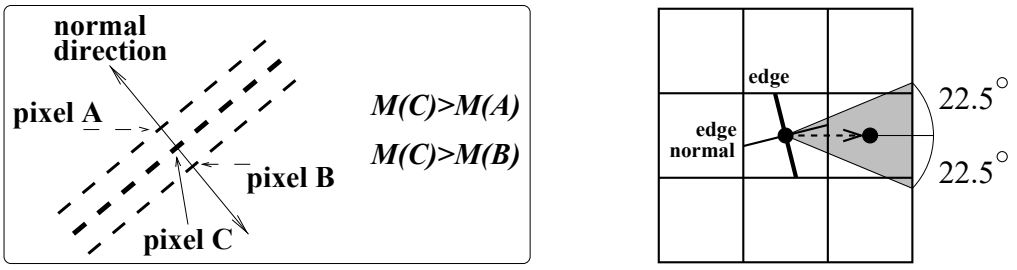
A szűrőt két 1D-s szűrő sorozataként alkalmazzuk, a $w'_G(x)$ alakját a 4.11. ábra szemlélteti.



4.11. ábra. A $w'_G(x)$ Gauss-derivált alakja növekvő σ-ra.

4.2.2. Élek lokalizálása

Ennek az utófeldolgozási műveletnek a bemenete az $M(x,y)$ élerősség (magnitúdó) és a $\Theta(x,y)$ élorientáció, kimenete pedig egy bináris *élkép*, amelynek az értéke 1, ha a pontban van él, és 0, ha nincs. Éllokalizálással a nagy élerősségű pontok közül kiválasztjuk a valós éleket. Hangsúlyozzuk, hogy a művelet nem csak gradiensszűrővel (pl. Canny, Prewitt), hanem minden olyan élszűrővel alkalmazható, amely élerősséget és -orientációt biztosít. Amint



ponttörlés élre merőleges lépés

4.12. ábra. Illusztrációk az NMS-művelethez.

már a Canny-éldetektor kapcsán említettük, az éllokalizálás két lépést szokott tartalmazni, de itt csak az egyikről, a *nem-maximumok törléséről* lesz szó. Az algoritmust az alábbiakban foglaljuk össze:

1. Algoritmus: A nem-maximumok törlése (NMS)

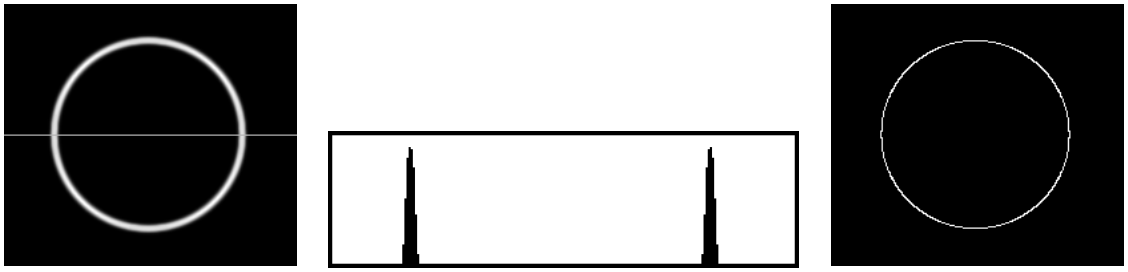
1. Az $M'(x, y)$ kimenetbe bemásoljuk az $M(x, y)$-t.
2. Az $M(x, y)$-ben minden pontból lépünk a két, élre merőleges irányban.
3. Legyen a kiinduló pixel C, a két, élre merőlegesen szomszédos pixel pedig A és B.
4. Ha $M(A) > M(C)$ vagy $M(B) > M(C)$, akkor az $M'(x, y)$-ben töröljük a C-t: $M'(x, y) = 0$.

Az algoritmus működését a 4.12. ábra magyarázza. A ponttörlés előtt az $M(x, y)$-ben levő kontúrok vastagok a fantomélek miatt. A NMS törli a nemmaximális erősségű pixeleket megőrizve a kontúrok folytonosságát. A C pixelt nem töröljük, mert $M(C) > M(A)$ és $M(C) > M(B)$. Az A és a B pixeleket viszont töröljük, mert $M(A) < M(C)$ és $M(B) < M(C)$. Fontos, hogy nem a kontúr mentén, hanem arra merőlegesen kell lépni. Például a középső pixel jobb- és baloldali szomszédját akkor kell megvizsgálni, ha a jobb képen az élnormálvektor a jelzett szögtartományba esik.

A 4.13. ábrán példát adunk a nem-maximumok törlésére. Az NMS vékonyítja a kontúrokat az élerősség képen, ahol egy sort kiemeltünk. A sorban az élerősség-metszetet a középső képen kinagyítva mutatjuk. Jól látható, hogy az igazi élpontok mellett nagy erősségű fantom pontok vannak, amelyeket eltávolítunk és vékonyabb élképet kapunk.

4.3. A zero-crossing éldetektor

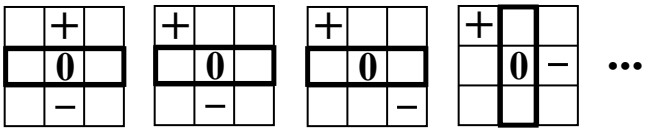
A zero-crossing éldetektor működési elvéről már szó volt a 4.5. ábrával kapcsolatban. Alkalmazzuk a Laplace szűrőt, utána a szűrt képen megkeressük a nulla átmeneteket. Az utóbbi



élerősség élerősség-metszet NMS eredménye

4.13. ábra. Példa a nem-maximumok törlésére.

művelethez felhasználhatjuk az 4.14. ábrán szereplő, egyszerű maszkokat és elforgatottjaikat. A Laplace-szűrt képen ott van nulla átmenet, ahol előjel-váltás tapasztalható a vonal két oldalán. Egy másik, korrektebb lehetőség az, hogy lineáris függvénnyel lokálisan közelítjük a Laplace-szűrt képfüggvényt, ami analitikus megoldást ad a problémára.



4.14. ábra. A nulla-átmenetek keresése.

Mint tudjuk, a Laplace-operátor zajérzékeny, ezért a gyakorlatban nem őt, hanem a már említett *Laplacian-of-Gaussian* (LoG) operátort alkalmazzák, amikor a deriválás előtt simítást, regularizációt végzünk a Gauss-szűrő segítségével. Az alábbiakban levezetjük a LoG konvolúciós maszkját és bevezetjük annak hatékony, szeparálható közelítését. A levezetésnél hasonlóan járunk el, mint a Canny-szűrő esetén, vagyis a Gauss-függvényre alkalmazzuk a Laplace-operátort.

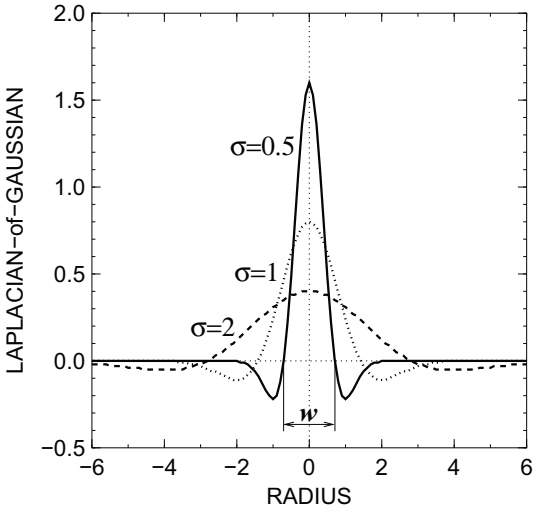
Ehhez először szükségünk lesz a Laplace-operátor polárkoordinátás alakjára, mert azzal felhasználhatjuk a Gauss-függvény körszimmetriáját. Ha az általunk ismert X, Y alakot alkalmaznánk, a levezetés sokkal bonyolultabb lenne, amiből azt a tanulságot vonhatjuk le, hogy a koordinátarendszer kiválasztásánál mindig célszerű figyelembe venni a vizsgált jelenség szimmetriáját.

Az ablak közepéből induló r, θ polárkoordinátákban

$$\Delta f = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial f}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 f}{\partial \theta^2}$$

Esetünkben $f = w_G$, amely nem függ a θ szögtől, ezért az utolsó tag kiesik, és a két, r szerinti deriválás után megkapjuk a LoG operátort:

$$w_{LoG}(r) = C\left(2 - \frac{r^2}{\sigma^2}\right)\exp\left\{\frac{-r^2}{2\sigma^2}\right\}, \tag{4.5}$$



4.15. ábra. A LoG szűrő alakja változó σ-ra.

ahol C egy normalizáló konstans.

A gyakran "mexikói kalap"-nak nevezett függvény alakját a 4.15. ábra szemlélteti. Az eddigekhez hasonlóan a σ paraméter a részletességet szabályozza úgy, hogy kisebb σ-val finomabb éleket detektálhatunk. Ugyanúgy, mint ahogy a Gauss szűrővel tettük, a LoG szűrőt is $k\sigma$-nál vágjuk el, így a pozitív középső rész mérete $w = 2\sqrt{2}\sigma$ lesz.

A LoG szűrő *nem szeparálható*, ezért nagy σ esetén az operátor nagyon lassú. Ezt a problémát úgy oldják meg, hogy az eredeti LoG szűrő helyett annak egy gyors, szeparálható közelítését használják. A *Difference-of-Gaussians* (DoG) szűrő megfelelő paraméter-választás mellett jól közelíti a LoG szűrőt:

$$w_{LoG}(r;\sigma) \approx w_G(r;\sigma_1) - w_G(r;\sigma_2) \doteq w_{DoG}(r;\sigma_1,\sigma_2), \tag{4.6}$$

ahol $w_G(r;\sigma_1)$, $w_G(r;\sigma_2)$ két különböző Gauss-szűrő, és általános esetben $\sigma \neq \sigma_1 < \sigma_2$. Gyakran használják azonban a $\sigma_1 = \sigma$, $\sigma_2 = 1.6\sigma$ beállítást:

$$w_{DoG}(r;\sigma) = w_G(r;\sigma) - w_G(r;1.6\sigma), \tag{4.7}$$

és a szűrőnek csak egy paramétere lesz.

A σ paraméterrel beállíthatjuk a szűrő méretét, ami részletesség-szabályozást jelent. A paraméter növelésével *élhierarchia* alakul ki a *mértéktérben* (*scale-space*). A részletesség csökkenésével az élek eltünnek vagy összeolvadnak, kialakítva egy fa-szerű élstruktúrát, amely támogatja a kép struktúrális elemzését.

A folytonos zero-crossing éldetektor *zárt kontúrokat* eredményez, mert a Laplace operátor alkalmazása után kialakuló folytonos felületet vízszintes (nulla szintű) síkkal metszük. Ez elvileg segíthet kontúrkövetésben, de a gyakorlatban, diszkrét esetben sok zajos kontúrt is kapunk, amelyet ki kell szűrni. Mivel azonban a detektor nem biztosítja az élorientációt, a

LoG feldolgozott DoG nyers

4.16. ábra. Példák 15×15-ös LoG és Dog szűrővel történő éldetektálásra.

nem-maximumok törlését nem lehet alkalmazni, és a gyenge, fantom és zajos éleket másképpen kell eltávolítani. Például, az élpontokban számított gradiens segítségével: a kis gradiens zajos élt jelent.

A 4.16. ábra példákkal illusztrálja a LoG és DoG szűrővel történő éldetektálást. A LoG esetén eltávolítottuk a gyenge éleket, a DoG esetén nem, hogy láthassuk, hogy a "nyers" eredmény sok zajos élet tartalmaz. Ha ezektől eltekintünk, megfigyelhetjük, hogy a fő kontúrokban a LoG és a DoG eredménye között nincs nagy eltérés.

4.4. Az éldetektálás összefoglalója

A 4.17. ábra további éldetektálási példákat mutat, amelyek az általunk megismert operátorok közötti kölönbseégeket és hasonlóságokat illusztrálják. A LoG esetén eltávolítottuk a zajos éleket. Az ábrán láthatjuk, hogy az azonos méretű Canny-éldetektor kevésbé zajos eredményt nyújt, mint a Prewitt-detektor. A nagy méretű Canny-operátor, a nagy méretű LoG operátorhoz hasonlóan, csak a főbb kontúrokat emeli ki. A LoG éldetektor vastagabb

Prewitt 3×3 Canny 3×3 Canny 25×25 LoG 21×21

4.17. ábra. Éldetektorok összehasonlítása.

és rendre zárt kontúrokat eredményez, amelyek egy része a zajos élek törlése ellenére nem hordoz érdemi információt.

Az alábbiakban összefoglaljuk az éldetektorok alkalmazhatóságát és az algoritmusok főbb tulajdonságait.

- A 3×3-*as gradiens operátorok* (Prewitt, Sobel) egyszerűek és gyorsak. Nincs paraméterük, így nem szabályozhatók. Akkor használják, amikor finom élekre van szükség, és a zajszint alacsony.

- A *Canny operátor* bonyolultabb, de robusztusabb és rugalmasabb. Van fontos szabályozható paramétere, σ. A Canny élszűrő alkalmazható finom és globális élek detektálására, zajos képek esetén is. Léteznek gyors, szeparálható, valamint rekurzív, közelítő Canny-implementációk.

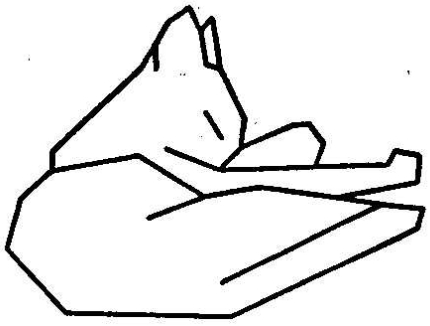
- Minden gradiens alapú detektor *élorientációt* biztosít és az élszűrés után éllokalizációt igényel.

- *A zero-crossing* éldetektor (LoG) nem eredményez élorientációt, csak az élpontokat jelöli meg. A zajos éleket utólag kell eltávolítani. Neurofiziológiai kísérletekkel alátámasztott, emiatt népszerű volt az 1980-as években. Bár van gyors, szeparálható változata (DoG) a gyakorlatban ritkábban használják, mint a gradiens alapú detektorokat.

5. fejezet

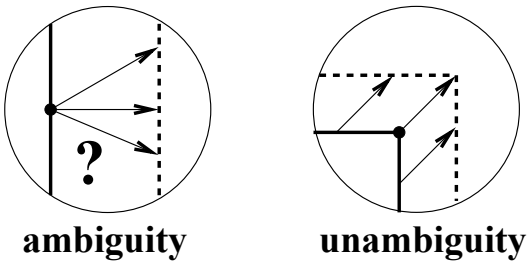
Sarokdetektálás

Ebben a fejezetben két sarokdetektálási algoritmust ismertetünk, amely közvetlenül képben keresi a sarkokat. Sarokpontokat mint lokális képi jellemzőket több területen, így kép-, alakzat- és mozgáselemzésben is használnak. Nem véletlenül, mert az emberi látásban is fontos szerepet játszanak. Ezt már az 1950-es években felfedezték, amikor Fred Attneave amerikai kutató publikálta az 5.1. ábrán látható rajzot, az Attneave-macskát. Az eredeti sima alakzatot kis számú, nagy görbületű pontból rekonstruálta úgy, hogy egyenesekkel kötötte össze a pontokat. A nagy adatvesztesség ellenére a macska még mindig könnyen felismerhető, ami demonstrálja a domináns sarokpontoknak az emberi alakzat-percepcióban játszott szerepét.



5.1. ábra. Sarkok emberi alakzat-percepcióban: az Attneave-macska.

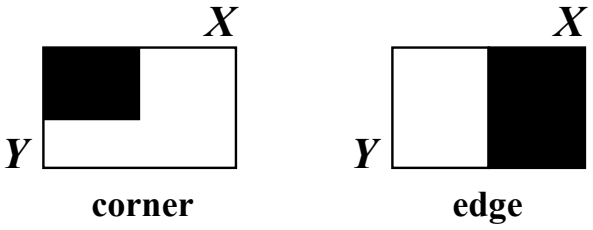
A sarokpontok mozgásbecslésben is kiemelt jeletőségűek, mert segítségükkel meg lehet oldani az 5.2. ábra által szemléltetett, alapvető mozgásbecslési problémát, amit *apertúra-problémának* szoktak nevezni. Ennek lényege, hogy egy sima határszakaszon a mozgásvektorok bizonytalanok, mert az elmozdulásvektornak csak a kontúrra merőleges, normális komponense határozható meg lokális megfigyelés alapján, egy kis ablakban. A tangenciális komponens nem határozható meg, mert kiemelt pontok hiányában nem látható a kontúrszakasz tangenciális irányú elmozdulása. Egy sarokpontban viszont a mozgásvektor egyértelmű, és ezt a szomszédos pontokra is ki lehet terjeszteni.



5.2. ábra. Sarkok mozgásbecslésben: az apertúra-probléma.

Két fajta sarokdetektálási feladatról szokás beszélni. Ha előzetesen kiemeltük a kontúrokat, amelyeket digitális görbékkel reprezentálunk, akkor sarkok keresése a *digitális görbékben* a feladat. Más esetekben viszont kontúrkiemelés nélkül, közvetlenül képben keressük a *képi sarokpontokat*. Ezzel a második feladattal most fogunk foglalkozni, a kontúrsarkok kiemeléséről később, a jegyzet végén lesz szó.

Mielőtt rátérnénk konkrét sarokkiemelési algoritmusokra, tisztázzuk az élek és a sarkok közötti különbséget, amelyet az 5.3. ábra illusztrál. A sarkok azok a pontok, ahol az $f(x, y)$ képfüggvény változása az X és az Y irányban is jelentős, azaz mind $|f_x|$, mind $|f_y|$ nagy. Az élek ezzel szemben azok a pontok, ahol az $f(x, y)$ változása az egyik irányban nagy, a merőleges irányban kicsi; például, függőleges élre $|f_x|$ nagy, $|f_y|$ kicsi.



5.3. ábra. Képi sarkok és élek.

Ezt felhasználva, lehetne elvben olyan triviális él- és sarokdetektálási algoritmusokat írni, amelyek küszöböket szabnak a két derivált abszolút értékére. Világos azonban, hogy egy ilyen algoritmus nem lenne invarians az élek illetve sarkok elforgatására. Például, a 45 fokkal elforgatott éleket nem is tudnánk detektálni, mert a két derivált értéke egyenlő. Az élek esetén az elforgatás-invarianciát a gradiensvektor biztosítja, amely forog az éllel együtt. A sarkok esetén is lesz olyan eszközünk, amely megoldja a problémát.

5.1. Sarokdetektálási algoritmusok

Többféle sarokdetektor létezik, ezekből kettővel foglalkozunk, a *Kanade-Lucas-Tomasi* (KLT) operátorral és a *Harris*-operátorral, amelyeket gyakran használnak és sok területen alkalmaznak (mozgáskövetés, sztereó illesztés, képi adatbázis-keresés). A két módszer aránylag

egyszerű, de hatékony és robusztus, és hasonló elven működik. Alapjuk a *lokális struktúramátrix* (tenzor).

5.1.1. A lokális struktúramátrix

Minden képpontban meghatározzuk a lokális struktúramátrixot:

$$C_{str} \doteq \begin{bmatrix} (f_x)^2 & f_x f_y \\ f_x f_y & (f_y)^2 \end{bmatrix} * w_G(r;\sigma) = \begin{bmatrix} \widehat{(f_x)^2} & \widehat{f_x f_y} \\ \widehat{f_x f_y} & \widehat{(f_y)^2} \end{bmatrix} \tag{5.1}$$

A mátrix meghatározásához gradiens maszkokkal kiszámítjuk a deriváltakat és a mátrix elemeit. Ezzel három kép (csatorna) alakul ki: $(f_x)^2$, $(f_y)^2$, $f_x f_y$. (*Nem* f_{xx}, f_{yy}, f_{xy}!) A három képet feldolgozzuk a $w_G(r;\sigma)$ Gauss-szűrővel, amelyben a σ paraméter szabályozza a sarkok finomságát. A Gauss-szűrő helyett szabályozható méretű dobozszűrőt is alkalmazhatunk. Az átlagszűrést a $\widehat{\ }$ kalappal jelöljük.

Ezek után, szükségünk lesz a lokális struktúramátrix sajátértékeire, amelyeket az alábbi módon határozzuk meg. A C_{str} mátrix szimmetrikus, ezért létezik olyan R mátrix, $R^T R = RR^T = \mathbf{I}$, amivel C_{str} diagonalizálható. Itt $\mathbf{I}$ a 2×2-es egységmátrix, R a lokális koordinátarendszer elforgatása.

A diagonalizálás után a diagonális elemek a C_{str} sajátértékei lesznek:

$$\tilde{C}_{str} = R^{-1} C_{str} R = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \tag{5.2}$$

és ezeket a karakterisztikus egyenlet adja:

$$\det\big(C_{str} - \lambda \mathbf{I}\big) = 0. \tag{5.3}$$

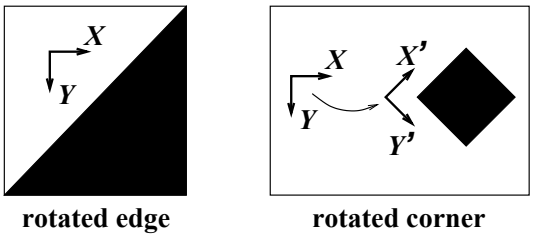
Tekintettel a mátrix kis méretére, használjuk fel inkább azt, hogy

$$D \doteq \det C_{str} = \prod_i \lambda_i = \lambda_1 \lambda_2,$$
$$T \doteq \operatorname{trace} C_{str} = \sum_i \lambda_i = \lambda_1 + \lambda_2.$$

Ebből megkapjuk a sajátértékeket:

$$\begin{aligned} \lambda_{1,2} &= \frac{1}{2}\left(T \pm \sqrt{T^2 - 4D}\right), \\ &= \frac{1}{2}\left(\widehat{f_x^2} + \widehat{f_y^2} \pm \sqrt{\left(\widehat{f_x^2} - \widehat{f_y^2}\right)^2 + 4\left(\widehat{f_x f_y}\right)^2}\right), \end{aligned} \tag{5.4}$$

ahol $\lambda_1 \geq \lambda_2 \geq 0$.



5.4. ábra. A diagonalizálás geometriai étrelmezése.

A sajátértékek jelentése a következő. Egy teljesen homogén képre $C_{str} = 0$, és $\lambda_1 = \lambda_2 = 0$. Az ideális élre $\lambda_1 > 0$, $\lambda_2 = 0$, és az első sajátvektor merőleges az élre. Az ideális sarokra pedig $\lambda_1 \geq \lambda_2 > 0$, és minél nagyobb a kontraszt, annál nagyobbak a sajátérték.

A diagonalizálás funkcióját az 5.3. ábra magyarázza. Él esetén a sajátvektorok megadják az élorientációt, az első sajátérték pedig az élerősséget. Egy sarokban két él találkozik, így a kisebbik sajátérték is nagy lehet. A C_{str} diagonalizálásakor a lokális koordinátarendszert a két élirányhoz igazítjuk, ami biztosítja a sarokdetektálás elforgatás-invarianciáját.

5.1.2. A Kanade-Lucas-Tomasi sarokdetektor

Az éldetektáláshoz hasonlóan a sarokkiemelés is két fázisban történik, ezek a sarokszűrés és az utófeldolgozás, amelynek célja a saroklokalizálás. A KLT-sarokdetektor két paraméterrel rendelkezik. Az egyik a D_w *ablakméret*, amelynek kettős a szerepe:

- sarokszűrésben beállítja a struktúramátrix definíciójában szereplő átlagszűrő méretét;
- utófeldolgozásban meghatározza a legkisebb távolságot két sarok között, amikor mind a kettőt még detektálni tudjuk.

A másik paraméter a λ_{thr} *küszöb*, a kisebbik sajátérték alsó küszöbe. Sarokpont csak ott lehet, ahol $\lambda_2 > \lambda_{thr}$. Az alábbiakban ismertetjük a KLT-algoritmus főbb lépéseit.

2. Algoritmus: KLT-sarokdetektor

1. Az egész képre kiszámítjuk az f_x és f_y deriváltakat.
2. Minden p képpontra:
   (a) $D_w \times D_w$ méretű ablakban kiszámítjuk a C_{str} mátrixot;
   (b) meghatározzuk a mátrix kisebbik sajátértékét, λ_2-t;
   (c) ha $\lambda_2 > \lambda_{thr}$, beírjuk a p-t az L **listába**.
3. Szortirozzuk az L listát csökkenő λ_2 szerint, megkapjuk az L_S **szortirozott listát**.

4. Az L_S **aljáról felfelé** haladva minden p_i pontra

   - megvizgáljuk az összes feljebb levő p_k pontot, $k > i$;
   - ha találunk olyan pontot, amely a p_i pont $D_w \times D_w$-es környezetében van, akkor a p_i-t töröljük a listából.

A λ_{thr} paraméter meghatározza a detektor érzékenységét. Beállítása függ a kép kontrasztjától úgy, hogy nagyobb kontraszt erősebb sarkokkal jár, ezért nagyobb λ_{thr} értékre van szükség.

A D_w paraméter szabályozza a detektor finomságát úgy, hogy kisebb D_w érték kisebb átlagszűrőt és finomabb sarkokat eredményez. A két detektált sarok közötti $D_w/2$ minimális távolság ennek megfelelően szintén kisebb lesz. Nagyobb D_w pedig nagyobb és ritkább sarkokat produkál. A paraméter szokásos értékektartománya $D_w = 5$–19. Beállításával vigyázni kell, mert ha túl kicsi, zajos sarkokat kaphatunk; ha túl nagy, elveszíthetünk sarkokat, vagy rossz helyen találhatjuk.

5.1.3. A Harris-sarokdetektor

Ez a detektor *sarokerősséget* vezet be, mértéke a következő:

$$H(x, y) = \det C_{str} - \alpha \left(\operatorname{trace} C_{str}\right)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2, \tag{5.5}$$

ahol α egy paraméter.

Ha $H > 0$, akkor a pontban sarok lehet; ha $H < 0$, akkor él. A mérték nemnegatív, amikor $0 \leq \alpha \leq 0.25$ (lásd lejebb).

Az operátor sarkot jelez, ha az erősség meghalad egy előre megadott értéket:

$$H(x, y) > H_{thr},$$

ahol $H_{thr} > 0$ a másik paraméter, a sarokerősség alsó küszöbe.

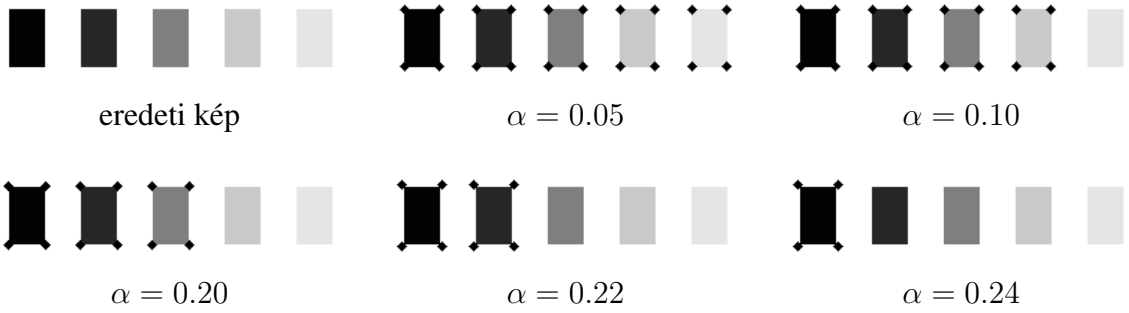
A KLT-detektorhoz hasonlóan, a Harris-sarokdetektor is utófeldolgozást, éllokalizálást igényel, amivel az erős sarkok környezetéből eltávolítjuk a gyenge sarkokat. Ennek a menete megegyezik a KLT-algoritmuséval.

Ahhoz, hogy a Harris-operátor paramétereit értelmezni tudjuk, felírjuk $\lambda_1 = \lambda$, $\lambda_2 = \kappa\lambda$, $0 \leq \kappa \leq 1$. Ekkor

$$H = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 = \lambda^2 \left[\kappa - \alpha(1 + \kappa)^2\right],$$

ezért a $H \geq 0$ feltétel akkor teljesül, amikor

$$0 \leq \alpha \leq \frac{\kappa}{(1 + \kappa)^2} \leq 0.25.$$

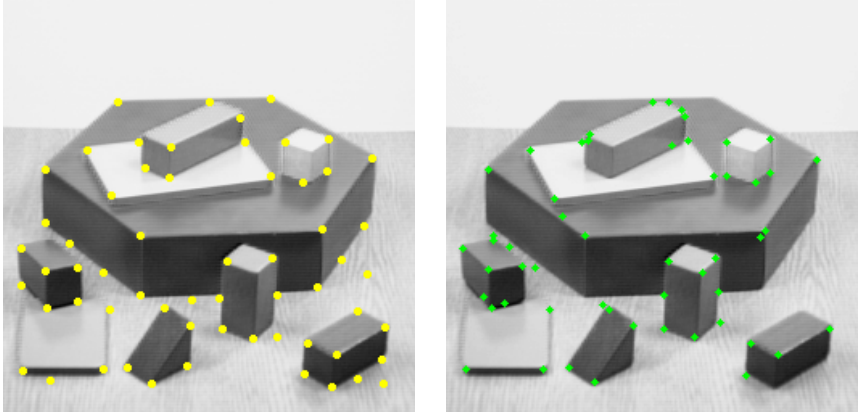


5.5. ábra. Az α paraméter hatása a Harris sarokdetektorra.

Az α funkciója hasonlít a KLT-féle λ_{thr} funkciójához. Nagyobb α kisebb H-t és kevésbé érzékeny detektort eredményez, tehát kevesebb sarkot találunk. Gyakran a H_{thr} paramétert beállítják kis értékre és rögzítik, így a sarokerősség egyetlen szabályozható paramétere α lesz. A paraméter hatását az 5.5. ábra szemlélteti: amikor kicsi, minden sarkot megtalálunk; amikor nagy, csak a legerősebbet.

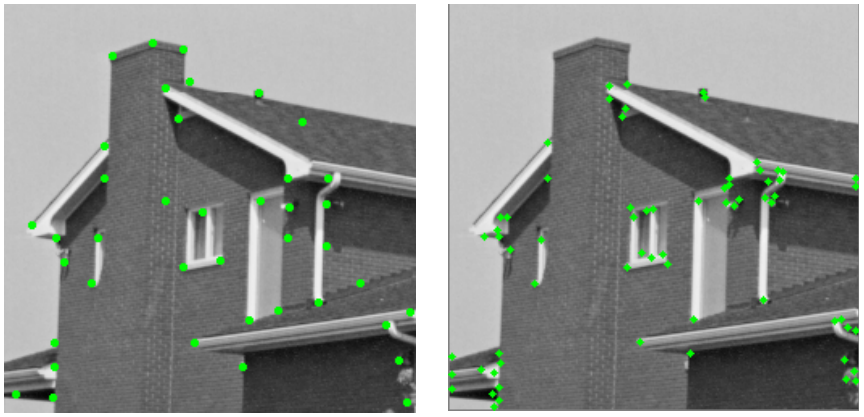
5.1.4. A két sarokdetektor összefoglalója

Az 5.6. ábrán egy KLT-sarokdetektálási példa látható. A sarkok többségét megtaláltuk, de vannak elveszített sarokpontok és hamisan detektált sarkok is. Az ábra egy Harris-sarokdetektálási példát is mutat. Megfigyelhető, hogy az operátorok nem kizárólag sarkokat, hanem más texturált régiókat is detektálnak.



5.6. ábra. Példa KLT- és Harris-sarokdetektálásra.

Az 5.7. ábra egy másik bemeneti kép segítségével illusztrálja a két módszer közötti hasonlóságot és különbséget. Ezzel az alábbiakban össze tudjuk foglalni a két sarokdetektor működését.



KLT, 40 sarok Harris, $\alpha = 0.2$

5.7. ábra. A két sarokdetektor összehasonlítása.

- A KLT és a Harris sarokdetektor között konceptuális hasonlóság van. A C_{str} lokális struktúramátrixot használják és olyan pontokat keresnek, ahol két ortogonális irányban van jelentős képváltozás.
- A KLT és a Harris sarokdetektor között különbségek is vannak. A KLT algoritmus *explicit* küszöböt állít fel a diagonalizált C_{str}-re, a Harris algoritmus ezzel szemben *implicit* módon, a $H(x, y)$ sarokerősségen keresztül küszöböli a mátrixot.
- A KLT sarokdetektor eredménye általában közelebb áll az emberi sarok-percepcióhoz. A módszer a népszerű KLT mozgáskövető (*KLT Tracker*) része.
- A Harris detektor stabil változó elforgatás és világítás mellett, azaz ugyanott talál sarkokat (*good repeatability*). Sztereó illesztésre és képi adabázis-keresésre szokták használni.
- A két detektor nemcsak sarkokat, hanem más hasznos, stabil pontokat (*interest points*) is detektál, ezek az erősen texturált kisrégiók.

6. fejezet

Küszöbölés

Az éldetektálás révén meghatározhatjuk a képen látható objektumok határait. Mint lokális művelet, az éldetektálás bizonyos előnyökkel és hátrányokkal rendelkezik. Az egyik hatékony alternatív megoldást a fejezet tárgyát képező *képküszöbölés* adja. Küszöböléssel nem a kontúrokat, hanem a belső képelemeket célozzuk meg feltételezve, hogy intenzításban lényegesen különböznek a külső, háttérpixelektől. Két módszert ismertetünk, amelynek a működési elve eltérő, de ugyanabból a képstatisztikából, a szürkeségi hisztogramból indulnak ki.

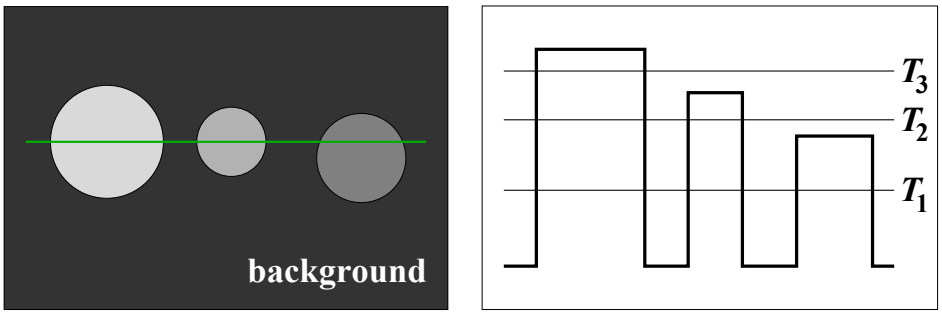
Jegyzetünkben csak az egész képre kiterjedő állandó küszöbértékkel történő képvágásról lesz szó. Léteznek adaptív megoldások is, de ezek bonyolultabbak és nem mindig megbízhatóak, ezért azokat nem ismertetjük. Az éldetektálással szemben a (rögzített értékű) küszöbölésnek előnyei, de hártanyai is vannak, ezeket a fejezet végén fogjuk megtárgyalni, amikor a hisztogram alapú küszöbölés elvi korlátairól is lesz szó.

6.1. Az intenzitás-küszöbölés elvei

Intenzitás-küszöbölés (*intensity thresholding*) egy alapvető képszegmentációs módszer osztály. Feltételezi, hogy a színtér lapos, egyenletesen megvilágított felületekből áll, ezért a képen több, megközelítően homogén régió van. A feltételezés a valóságban legtöbbször nem teljesül szó szerint, ennek ellenére az elképzelés működőképes és gyakran alkalmazzák.

A küszöbölési eljárások célja egy vagy több küszöb automatikus beállítása. A küszöbök intevallumokra bontják az intenzitás tartományt, ezzel *intenzitás-osztályokat* határoznak meg. A küszöbölés végeredménye az intenzitás-osztályokba sorolt képelemek. Egy osztályhoz tartozó képelemek várhatóan hátteret vagy objektumokat alkotnak, ezért a háttértől és az egymástól megkülönböztetett objektumokat kapjuk.

Az N*-szintű küszöbölés* definíciója a következő. Állítsunk be $N-1$ darab T_k küszöböt, $N \geq 2$, ahol $0 < T_k < 255, k = 1, \ldots, N-1$ és $T_k < T_{k+1}$. Egészítsük ki a küszöbhalmazt két konstans, szélső értékkel: $T_0 = 0$ és $T_N = 255+1 = 256$, így összesen $N+1$ küszöbérték



kép képmetszet a vonal mentén

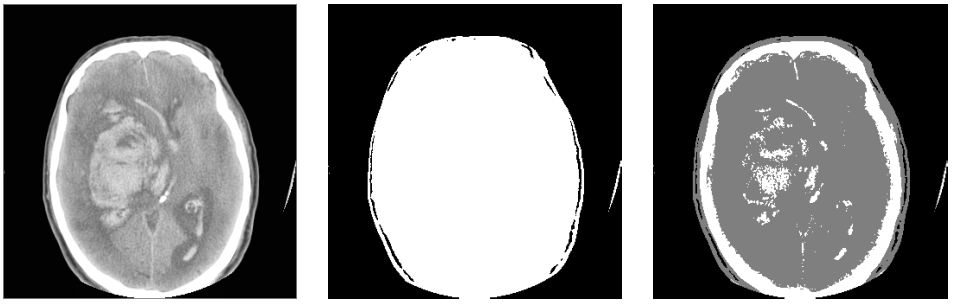
6.1. ábra. A négyszintű küszöbölés illusztrációja.

lesz. Akkor sorolunk be az $f(x, y)$ képelemet az n-ik osztályba, ha

$$T_n \leq f(x, y) < T_{n+1}, \quad n = 0, \ldots, N \tag{6.1}$$

A definíciót a 6.1. ábra szemlélteti, ahol három (érdemi) küszöbbel négy szintre vágjuk a szintetikus képet úgy, hogy $T_0 = 0$ és $T_4 = 256$. Az első szint a sötét háttér, a negyedik szint a legvilagosabb folt.

A 6.2. ábra két valós küszöbölési példát mutat. A *kétszintű* (bináris) küszöbölés, azaz a *binarizálás* esetén $N = 2$, így a teljes objektumot emeljük ki a háttérből. A *többszintű* küszöbölés esetén $N > 2$, ami lehetőséget nyújt arra, hogy háromszintű vágással (*trilevel thresholding*-gal) az objektum kűlső részét elkülönítsük. Jegyzetünkben csak kétszintű eljárásokkal foglalkozunk, de megtárgyaljuk az kiterjesztés lehetőségét is.

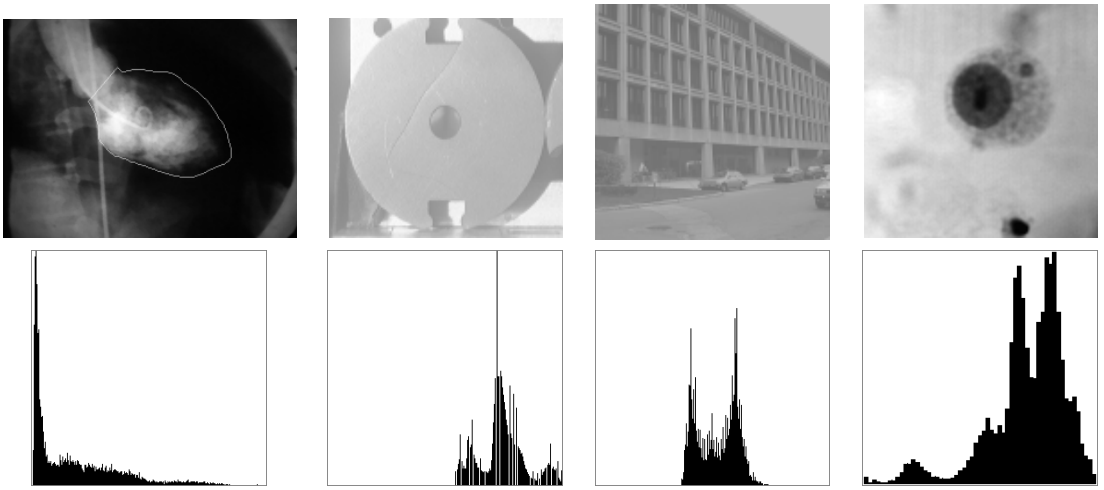


eredeti kép kétszintű küszöbölés 3-szintű küszöbölés

6.2. ábra. Példák két- és háromszintű automatikus küszöbölésre.

6.2. Hisztogram alapú küszöbölés

A küszöbértékeket sokféleképpen lehet meghatározni. Az automatikus küszöb-beállítás legelterjedtebb alapja az *intenzitás-hisztogram*, amely azt mutatja, hogy egy k intenzitás milyen



túl sötét túl világos alacsony kontraszt jó kontraszt

6.3. ábra. Hisztogram példák.

gyakran fordul elő a képben, azaz mi az előfordulási valószinűsége:

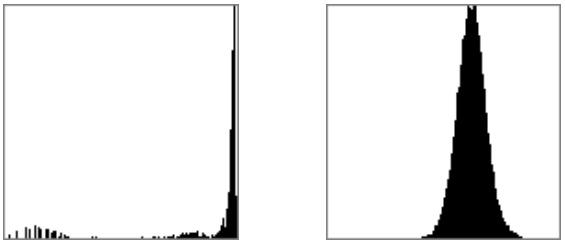
$$P(k) = \frac{n_k}{n}, \tag{6.2}$$

ahol n_k a $k = 0, 1, \ldots, 255$ intenzitású képelemek száma, $n = \sum_k n_k$ a képelemek teljes száma. A hisztogram kiszámítása egyszerű és gyors:

1. Inicializáljuk az n hosszúságú p tömböt: $p[k] = 0$.
2. Bejárjuk a képet és kitöltjük a p-t: k pixelértékre $p[k] \leftarrow p[k] + 1$
3. A végén normalizálunk: $P[k] = \frac{p[k]}{n}$.

A 6.3. ábra négy példát mutat valós hisztogramra. A bal szélső kép túl sötét, ezért a pixelek nagy része az alacsony tartományban csoportosul. A második túl világos képen megfordul a helyzet, és a hisztogram eltolódik a magas intenzitások felé. A harmadik esetben az intenzitás tartomány jelentős része nincs kitöltve, ami alacsony kontraszthoz vezet. Végül az utolsó kép jó kontraszttal rendelkezik, és az intenzitási osztályok is jól kivehetők, mint a hisztogram egymástól elkülönülő modusai.

Olyan esetekben, amikor a modusok nem egyértelműek, vagy rossz helyen vannak, vagy esetleg csak egy modus van, a képvágás nehéz, de nem feltétlenül reménytelen. A kedvezőtlen hisztogramtípusokat a 6.4. ábra szemlélteti. A baloldali képen a kiemelkedő modus az intenzitás tartomány szélén van, ezért a hisztogramot várhatóan nehezebb lesz modellezni. A többi feltételezett modus és a maximumok közötti völgy nehezen azonosítható. A másik kép egymodusu, ami ellentmond annak a feltételezésnek, hogy a képet több intenzitási osztály alkotja.



nehezen küszöbölhető egymodusú

6.4. ábra. Kedvezőtlen hisztogramtípusok.

Az esetek többségében nem egy, hanem több elfogadható küszöbérték van, ezt a 6.5. ábrával illusztráljuk. A hisztogramon a két csúcs közötti völgyben a G (good, jó) betűvel jelöltük azt az intervallumot, ahol az elfogadható értékek vannak. Ebből az intervallumból ered az a két jó küszöb, amelynek az eredményét az ábra mutatja. A túl alacsony (L, low) küszöb megtöri a vonalakat, a túl magas (H, high) pedig összemossa a vonalakat.

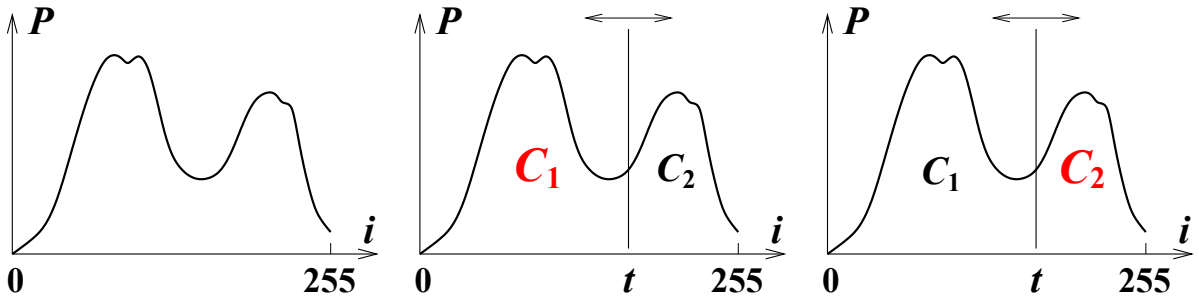
kép jó 1 jó 2 alacsony magas hisztogram

6.5. ábra. Példák jó és rossz küszöbválasztásra.

6.3. Két küszöbölési módszer

6.3.1. Otsu-módszer

Az egyik legelterjedtebb küszöb-beállítási módszert Nabuyuki Otsu japán kutató javasolta 1978-ben. (A nevet "Otszunak" vagy "Occunak" kell kiejteni.) A eljárás lényege egyszerű. Minden lehetséges t küszöbérték két intenzitás-osztályt hoz létre. Definiáljuk a két osztály *elválasztásának* statisztikai mértékét, azaz a két osztály közti távolságot a t függvényében, aztán keressük meg az elválasztást maximalizáló t_{opt} küszöbértéket! A módszert a 6.6. ábra szemlélteti.



a teljes hisztogram a C_1 osztály a C_2 osztály

6.6. ábra. A teljes hisztogram és a két részhisztogram.

A teljes hisztogram átlaga és szórása:

$$\mu = \sum_{i=0}^{255} iP(i) \qquad \sigma^2 = \sum_{i=0}^{255} (i-\mu)^2 P(i),$$

intervalluma $[0, 255]$. $P(i)$ normalizálva van: $\sum_{i=0}^{255} P(i) = 1$.

A C_1 osztály átlaga és szórása:

$$\mu_1(t) = \frac{1}{q_1(t)} \sum_{i=0}^{t} iP(i) \qquad \sigma_1^2(t) = \frac{1}{q_1(t)} \sum_{i=0}^{t} [i-\mu_1(t)]^2 P(i),$$

intervalluma $[0, t]$. $q_1(t) = \sum_{i=0}^{t} P(i)$ az osztály súlya (relatív mérete).

A C_2 osztály átlaga és szórása:

$$\mu_2(t) = \frac{1}{q_2(t)} \sum_{i=t+1}^{255} iP(i) \qquad \sigma_2^2(t) = \frac{1}{q_2(t)} \sum_{i=t+1}^{255} [i-\mu_2(t)]^2 P(i),$$

intervalluma $[t+1, 255]$. $q_2(t) = \sum_{i=t+1}^{255} P(i)$ az osztály súlya és $q_2(t) = 1 - q_1(t)$.

Minden t-re a σ^2 teljes szórásnak két komponense van:

- Az **osztályokon belüli szórás** (*within-class variance*), amely az osztály-szórások súlyozott összege:
$$\sigma_W^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \tag{6.3}$$
- Az **osztályok közti szórás** (*between-class variance*), amely az osztályok közti távolság. Ez σ^2 teljes szórás maradék része:
$$\sigma_B^2(t) = \sigma^2 - \sigma_W^2(t) \tag{6.4}$$

Érdemes megjegyezni, hogy – szemben a szórással – a részátlagok súlyozott összege konstans:

$$q_1(t)\mu_1(t) + q_2(t)\mu_2(t) = \mu.$$

Könnyű levezetni, hogy

$$\begin{aligned}\sigma_B^2(t) &= q_1(t)q_2(t)\left[\mu_1(t) - \mu_2(t)\right]^2 \\ &= q_1(t)\left[1 - q_1(t)\right]\left[\mu_1(t) - \mu_2(t)\right]^2.\end{aligned} \tag{6.5}$$

Az optimális küszöb a legjobban választja el a 2 osztályt. Mivel $\sigma_W^2(t) + \sigma_B^2(t)$ konstans, két ekvivalens feladatunk van:

- **minimalizáljuk** az osztályok átfedését, $\sigma_W^2(t)$-t, **vagy**
- **maximalizáljuk** az osztályok közti távolságot, $\sigma_B^2(t)$-t.

A második változatot fogjuk használni. Ehhez minden t-re ki kell számítani a $\sigma_B^2(t)$ célfüggvényt. Ezt a definíció szerint is megtehetjük úgy, hogy minden t-re kiszámítjuk a (6.5) egyenlet $q_1(t), \mu_1(t), \mu_2(t)$ komponenseit.

Van azonban gyorsabb *rekurzív megoldás* a komponensek meghatározására:

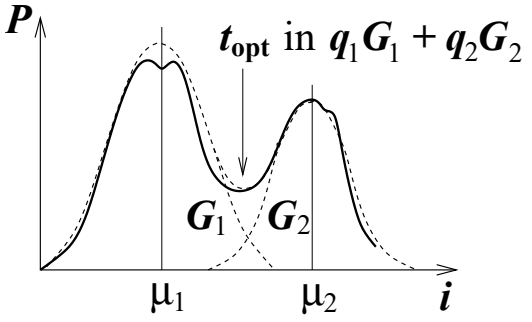
$$\begin{aligned}q_1(t+1) &= q_1(t) + P(t+1), && q_1(0) = P(0), \\ \mu_1(t+1) &= \frac{q_1(t)\mu_1(t) + (t+1)P(t+1)}{q_1(t+1)}, && \mu_1(0) = 0, \\ \mu_2(t+1) &= \frac{\mu - q_1(t+1)\mu_1(t+1)}{1 - q_1(t+1)}.\end{aligned} \tag{6.6}$$

Megjegyzés: A hisztogram elején esetleg levő nullákat, üres részt ki kell hagyni!

Az alábbiakban összefoglaljuk az Otsu-algoritmust.

3. Algoritmus: Otsu küszöbválasztás

1. Meghatározzuk a $P(i)$ hisztogramot, kiszámítjuk μ-t.
2. Minden t-re a (6.6) egyenlet szerint rekurzívan kiszámítjuk $q_1(t)$-t, $\mu_1(t)$-t és $\mu_2(t)$-t.
3. A (6.5) egyenlet szerint kiszámítjuk $\sigma_B^2(t)$-t.
4. Kiválasztjuk $t_{opt} = \arg\max_t \sigma_B^2(t)$.



6.7. ábra. A hisztogram-modellezés elve.

Az Otsu-módszernek *előnyei* a következők:

- Nem tételez fel konkrét, speciális hisztogram alakot.
- Megbízható és stabil.
- Kiterjeszthető többszintű küszöbölésre. N küszöbre egy 256^N méretű tömbben kell maximumot keresni.

A módszernek *hátrányai* is vannak:

- Feltételezi, hogy $\sigma_B^2(t)$ egymodusú, ami nem mindig igaz.
- $\sigma_B^2(t)$ gyakran lapos, ezért hamis maximumok lehetségesek.
- Hajlamos kis osztályokat mesterségesen kitágítani.

6.3.2. Hisztogram-modellezés Gauss-eloszlásokkal

A hisztogram-modellezésen alapuló módszert a 6.7. ábra szemlélteti. Az eljárás elve a következő. Felételezzük, hogy a $P(i)$ normalizált hisztogram *két Gauss-eloszlás súlyozott átlaga*, keveréke:

$$P(i) \approx G(i) \doteq q_1 G_1(i) + q_2 G_2(i). \tag{6.7}$$

A $G(i)$ modellfüggvénynek több paramétere van. A paramétereket megvariálva, $G(i)$-t illesztjük $P(i)$-hez, ezzel becslést adunk az optimális paraméterértékekre. Ha ez sikerült, azaz a modell érvényes, analítikus megoldást adunk az osztályozási hibát minimalizáló, statisztikai értelemben optimális küszöbre.

A modelleloszlás részletes alakja:

$$G(i, \mathbf{p}) = \frac{q_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{1}{2}\left(\frac{i-\mu_1}{\sigma_1}\right)^2} + \frac{q_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{1}{2}\left(\frac{i-\mu_2}{\sigma_2}\right)^2}, \tag{6.8}$$

ahol $\mathbf{p}$ a paramétervektor, amely hat paramétert tartalmaz: $\mathbf{p} = (q_1, q_2, \mu_1, \mu_2, \sigma_1, \sigma_2)$. Ezen belül q_1 és q_2 a két Gauss-eloszlás súlya és $q_1 + q_2 = 1$. Tehát, valójában csak *öt szabad paraméter* (szabadságfok) van. Legyen a független paraméterek vektora $\mathbf{p}' = (q_1, \mu_1, \mu_2, \sigma_1, \sigma_2)$.

A paraméteres modelleloszlás illesztéséhez a következő ötváltozós hibafüggvényt kell minimalizálni:

$$C(\mathbf{p}') = \sum_{i=0}^{255} \left[f(i, \mathbf{p}') - P(i) \right]^2 . \tag{6.9}$$

Ehhez többféle nemlineáris minimalizációs módszer áll rendelkezésre: Newton, Marquard-Levenberg, vagy a sztochastikus. Előfordulhat, hogy a kiválaszott módszer *nem hoz eredményt*, például, nem konvergál. Ez azt jelenti, hogy a Gauss-keverék modellünk az adott hisztogramra nem érvényes, így segítségével nem kapunk megoldást, küszöböt.

A fenti módszerek iteratívak, ezért fontos lehet a kindulópont kiválasztása. A 6.8. ábrán illusztrálunk néhány lehetséges megoldást, amely jól meghatározható modusok és völgyek esetén működik, amikor nem túl nagy a modusok közötti átfedés. A μ_1, μ_2 átlag a $P(i)$-ben a két domináns maximum x-helye, a $P''(i)$-ben pedig a két domináns völgy x-helye. A σ_1, σ_2 szórás becsléséhez felhasználhatjuk, hogy $P'(i)$-ben a maximum és a völgy közti x-távolság 2σ. Ha nincs más információnk, a súly kezdő értéke $q_1 = 0,5$ lehet.

Ha a modelleloszlás illesztése sikerült és megkaptuk az optimális paraméterértékeket $(\hat{q}_1, \hat{q}_2, \hat{\mu}_1, \hat{\mu}_2, \hat{\sigma}_1, \hat{\sigma}_2)$, sor kerülhet az optimális küszöbérték meghatározására. A 6.9. ábra szerint a hibás osztályozás valószínűsége

$$E(t) = E_1(t) + E_2(t) = \int_{-\infty}^{t} G_2(x)\, dx + \int_{t}^{\infty} G_1(x)\, dx, \tag{6.10}$$

ahol $E_1(t)$ annak a valószínűsége, hogy egy C_1-pixelt C_2-ként osztályozunk, $E_2(t)$ pedig annak, hogy egy C_2-pixelt C_1-ként osztályozunk.

A hiba akkor minimális, ha

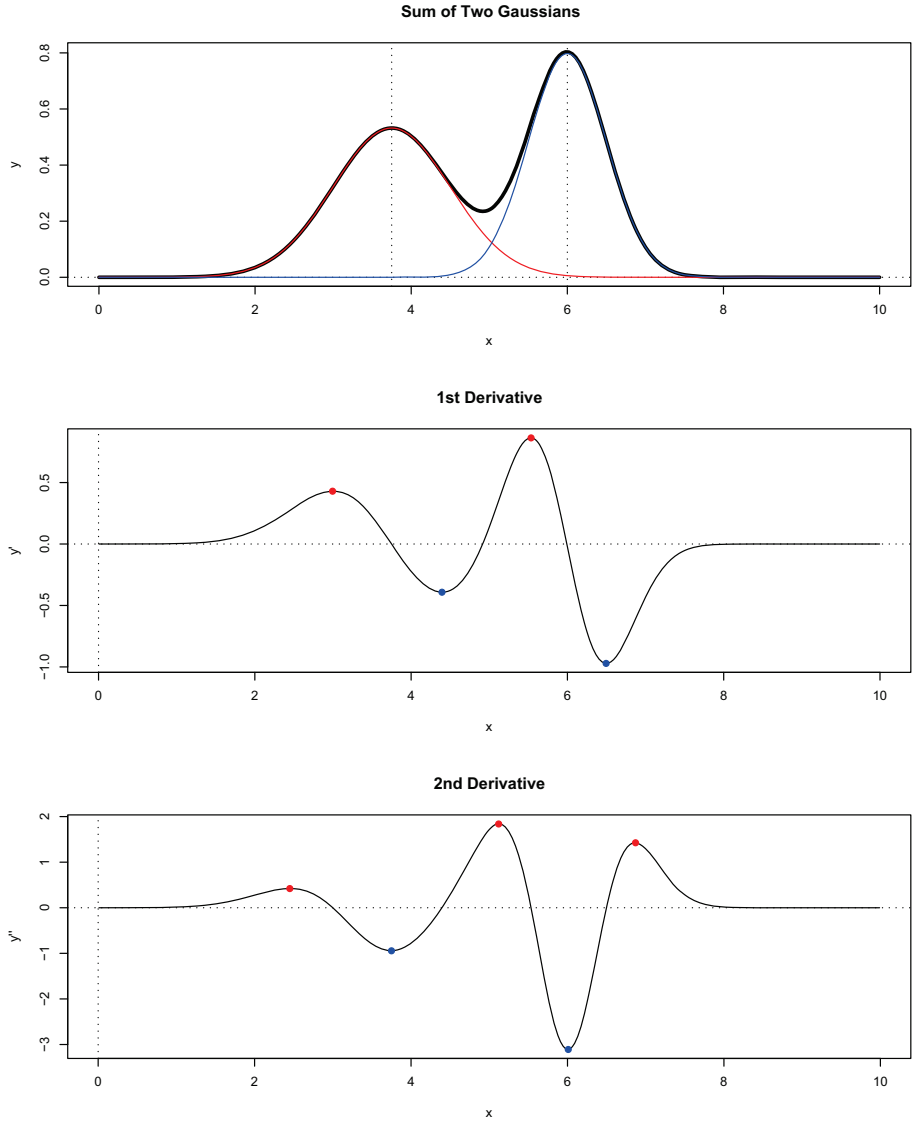
$$\frac{\partial E}{\partial t} = 0.$$

Elvégezve a deriválást, be lehet bizonyítani, hogy a t_{opt} **optimális küszöb** a következő egyenlet megoldása:

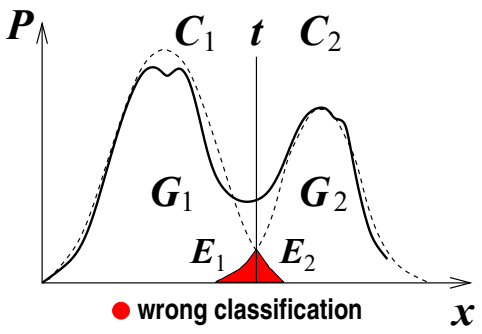
$$At^2 + Bt + C = 0, \tag{6.11}$$

ahol

$$\begin{aligned} A &= \hat{\sigma}_1^2 - \hat{\sigma}_2^2, \\ B &= 2(\hat{\mu}_1 \hat{\sigma}_2^2 - \hat{\mu}_2 \hat{\sigma}_1^2), \\ C &= \hat{\sigma}_1^2 \hat{\mu}_2^2 - \hat{\sigma}_2^2 \hat{\mu}_1^2 + 2\hat{\sigma}_1^2 \hat{\sigma}_2^2 \ln \left(\frac{\hat{\sigma}_2 \hat{q}_1}{\hat{\sigma}_1 \hat{q}_2} \right). \end{aligned}$$



6.8. ábra. A Gauss-paraméterek kezdeti becslése. Forrás: Earl F. Glynn, Stowers Institute for Medical Research.



6.9. ábra. A hibás osztályozás valószínűsége.

Bár az egyenlet mindig megoldható, nem biztos, hogy a kapott t_{opt} értékeket fel tudjuk használni. A következő esetek lehetségesek:

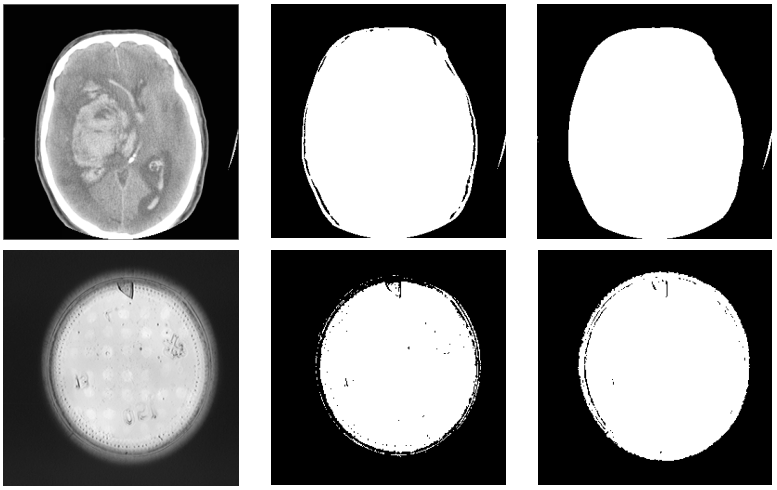
- Az egyenletnek *egy valós gyöke* van a $[0, 255]$ tartományban. Akkor ez az optimális küszöb.
- Az egyenletnek *két valós gyöke* van a $[0, 255]$ tartományban. Ebben az esetben válasszuk azt, amelyre az $E(t)$ hiba kisebb!
- Az egyenletnek *nincs valós gyöke* a $[0, 255]$ tartományban. Akkor nincs optimális küszöb.

Az alábbiakban összefoglaljuk a Gauss-keverék algoritmust.

4. Algoritmus: Gauss küszöbválasztás

1. Meghatározzuk a $P(i)$ normalizált hisztogramot.
2. Minimalizáljuk a (6.9) egyenlet által definiált $C(\mathbf{p}')$ hibafüggvényt és megkapjuk a optimális paraméter-becsléseket.
3. Megoldjuk a (6.11) egyenletet, két gyököt kapunk.
4. Csak a $[0, 255]$ tartományban levő, valós gyök számít. Ha csak egy van, akkor ez a megoldás. Ha kettő, akkor válasszuk azt, amelyre az $E(t)$ hiba kisebb!

Az algoritmust nem kizárólag a küszöbölés miatt volt érdemes megismerni. A több Gauss-eloszlást tartalmazó modellt sok más területen is használják, például, a videó alapú változás-detektálásban. A modell angol neve *Multiple Gaussian Model*, vagy *Gaussian Mixture*. Küszöbölés esetén a módszer az alábbi *előnyökkel* rendelkezik:



6.10. ábra. Sikeres küszöbölés példái.

- A hisztogram-modell viszonylag általános.
- Ha érvényes a modell, optimális megoldást kapunk.
- Alkalmazható kisebb osztályokra is.

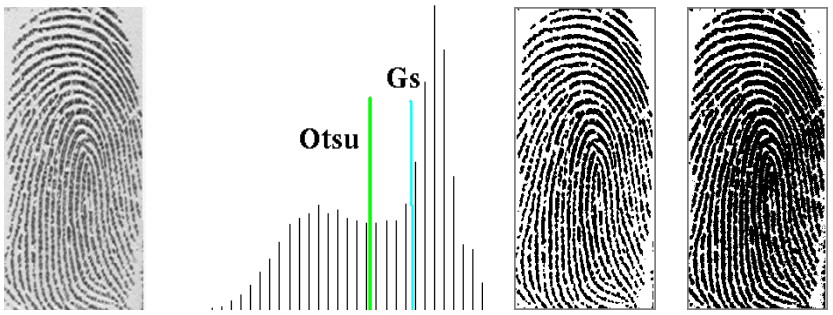
A *hátrányok* a következők:

- A valós hisztogram nem mindig követi a normáleloszlást. Többek között azért, mert az intenzitások végesek és nemnegatívok. A 0 vagy 255 közeli hisztogramcsúcsokat emiatt nehéz modellezni.
- A többszintű küszöbölésre való kiterjesztés csak lényeges modell-egyszerűsítés árán lehetséges. Például, azzal a feltételezéssel, hogy a modusok jól elkülönülnek.
- Az egymáshoz közeli vagy lapos modusokat nehéz detektálni.

6.4. Küszöbölés példái és elemzése

6.4.1. Küszöbölési példák

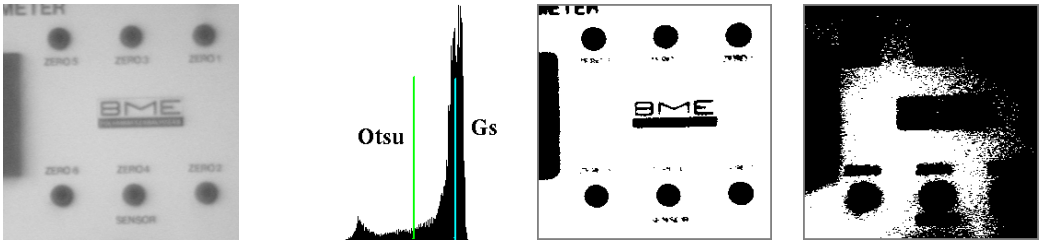
A 6.10. ábra két olyan példát mutat, amikor mind az Otsu, mind a Gauss-módszer jó eredményt nyújt. Az utóbbi valamivel alacsonyabb küszöböt produkál, ezért egy kicsit jobban tölti ki a kontúrokat, mint az előbbi, ami adott esetben előnyös lehet.



6.11. ábra. Elfogadható küszöbölés példái.

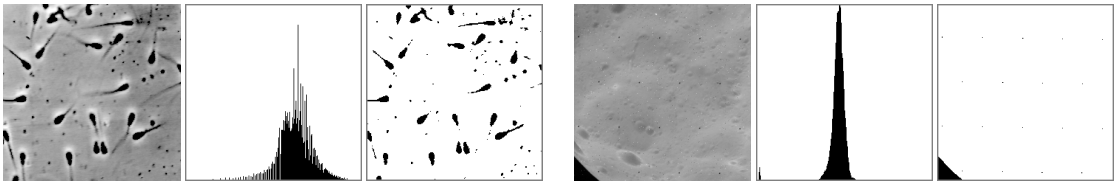
A 6.11. ábrán egy ujjlenyomat a bemeneti kép. Itt az Otsu-módszer eredménye megint jó, pontosan a völgy legmélyebb pontjába helyezi a $T = 158$ küszöböt és a vonalak jól elkülönülnek. A Gauss-módszer ezzel szemben egy épp hogy elfogadható, vagy inkább nem egészen jó eredményt ad, mert a $T = 199$ küszöb túl magas és a vonalak egy része összeolvad.

A 6.12. ábrán látható kép esetén az objektum osztály sokkal kisebb, mint a háttér. Ráadásul, a háttér nem teljesen egyenletes. Ennek ellenére az Otsu-algoritmus eredménye elfogadható, amihez talán egy kis szerencse is kellett. A Gauss-eredmény teljesen rossz (itt látszik egyébként, hogy a háttér változó). Az algoritmus megpróbálja szétválasztani a háttér két modusát, mert az objektum osztály kicsi és messze van. A hibát a rossz inicializálás is okozhatta.



6.12. ábra. Hibás Gauss-féle küszöbölés példája.

Végül a 6.13. ábrán bemutatott esetekben csak az Otsu-módszer ad eredményt, és az eredmény értelmezhető, annak ellenére, hogy a hiszogramok alakja kifejezetten kedvezőtlen. A második képen az egyik osztály nagyon kicsi, itt az értelmezhető eredményhez nyilvánvalóan nagy szerencse kellett, mert egyik módszernek sem kedvez az ilyen hatalmas méretkülöbség a két osztály között. A Gauss-módszer nem ad semmi eredményt. A baloldali képre nem sikerült az illesztés, mert a hisztogram egymodusú. A másik képre az illesztés ugyan sikerült, de a (6.11) egyenletnek nem volt valós gyöke.

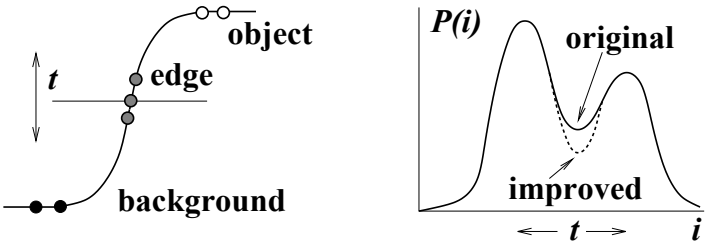


6.13. ábra. Sikertelen Gauss-féle küszöbölés példája.

6.4.2. Küszöbölés elemzése

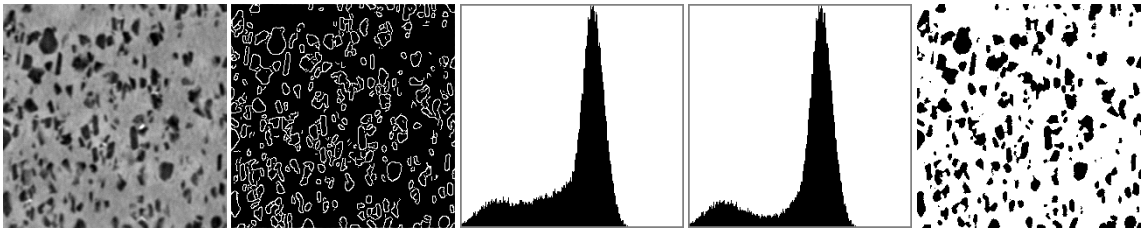
Amikor a hisztogram alakja nem kedvező, felmerül a kérdés, hogy mivel tudnánk ezen javítani. Célunk az objektum és a háttér jobb elválasztása, ehhez el kell gondolkodnuk azon, hogy az elválasztásnak mi az akadálya. A problémát a 6.14. ábra szemlélteti. Láthatjuk, hogy az élek körüli intenzitások a két osztály között helyezkednek el és elmossák a hisztogramban az osztályok közötti határt.

Így merül fel a *hisztogram-javítás* ötlete, amely egyszerű és természetes. Az élek körüli pontokban a gradiens magas, a háttér- és objektum-pontokban alacsony. Zárjuk ki hisztogramból a magas gradiensű pontokat, az éleket, abban a reményben, hogy a hisztogram alakja kedvezőbb lesz! A 6.15. ábra példát ad az elképzelés sikeres működésére.



6.14. ábra. A gradiens felhasználása hisztogram-javításra.

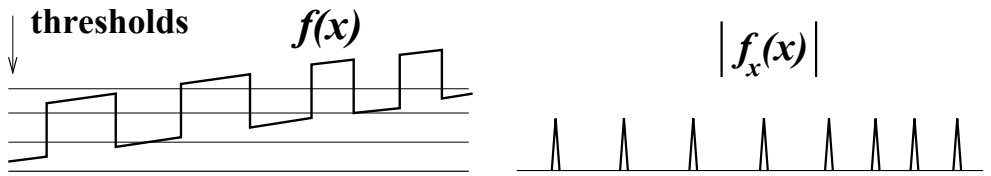
Mint már említettük a küszöbölés és az éldetektálás a képszegmentáció két alternatív megközelítése. Rögzített értékű küszöbölés nem adaptív művelet, ezért nem működik vál-



6.15. ábra. Példa hisztogram-javításra.

tozó háttérű képekre, amikor $T(x, y)$ adaptív küszöbre lenne szükség. Ezt a hátrányt a 6.16. ábra illusztrálja, ahol a küszöböléssel ellentétben az éldetektálás mint lokális, adaptív művelet megoldja a problémát, hiszen a módszernek elég a lokális kontraszt.

Azonban a küszöbölésnek is megvan a maga előnye az éldetektálással szemben, nevezetesen az, hogy zárt kontúrokat garantál. Az éldetektálás erre nem képes, ha a kontúr egy része nem elég kontrasztos.

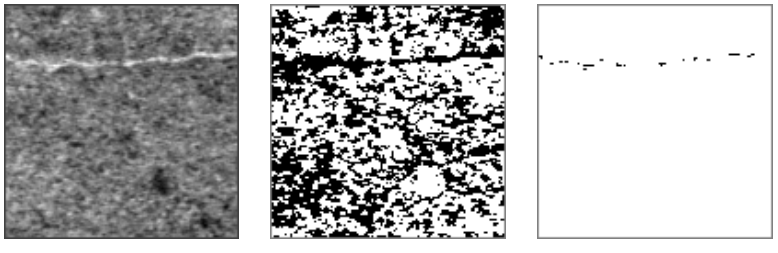


változó háttérű kép nem küszöbölhető de az éleket detektáljuk

6.16. ábra. Küszöbölés kontra éldetektálás.

A változó háttérű képek gondot jelentenek a hisztogram alapú küszöbölés számára, de nem ezek szabják az *elvi korlátját*, hanem az a tény, hogy a hisztogram nem tartalmaz semmi geometriai információt, nem tükröz szomszédsági vagy más strukturális relációkat a pixelek között. Ennek egyszerű, de szemléletes példája az, hogy ha egy periodikus képen véletlenszerűen felcseréljük a képelemeket és egy random képet kapunk, annak a hisztogramja pontosan ugyanaz marad, mint a periodikus képé.

A küszöböléssel szembeni elvárások feladatfüggőek, köztük geometriai tulajdonságok is lehetnek. Például a 6.17. ábrán a repedést szeretnénk kiemelni, amiről tudjuk, hogy vonalszerű. Ezt az információt az Otsu-algoritmus nem tudja hasznosítani, és az emberi szemmel szemben a vonalat nem tudja detektálni. Egy másik hisztogram alapú küszöbölési eljárás szerencsés módon kiemeli a vonalpixeleket, de nem azért, mert tudja, hogy mi a vonal, hanem kizárólag annak eltérő intenzitása miatt. A tanulság az, hogy olyan *régió alapú* szegmentációs módszerekre van szükség, amelyek figyelembe veszik az intenzitást is és a struktúrát is. Ezekről a következő fejezetben lesz szó.



kép Otsu más módszer

6.17. ábra. Küszöbölés korlátai.

7. fejezet

Régió alapú szegmentálás

Ebben a fejezetben olyan képszegmentációs algoritmusokkal foglalkozunk, amelyek – szemben a küszöböléssel – biztosítják a kapott szegmensek, régiók összefüggőségét. Diszkrét képekben az összefüggőség fogalma nem teljesen triviális és a fejezetben esetenként használt "határpixel" fogalom sem az. Egyelőre azonban nem adjuk meg a precíz definíciókat, hogy feleslegesen ne bonyolítsuk a módszerek ismertetését. Aki kíváncsi ezekre, elolvashatja a 8. fejezet bevezető részét.

7.1. A régió alapú szegmentálás elvei

A régió alapú szegmentálás célja felosztani az I képet n darab $R_1, \ldots, R_n$ összefüggő és homogén regióra. Ehhez definiálunk egy $P(R)$ *homogénségi kritériumot*, amely minden $R \subset I$ régióra alkalmazható. $P(R) = TRUE$, ha minden R-beli képelemnek *hasonló tulajdonságai* vannak, azaz R homogén. Egyébként, $P(R) = FALSE$, vagyis R inhomogén (nem homogén).

A homogénségi kritériumok különbözőek lehetnek. Egy R régiót, például, akkor nevezhetünk homogénnek, ha az alábbi feltételek valamelyike teljesül:

- $|I_{max} - I_{min}|$ kicsi;
- bármelyik $I(x, y) \in R$ pixelre $|I(x, y) - I_{mean}|$ kicsi, ahol I_{mean} a régió átlaga;
- a σ_R intenzitás szórás a régióban kicsi.

A szegmentálás eredménye így attól függ, hogy milyen képi tulajdonságokat használunk (pl. intenzitás, szín, textúra), hogyan hasonlítjuk össze a tulajdonságokat, és mekkora változásokat tolerálunk egy régión belül.

A szegmentálás matematikai definíciója a következő. Az I képet n darab $R_1, \ldots, R_n$ régióra bontjuk úgy, hogy:

1. $\bigcup_{i=1}^{n} R_i = I$, azaz minden pixel valamely régióhoz tartozik.

2. Minden R_i régió topológiailag összefüggő.
3. $R_i \cap R_j = \emptyset$ minden i, j-re, $i \neq j$, azaz nincs közös pixel.
4. $P(R_i) = TRUE$ minden i-re, azaz minden régió homogén.
5. $P(R_i \cup R_j) = FALSE$ minden szomszédos R_i, R_j-re, $i \neq j$, azaz bármelyik két szomszédos régió uniója inhomogén.

Az utolsó feltétellel azt akarjuk elérni, hogy a regiók száma minimális legyen, különben az egyéni pixelekből álló, triviális szegmentálás is megfelelne a definíciónak.

7.2. Régió alapú szegmentálási eljárások

Az alábbiakban ismertetett eljárások lényege viszonylag egyszerű, de korrekt, hatékony megvalósításuk már nem.

Régió-növesztés (*region growing*) esetén képelemeket vagy kisebb régiókat nagyobb régiókba csoportosítunk, amelyeket addig növesztünk, amíg a homogénségi kritériumunk engedi.

Pixel-felhalmozás (*pixel aggregation*) a régió-növesztés egyszerű formája. Kiválasztunk *magpontokat* (*seed points*) és egy homogénségi kritériumot. Minden magpontból régiót növesztünk úgy, hogy hozzáadunk olyan szomszédos képelemeket, amelyek nem sértik az eddig kialakul régió homogénségét.

Régió-egyesítés (*region merging*) szintén a régió-növesztés egyik formája. Külön műveletként is használják (pl. pixel-felhalmozás után), vagy beépítik egy iteratív szegmentációs algoritmusba.

7.2.1. Régió-növesztés

A pixel-felhalmozással történő régió-növesztési algoritmus vázlata a következő:

5. Algoritmus: Pixel-felhalmozás

1. **Inicializálás**
   - Kiválasztunk N darab s_i magpontot és egy T küszöböt.
   - Magpontokkal inicializálunk N régiót: $R_i^{(0)} = s_i$.
   - Inicializáljuk a régiók átlagértékeit: $M_i^{(0)} = I(s_i)$.
2. **Iteráció**, k-ik lépés
   - Megvizsgáljuk az összes $R_i^{(k)}$ minden **határpixelének** 8-szomszédjait.

- Ha van olyan új szomszéd, p, amelyre $|I(p) - M_i^{(k)}| \leq T$, akkor p-t hozzáadjuk $R_i^{(k)}$-hez.

3. **Megállunk**, ha nem tudunk tovább növeszteni; különben, felfrissítjük az összes $M_i^{(k)}$-t és iterálunk.

A magpontok kiválasztása nem egyszerű feladat, és az eredmény függ a magpontoktól. Ha nem tudjuk, hogy hány szegmens van a képen és honnan érdemes kiindulni, akkor lehet sok magpontot véletlenszerűen szétszórni és végrehajtani a pixel-felhalmozást. Ez nyilvánvalóan *túlszegmentáláshoz* vezet, amelyet utólagos régió-egyesítéssel célszerű kompenzálni.

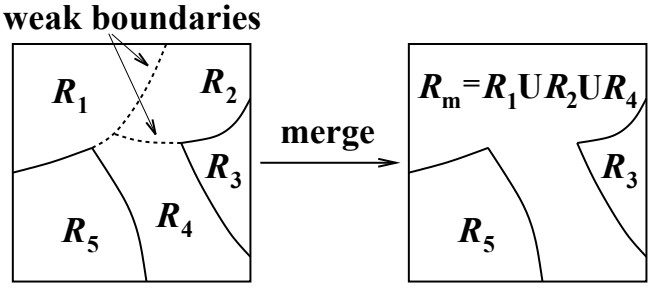
A régió-növesztés *előnyei* a garantáltan összefüggő régiók és a régiókkal szembeni elvárásoknak a folyamatba történő beépíthetősége. Az elvárások között alak, méret, vagy textúra is lehet, ez által kívánt tulajdonságokkal rendelkező régiókat kapunk.

A módszer *hátrányai* a következők:

- A magpixelek kijelölése nem egyszerű.
- Az eljárásban sok a heurisztika (egyesítési szabályok, gyenge határok).
- A módszer eredendően soros és nem könnyen párhuzamosítható.
- Az eljárás sorrend-függő, azaz az eredmény függ a feldolgozási sorrendtől. Erre ugyan vannak részmegoldások, de azok nem tökéletesek.

7.2.2. Régió-egyesítés

Régió-egyesítés felhasználható, amikor a szomszédos régiók hasonló tulajdonságokkal rendelkeznek, ezért feltehetően egy szegmens részei és egybeolvaszthatók. Ennek egy tipikus példája, hogy régió-felhalmozáskor több mag került egy szegmensbe, és emiatt a szegmens több régióra bomlott, amelyet egybe kell olvasztani. Az eljárást a 7.1. ábra illusztrálja.



7.1. ábra. Régió-egyesítési kritériumok.

Két szomszédos régió (R_i és R_j) egyesíthető, ha uniójuk homogén: $P(R_i \cup R_j) = TRUE$. Egy másik egyesítési lehetőség az, hogy a két régió közti határ "gyenge": nincs rajta erős él (nagygradiensű pixel), vagy sok a kisgradiensű pixel.

Régió-növesztés egyesítés révén is lehetséges. Ehhez a képet sok kisebb homogén *magrégióra* bontjuk, amelyek megközelítőleg állandó intenzitással vagy más lokális tulajdonsággal bírnak. Ezek után iteratívan egyesítjük a szomszédos régiókat és akkor állunk meg, amikor további egyesítés már nem lehetséges.

Bár vannak olyan (pl. orvosi) alkalmazások, amikor a magpixeleket kézzel is meg lehet adni, az esetek többségében az automatikus megoldásokat preferálják. Ehhez használhatunk egy sűrű, véletlenszerűen vagy szabályosan elhelyezett magpixel-halmazt, amiből kiindulva felhalmozással sok magrégiót kapunk. (Az esetleg be nem járt pixeleket külön kell feldolgozni.) A másik lehetőség, hogy végigjárjuk a képet és minden fel nem dolgozott pontból indítunk felhalmozást, a kapott magrégiókat pedig iteratívan egyesítjük. Szem előtt kell tartani azonban, hogy az eredmény függ a végigjárás sorrendjétől

7.2.3. Vágás-és-egyesítés

A vágás-és-egyesítés (*split-and-merge*) képszegmentálási algoritmust és a négyesfa (*quadtree*) fogalmát a 7.2. ábrával szemléltejük. Az algoritmus vázlata a következő:

6. Algoritmus: Vágás-és-egyesítés algoritmus

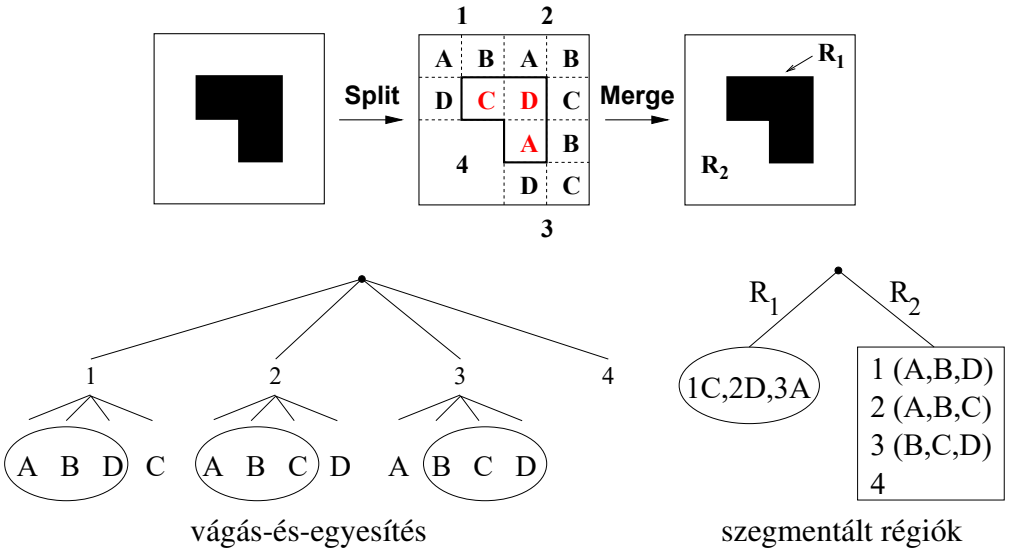
1. Föntről lefelé (*top-down*)
   - felosztjuk a képet egyre csökkenő méretű R_i kockákra
   - megállunk, ha az összes kocka homogén: $P(R_i) = TRUE$

   $\Rightarrow$ az eredmény egy négyesfa
2. Lentről fölfelé (*bottom-up*)
   - minden szinten egyesítünk két szomszédos R_i és R_j régiót, ha $P(R_i \cup R_j) = TRUE$
3. Iteráljuk a két lépést, amíg van új felosztás vagy egyesítés

---

A módszer összefüggő régiókat eredményez. *Előnye*, hogy beépíthetjük az elvárásokat, bár kisebb mértékben, mint az előző módszerek esetén. Tipikus alkalmazási területe a Földrajzi Informaciós Rendszerek (GIS). Az algoritmus *hátrányai* a következők:

- Ha egy képet elforgatva vagy eltolva digitalizálunk, más szegmentálási eredményt kaphatunk.



7.2. ábra. Szegmentálás vágás-és-egyesítéssel.

7.3. ábra. Pixel-felhalmozás növekvő küszöbbel.

- A négyzetes struktúra látszik az eredményképen.
- Az algoritmus csak vertikális adatforgalmat támogat. Ezért két szomszédos régió messze lehet egymástól a négyesfában és akkor egyesül, amikor a két ág találkozik.

7.3. Példák és összefoglaló

A 7.3. ábra a növekvő T küszöbbel történő pixel-felhalmozást illusztrálja. A baloldalon a bemeneti kép látható, rajta a kézzel elhelyezett magpontokkal.

Amikor $T = 20$ – ami kevés mozgásszabagságot ad – a változó intenzitású arcrégió nem tud nőni, és a kisváltozású hajrégió nagy területet foglal el. Ahogy nő a küszöb, az arcrégió tovább tud terjedni, a hajrégió ennek megfelelően zsugorodik. Az eredmény természetesen függ a magpixelektől.

A 7.4. ábra régió-növesztési példát mutat, szintén növekvő küszöbbel. Ebben az esetben végigpásztáztuk a képet és minden új pontból indítottunk magrégiót. A növekvő küszöbnek itt ellenkező hatása van, mert a régiók száma egyre csökken. A szegmentálás kevésbé lesz finom, a hajrégió pedig egyre nő.

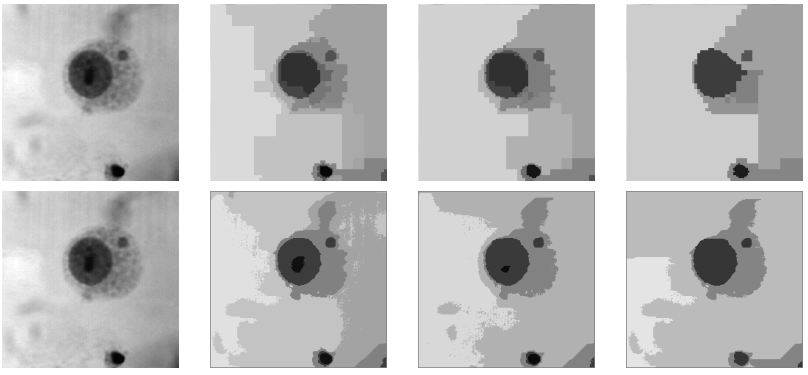
$T = 20$ $T = 30$ $T = 40$

7.4. ábra. Régió-növesztés egyesítéssel növekvő küszöbbel.

Az utolsó, 7.5. ábrán összehasonlítjuk a vágás-és-egyesítés és az egyesítés algoritmusokat. A felső sorban szereplő vágás-és-egyesítési eredményen látszik a négyesfára jellemző, négyzetes struktúra. A másik algoritmus eredménye finomabb.

Végezetül, a teljesség igénye nélkül felsorolunk több olyan szegmentálási módszert is, amely más elveken működik.

- *Variációs*, pl. aktív kontúrok, Level Set módszerek.
- *Statisztikai*, pl. Markov Random Mezők (*Markov Random Field*, MRF).
- *Modell* alapú, azaz adott alakú és más tulajdonságú régiók keresése.
- *Él* alapú, vagyis az objektumok behatárolása élkövetéssel.



sejtkép $T = 40$ $T = 50$ $T = 60$

7.5. ábra. Vágás-és-egyesítés (felső sor) kontra egyesítés.

- *Textúra, szín* alapú.
- *Mozgás* alapú.

8. fejezet

Középvonal és váz

A jegyzet maradék részét bináris képek feldolgozásának és az általuk megjelenített kétdimenziós alakzatok leírásának szenteljük. A küszöbölési módszerek segítségével két osztályba tudjuk sorolni a pixeleket, ezek az *objektum* (*figure*) és a *háttér* (*ground, background*). A továbbiakban az **algoritmusokban** az objektumpixelek értéke 1 lesz, háttérpixeleké pedig 0. Az **illusztrációkban** hagyományt követve az objektumot feketével, a hátteret fehérrel ábrázoljuk.

A fejezetben megismerünk két fontos, konceptuálisan hasonló, de algoritmikusan különböző alakzatleírási módszert. A módszerek olyan vázszerű szerkezeteket hoznak létre, amelyek jól tükrözik az alakzat struktúráját és lehetővé teszik annak elemzését. Ahhoz azonban, hogy megértsük az algoritmusok működését, először at kell tekintenünk a digitális topológiá elemi fogalmait, amivel az előző fejezetben adósak maradtunk.

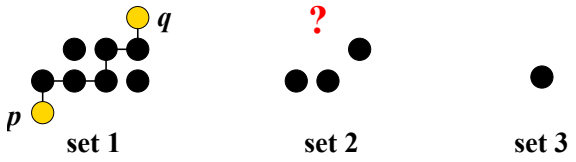
8.1. Egy kis digitális topológiá

Digitális képekben egy képelemnek négy vagy nyolc szomszédja lehet attól függően, hogy melyik változatot választjuk. A két esetet a 8.1. ábra szemlélteti.

Azt mondjuk, hogy két $p, q \in S$ pixel **összefügg az** S **halmazban**, ha létezik egy $p_0 = p, p_1, \ldots, p_n = q$, $p_i \in S$ pixelsorozat, amely összeköti p-t és q-t úgy, hogy p_i és p_{i-1} szomszédok. Egy S képelemhalmaz pedig akkor **összefüggő régió**, ha az összes képeleme összefügg az S-ben. A fogalmakat a 8.2. ábra illusztrálja, ahol az 1. halmaz 4- és 8-összefüggő, és a p és q pixel 4- és 8-összefüggő az 1.halmazban. Világos, hogy egy 4-

négy szomszéd nyolc szomszéd

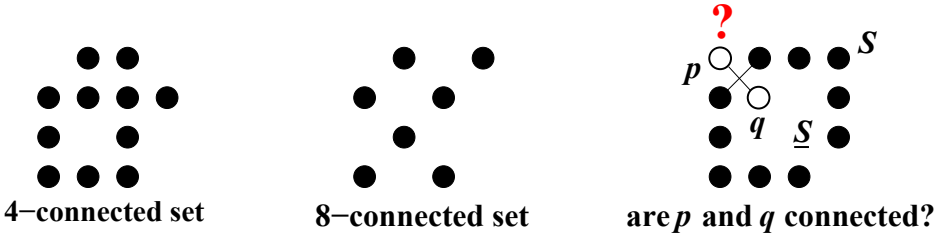
8.1. ábra. A diszkrét szomszédság két típusa.



8.2. ábra. Példák összefüggő és össze nem függő ponthalmazokra.

összefüggő régió egyben 8-összefüggő is, de fordítva ez már nem mindig igaz. A 2. halmaz például 8-összefüggő, de nem 4-összefüggő, hanem két két 4-összefüggő régióból áll. Az egy pixelből álló 3. halmaz értelemszerűen 4- és 8-összefüggő.

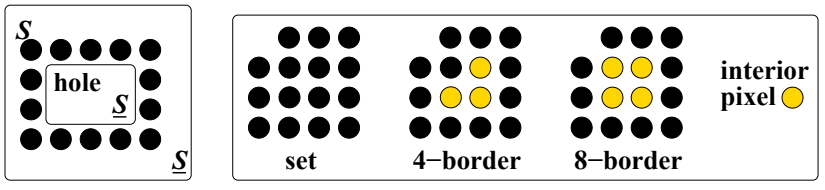
Amikor azt mondtuk, hogy a digitális képekben az összefüggőség definíciója nem teljesen egyszerű, a 8.3. ábrán illusztrált helyzetre gondoltunk. Ugyanis *ellentmondást* kapunk, ha ugyanazt az összefüggőséget alkalmazzuk az S objektumra és az $\underline{S}$ háttérre: az S és $\underline{S}$ pixelpárjai "kereszbe" lesznek összekötve.



8.3. ábra. Összefüggőség objektumra és háttérre.

A *megoldás* az, hogy más összefüggőséget alkalmazunk a két halmazra: ha a 8-ast az objektumra, akkor a 4-est a háttérre; ha a 4-est az objektumra, akkor a 8-ast a háttérre.

A fejezetben ismertetésre kerülő algoritmusok szempontjából két további fontos fogalom a lyuk és a határpixel (8.4. ábra). Az S halmazban levő **lyuk** egy, az S által körülvett $\underline{S}$-komponens. Az S **határpixele** egy olyan S-pixel, amelynek $\underline{S}$-beli szomszédja van, mégpedig az $\underline{S}$*-re alkalmazott összefüggőség* értelmében. A **belső pixel** minden nem határpixel.



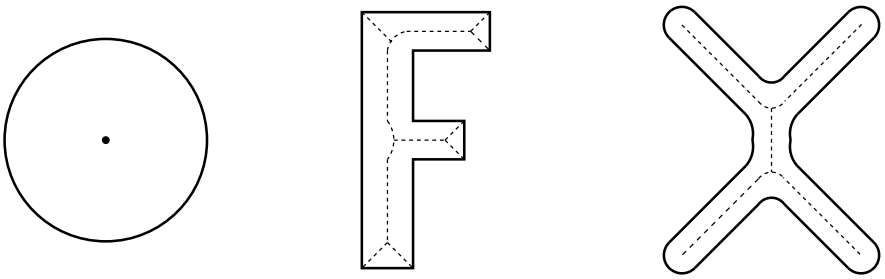
8.4. ábra. Lyukak és határok.

8.2. Középvonal

Folytonos esetben egy alakzat **középvonala** (*medial axis*, MA) az alábbi feltételeket teljesítő $p(x, y)$ pontok halmaza:

1. $p(x, y)$ az alakzat egyik belső pontja;
2. $p(x, y)$ egyenlő távolságra van *két vagy több legközelebbi* kontúrponttól.

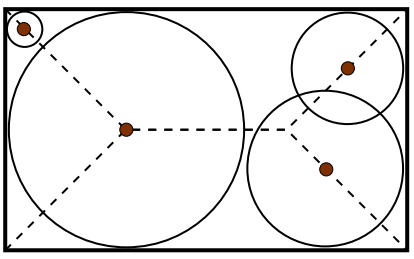
A középvonalat előállító műveletet **középvonal-transzformációnak** (*medial axis transform*, MAT) hívják. A 8.5. ábra három egyszerű alakzat középvonalát mutatja. Egy lemez középvonala a lemez középpontja. Megfigyelhető, hogy minden szögfelező a középvonal ága, hiszen eleget tesz a fenti definíciónak.



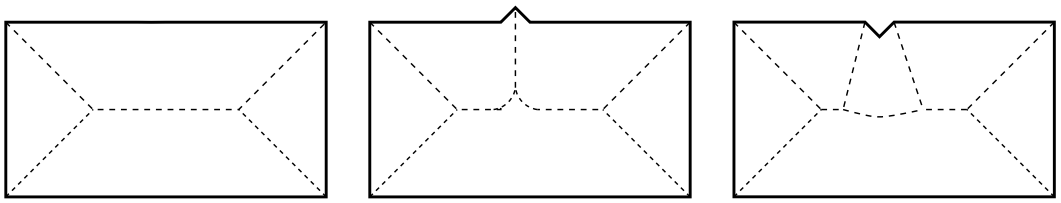
8.5. ábra. Egyszerű alakzatok középvonala.

Egy *folytonos* alakzat középvonala mindig *összefüggő*. Ezzel szemben, mint később látni fogjuk, egy diszkrét alakzatra ez nem érvényes. Egy alakzat *visszaállítható középvonalából*, ha tudjuk az MA minden pontja és a legközelebbi kontúrpont közti távolságot; egyébként, nem. A visszaállítást a 8.6. ábra szemlélteti, ahol az MA a beírt körök középpontjainak halmaza. Az alakzat visszaállítható a lemezek uniójaként, ha ismerjuk azoknak a sugarát. Ha nem ismerjuk, a rekonstrukció nem lehetséges.

A középvonal egy alakzat kompakt és hatékony reprezentációja, ha az alakzat hosszúkás részekből áll, pl. egy betű vagy kromoszóma. Sajnos, az MA *torzítás- és zaj-érzékeny*, mert egy kis alakzat-torzítás egy nagy középvonal-változást eredményezhet. A jelenséget



8.6. ábra. Alakzat visszaállítása középvonalából.



8.7. ábra. Középvonal torzítás-érzékenysége.

a 8.7. ábrán illusztráljuk, ahol minden konvex sarokból indul egy középvonal ág, amely csatlakozik a többi ághoz, hiszen a folytonos MA összefüggő.

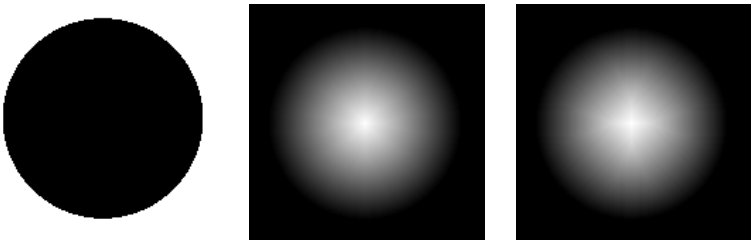
Ha az alakzatban megjelenik egy kis szögletes torzulás, az MA struktúrája jelentősen megváltozik úgy, hogy a változás függ a torzulás irányától. Emiatt a transzformációt regularizálni kell, aminek a legegyszerűbb módja a kontúrsimítás, hatékonysága azonban korlátozott.

8.3. Távolság-transzformáció

A távolság-transzformáció (*distance transform*, DT) egy fontos eszköz, amelyet több célra is fel lehet használni. Segítségével kinyerhető a középtengely is, ennek az algoritmusát rövidesen megismerjük.

Folytonos esetben a DT bemenete egy vagy több alakzat, az eredmény pedig az euklideszi távolság minden belső pont és a legközelebbi kontúrpont között; külső pontokra és kontúrpontokra az eredmény nulla.

Diszkrét esetben a bemenet egy vagy több alakzatot tartalmazó bináris kép, a kimenet távolság minden objektumpixel és a legközelebbi háttérpixel között, ezen belül a háttérpixelekre 0, határpixelekre 1 vagy $\sqrt{2}$. Eredményképekben a távolságot intenzitással fogjuk kódolni úgy, hogy világosabb pixel nagyobb távolságot jelent.



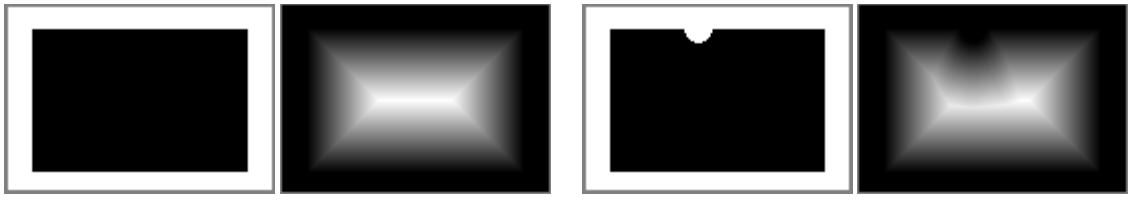
lemez finom közelítés durva közelítés

8.8. ábra. Távolság-transzformáció függősége a távolság közelítésétől.

Digitális képekre az euklideszi távolság diszkrét közelítését fogjuk használni. Minél pontosabb a közelítés, annál pontosabb az eredmény. A 8.8. ábrán az euklideszi távolság

finomabb approximációja nagyobb maszkot használ és pontosabb eredményt nyújt, de a számításigény is nagyobb.

A MAT-hoz hasonlóan a DT is torzítás-érzékeny, amint a 8.9. ábra illusztrálja. A távolság-transzformáció továbbterjeszti a kisebb méretű hibát, így az eredeti és torzított DT lényegesen különbözik.

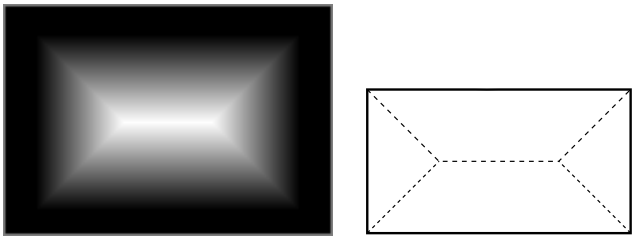


eredeti téglalap eredeti DT hibás téglalap torzított DT

8.9. ábra. Távolság-transzformáció torzítás-érzékenysége.

8.3.1. Távolság-transzformáció és középvonal

Ez eddigiek alapján világos, hogy a távolság-transzformáció kapcsolatban áll a középvonallal. A kapcsolatot a 8.10. ábra szemlélteti, ahol láthatjuk, hogy a DT gerincei a MAT ágai. Ez azért van, mert egy DT-gerinc a lokálisan legbelsőbb pontok sorozata, olyan pontoké, amelyek egyenlő távolságra vannak két vagy több legközelebbi háttérponttól. Ha végrehajtjuk a távolság-transzformációt, az ilyen pontokból össze tudjuk állítani az alakzat középvonalát.



téglalap, DT téglalap, MAT

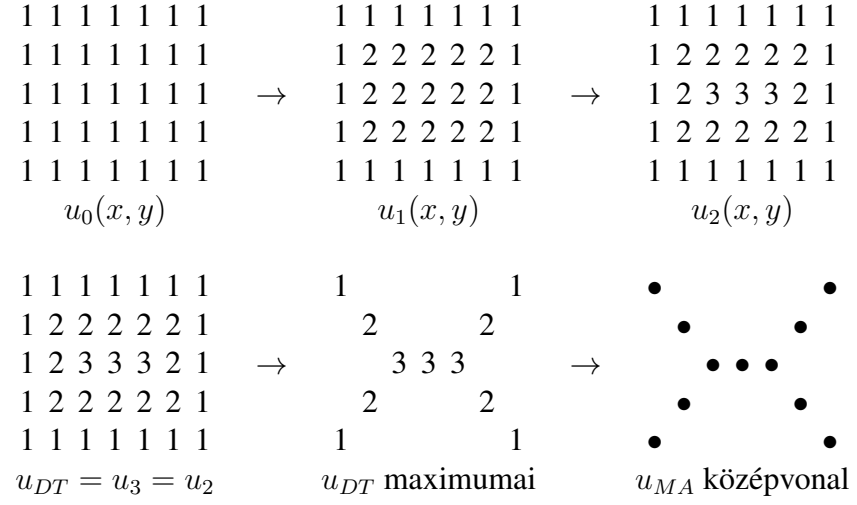
8.10. ábra. A DT és a MAT közötti kapcsolat.

Az alábbiakban egy *egyszerűsített* iteratív algoritmust adunk, mely kap egy $u(x, y)$ bináris képet és kiszámítja az $u_{DT}(x, y)$ távolság-transzformációt, azaz az (x, y) objektumpixel és a legközelebbi háttérpixel közti távolságot. Ezzel véget ér az algoritmus első DT-része. A második részben az eljárás a DT-ből kiindulva meghatározza az $u_{MA}(x, y)$ középvonalat. Az algoritmus 4-összefüggést használ az objektum-pixelekre. Az egyszerűség kedvéért nem az euklideszi, hanem a $\Delta(x, y; i, j)$ *city-block* távolságot alkalmazza. A gyakorlatban finomabb közelítést használnak, de az elv ugyanaz.

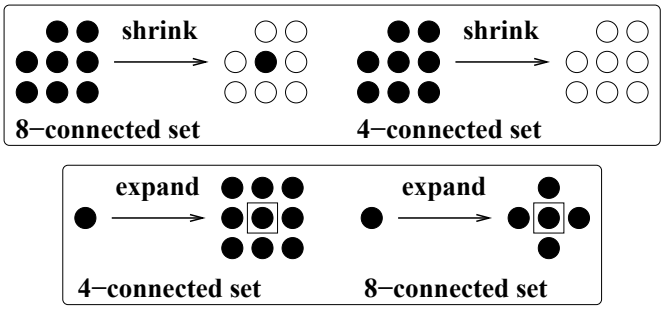
7. Algoritmus: Egyszerű diszkrét DT és MAT

1. DT inicializálása: $u_0(x,y) = u(x,y)$
2. DT meghatározása
   - Minden $k = 1, 2, \ldots$-ra rekurzívan kiszámítjuk a DT-t:
   $$u_k(x,y) = u_0(x,y) + \min_{i,j} \{u_{k-1}(i,j); (i,j) : \Delta(x,y;i,j) \leq 1)\} .$$
   - Megállunk, ha nincs több változás:
   $$u_k(x,y) = u_{k-1}(x,y) \quad \text{minden } (x,y)\text{-ra.}$$
   - Az eredmény: $u_{DT}(x,y) = u_k(x,y)$.
3. $u_{MA}(x,y)$ középvonal meghatározása DT-ből:
   $$\{(x,y) : u_{DT}(x,y) \geq u_{DT}(i,j); \Delta(x,y;i,j) \leq 1\}$$

Az eljárást a 8.11. ábra illusztrálja, ahol egy numerikus példát mutatunk. A DT megáll, amikor k eléri az alakzat maximális méretének a felét. Az algoritmus "megfordításával" az alakzat visszaállítható u_{DT} maximumaiból.



8.11. ábra. Numerikus példa DT-MAT-ra.



8.12. ábra. Numerikus példák hámozásra és kiterjesztésre.

8.4. Vékonyítás és váz

Vékonyítás egy speciális hámozási művelet, amelynek az eredményét váznak nevezik. A váz hasonlít a középvonalra, de nem azonos vele. Az alábbiakban először definiáljuk az egyszerű hámozás és kiterjesztés műveleteket, utána megadjuk a vékonyítás definícióját.

Az S halmaz egyszeri **hámozása** az S határának törlése, jelölése $S^{(-1)}$. Az n**-szeri hámozást**, azaz a hámozás n-szeri megismétlését $S^{(-n)}$-ként jelölik.

$S^{(1)}$ az S halmaz **kiterjesztése**, amit úgy kapunk, hogy felcseréljük S és $\underline{S}$ összefüggőségét és hámozzuk az $\underline{S}$ halmazt. $S^{(n)}$ az S halmaz n**-szeri kiterjesztése**, azaz a kiterjesztés n-szeri megismétlése.

Angolul a két fogalom neve az *(n-step) shrinking* és az *(n-step) expanding*. A 8.12. ábra numerikus példákat ad egyszeri hámozásra és kiterjesztésre. Amikor az első halmazt 4-összefüggőnek tekintjük, az összes pontja határpixel, amelyet a hámozás el is tüntet, törölve ezzel az egész objektumot.

Mint látjuk, a hámozás nem őrzi meg az összefüggőséget és eltüntethet objektumokat. Vékonyításhoz ennél kifinomultabb műveletre lesz szükségünk, hogy egy, az eredeti alakzat struktúráját tükröző, vázszerű struktúrát kaphassunk.

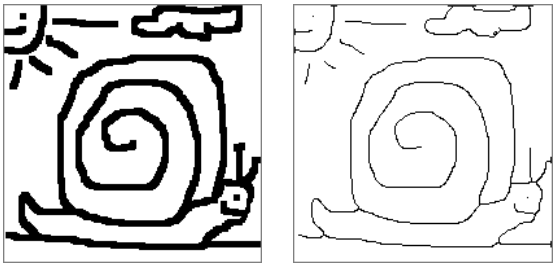
A **vékonyítás** (*thinning*) a topológiát megőrző iteratív hámozás, amelynek **váz** (*skeleton*) az eredménye. Definíció szerint az S halmazt iteratív hámozással úgy vékonyítjuk vázzá, hogy a következő két feltételnek teszünk eleget:

1. megőrizzük az S összefüggőségét;

2. nem törlünk végpontokat.

A 8.13. ábra egy vékonyítási példát mutat. Definíciójából kifolyólag a váz *összefüggő halmaz*, amely a középvonalhoz hasonlóan több ágból áll. A váz követi egy, hosszúkás részekből álló objektum alakját. Más esetekben egy objektum és a váza közti kapcsolat nem ennyire egyszerű.

A váz általában *hasonlít a középvonalra*, emiatt gyakran nem tesznek különbséget az MA és a váz között, és a középvonalat is "váznak" nevezik. *Vázszerű reprezentációról* beszélnek,



8.13. ábra. Vékonyítás példája. Baloldalon egy bináris kép, jobboldalon a váza.

amely MAT-tal és vékonyítással is nyerhető. Nem szabad azonban elfelejteni, hogy vannak esetek, amikor a váz egészen más is, mint a középvonal, például, egy téglalap esetén.

p_3	p_2	p_9
p_4	$\mathbf{p_1}$	p_8
p_5	p_6	p_7

8.14. ábra. p_1 pont és környezete.

Az alábbiakban megadunk egy olyan vékonyító algoritmust 8-összefüggő vázra, amelyet magunk is használunk. Tekintsünk egy p_1 pontot és 3×3-as környezetét (8.14. ábra). Legyen a háttér értéke 0, az objektumé 1. Legyen továbbá $TR(p_1)$ a $0 \to 1$ átmenetek száma a $\{p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_2\}$ sorozatban. $TR(p_1)$ meghatározásához körbejárjuk a p_1 pontot és megszámoljuk a $0 \to 1$ átmeneteket (a $1 \to 0$ átmenetek nem számítanak). Végül legyen $NZ(p_1)$ a p_1 pont 1-es szomszédainak száma.

8. Algoritmus: Vékonyítás

1. Az $u_1(x, y)$ és $u_2(x, y)$ segédképeket inicializáljuk az $u_0(x, y)$ bemeneti képpel.

2. Letapogatjuk u_1-et, pontokat törlünk u_2-ben. Akkor törlünk egy $p_1 = 1$ pontot, ha **egyszerre teljesülnek** a következő feltételek:

$$2 \leq NZ(p_1) \leq 6 \tag{8.1}$$

$$TR(p_1) = 1 \tag{8.2}$$

$$p_2 \cdot p_4 \cdot p_8 = 0 \ \textbf{vagy} \ TR(p_2) \neq 1 \tag{8.3}$$

$$p_2 \cdot p_4 \cdot p_6 = 0 \ \textbf{vagy} \ TR(p_4) \neq 1 \tag{8.4}$$

3. Ha nem volt törlés, megállunk. Egyébként, másolunk $u_1 = u_2$ és a 2.lépésre ugrunk.

1	1	0
1	p_1	1
0	0	0

0	0	0
1	p_1	0
0	0	0

1	0	1
0	p_1	0
1	1	1

8.15. ábra. Három példa, amikor $p_1 = 1$ nem törölhető.

A 8.15. ábra példákat mutat, amikor $p_1 = 1$ nem törölhető. A ponttörlés a bal szélső esetben ketté szakíthat egy régiót, a másodikban lerövidíthet egy vonalvéget. A 3. esetben $2 \leq NZ(p_1) \leq 6$, és (8.3, 8.4) is teljesül, de p_1 mégsem törölhető, mert $TR(p_1) = 3 \neq 1$.

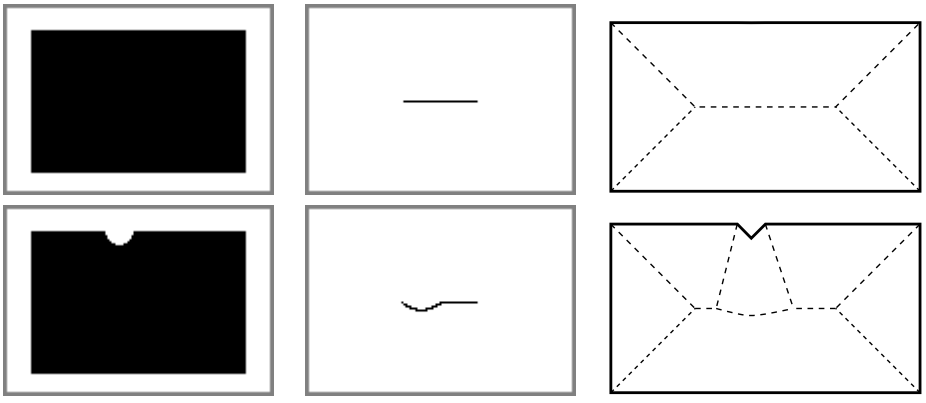
A vékonyító algoritmushoz több megjegyzést is szeretnénk tenni. Egy pixel körbejárása és az átmenetek megszámolása egyébként is hasznos művelet. Például vékonyított képek elemzésére alkalmazzák, hogy megtalálják a vonalak végpontjait vagy kereszteződéseit.

Az algoritmus igazából nem 3×3-as, hanem $\mathbf{4} \times \mathbf{4}$-es ablakot használ, mert körbejárja p_2-t és p_4-t, amikor $TR(p_2)$-t illetve $TR(p_4)$-t számolja. Ilyenkor szándékos *aszimmetriát* visz be azzal, hogy $TR(p_6)$-t és $TR(p_8)$-t nem számítja ki (lásd (8.3, 8.4)). Az aszimmetria révén *egypixelnyi vastagságú* vázat kapunk páros vastagságú vonalak esetén is, ami félpixelnyi eltolást jelent az "igazi" vázhoz képest.

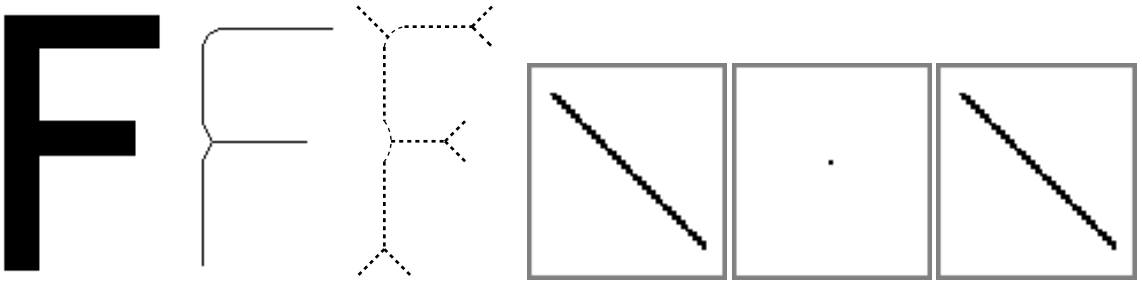
Amint rövidesen látni fogjuk, ennek elvben lehet mellékhatása, amikor bizonyos vonalak teljesen eltűnnek. A gyakorlatban azonban ez nagyon ritkán fordul elő. Létezik egy olyan, bonyolultabb változata az eljárásnak, amely mentes a mellékhatástól.

8.5. Vázszerű reprezentációk összefoglalója

A 8.16. ábrán összehasonlítjuk a vékonyítást és MAT-ot téglalap esetén. A két eredmény igen különböző. Az alakhiba mind a két esetben továbbterjed, de a MAT struktúra-változása drasztikusabb. A vékonyítás láthatóan robusztusabb.



8.16. ábra. Vékonyítás és MAT összehasonlítása téglalapra.



8.17. ábra. Vékonyítás és MAT további összehasonlítása.

A 8.17. ábra további összehasonlító példákat mutat. Az F betűre a két eredmény hasonló. de a MAT kisebb "parazita" sarokágakat produkál. Egy tökéletes, egypixelnyi vastagságú diagonális vonalat a vékonyító algoritmusunk egy szem pixelre redukál. Amint az előbb említettük, ez ritka mellékhatás, hiszen a valós képekben az ehhez hasonló, ideális vonalak ritkák.

Összefoglalva a fejezetet, a vázszerű reprezentáció *előnyei* a következők:

- Adattömörítést eredményez, amely vesztességmentes, ha a távolságokat is megtartjuk.
- Tükrözi az alakzat struktúráját.
- Elforgatás-invariáns, bár diszkrét esetben csak közelítően.

A *hátrányok* közé az alábbiak tartoznak:

- Főleg hosszúkás részekből álló alakzatok esetén hasznos.
- Zaj- és torzítás-érzékeny lehet.

A vékonyítás valamivel robusztusabb, ezért gyakrabban alkalmazzák.

9. fejezet

Morfológiai képfeldolgozás

A morfológiai képfeldolgozás egy önálló, sajátos szakterület, amelynek csak töredékét tudjuk itt bemutatni. Célunk nem a részletes ismertetés, hanem az elvek és fő műveletek, algoritmusok rövid, de tárgyszerű és a gyakorlatban is felhasználható leírása. Morfológiai módszerekkel is kaphatunk vázszerű reprezentációkat, ezért a fejezet végén összehasonlítjuk őket az elöző fejezetben bevezetett algoritmusokkal.

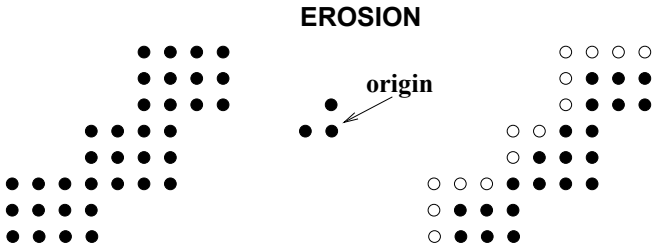
A félreértések elkerülése érdekében már az elején érdemes hangsúlyozni, hogy a morfológiai képfeldolgozási elméletet és módszereket arra fejleszették ki, hogy a képen levő sok, kisebb méretű alakzat statisztikai leírását kapjuk. Eredetileg fémmetszeteket ábrázoló mikroszkópképek strukturális elemzéséről volt szó, azóta sok minden másra is alkalmazták, de nem az egyéni, nagy alakzatok precíz leírására. (Persze, ez nem jelenti azt, hogy a módszerekkel nem lehet, például nagyobb alakzatokra zajszűrést vagy simítást végezni.) Ezt szem előtt kell tartanunk, amikor megtapasztaljuk, hogy egy-egy alakzatra kapott eredmény függhet annak elforgatásától vagy a műveletek sorrendjétól. Ezzel együtt a morfológiai képfeldolgozás egy igen hasznos eszköztár, amelyet meg kell ismerni.

9.1. Morfológiai képfeldolgozás alapjai

Hagyományos szóhasználatban a morfológia az állatok, növények, vagy szavak struktúra- és alakelemzése. Ezen belül, fontos mozzanat a **strukturáló elemekkel** (*structuring elements*) történő leírás.

A **morfológiai képfeldolgozás** ezzel szemben a képek olyan jellegű struktúra- és alakelemzése, amikor a képet összehasonlítjuk egy csúsztatott strukturáló elemmel. A strukturáló elemet "megütköztetjük" a képi objektumokkal, ez által kifejezőbb alakba hozva őket. A folyamat mintaillesztésre hasonlít.

A legtöbb morfológfiai művelet két műveleten keresztül fejezhető ki, ezek az erózió (*erosion*) és a dilatáció (*dilation*). A definíciókban B a **strukturáló elem** (SE), amelynek egy c origója van. A strukturáló elem egy, általunk definiált, adott műveletre szánt diszkrét ponthalmaz, az origó pedig nem feltétlenül a B egyik pontja. A bemeneti bináris kép jelölése



9.1. ábra. Erózió példája.

X. B_x a B strukturáló elem olyan eltolása, hogy a B origója az x pontban van. A művelet eredményét a kimeneti kép x pontjába írjuk be.

9.1.1. Erózió és dilatáció

Az X kép **eróziója** a B strukturáló elemmel az összes olyan x pont halmaza, amikor B_x benne van X-ben:

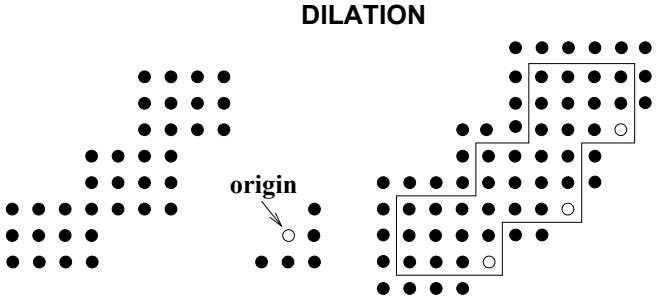
$$X \ominus B = \{x : B_x \subseteq X\}. \tag{9.1}$$

A művelet eredménye 1, ha a strukturáló elem *minden pontja* egybeesik X valamely objektum-pixelével. Egyébként nulla az eredmény. A 9.1. ábrán egy példát láthatunk erózióra. A strukturáló elemet az illusztráció közepén mutatjuk. Egy üres kör egy *törölt pont*, ami ebben az esetben minden olyan pont, ahol $B_x \nsubseteq X$.

Az X kép **dilatációja** a B strukturáló elemmel az összes olyan x pont halmaza, amikor B_x és X metszete nem üres:

$$X \oplus B = \{x : B_x \cap X \neq \emptyset\} \tag{9.2}$$

A művelet eredménye 1, ha a strukturáló elem *legalább egy pontja* egybeesik X valamely objektum-pixelével, azaz B_x "ütközik" X-szel (B_x *hits* X). A 9.2. ábra a dilatáció egy példáját mutatja. Itt egy üres kör egy törölt pont és az origó. A kontúr az eredeti alakzat helyét mutatja. Megfigyelhető, hogy ponttörlés ellenére az alakzat nagyobb lett, de az origó eltolása miatt elmozdult a bal felső irányba.



9.2. ábra. Dilatáció példája.

Az alábbiakban bizonyitás nélkül felsoroljuk a két alapművelet fő tulajdonságait. Először a **műveletek sorrendjére** vonatkozó tételek következnek.

1. A *dilatáció kommutatív a képpel*:

$$X \oplus B = B \oplus X$$

   Szűréshez hasonlóan, $B \oplus X$-ben B-t képként, X-t strukturáló elemként kezeljük.

2. Az *erózió nem kommutatív a képpel*:

$$X \ominus B \neq B \ominus X$$

   Például, egy nagy alakzatra és kis strukturáló elemre $B \ominus X$ üres, $X \ominus B$ nem.

3. Viszont az *eróziók sorrendje tetszőleges*:

$$(X \ominus A) \ominus B = (X \ominus B) \ominus A$$

4. A *dilatációk sorrendje is tetszőleges*:

$$(X \oplus A) \oplus B = (X \oplus B) \oplus A$$

5. Erózió és dilatáció nem cserélhető fel és *nem inverze egymásnak*. Például, ha erózió után üres halmazt kapunk, nem tudjuk "visszadilatálni".

A **disztributivitással** kapcsolatos tulajdonságok a következők:

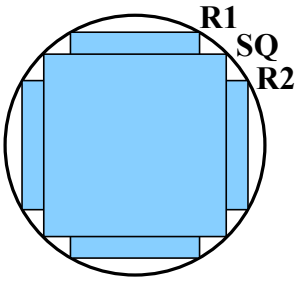
1. A *strukturáló elem* szerinti disztributivitás:

$$X \ominus (B \cup B') = (X \ominus B) \cap (X \ominus B')$$
$$X \oplus (B \cup B') = (X \oplus B) \cup (X \oplus B')$$

2. A *kép* szerinti disztributivitás:

$$(X \cap Y) \ominus B = (X \ominus B) \cap (Y \ominus B)$$
$$(X \cup Y) \oplus B = (X \oplus B) \cup (Y \oplus B)$$

Az SE szerinti disztributivitás egyik lehetséges felhaszálását a 9.3. ábrán szemlélteti. Egy bonyolult strukturáló elem több egyszerű elem *uniójaként* reprezentálható vagy közelíthető abban a reményben, hogy a számításigény csökken. Például egy lemez közelíthető egy SQ négyzettel és két R1, R2 téglalappal. Ennek az az előnye, hogy a lemezzel végrehajtott műveletek gyorsabbak lesznek, mert a téglalappal való műveletek hatékonyan, futószűrő-szerűen implementálhatók.



9.3. ábra. A strukturáló elem szerinti disztributivitás felhasználása.

További tulajdonságok elsősorban az **asszociativitásra** és a vele kapcsolatos iterációra vonatkoznak.

1. A műveletek *iterációja*:

$$(X \ominus B) \ominus B' = X \ominus (B \oplus B')$$
$$(X \oplus B) \oplus B' = X \oplus (B \oplus B')$$

   A dilatáció tehát associatív, az erózió nem.

2. Ha egy SE egy másik SE részhalmaza, $B \subset B'$, akkor

$$X \ominus B' \subset X \ominus B$$
$$X \oplus B \subset X \oplus B'$$

3. *Dualitás* komplemensre:

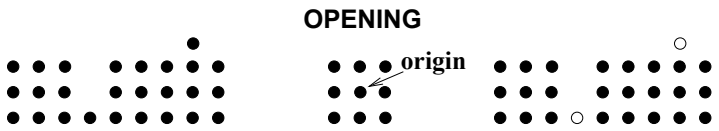
$$\underline{X} \oplus B^{\sim} = \underline{X \ominus B},$$

   ahol $\underline{X}$ az X komplemense, $B^{\sim}$ a B-nek az origóra való tükrözése.

Az iteráció felhaszálható műveletek felgyorsítására, ha egy nagy SE több kisebb elem dilatációjára "bontható" (9.4. ábra). Ez nagyon hasonlít arra, amit szűrés esetén megtanultunk.

$$\begin{array}{ccccc} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{array} = \begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array} \oplus \begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array}$$

9.4. ábra. Egy 5×5-ös művelet azonos két 3×3-as művelettel.



9.5. ábra. Nyitás példája.

9.1.2. Nyitás és zárás

Az X halmaz **nyitása** (*opening*) B strukturáló elemmel:

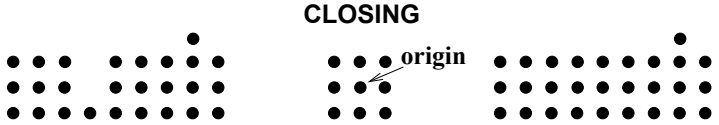
$$X_B = (X \ominus B) \oplus B \tag{9.3}$$

A nyitás hatását a 9.5. ábra illusztrálja. Elsimít kontúrokat és eltávolítja a kis foltokat és dudorokat. A művelet felhasználható az objektumméret-eloszlás elemzésére, ehhez növekvő méretű B-vel történő iteratív nyitást kell alkalmazni.

Az X halmaz **zárása** (*closing*) B strukturáló elemmel:

$$X^B = (X \oplus B) \ominus B \tag{9.4}$$

A zárás hatását a 9.6. ábra szemlélteti. Kitölti a keskeny csatornákat és a kis lyukakat. A művelet felhasználható az objektumok közti távolságeloszlás elemzésére, ehhez növekvő méretű B-vel történő iteratív zárást kell alkalmazni.



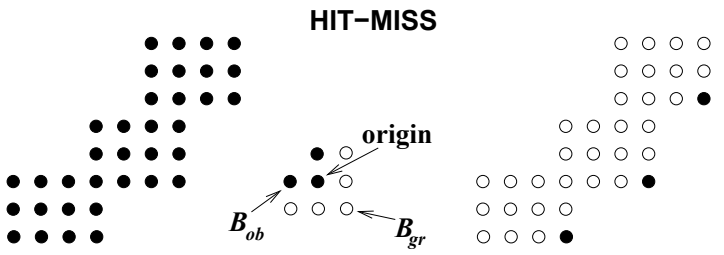
9.6. ábra. Zárás példája.

Nyitás és zárás **idempotens** műveletek:

$$(X_B)_B = X_B$$
$$\left(X^B\right)^B = X^B$$

Ez azt jelenti, hogy a műveletet *ugyanazzal* a strukturáló elemmel nem érdemes egymás után többször alkalmazni. Ezen kívül a két művelet **duális** a komplemensre:

$$\underline{(X_B)} = (\underline{X})^B$$
$$\underline{\left(X^B\right)} = (\underline{X})_B$$



9.7. ábra. Sarokkeresés hit-miss segítségével.

9.1.3. Hit-miss

A hit-miss művelet lényegében bináris mintaillesztés, azaz adott objektum- és háttérpixel konfigurációk keresése. A B strukturáló elemnek mind az objektum-, mind a háttérpixeleit meg kell adni:

- $B_{ob} \subset B$ a strukturáló elem objektumrésze,
- $B_{gr} \subset B$ a strukturáló elem háttérrésze.

A hit-miss operátor eredménye objektumpixel, ha

- B_{ob} a kép objektumpixeleihez illeszkedik **és**
- B_{gr} a kép háttérpixeleihez illeszkedik.

A művelet definíciója:

$$X \otimes B \doteq \{x : B_{ob} \in X \text{ and } B_{gr} \in \underline{X}\} = (X \ominus B_{ob}) \cap (\underline{X} \ominus B_{gr}), \tag{9.5}$$

ahol B_{gr}-t *objektumpixelekből* álló halmazként kezeljük. Ezeknek illeszkedniuk kell az $\underline{X}$ komplemenshez.

A hit-miss bináris mintaillesztés bináris kimenettel. Emlékszünk, hogy a szürke képek illesztésénél az eredmény korrelációs érték volt, ami egy valós szám. Léteznek a fenti képlettől eltérő, de vele *ekvivalens* definíciók. A műveletre néha a $\odot$ jelölést is használják.

A 9.7. ábrán bemutatjuk, hogyan lehet sarkokat detektálni a hit-miss segítségével. A strukturáló elem objektum- és háttérpixeleinek is illeszkedniük kell. Csak bizonyos orientációjú és alakú sarkokat találunk, az elforgatott sarkok detektálásához az elemet meg kell forgatni.

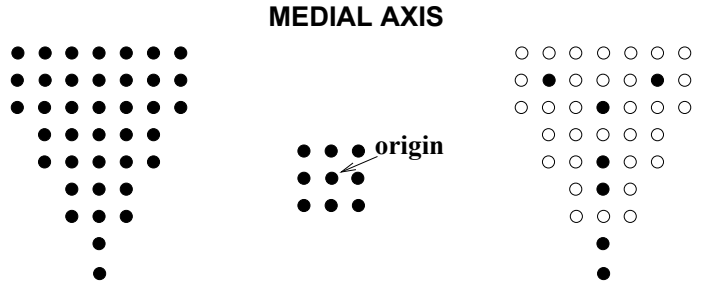
9.2. További morfológiai műveletek

9.2.1. Morfológiai középvonal

Az X halmaz $S(X)$ **középvonala**

$$S(X) = \bigcup_{n=0}^{n_{max}} (X \ominus nG) \setminus (X \ominus nG)_G \doteq \bigcup_{n=0}^{n_{max}} s_n(X), \tag{9.6}$$

ahol a G strukturáló elem a 3×3-as négyzet ("lemez"), n_{max} a maximális méret, ami után X üresre erodálódik, $(X \ominus nG)$ az X n-szeres eróziója, $(X \ominus nG)_G$ az $(X \ominus nG)$ nyitása G-vel, $X \setminus Y$ pedig a két halmaz különbsége. A középvonalat az $s_n(X)$ részek uniójaként kapjuk. Egy példa a 9.8. ábrán látható. A 3×3-as G négyzet egy kis lemez közelítése, amit a morfólogiai feldolgozásban gyakran használnak. Mint láttuk, nagyobb "lemezeket" (négyzeteket) iterációval kaphatunk.



9.8. ábra. Példa morfológiai középvonalra.

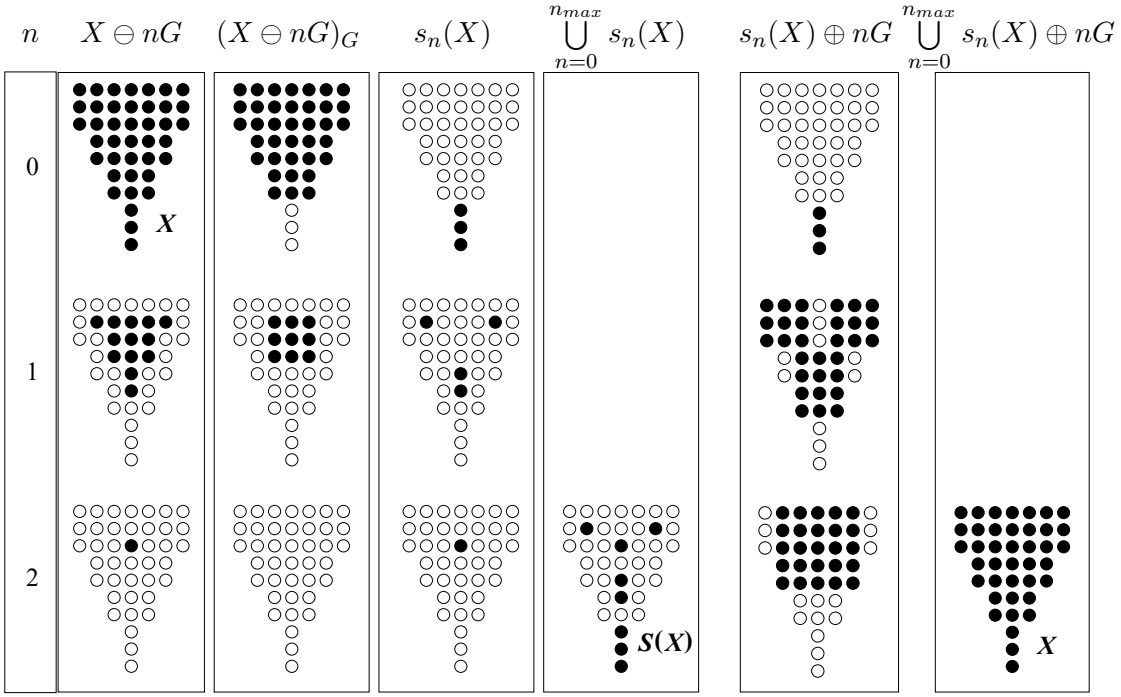
A morfológiai MA több részből tevődik össze, és a folytonos középvonallal ellentétben ezek nem mindig függnek össze. Az X halmazt az alábbi módon állíthatjuk vissza a morfológiai MA-ból:

$$X = \bigcup_{n=0}^{n_{max}} s_n(X) \oplus nG, \tag{9.7}$$

ahol $(X \oplus nG)$ az X n-szeres zárása.

A folyamat ugyanaz, mint amit az előző fejezetben a 8.6. ábrával illusztráltunk. A középvonal azon lemezek középpontjaiból áll, amelyek benne vannak az X-ben és két vagy több pontban érintik X határát. Az MA meghatározáshoz iteratív eróziót alkalmatunk. A visszaállításban az eredeti X halmaz azon lemezek uniója, amelyeknek a középpontjai a MA pontja és a sugarai a határtól való távolságok. A visszaállításhoz iteratív dilatációt alkalmatunk.

A fenti példát a 9.9. ábra részletezi. Az első oszlopban iteratívan erodáljuk a halmazt, a másodikban a megfelelő iterációkra alkalmazzuk a nyitást. Levonva az első oszlopból a másodikat, megkapjuk azokat a részeket, amelyeknek az uniója képezi a morfológfiai középvonalat.



9.9. ábra. Morfológiai középvonal előállítása.

Az ábra jobboldalán a visszaállítás lépései láthatók. Csak akkor tudjuk visszaállítani az eredeti halmazt, ha ismerjük az $s_n(X)$ részeket, tehát tudjuk, hogy a rész hányadik iterációban született. A részeket iteratívan visszadilatáljuk és összeadjuk a részeredményeket.

9.2.2. Morfológiai határkiemelés, vékonyitás és ágmetszés

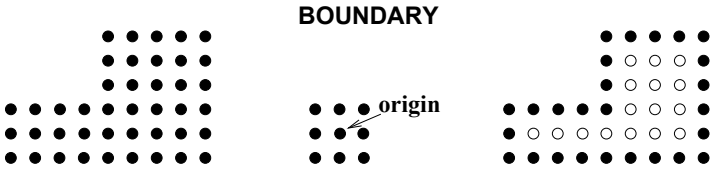
Az X halmaz **határa** (9.10. ábra)

$$\partial X = X \setminus (X \ominus G), \qquad G = \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \tag{9.8}$$

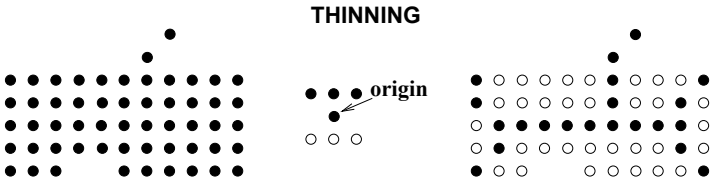
Itt az $(X \ominus G)$ erózióval belső pontokat kapunk, levonjuk őket az eredeti halmazból és megkapjuk az alakzat határát.

A **morfológiai vékonyitás** a hit-miss segítségével vázzá redukál egy alakzatot. A váz az ágak halmaza, amelyeket iteratívan határozunk meg. Minden iterációnál a strukturáló elem nyolc elforgatását használjuk. Egy példa a 9.11. ábrán látható.

A 9.12. ábra összehasonlítja a hagyományos és a morfológiai vékonyitást. Az ábra demonstrálja, hogy a morfológiai eredmény függhet az elforgatások sorrendjétől, vagyis érzé-

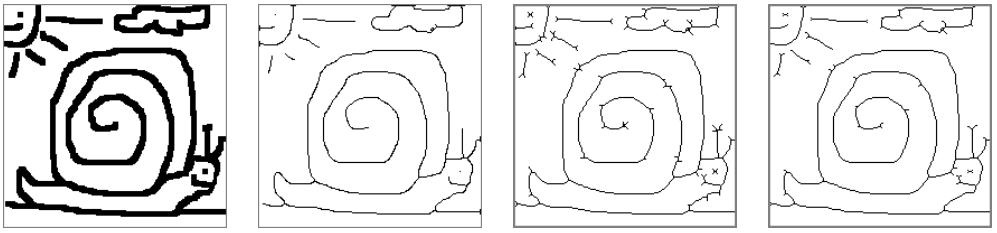


9.10. ábra. Példa határkiemelésre.



9.11. ábra. Morfológiai vékonyitás.

keny az alakzat orientációjára. Ezen kívül, fölösleges kis ágakat tartalmaz. A hagyományos vékonyitás eredménye jobb.



kép hagyományos 1. morfológiai 2. morfológiai

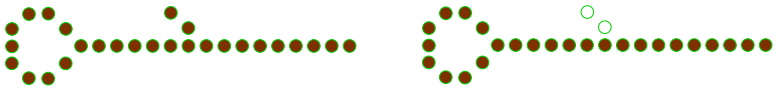
9.12. ábra. Különböző vékonyitások összehasonlítása.

A morfológfiai **ágmetszés** (*pruning*) eltávolít kisebb "zajos" ágakat. Ezek tipikusan a MAT vagy vékonyitás következtében jelennek meg, ezért szokták e két művelet után alkalmazni. Az eredményt a 9.13. ábra szemlélteti.

Az ágmetszés több lépésből áll, ezeket a 9.14. ábrával illusztráljuk. Először forgó strukturáló elemmel rekurzívan rövidítjük az ágakat. A rekurziós lépések száma meghatározza a törlendő ágak maximális hosszát; példánkban két rekurziós lépés van. Ezek után megtaláljuk a megmaradt ágak végpontjait. Egy másik strukturáló elemmel rekurzívan "visszanövesztjük" a megmaradt ágakat. Ehhez **geodetikus dilatációt** (*geodesic dilation*) alkalmazunk, ami az eredeti halmazra korlátozott dilatáció, azaz a hagyományos dilatáció és az eredeti halmaz metszete.

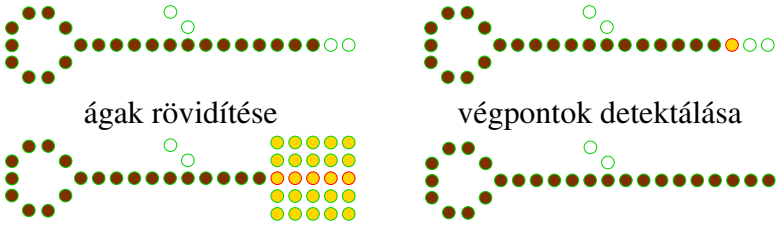
9.2.3. A morfológiai feldolgozás összefoglalója

Morfológiai feldolgozást szürke képekre is ki lehet terjeszteni. Leginkább az alábbi feladatokra alkalmazzák:



eredeti alakzat metszett alakzat

9.13. ábra. Kis ágak morfológiai metszése.



ágak rövidítése végpontok detektálása

végpontok visszanövesztése metszett alakzat

9.14. ábra. Morfológiai ágmetszés példája.

- Képek előfeldolgozása, például zajszűrés, alakzat-egyszerűsítés.
- Alakzatok struktúraelemzése, tipikusan a középvonal és a váz kiemelése.
- Objektumok elkülönítése háttértől.
- Objektumok számszerű leírása, például területek, kerületek, vetületek, lyukak számának meghatározása.

Morfológiai feldolgozás *hatékony* sok kis alakzatot tartalmazó képek esetén, különösen az ilyen képek statisztikai leírására, méretek és más alakparaméterek eloszlásának gyors kiértékelésére. Nem alkalmazható nagy, összetett alakzatok precíz leírására.

A *hátrányok* között megemlíthetjük az elforgatás-érzékenységet, amely miatt sok morfológiai eredmény csak közelítő jellegű. Ezek közé tartoznak például a morfológiai középvonal és a váz.

10. fejezet

Kétdimenziós alakelemzés

A jegyzet utolsó fejezetében olyan módszerekkel foglalkozunk, amelyek képesek kétdimenziós alakzatok leírására, ezzel lehetővé téve az objektumok strukturális elemzését, megkülönböztetését és osztályozását. Nem csak a lapos objektumokra gondolunk, hanem a háromdimenziós tárgyak vetületeire is.

Az alakelemzés (*shape analysis*) feladatai alakzatleírás, beleértve a számszerű leírásokat és alakzat-sajátságok (pl. kontúrsarkok) kiemelését, valamint alakzat-szegmentálás és -illesztés, továbbá pozíció- és orientáció-meghatározás.

Egy 2D-s alakzatot kétféleképpen lehet reprezentálni, az alakzat kontúrjával és az alakzat teljes területével, azaz a belső résszel és a kontúrral. Ennek megfelelően két fő módszercsoportot szoktak említeni. A **terület alapú** alakelemzési módszerekre az alábbiak a jellemzők:

- az alakzat teljes területén operálnak;
- a pontokat 2D-ben, a képsíkon rendezik;
- támogatják a 2D-s lokális műveleteket;
- számításigényük az alakzat területétől függ.

A **kontúr alapú** alakelemzési módszerekre ezzel szemben az a jellemző, hogy

- az alakzat kontúrján operálnak;
- a pontokat a kontúr mentén rendezik;
- nem támogatják a 2D-s lokális műveleteket;
- számításigényük az alakzat kerületétől függ.

Ennek megfelelően többféle *adatstruktúra* létezik, amely vagy az egyik, vagy a másik módszercsoporhoz illeszkedik jobban. A három legelterjedtebb adatstruktúra maga a képmátrix, a futam-hossz kód (*run-length code*) és a lánc-kód (*chain code*). Az első kettő a terület

alapú módszereket támogatja, a harmadik a kontúr alapú eljárásokat. Mindegyik adatstruktúra esetén a kiinduló lépés az *összefüggőség-elemzés*, vagyis az összefüggő objektum-régiók (komponensek) meghatározása.

10.1. Bináris képek adatstruktúrái

10.1.1. Futam-hossz kód és komponens-analízis

A 8. fejezetben egy p, q pixelpárra definiáltuk a $p \leftrightarrow q$ összefüggőségi relációt, amely

- reflexív: $p \leftrightarrow p$;
- szimmetrikus: ha $p \leftrightarrow q$, akkor $q \leftrightarrow p$;
- tranzitív: ha $p \leftrightarrow q$ és $q \leftrightarrow r$, akkor $p \leftrightarrow r$,

tehát ekvivalencia-reláció. Egy bináris képet a reláció **összefüggő komponensekre**, azaz maximális összefüggő pixelhalmazokra bont.

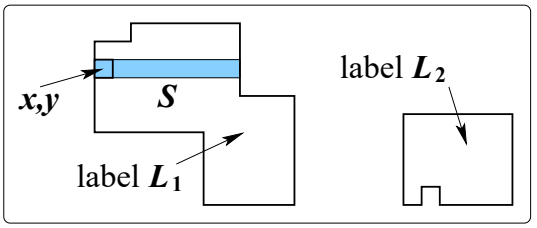
A komponenseket közvetlenül a képen lehet keresni úgy, hogy minden képelemhez hozzárendelünk egy komponenscímkét. A *rekurzív algoritmus* egyszerű és rövid, vázlata a következő:

- A képmátrixban megkeressük a következő be nem járt pontot.
- Rekurzív függvényhívással bejárjuk a szomszédokat, ezzel megkapjuk a kiinduló pont összefüggő komponensét.
- A bejárt pixelekhez hozzárendeljuk az aktuális címkét.
- Növeljük a címkét és megismételjük az eljárást.

Sajnos nagy komponensek esetén az algoritmus a mély rekurzió miatt nem hatékony, mert lassú és el is szállhat, ha túlcsordul a verem (ez utóbbi ügyes programozással elkerülhető). Csak akkor érdemes használni, amikor a komponensek viszonylag kicsik.

Ennél bonyolultabb, de hatékonyabb a **futam-hossz kód** (*run-length code*, RLC) alapú komponens-analízis. A kódolás elvét a 10.1. ábra illusztrálja. Egy bináris képet *vízszintes futamokkal* reprezentálunk. A futam az objektum-pixelek maximális összefűggő sorozata. Egy R futamot az alábbiakkal kódolunk:

- x, y, az *első pixel koordinátái*
- S, a *futam hossza*, pixelben
- L, a *futam címkéje*: R az L-ik komponenshez tartózik.



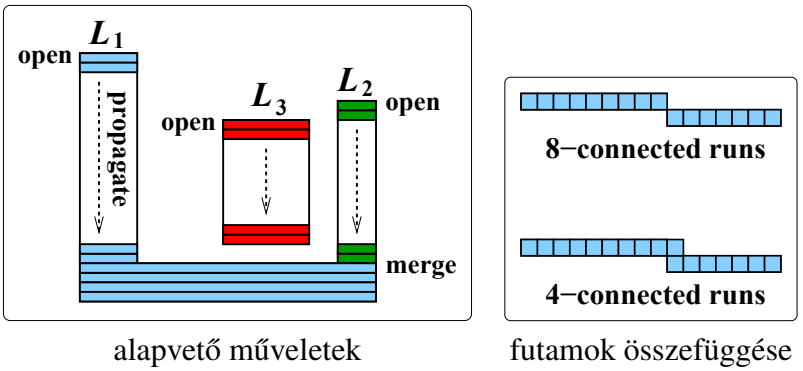
10.1. ábra. Futam-hossz kód.

Címke nélküli futam-hossz kód a bináris kép letapogatásával nyerhető, egyszerűen és gyorsan. Ez nagy, információveszteség nélküli tömörítéssel jár, kivéve az egészen különleges eseteket, amikor a képen rengeteg apró komponens van. Egy kép simán visszaállítható az RLC-jéből, csak be kell írni a képmátrixba a futamokat.

A komponens sorszámával címkézett futam-hossz kód komponens-analízissel kapható a címke nélküli RLC alapján. Egy tipikus algoritmus *címke-ekvivalencia táblát* használ és két menetben működik. Az **első menet** három alapvető műveletet tartalmaz:

- új címke megnyitása (*open*);
- létező címke továbbterjesztése (*propagate*);
- ekvivalens címkék egyesítése (*merge*), az ekvivalencia-tábla kitöltése.

Az első menetet a 10.2. ábra szemlélteti. A **második menetben** az ekvivalencia-tábla alapján módosítjuk a címkéket, hogy folytonos növekvő címkesort kapjunk.



alapvető műveletek futamok összefüggése

10.2. ábra. Az első RLC-menet áttekintése.

Az első menet előtt ki kell választanunk az objektum-összefüggőséget. Ezek után, inicializáljuk a címke-ekvivalencia táblát és a címke-számlálót:

$$E_{ij} = \delta_{ij},$$
$$L = 0,$$

ahol δ_{ij} a Kronecker delta függvény.

Az első menet során, amikor egy új, címke nélküli R futamot találunk, *az előző sorban* keressük az R-rel összefüggő futamokat.

- Ha **nincs összefüggő futam**, új címkét nyitunk: hozzárendeljük L-t az R-hez és növeljük L-t.
- Ha **egy összefüggő futam van** $L' \leq L$ címkével, továbbterjesztjüuk a címkét: hozzárendeljük L'-t az R-hez.
- Ha **több összefüggő futam van** $L'_k \leq L$ címkékkel, egyesítjuk az ekvivalens címkéket. Kiválasztjuk a legkisebbet $L'_m < L'_k, k \neq m$, hozzárendeljük L'_m-t az R-hez, és kitöltjük az ekvivalencia-táblát: minden k-ra $E_{km} = 1$.

10.1.2. Lánckód

Amikor csak **egy alakzat** (összefüggő komponens) van, lánckódolása a következő lépésekből áll. Kiválasztjuk a *kezdő kontúrpixelt* és a *körbejárási irányt*. Végigkövetjük az alakzat kontúrját, amíg vissza nem térünk a kezdőpontba. Közben eltároljuk a *lánckódokat*, azaz a következő kontúrpontra mutató vektor irányát. Diszkrét képen nyolc irány van, ezeket a 10.3. ábra mutatja (néha másképpen rendelik hozzá az irányokhoz a számokat). Egy alakzat teljes lánckódja tehát két részből áll:

1. a kezdőpont koordinátái, (x_0, y_0);
2. a következő kontúrpontra mutató irányok sorozata, $\{c_1, c_2, \ldots\}$.

Egy **bináris kép** több összefüggő komponensből is állhat. A kép lánckódját úgy kapjuk meg, hogy végigpásztázzuk és ha találunk egy körbe nem járt komponenst, körbejárjuk és lekódoljuk. Egy kép lánckódja az összes kontúr kódjából áll. Benne lehetnek a **belső kontúrok** (lyukak) lánckódjai, amelyket külön kell megjelölni, de lehetnek egymásba ágyazott (*nested*) komponensek is.

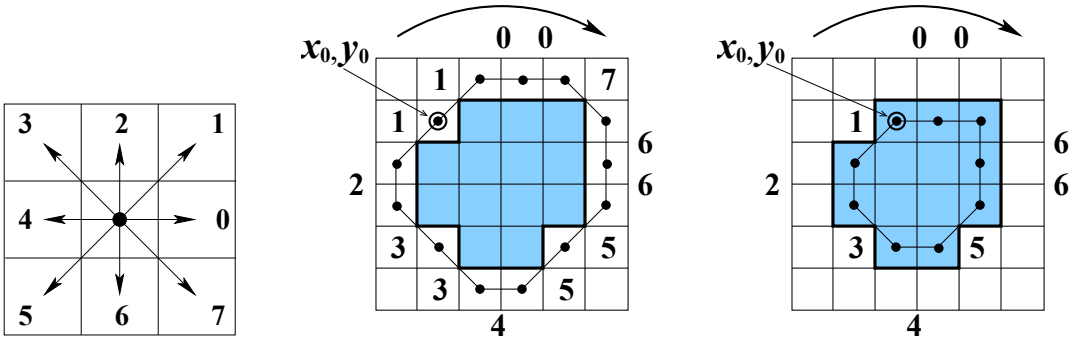
A 10.3. ábrán két lánckódolási példa látható. Egy alakzatot kívül is, belül is körbe lehet járni, ez szabadon választható. A példákban a 8-összefüggő háttér esetén az előbbi, a 8-összefüggő objektum esetén az utóbbi lehetőséget választattuk, de megtehettük volna másképpen is.

A 8-összefüggő háttérre a komponens 4-összefüggő. A komponenst kívül járjuk körbe, a kezdőpont az első olyan háttérpont, amelytől jobbra objektumpont van. A kapott lánckód:

$$1, 0, 0, 7, 6, 6, 5, 5, 4, 3, 3, 2, 1.$$

A 4-összefüggő háttérre a komponens 8-összefüggő. A komponenst belül járjuk körbe, a kezdőpont az első olyan objektumpont, amelytől balra háttérpont van. A kapott lánckód:

$$0, 0, 6, 6, 5, 4, 3, 2, 1.$$

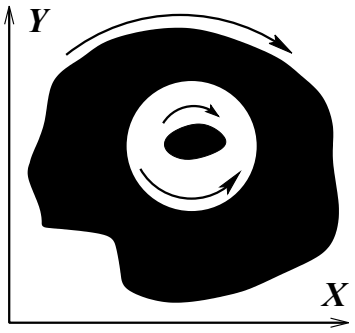


nyolc lánckód 8-összefüggő háttér 8-összefüggő objektum

10.3. ábra. Példák lánckódolásra.

Hogyan tudunk különbséget tenni **külső és belső kontúrok** (lyukak) között? A kérdés nem olyan egyszerű, mint amilyennek tűnik, mert egy kép letapogatása során nem tudjuk, hogy lyukban vagyunk-e vagy sem, és ez a körbejárás közben sem derül ki. Az egyik lehetséges megoldás az, hogy minden kontúrra kiszámítjuk a **komponens területét**, amelynek az **előjele** megadja a választ.

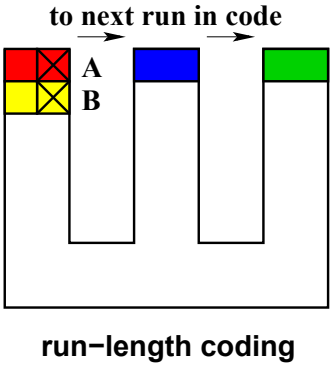
Az ötletet a 10.4. ábra illusztrálja, ahol a külső kontúrokat úgy járjuk körbe, hogy mindig **jobbra** fordulunk, az óramutatóval megegyező irányban haladva. A jelzett koordinátarendszerben a külső kontúrok területe ilyenkor **pozitív**. A belső kontúrok körbejárási iránya automatikusan az ellenkező lesz, példánkban az óramutatóval ellentétes irányú. Ez a körbejárás elvéből következik. A belső kontúrok területe emiatt **negatív**. Természetesen, a helyzet megfordulhat, ha másképpen választunk, de a lényeg megmarad: a külső és belső kontúrok közötti különbséget a terület előjele jelzi.



10.4. ábra. Külső és belső kontúrok egymásba ágyazva.

10.1.3. Adatstruktúrák összefoglalója

Gyakran előfordul, hogy egy alakzat helye és orientációja változik. Vizsgáljuk meg, hogyan változik az eltolt és elforgatott alakzat kódja! Az **eltolt** alakzat futam-hossz kódjában minden

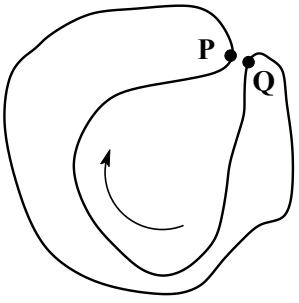


10.5. ábra. Az RLC nem támogatja a kontúrelemzést.

futam kezdőpontját kell módosítani. A futamok hosszai nem változnak. Az eltolásnál a lánckódjában csak a kezdőpontot kell módosítani, maguk a lánckódok nem változnak. Az **elforgatott alakzat** futam-hossz és lánckódját viszont újra kell kiszámítani, mert változásuk csak triviális esetekben követhető.

A terület alapú alakreprezetációk *támogatják* az integrális alakjellemzők kiszámítását, olyanokét, mint például a terület és a nyomatékok. Viszont *nem támogatják* a kontúrelemzést. Ezt a 10.5. ábra szemlélteti, ahol az A és B pont közel van egymáshoz a kontúron, de távol az RLC-ben, mert több futam elválasztja. A képmártixra jellemző síkbeli szomszédsági reláció itt nehezebben érvényesíthető.

A kontúr alapú alakreprezetációk ezzel szemben *támogatják* a kontúrelemzést és egyes integrális alakjellemzők kiszámítását, például a kerületet és a területet. Viszont, ahogy a 10.6. ábrán láthatjuk, *nem támogatják* a lokális képműveleteket. Az ábrán a P és Q pont közel van egymáshoz a képen, de távol a kontúron. Itt a síkbeli szomszédság még nehezebben érvényesíthető, mert megállapításához végig kell járni a kontúrt.



10.6. ábra. Kontúr alapú alakreprezetációk hátránya.

10.2. Terület alapú alakelemzési módszerek

10.2.1. Invariáns alaknyomatékok

Egy Q alakzat pq-rendű **centrális nyomatékai** (*shape moments*) folytonos esetben:

$$\mu_{pq} = \frac{1}{S} \iint\limits_{Q} (x - x_C)^p (y - y_C)^q \, dx \, dy, \qquad p, q = 0, 1, \ldots \tag{10.1}$$

diszkrét esetben pedig

$$\mu_{pq} = \frac{1}{S} \sum_{x,y \in Q} (x - x_C)^p (y - y_C)^q , \tag{10.2}$$

ahol S a Q területe. A centrális szó azt jelenti, hogy a nyomatékokat a centroidhoz (súlyponthoz) képest definiáljuk, amelynek a koordinátái diszkrét esetben:

$$x_C = \frac{1}{S} \sum_{x,y \in Q} x, \qquad y_C = \frac{1}{S} \sum_{x,y \in Q} y. \tag{10.3}$$

Az alaknyomatékok felhasználása a gyakorlatban információ veszteséggel jár. Elméletileg ugyan a nyomatékok megőrzik az alakinformációt olyan értelemben, hogy az alakzat visszaállítható *az összes* μ_{pq}-ből. A gyakorlatban azonban kevés számú, kisebb rendű (tipikusan, $p + q \leq 4$) nyomatékot használnak számszerű alakzat-jellemzőként, ami nem kontrollálható információ veszteséget jelent.

A magasabb rendű nyomatékok mellőzésének oka a zaj- és torzítás-érzékenységük. Nagy p-re az x^p deriváltja ugyanis nagy, ezért x^p felerősíti a zajt az x-ben. Az alacsonyabb rendű nyomatékokat viszont gyakran és hatékonyan alkalmazzák.

A fejezetben elsősorban a másodrendű nyomatékokról lesz szó, amikor $p+q = 2$. Ezeket gyakran használják elforgatás-invariáns alakleírásra. A másodrendű nyomatékok segítségével **két invariáns jellemző** definiálható:

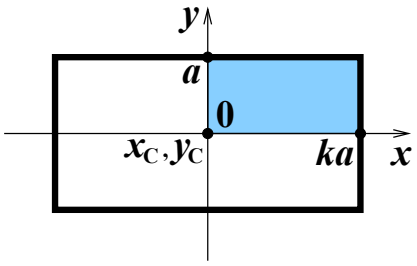
$$M_{cmp} = \frac{1}{2\pi} \cdot \frac{S}{\mu_{20} + \mu_{02}}, \tag{10.4}$$

$$M_{ect} = \frac{\sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}}{\mu_{20} + \mu_{02}}, \tag{10.5}$$

ahol a centrális nyomatékok

$$\mu_{20} = \frac{1}{S} \sum_{x,y \in Q} (x - x_C)^2 , \tag{10.6}$$

$$\mu_{11} = \frac{1}{S} \sum_{x,y \in Q} (x - x_C)(y - y_C) . \tag{10.7}$$



10.7. ábra. Változó méretű téglalap.

Az első jellemző (M_{cmp}) neve **kompaktság** (*compactness*). Értéktartománya $0 \leq M_{cmp} \leq 1$ úgy, hogy egy végtelen hosszú alakzatra $M_{cmp} = 0$, egy lemezre $M_{cmp} = 1$. M_{ect} neve **excentricitás** (*eccentricity*). Ugyanarra az értéktartományra van normalizálva, de pontosan fordítva viselkedik: egy végtelen hosszú alakzatra $M_{ecc} = 1$, egy lemezre $M_{ecc} = 0$.

Folytonos esetben M_{cmp} és M_{ect} **invariáns** minden hasonlósági transzformációra, azaz eltolásra, elforgatásra és izotrop skálázásra (méretváltozásra). Diszkrét esetben a jellemzők invariánsak a digitális rácson való eltolásra, az elforgatás- és skálázás-invariancia csak közelítő.

Az alábbiakban változó méretű téglalap példáján (10.7. ábra) bemutatjuk a két invariáns leíró (*descriptor*) jelentését. A téglalap méretaránya $k \geq 1$, tehát minél nagyobb az arány, annál hosszabb az alakzat. A téglalap területe $S = 2a \cdot 2ka = 4ka^2$, centrális nyomatékai pedig

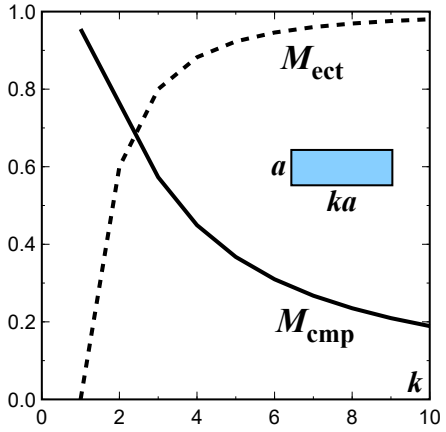
$$\begin{aligned}
\mu_{20} &= \frac{4}{S} \int_0^a \int_0^{ka} x^2 \, dx \, dy = \frac{k^2 a^2}{3}, \\
\mu_{02} &= \frac{4}{S} \int_0^a \int_0^{ka} y^2 \, dx \, dy = \frac{a^2}{3}, \\
\mu_{11} &= \frac{1}{S} \int_{-a}^a \int_{-ka}^{ka} xy \, dx \, dy = 0.
\end{aligned} \tag{10.8}$$

Egy hosszú téglalapra ($k \gg 1$)

$$M_{cmp} = \frac{1}{\pi} \cdot \frac{6k}{k^2 + 1} \approx \frac{6}{\pi k}, \tag{10.9}$$

$$M_{ect} = \frac{k^2 - 1}{k^2 + 1} \approx 1 - \frac{2}{k^2}, \tag{10.10}$$

vagyis a kompaktság csökken, az excentricitás pedig nő (10.8. ábra). Mind a két esetben a változás üteme egyre csökken, de az excentricitás hamarabb laposodik el. Emiatt hosszú



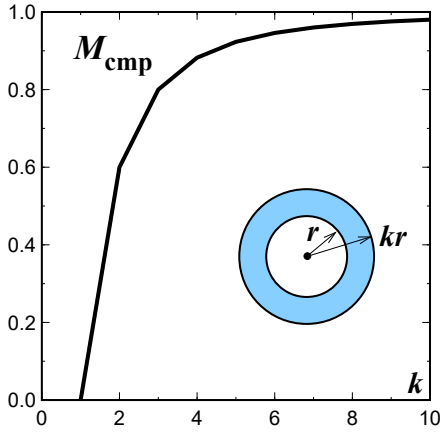
10.8. ábra. Téglalap-nyomatékok változása növekvő k-ra.

téglalapokat ($k > 5$) M_{ect} nem tud hatékonyan megkülönböztetni, mert értéke alig változik. Négyzetre ($k = 1$) $M_{cmp} \approx 0.96$, azaz közel áll a lemezre jellemző, maximális értékhez. Mivel azonban k növekedésével a kompaktság gyorsan csökken, ebben a tartományban vele még meg tudjuk különböztetni az alakzatokat.

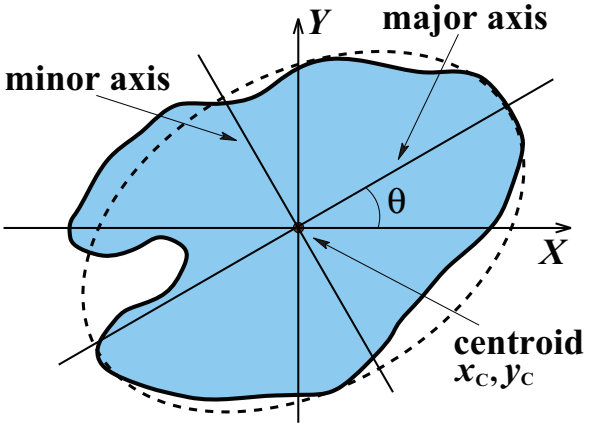
A 10.9. ábrán látható grafikon a gyűrű kompaktságát mutatja, amely

$$M_{cmp} = \frac{k^2 - 1}{k^2 + 1}. \tag{10.11}$$

Az alakzat körszimmetrikus, ezért $M_{ect} = 0$. Vastagságát a $k \geq 1$ paraméter szabályozza: minél nagyobb a paraméter, annál vastagabb a gyűrű. Elemezve az görbe deriváltját, megállapíthatjuk, hogy M_{cmp} megkülönböztet gyűrűket, ha $1 < k < 4$. Vastag gyűrűket ($k > 5$) a kompaktság nem tud meg különböztetni, mert túl kicsi a változás. Lyukas lemezre ($k \gg 1$) $M_{cmp} \approx 1$, vagyis közel áll a lemezhez ($M_{cmp} = 1$).



10.9. ábra. Gyűrű kompaktsága.



10.10. ábra. Inerciatengelyek és orientáció.

A kompaktság jellemzi a pontok *radiális eloszlását*: lemezre $M_{cmp} = 1$, gyűrűre $M_{cmp} \leq 1$. M_{cmp} robusztus zajra és a diszkrét képen történő elforgatásra. Az excentricitás jellemzi az alakzat *elnyújtottságát*: lemezre és gyűrűre $M_{ect} = 0$, vonalra $M_{ect} = 1$. M_{ect} nagyobb mértékben érzékeny zajra és elforgatásra.

10.2.2. Inerciatenzor és orientáció

μ_{20}, μ_{02}, és μ_{11} az I_{ij} **inerciatenzor** elemei, amely leírja a objektum forgását a súlyponton átmenő tengelyek körül. A tenzor a legjobban közelítő *ellipszissel* modellezi az alakzatot, ezt a 10.10. ábrával illusztráljuk. Az ellipszis *főtengelye* a hosszú tengely, amelyre az inercia a minimális: I_{min}. A *melléktengely* az ellipszis rövid tengelye, erre maximális az inercia: I_{max}. Az excentricitás a két inerciaérték különbségével arányos: $M_{ect} \propto I_{max} - I_{min}$, ezért nagyobb excentricitás hosszabb ellipszist jelent.

Egy alakzat helyét, pozícióját az x_C, y_C súlypont határozza meg, az orientációt pedig a főtengely és az X tengely által bezárt θ szög, amelynek a képlete

$$\theta = \frac{1}{2} \arctan \frac{\mu_{02} - \mu_{20}}{2\mu_{11}} + (\text{sign}\ \mu_{11}) \cdot \frac{\pi}{4} + \pi n, \quad n = 0, 1. \tag{10.12}$$

θ cirkuláris mennyiség, csak $\mod \pi$ határozható meg. *Hosszúkás* alakzatok esetén a képlet jól használható, mert hosszabb ellipszis pontosabb orientáció-becslést tesz lehetővé. *Kompakt*, körhöz közeli alakzatok esetén viszont a szög pontatlan, mert a képlet numerikusan instabil, amikor $\mu_{02} - \mu_{20}$ és μ_{11} kicsi. A főtengely *nem definiált* és a képlet nem alkalmazható, ha az alakzatnak *három vagy több szimmetriatengelye* van.

10.2.3. Az alaknyomatékok összefoglalója

A nyomatékokat az alábbi **adatstruktúrákban** lehet kiszámítani:

- szegmentált kép, a definíció szerint;
- futam-hossz kód, kicsit bonyolultabb;
- lánckód, még bonyolultabb.

Amikor $p + q = 3$, öt invariáns nyomaték-kombináció van, ezek az alakzat *aszimmetriáját* tükrözik. Zaj- és torzítás-érzékenyek, ezért ritkábban használják, mint M_{cmp}-t és M_{ect}-t. A nyomatékokat többféleképpen lehet kiterjeszteni, többek között, szürke és színes képekre és affin-invariáns módon.

Az invariáns nyomatékok **alkalmazási lehetőségei** a következők:

- Viszonylag kis számú (10–20), egymástól jól elkülönülő alakzat felismerése.
- Előszelektálás nagy számú alakzat felismerése esetén, azaz a szóba jöhető jelöltek gyors kiválogatása, például, kontúrillesztés vagy más számításigényes eljárás előtt.
- Más alakjellemzőkkel kombinálva.
- Tipikus alkalmazások: robot látás, karakter felismerés, affin-invariáns megfeleltetés.

Az invariáns nyomatékok **előnyei**:

- Elforgatás- eltolás- és nagyítás-invariánsak.
- Viszonylag egyszerűen kiszámíthatók.
- Különböző adatstruktúrákban meghatározhatók.
- M_{cmp} robusztus zajra és kisebb torzításra.
- Elég jól különböztetnek meg alakzatokat.
- Az orientáció is meghatározható.

A **hátrányok** az alábbiak:

- Nem tükröznek lokális kontúrsajátságokat.
- M_{ect} és a magasabb rendű nyomatékok kevésbé robusztusok.
- Az orientáció pontatlan vagy definiálatlan lehet.

10.3. Kontúr alapú alakelemzés

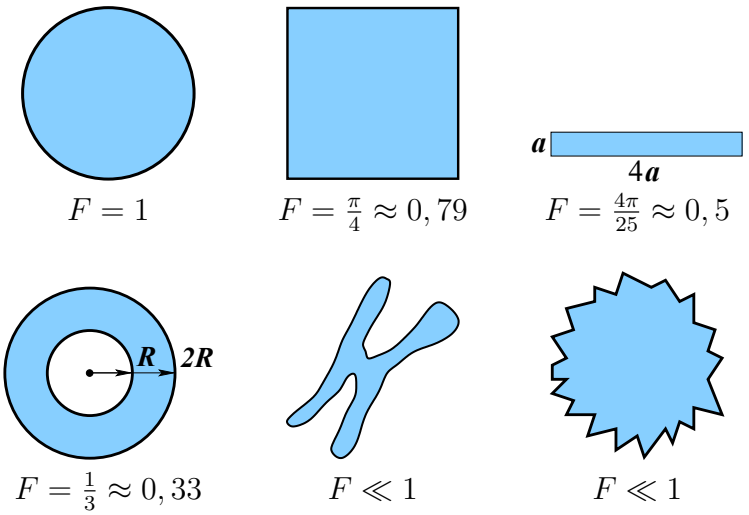
10.3.1. Alakténye ző

Az alakténye ző (*shape factor*) definíciója

$$F = \frac{4\pi S}{L^2}, \tag{10.13}$$

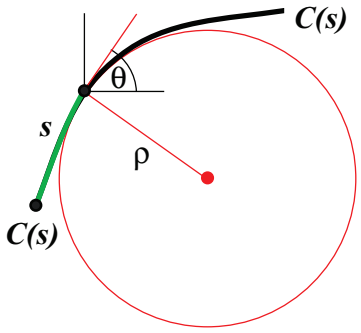
ahol S a terület, L a kerület és $0 \leq F \leq 1$. A jellemzőt körszerűségnek (*circularity*) is hívják, mert a körszerűség, kompaktság mértéke: lemezre $F = 1$, hosszú keskeny alakzatra $F \ll 1$. Az alakténye ző a kontúr *simaságát* is tükrözi.

Folytonos esetben az alakténye ző, mint a nyomatékok, invariáns minden hasonlósági transzformációra. Diszkrét esetben F invariáns a digitális rácson való eltolásra. Az elforgatás- és skálázás-invariancia csak közelítő. A jellemző függ a felbontástól, mert a kontúrrészletek megjelenésével a kerület lényegesen változik. A leíró viszonylag robusztus, de zajos kontúrok esetén simításra vagy approximációra lehet szükség.



10.11. ábra. Különböző alakzatok alakténye ző i.

A 10.11. ábra példákat mutat különböző alakzatok alakténye ző ire. A kompaktsághoz hasonlóan az alakténye ző értéke lemezre a maximális. Cikcakkos határú lemezre viszont a két jellemző viselkedése nagyon eltérő: a kompaktság csak kisebb mértékben csökken, miközben az alakténye ző a kerület lényeges növekedése miatt nagyot zuhan. Hasonló okok miatt a gyűrűre a két jellemző értéke közel kétszeres eltérést mutat. A hosszú téglalapra ezzel szemben a két érték alig különbözik egymástól.



10.12. ábra. Görbület definíciója.

10.3.2. Görbület-elemzés

Legyen $C(p) = \{x(p), y(p)\}$ egy p paraméterrel paraméterezett görbe. Például, p lehet x, akkor $C(x) = \{x, y(x)\}$; vagy az s ívhossz, akkor $C(s) = \{x(s), y(s)\}$. Definíció szerint a **görbület**

$$\kappa = \frac{d\theta}{ds} = \frac{1}{\rho}. \tag{10.14}$$

Itt θ a meredekség

$$\tan\theta = \frac{dy}{dx},$$

s az ívhossz

$$ds^2 = dx^2 + dy^2,$$

ρ pedig az érintőkör sugara, a **görbületi sugár**. A fogalamakat a 10.12. ábra szemlélteti[1].

Tetszőleges $\{x(p), y(p)\}$ paraméterezésre

$$\kappa = \frac{x_p y_{pp} - y_p x_{pp}}{\left(x_p^2 + y_p^2\right)^{3/2}}, \quad \text{ahol } x_p = \frac{\partial x}{\partial p}. \tag{10.15}$$

Az $\{x, y(x)\}$ paraméterezésre

$$\kappa = -\frac{y_{xx}}{\left(1 + y_x^2\right)^{3/2}}. \tag{10.16}$$

Ezzel a paraméterezéssel gond lehet, ha $y(x)$ többértékű; ezen kívül függőleges érintőkkel is lehet probléma, mert $|y_x| \to \infty$. Ezzel szemben az s ívhosszal paraméterezett folytonos görbékre ilyen gondok nincsenek, mert $x(s)$ és $y(s)$ egyértékű és x_s és y_s korlátos.

A görbület pozitív és negatív is lehet, mert az érintőkör a görbétől jobbra, balra is eshet. A deriválás miatt a görbület *zajérzékeny*, ezért a deriválás előtt a görbét el kell simítani, amire

[1] Angolul görbe *curve*, görbület *curvature*, meredekség *slope*, ívhossz *arc length*, érintőkör *osculating circle*, görbületi sugár pedig *radius of curvature*.

átlagszűrőket, tipikusan Gauss-szűrőket használnak. Növekvő σ-val történő Gauss-szűrés utáni a görbület számítás *görbületi mértékteret, scale-space*-t eredményez. Az élszűréshez hasonlóan, ez fokozatosan csökkenő részletességet jelent.

A nagy görbületű pontok **sarkok**, kontúr-töréspontok. Már tudjuk, hogy a sarkok kiemelt szerepet játszanak az emberi és a gépi látásban egyaránt, mert jól lokalizálhatók és fontosak az alak- és mozgáselemzésben.

A görbületi **nulla átmenetek** (zero-crossings) inflexiós pontok, amelyek kevésbé jól lokalizálhatók és kevésbé fontosak az emberi látásban, a gépi látásban is ritkábban alkalmazzák.

Diszkrét esetben a görbület számításához megfelelő kontúr adatstruktúrára van szükség, ami gyakran a lánckód lesz, hiszen kontúrkiemeléshez körbe kelle járni az alakzatot. Itt az lehet a probléma, hogy zajos képek zajos görbéket produkálnak, és a lánckód túlságosan "cikcakkos" lesz. Beleszólhatnak a diszkrétizálási hibák is, amelyek miatt megjelenhetnek nagyon keskeny részek, szélsőséges esetben önmagát érintő kontúrszakaszok. Ezek a tényezők pontatlan görbületet és hamis sarkokat eredményezhetnek.

Az alábbiakban adunk egy lehetséges megoldást **görbület számításra**. Célszerű ívhosszal paraméterezett görbékkel dolgozni.

1. Eltávolítjuk a zajt, elsimítjük a görbét, közben ügyelünk a potenciális sarkokra, hogy ne tegyük tönkre.

2. Szabályos, konstáns Δs lépésenként újramintavételezzük az $x(s), y(s)$ görbét. Ehhez interpolációra lesz szükség.

3. Kiszámítjuk a görbületet a (10.15) képlet szerint, felhasználva az deriváltak közelítését:

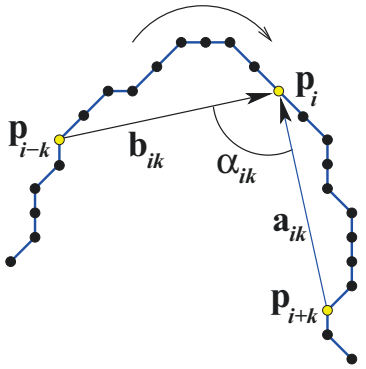
$$x_s \approx \frac{x_{i+1} - x_{i-1}}{2\Delta s},$$
$$x_{ss} \approx \frac{x_{i+1} - 2x_i + x_{i-1}}{(\Delta s)^2}.$$

Kontúrsarkok detektálásához a lánckód alapú **sarokerősség** bevezetése szükséges. Az egyik lehetőség a k-koszinuszok alkalmazása, ezt a 10.13. ábrával illusztráljuk.

1. Beállítjuk a sarok finomságát szabályozó skálaparamétert: $k = 1, 2, \ldots$ Minél kisebb a paraméter, annál finomabbak a sarkok és zajérzékenyebb az algoritmus.

2. Kiszámítjuk a k**-vektorokat**:

$$\mathbf{a}_{ik} = (x_i - x_{i+k}, y_i - y_{i+k}), \tag{10.17}$$
$$\mathbf{b}_{ik} = (x_i - x_{i-k}, y_i - y_{i-k}). \tag{10.18}$$



10.13. ábra. Sarokerősség k-koszinuszokkal.

3. Sarokerősségként felhasználjuk a k**-koszinuszokat**:

$$c_{ik} = \cos\alpha_{ik} = \frac{(\mathbf{a}_{ik}\mathbf{b}_{ik})}{|\mathbf{a}_{ik}||\mathbf{b}_{ik}|}, \tag{10.19}$$

ahol $(\mathbf{a}_{ik}\mathbf{b}_{ik})$ a két vektor skalárszorzata.

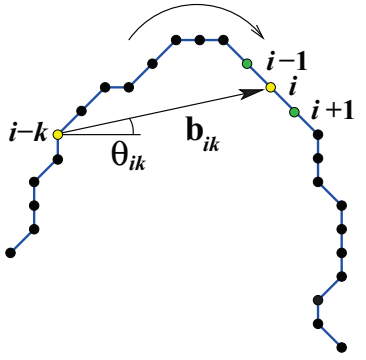
Sarokdetektálás másképpen is elvégezhető (10.14. ábra). A $\mathbf{b}_{ik}$ k-vektor komponensei

$$(X^-_{ik}, Y^-_{ik}) = (x_i - x_{i-k}, y_i - y_{i-k})$$

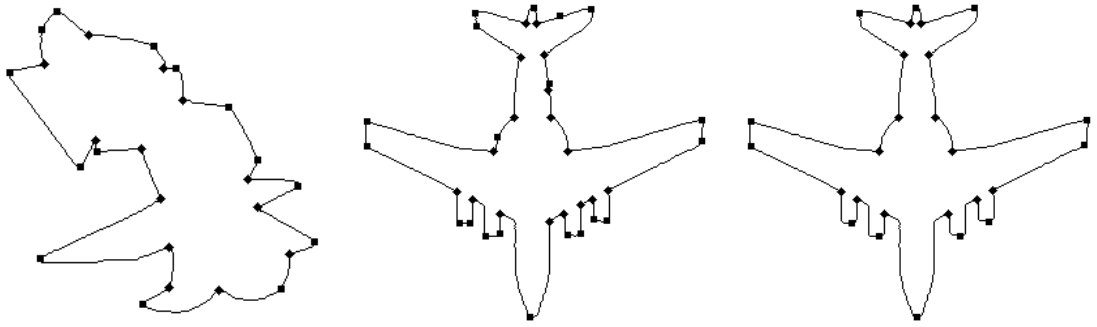
Az alternatív sarokerősség a k**-vektor szögváltozása** lehet:

$$\Delta\theta_{ik} = \theta_{i+1,k} - \theta_{i-1,k}, \quad \text{ahol} \tag{10.20}$$

$$\theta_{ik} = \arctan\left(\frac{Y^-_{ik}}{X^-_{ik}}\right). \tag{10.21}$$



10.14. ábra. k-vektor szögváltozása.



10.15. ábra. Példák sarokdetektálásra.

Az éldetektáláshoz hasonlóan a sarokerősségre alkalmazzák a **non-maximumok elnyomását**. A helyzet annyiban egyszerűbb, hogy a kontúron a szomszédos pontokat könnyebb megkeresni.

A 10.15. ábra több sarokdetektálási példát mutat változó részletességi paraméter mellett. Kis k esetén zajos, hamis detektálás is van, viszont a repülő kontúrján megtaláljuk az összes sarkot, az egymáshoz közelit is. Amikor k nagy, a non-maximumok elnyomása miatt az egymáshoz közeli sarkok közül az egyik elvész.