

A gépi látás és képfeldolgozás párhuzamos modelljei és algoritmusai

**Rövid András
Sergyán Szabolcs
Vámossy Szabolcs**

A gépi látás és képfeldolgozás párhuzamos modelljei és algoritmusai

írta Rövid András , Sergyán Szabolcs , és Vámossy Szabolcs

Szerzői jog © 2013 Rövid András, Sergyán Szabolcs, Vámossy Zoltán, Óbudai Egyetem

Tartalom

1. Párhuzamos feldolgozás eszközei	1
1. CUDA kernelek és szálak	1
2. CUDA - Sokelemű vektorok összeadása:	3
2. Képfeldolgozás és gépi látás bevezető	4
1. Digitális képfeldolgozás és a rokon területek	4
1.1. Bevezető	4
1.2. Történeti bevezető – kezdetek	5
1.3. Mi a digitális kép fogalma?	5
1.4. Mintavételezés és kvantálás	5
1.5. Digitális kép	6
1.6. Mi a képfeldolgozás?	6
1.7. Vázlatos definíciók	6
1.8. Képfeldolgozás (Image Processing)	7
1.9. Számítógépes grafika (Computer Graphics)	8
1.10. A digitális képfeldolgozás szintjei	8
1.11. A három feldolgozási szint	9
1.12. Mi a gépi látás (Computer Vision)?	10
1.13. minden kép egy történet	10
1.14. Számítógépes látórendszer általános modellje	11
1.15. A számítógépes látás a következő területekre koncentrál	11
1.16. Számítógépes látás	12
1.17. Mintafelismerés (Pattern Recognition)	12
1.18. Mesterséges intelligencia (AI)	12
2. Miért bonyolult a számítógépes látás?	12
2.1. Felismerés nehézségei	13
2.2. A szín szerepe	13
2.3. A textúra szerepe	14
2.4. Az alak szerepe	15
2.5. A csoportosítás szerepe	16
2.6. Praktikus megfontolások	17
2.7. Gépi látás eléri-e, megelőzi-e az emberi látást?	18
2.8. Emberi érzékelés korlátai	18
3. Hol tart a képfeldolgozás és a gépi látás?	20
3.1. Hol tart ma a gépi látás?	20
3.2. Föld megjelenítők (3D modell)	20
3.3. Fotószintézis	20
3.4. Optikai karakterfelismerés (OCR)	21
3.5. Arcdetektálás	21
3.6. Mosoly detektálás	22
3.7. Arcfelismerés	22
3.8. Biometria	23
3.9. Biometrikus azonosítás	24
3.10. Objektum felismerés (mobil telefonokban)	24
3.11. Speciális effektusok	25
3.12. Speciális effektusok: motion capture technika	25
3.13. Sport	26
3.14. Okos autók	27
3.15. Google autó	27
3.16. Látás alapú interaktivitás	27
3.17. Úralkalmazás	28
3.18. Robotlátás (Robot Vision)	29
3.19. Orvosi alkalmazás	29
3.20. „State of the art”	30
3. Adatstruktúrák a képfeldolgozásban	31
1. Kép	31
1.1. Képek ábrázolása	31

1.2. Digitális kép	32
1.3. Számítási igény	33
1.4. Színes képek	33
1.5. Hisztogram	34
1.6. Integrál kép	35
1.7. Adott téglalapra a számítás	36
1.8. Haar-szerű jellemzők számítása integrál képből	36
1.9. Jellemzők felhasználása	37
4. Intenzitás transzformációk	39
1. Pont alapú műveletek	39
1.1. Képműveletek osztályozása	39
1.2. Pont alapú műveletek	39
1.3. Pont alapú: identitás, vagy egység	40
1.4. Pont alapú: negálás	40
1.5. Pont alapú: intenzitás szintre vágás	41
1.6. Pont alapú: kontraszt nyújtás	42
1.7. Pont alapú: kontraszt növelés/csökkentés	42
1.8. Pont alapú műveletek	43
1.9. Pont alapú: nem lineáris transzformációk	43
1.10. Logaritmikus skálázás példa	45
1.11. Hatvány és gyök függvények	45
1.12. Gamma korrekció	46
1.13. Pont alapú műveletek: küszöbölés	47
1.14. Bit-síkonkénti vágás	47
1.15. Bit-síkonkénti vágás (példa)	48
1.16. Hibadiffúziós algoritmus (Floyd 1975)	48
1.17. Algoritmus	49
1.18. Párhuzamosított hibadiffúzió	50
2. Hisztogram transzformációk	50
2.1. Hisztogram műveletek	50
2.2. Hisztogram széthúzás	51
2.3. Kontraszt széthúzás	52
2.4. Példa: hisztogram és kontraszt széthúzás	53
2.5. Hisztogram kiegyenlítés	53
2.6. Hisztogram kiegyenlítés példa	54
3. Teljes képes és geometriai transzformációk	55
3.1. Teljes képre vonatkozó transzformációk	55
3.2. Teljes képre: átlagolás	55
3.3. Teljes képre: kivonás	56
3.4. Teljes képre: AND/OR	57
3.5. Geometriai transzformációk	57
3.6. Geometriai transzformáció: problémák	58
3.7. Geometriai transzformáció: 2. probléma	59
3.8. Geometriai transzformáció: 3. probléma	59
5. Szűrés képtérben	61
1. A szűrés elve, konvolúció, korreláció, lineáris, nem lineáris szűrők	61
1.1. Maszk, vagy ablak alapú műveletek	61
1.2. Mit jelent a képszűrés (filtering)?	62
1.3. Maszkok	63
1.4. Lineáris szűrők	63
1.5. Megjegyzés: Nem lineáris szűrők	63
1.6. Képtérben történő szűrések csoportosítása	64
1.7. Maszk használata	64
1.8. Maszk	64
1.9. Maszk használata és konvolúció	65
1.10. Az 1D konvolúcióról	65
1.11. 1D konvolúció példa	66
1.12. Megjegyzés: korreláció	68
1.13. Konvolúció és korreláció 1D – példa	68
1.14. Konvolúció és korreláció 2D – példa	69

1.15. Maszksúlyok normalizálása	69
1.16. Gyakorlati problémák	69
1.17. Szűrő példák	70
1.18. Példa: simítás (blurring) és kivonás	71
2. Simító szűrés	71
2.1. Maszk alapú műveletek	71
2.2. Átlagoló szűrő	72
2.3. Simító ablak méretének hatása	74
2.4. Példa: simítás átlagolással	75
2.5. Párhuzamos átlagoló szűrő	76
2.6. Simítás Gauss szűrővel	76
2.7. Gauss szűrő	77
2.8. Gauss szűrő szeparált megvalósítása	78
2.9. Gauss szűrő	79
2.10. Binomiális szűrő	80
2.11. Medián szűrő (nem lineáris)	82
2.12. Medián szűrő	82
2.13. Medián szűrő megjegyzés	83
2.14. Medián szűrő negatív hatásai	83
2.15. Medián szűrő	84
2.16. Közelítő medián szűrő – párhuzamosítás	84
3. Élesítő szűrés	85
3.1. Élesítés (sharpening)	85
3.2. Élesítés deriváltak használatával	86
3.3. Első és másodrendű differenciák példa	86
3.4. Élesítés	87
3.5. Laplace szűrő	87
3.6. Életlenítés (unsharpling) és felül erősítés	89
3.7. Felül erősítés	90
6. Élek, sarokpontok, speciális szakaszok	92
1. Éldetektálás elve és éldetektorok	92
1.1. Élek (edges)	92
1.2. Élek: Mi okoz hirtelen változást?	93
1.3. Tipikus élprofilok	93
1.4. Hogyan találhatunk éleket?	94
1.5. Definíciók	94
1.6. Éldetektálás deriválással	95
1.7. Változás detektálás 1D-ben	95
1.8. Deriváltak, differenciák 2D-ben	96
1.9. Gradiens nagyság és irány	96
1.10. Differenciák	97
1.11. Éldetektáló maszkok	97
1.12. Prewitt maszk	98
1.13. Sobel éldetektáló	98
1.14. Prewitt és Sobel éldetektálás	99
1.15. Prewitt éldetektor párhuzamosítás	100
1.16. Sobel éldetektáló párhuzamosítás	100
1.17. Sobel maszk: összefoglalás	100
1.18. Robinson iránytú maszk	101
1.19. További maszkok	101
1.20. Laplace éldetektálás	102
1.21. Gauss simítás + Laplace (LoG)	102
1.22. LoG – Marr-Hildreth éldetektáló	103
1.23. LoG példa	104
1.24. Canny éldetektor	105
1.25. Ideális éldetektáló	105
1.26. I. Canny éldetektor	105
1.27. I. Canny éldetektor – első két lépés példa	106
1.28. II. Non-maxima suppression	106
1.29. II. Non-maxima suppression algoritmus	107

A gépi látás és képfeldolgozás
párhuzamos modelljei és
algoritmusai

1.30. III. Canny – harmadik lépés oka	107
1.31. III. Hysteresis thresholding	107
1.32. III. Canny alacsony, illetve magas küszöb	108
1.33. III. Hysteresis thresholding	108
1.34. Canny eredmények	109
1.35. Canny: sarok effektus	109
1.36. SUSAN algoritmus	110
2. Jellemző pontok keresése	112
2.1. Sarokpont detektálás	112
2.2. Moravec operátor	112
2.3. Moravec - példa	113
2.4. Harris sarokdetektor	114
2.5. Harris sarokdetektáló	114
2.6. Harris sarokdetektáló - példa	114
2.7. Kanade-Lucas-Tomasi algoritmus	115
3. Elvárt helyzetű szakaszok detektálása	116
3.1. Átlagos intenzitás számolása	117
3.2. Élerősség tömb elkészítése	117
3.3. Élfeltételek vizsgálata	118
7. Képpiramisok	120
1. Képpiramisok bevezető	120
1.1. Skálázás	120
1.2. Sub-sampling (downsampling)	121
1.3. Image sub-sampling - visszanagyítva	122
1.4. Sub-sampling minden második pixellel	122
1.5. Mintavételezés - 2D példa	123
1.6. Simítás	123
1.7. Sub-sampling Gauss szűrővel	124
1.8. Sub-sampling Gauss szűrővel - visszanagyítva	124
1.9. Csak sub-sampling	125
1.10. Képpiramisok	125
1.11. Képpiramis	125
1.12. Piramisok készítése	126
1.13. Közelítő piramis és maradék piramis	127
1.14. Alkalmazási területek	128
2. Gauss piramis	128
2.1. Gauss szűrő - emlékeztető	128
3. Gauss piramis	129
3.1. Gauss piramis 1D-ben	129
3.2. Redukáló függvény, konvolúciós maszk w	129
3.3. Konvolúciós maszkok (5×1)	130
3.4. Gauss piramis megvalósítása képre	130
3.5. Gauss piramis példa	131
4. Laplace piramis	132
4.1. Laplace piramis példa	133
4.2. Képrekonstrukció piramisokból	134
4.3. Alma-narancs összeolvastás	134
4.4. Összeolvastás maszkkal	135
8. Képek szegmentálásának módszerei	137
1. Küszöbölés	137
1.1. Hisztogram	137
1.2. Hisztogram alapú technika	138
1.3. Küszöb meghatározása	138
1.4. Küszöbölés eredménye	139
1.5. Küszöb meghatározása	139
1.6. Iteratív küszöbölés megvalósítása	140
1.7. Küszöb meghatározása	141
1.8. Élpixelek használata	142
1.9. Hiszterézises küszöbölés	142
1.10. Otsu algoritmus	142

A gépi látás és képfeldolgozás
párhuzamos modelljei és
algoritmusai

1.11. Entrópia használata	143
1.12. Minimális hibájú küszöbölés	144
1.13. Több küszöb használata	146
1.14. Peakiness teszt	146
1.15. Adaptív (alkalmazkodó) küszöbölés	147
1.16. Adaptív küszöbölés	148
1.17. Niblack algoritmus	149
1.18. Hisztogram klaszterezés	149
1.19. Csúcsok és természetes intervallumok	149
1.20. Színes képek hisztogramjának klaszterezése	150
2. Határvonal alapú szegmentálás	151
2.1. Belső határvonal bejárás	151
2.2. Hough transzformáció	152
2.3. Egyenes	152
2.4. Koordináta-sík és paraméter-sík	152
2.5. A Hough transzformáció alapötlete	152
2.6. Egyenes szakaszok detektálása Hough transzformációval	153
2.7. Probléma	154
2.8. Megoldás: Polár koordinátás reprezentáció	154
2.9. A végleges algoritmus	154
2.10. Kör	155
2.11. Körök detektálása Hough transzformációval	155
2.12. Ellipszis	156
2.13. Algoritmus általános paraméteres görbe esetén	157
3. Összefüggő komponens analízis	157
3.1. Segmentálás küszöböléssel	157
3.2. Rekurzív régió növelő algoritmus	158
3.3. Rekurzív régió növelő algoritmus (példa)	158
3.4. Szekvenciális algoritmus	161
3.5. Szekvenciális algoritmus (példa)	161
4. Régió alapú szegmentáló algoritmusok	171
4.1. Régió növesztés (Region Growing)	171
4.2. Régió növesztés (példa)	172
4.3. Split and Merge	177
4.4. Watershed algoritmus	180
9. Alakleírók, alakfelismerés módszerei	191
1. Határvonal alapú leírók	191
1.1. Lánckód	191
1.2. Sztigmatúra	193
1.3. Határvonalak Fourier transzformáltja	194
1.4. Fourier leírók (MATLAB példa)	195
2. Régió alapú alakleírás	196
2.1. Terület	196
2.2. Momentumok	197
10. Textúra	201
1. Statisztikai textúra leírók	201
1.1. Frekvencián alapuló módszerek	201
1.2. Autokorrelációs textúra leíró	201
1.3. Autokorrelációs függvény a frekvencia tartományban	201
1.4. Együttes előfordulási (co-occurrence) mátrixok	202
1.5. Együttes előfordulási mátrixok	202
11. Optikai áramlás	204
1. Optikai folyamok alkalmazása	204
1.1. Mozgás	204
1.2. minden egyes pixel mozgását mérjük	204
1.3. Hol használjuk a mozgást a gépi látás területein?	205
1.4. Mozgásdetektálás	205
1.5. Mozaikozás	205
1.6. Képszegmentálás	206
1.7. Struktúra meghatározás mozgás alapján	207

1.8. Optikai áramlás (Optical flow)	207
2. Jellemzők és egyenletrendszerek	207
2.1. Mi az optikai folyam?	207
2.2. Az optikai folyam speciális esetei	208
2.3. Apertúra probléma	209
2.4. Kiinduló feltevések (1): konstans fényesség	210
2.5. Kiinduló feltevések (2): térbeli összetartozás	210
2.6. Kiinduló feltevések (3): időbeli összetartozás	210
2.7. Optikai folyam egyenletek	211
2.8. Apertúra probléma	213
3. Megoldási módszerek	214
3.1. Megoldási technikák	214
3.2. Horn & Schunck megoldása	214
3.3. Schunck módszere	215
3.4. Lucas & Kanade módszere	216
3.5. Fontos megjegyzések	216
3.6. Optikai folyam számítása piramis módszerrel	217
3.7. Optikai folyam meghatározása nagy mozgás esetén – hiba	217
3.8. Optikai folyam meghatározása piramis módszerrel	217
3.9. Optikai folyam példa	218
4. Irodalomjegyzék	219
12. Sztereó látás, 3D rekonstrukció	220
1. Perspektív vetítés és a kamera paraméterei	220
1.1. "Pinhole" kamera modell	220
1.2. Perspektív vetítés és a kamera paraméterei	221
1.3. Kamera belső paraméterei	221
2. Kamera kalibráció	223
2.1. Kamera kalibráció feladata	223
2.2. Zhang-féle kamera kalibráció	223
2.3. Felhasznált és javasolt irodalom	225
2.4. Hand-eye kalibráció	225
2.5. "Hand-eye" kalibráció	226
2.6. Forgató mátrix és a quaterniók kapcsolata	227
2.7. Felhasznált és javasolt irodalom	228
3. EPIPOLÁRIS geometria	228
3.1. EPIPOLÁRIS geometria - Fundamentális mátrix	229
3.2. Felhasznált és javasolt irodalom	230
4. 3D rekonstrukció	230
4.1. Kamera-lézer alapú 3D rekonstrukció	230
4.2. Kamera-projektor alapú 3D rekonstrukció	234
13. Fourier transzformáció és alkalmazásai	238
1. Fourier sorok és a Fourier transzformáció	238
1.1. Fourier transzformáció - kétváltozós eset	240
2. Diszkrét Fourier transzformáció (DFT)	241
2.1. 2D Diszkrét Fourier transzformáció (DFT)	241
3. Gyors Fourier transzformáció (FFT)	241
3.1. Fourier transzformáció - példa	243
3.2. Felhasznált és javasolt irodalom	244
14. Diszkrét koszinusz transzformáció	245
1. Felhasznált és javasolt irodalom	246
15. Waveletek és alkalmazásaiik	247
1. Rövid idejű Fourier transzformáció (STFT)	247
2. Wavelet transzformáció	247
2.1. Wavelet transzformáció - skála	248
2.2. Wavelet transzformáció - skála és frekvencia	249
2.3. Wavelet transzformáció - felbontás	249
2.4. Wavelet transzformáció - inverz FT	249
2.5. Wavelet transzformáció - skála és wavelet	250
3. Diszkrét Wavelet transzformáció	251
4. Gyors Wavelet transzformáció	251

A gépi látás és képfeldolgozás
párhuzamos modelljei és
algoritmusai

4.1. Wavelet transzformáció - példa	252
4.2. Inverz Wavelet transzformáció - példa	252
4.3. Diszkrét Wavelet transzformáció kétváltozós esetben	252
4.4. 2D Wavelet transzformáció - példa	253
4.5. 2D Inverz Wavelet transzformáció - példa	253
4.6. 2D Wavelet transzformáció - párhuzamos végrehajtás GPU-n	254
4.7. Felhasznált és javasolt irodalom	254
16. Képtömörítési eljárások párhuzamos környezetben	256
1. JPEG tömörítő eljárás	256
1.1. JPEG tömörítő eljárás - példa	257
1.2. Felhasznált és javasolt irodalom	259
2. HOSVD-alapú tömörítés	259
2.1. Felhasznált és javasolt irodalom	261
17. Letölthető melléklet	262

Az ábrák listája

1.1. GPGPU architektúra szemléltetése	1
1.2. "kerneleket" tartalmazó alkalmazás végrehajtásának folyamata	2
1.3. Szálak, Blokkok és a Grid kapcsolata	2
12.1. Kamera-lézer alapú 3D mérés	230
12.2. Lézersík meghatározása	232
12.3. Kamera-lézer alapú rendszer sztereó kamerákkal való követése	233
12.4. Kamera-lézer alapú 3D rekonstrukció párhuzamos megvalósításának folyamatábrája	233
12.5. A projektor inverz kamera modellként való kalibrálása	234
12.6. 4 bites Gray code	235
12.7. A 3D rekonstruções rendszer architektúrája	236
12.8. Szkennelés eredménye	236
13.1. Minél több a közelítésben résztvevő trigonometrikus függvények száma, annál pontosabb a közelítés	238
13.2. 16 pontos FFT 4 feldolgozó egység használatával	243
13.3. Eredeti kép (balra), A kép Fourier transzformáltja (jobbra)	243
13.4. A nagyobb frekvenciák elhagyásának hatása	244
15.1. Nemstacionárius jel (balra); Stacionárius jel (jobbra)	247
16.1. A HOSVD illusztrációja három változós esetben. D az ún. magtenzor a mátrixok pedig a dimenzióinkénti ortonormált mátrixokat jelölik, melyek oszlopai az approximációnál alkalmazott egyváltozós függvények diszkretizált változatai.	260

1. fejezet - Párhuzamos feldolgozás eszközei

Rövid András

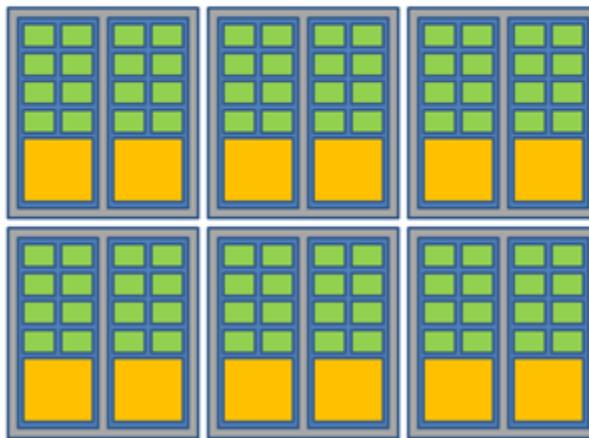
Párhuzamos feldolgozás eszközei

- Sokmagos processzorokon való feldolgozás
 - Többszálú programozás
- GPGPU-n (General Purpose GPU)
 - Több száz maggal rendelkeznek, melyek kollektívan több ezer szálat futtathatnak
 - Ezen magok megosztott erőforrásokkal rendelkeznek ("register file", "shared memory")
 - Az "on-chip" megosztott memória lehetővé teszi a magokon futó párhuzamos feladatok adatmegosztását anélkül, hogy a "system memory bus"-on átküldenék egymásnak.
 - NVIDIA CUDA - párhuzamos számítási platform és programozási modell

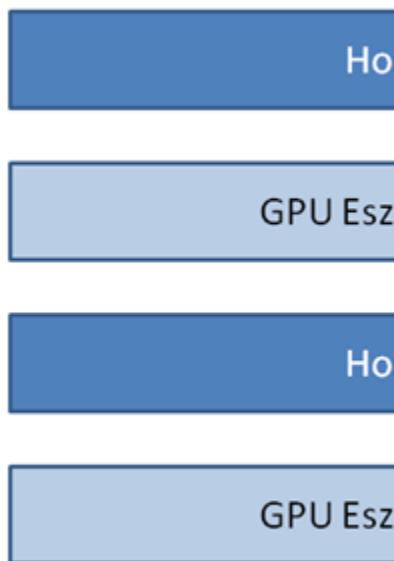
1. CUDA kernelek és szálak

- Az alkalmazások párhuzamos részei az eszközön ún. kernelként futnak.
- CUDA vs. CPU szálak
 - A CUDA szálak létrehozási "költsége" kicsi
 - Gyors váltás a szálak között
 - Több ezer szál futtatása
- minden szál ugyanazt a kódot futtatja
- A szálak blokkokba vannak szervezve
- minden szál és blokk azonosítóval van ellátva
 - így minden szál tudja, hogy melyik adaton kell dolgozni
 - A szál ID-k egy, két vagy három dimenziósak
 - A kernel, szál blokkok csoportját (Grid) futtatja.
 - A Grid ID-k egy vagy két dimenziósak

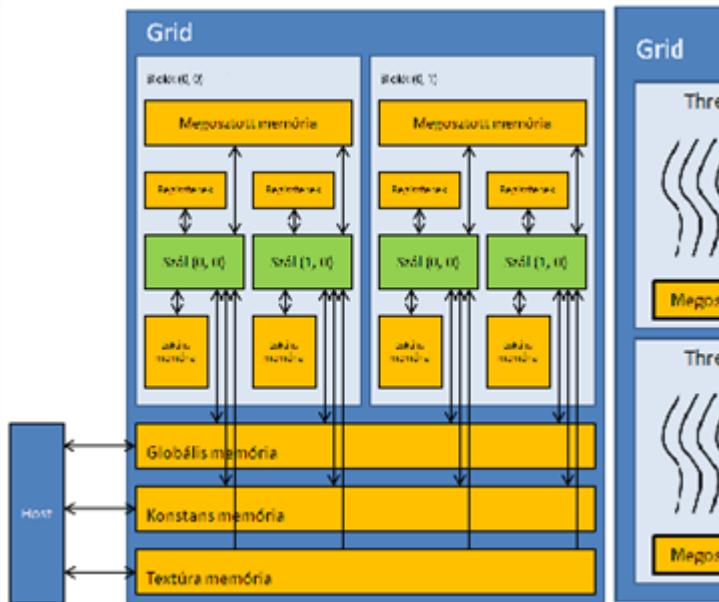
1.1. ábra - GPGPU architektúra szemléltetése



1.2. ábra - "kerneleket" tartalmazó alkalmazás végrehajtásának folyamata



1.3. ábra - Szálak, Blokkok és a Grid kapcsolata



2. CUDA - Sokelemű vektorok összeadása:

NVIDIA CUDA alapú megvalósítás

```

1  __global__ void add(int *a, int *b, int *c) {
2      int tid = threadIdx.x + blockIdx.x * blockDim.x;
3      while(tid < N) {
4          c[tid] = a[tid] + b[tid];
5          tid += blockDim.x * gridDim.x;
6      }

```

CPU-n történő futtatáshoz

```

1  void add(int *a, int *b, int *c) {
2      int tid = 0;
3      while (tid < N) {
4          c[tid] = a[tid] + b[tid];
5          tid += 1;
6      }

```

Felhasznált és javasolt irodalom

- [1] Zeller, Cyril: *Tutorial CUDA (slides)*, NVIDIA Developer Technology, 2008.
- [2] Sanders, Janson:Kandrot, Edward: *CUDA by Example, An Introduction to General-Purpose GPU Programming*, Addison-Wesley, ISBN-13: 978-0-13-138768-3, 2010.
- [3] *CUDA Programming Model Overview (slides)*, NVIDIA, 2008.

2. fejezet - Képfeldolgozás és gépi látás bevezető

Vámossy Zoltán

A fejezet nagyrészt Steve Seitz és Richard Szeliski [1] prezentációján, valamint Szeliski könyvén [2] alapszik, de az egyes részeknél merítettünk a képfeldolgozás és gépi látás kurzusokban gyakran használt Gonzales-Woods [4] és Trucco-Verri [3] könyvekből.



Terminator 2

1. Digitális képfeldolgozás és a rokon területek

1.1. Bevezető

- A számítástechnikában korábban az adat numerikus érték volt
- Később szöveges
- Ma sok más forma: hang, zene, beszéd, kép, ...
- Ezek az adatok mind jelek
- A jel tartalmazhat információt, azonban annak értelme (szemantikája) függ a környezettől (kontextustól), amelyben a jelet értelmezni szeretnénk, illetve a feldolgozástól (szubjektumtól), amely az értelmezést végzi:
Hell - németül fényes, angolul pokol

Die - németül "a", angolul kocka, de igeként meghalni

Red - angolul piros, spanyolul "net"

Tea - angolul tea, spanyolul fáklya

Chat - angolul csevegés, francia macska

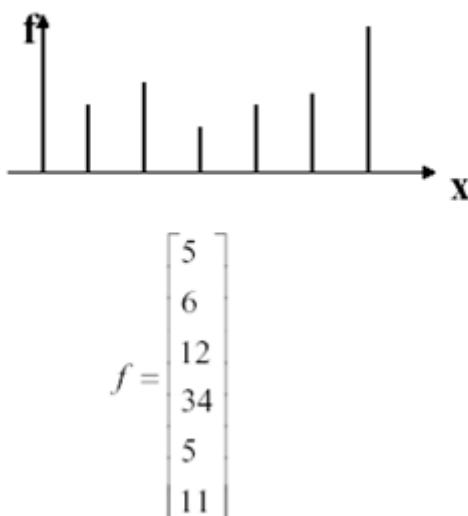
Hold - magyarul a Hold, angolul tartani

1.2. Történeti bevezető – kezdetek

- A digitális képfeldolgozás története a számítógépek fejlődéséhez igazodott
- Az első képfeldolgozáshoz elegendő teljesítménnyel rendelkező számítógép: 1960 (űrprogramok kezdetének ideje)
- 1964: úrból érkező képek fokozása számítógéppel
- Digitális képfeldolgozás ugyanakkortól az orvoslásban, a Föld megfigyelésében és a csillagászatban
- Computerized Tomography (CT) az egyik legfontosabb eredménye a képfeldolgozásnak

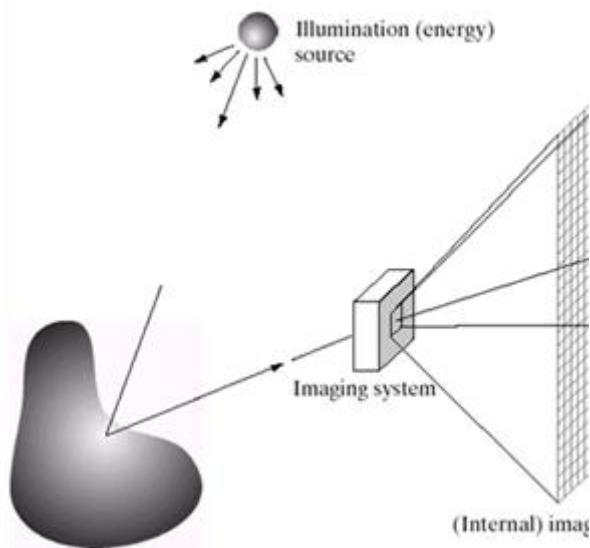
1.3. Mi a digitális kép fogalma?

- Kép (image): kétdimenziós $f(x, y)$ függvény, ahol az x és y koordináták; f amplitúdó az (x, y) koordinátákban az intenzitás vagy a szürkeségi szint
- Ha x , y és f diszkrét mennyiségek, akkor a képet digitálisnak mondjuk
- Mintavételezés és kvantálás eredménye 1D-ben és 2D-ben



1.4. Mintavételezés és kvantálás

- Mintavételezés (rácspontokban): folytonos képből diszkrét mennyiségek
- Kvantálás (intenzitások reprezentálása): amplitúdó nagyság diszkretizálása



1.5. Digitális kép

Digitalizált kép és intenzitás reprezentációja



1.6. Mi a képfeldolgozás?

Szűkebb értelmű megközelítés:

- A képfeldolgozás a jelfeldolgozás része, amely képekkel foglalkozik
- Célja: a kép minőségének javítása az ember, vagy további számítógépes feldolgozás számára
- Kép → Képfeldolgozás (képjavítás – image enhancement) → “Jobb” kép

Bővebb értelmű megközelítés:

- Segmentálás (részekre bontás), leírók kinyerése
- Osztályozás, analízálás, megértés

1.7. Vázlatos definíciók

Digitális képfeldolgozás (Digital image processing, DIP):

- digitális képek feldolgozása digitális számítógépekkel;

- képek fokozása, vagy más manipulálása, az eredmény általában másik kép (és valamelyen jellemzők)

Számítógépes látás, vagy röviden gépi látás (Computer Vision, CV):

- számítógép használata az emberi látás emulációjára, amely magába foglalja a tanulást, a következtetést és a reagálást (leírás, analízis, megértés)

A mesterséges intelligencia (Artificial Intelligence, AI) több részét használják a CV-ben, mint a DIP-ben

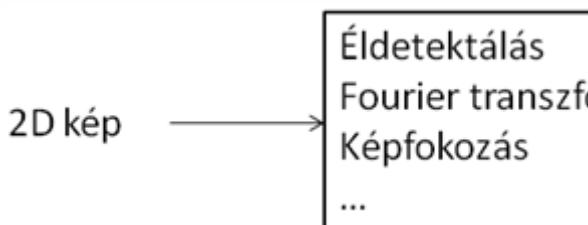
Képekkel foglalkozó más terület a Számítógépes grafika (Computer Graphics):

- képek készítése modellekiből

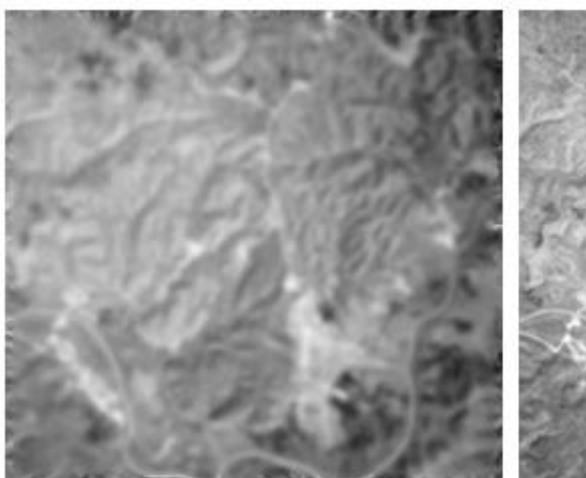
Bemenet/Kimenet	Kép	Leírás
Kép	Képfeldolgozás	Gépi látás
Leírás	Számítógépes grafika	Mesterséges intelligencia

1.8. Képfeldolgozás (Image Processing)

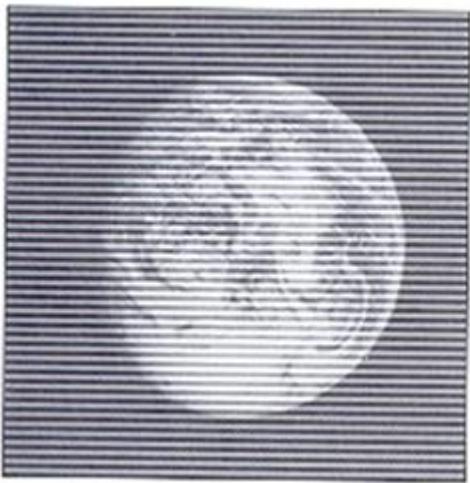
- Képfeldolgozás



- Képfokozás (Image Enhancement)

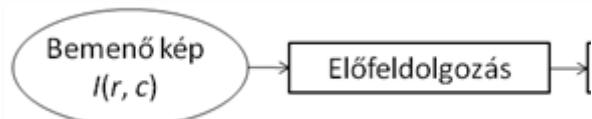


- Kép helyreállítás (Image Restoration) (pl. rosszul fókuszált képek korrekciója)
- Képre rakódott ismétlődő zaj eltávolítása

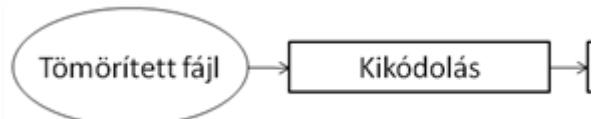


Képtömörítés (Image Compression)

- Tömörítés

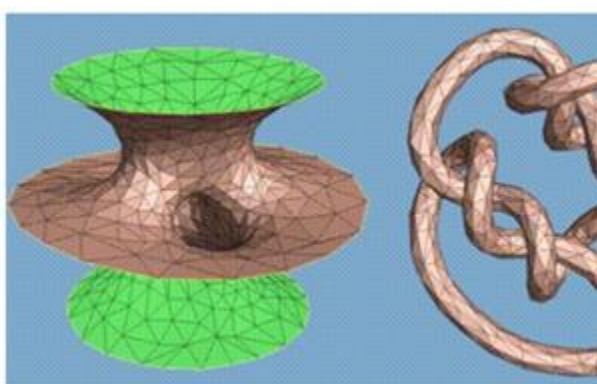


- „Kicsomagolás”



1.9. Számítógépes grafika (Computer Graphics)

Geometriai modellezés



1.10. A digitális képfeldolgozás szintjei

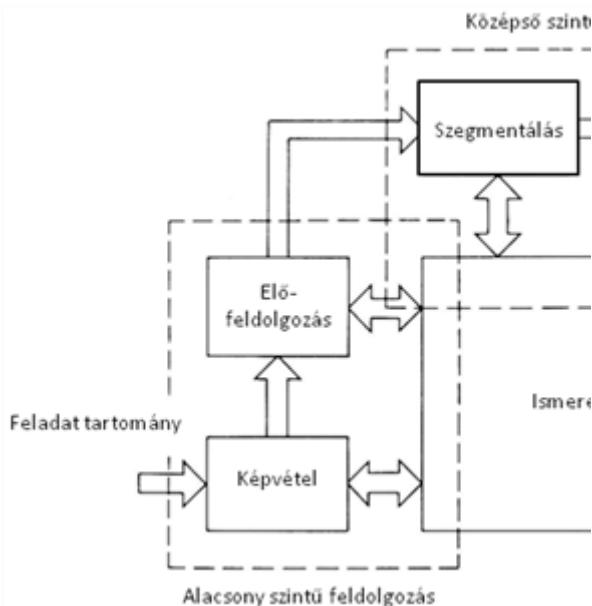
A képek számítógépes feldolgozását három szintre lehet osztani: alacsony, közép és magas szintű feladatok (low-level, intermediaite-level, high-level)

- Alacsony szint: minden az input minden az output kép
- Közép szint: az inputok általában képek, de az outputok a képekből nyert attribútumok (pl. egy objektum azonosítói a képen)
- Magas szint: a felismert objektumok együttesének érzékelése

1.11. A három feldolgozási szint

Alacsony szintű (low-level) feldolgozás

- Sztenderd eljárások alkalmazása a kép minőségének javítása érdekében – adatvezérelt, jellemzően előfeldolgozás (zajszűrés, élesítés, ...)



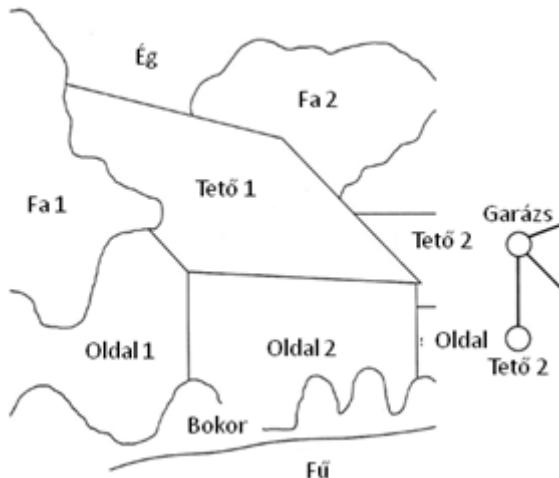
Középső szintű (intermediate-level) feldolgozás

- A kép komponenseinek kiemelése (szegmentálás) és azok jellemzése
- Bizonyos mértékű mesterséges intelligencia szükséges



Magas szintű (high-level) feldolgozás

- Felismerés és értelmezés (interpretáció)
- Mesterséges intelligencia módszerek szükségesek



1.12. Mi a gépi látás (Computer Vision)?

Olyan elméleti és algoritmikus alapok kifejlesztését jelenti, amelyek segítségével a 3D világról automatikusan nyerhető ki és analizálható hasznos információ - a világ 2D képének egyetlen vagy több példányát felhasználva

Emberi mozgások áttranszformálása avatarokra „motion capture” technikával

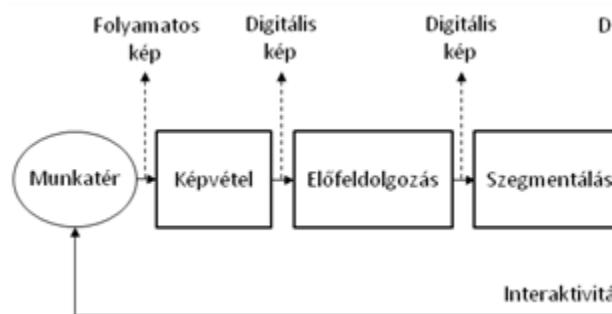


1.13. minden kép egy történet

A gépi látás célja, hogy olyan programot írunk, ami értelmezi a képet



1.14. Számítógépes látórendszer általános modellje



1.15. A számítógépes látás a következő területekre koncentrál

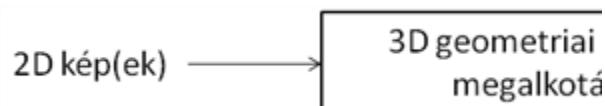
- Milyen információt kell kinyerni a vizuális szenzorokból?
- Hogyan történik a kinyerés?
- Hogyan kell a kinyert adatot reprezentálni?
- Hogyan kell az információt használni, annak érdekében, hogy a rendszer a feladatát ellássa?

Számítógépes látáshoz hasonló, rokon fogalmak, elnevezések:

- Képanalízis (Image Analysis)

- Jelenet analízis (Scene Analysis)
- Képmegértés (Image Understanding)

1.16. Számítógépes látás



1.17. Mintafelismerés (Pattern Recognition)

- Tradicionális terület (60-as évek óta kutatási terület)
- 2D képekből származó 2D objektumok felismerésével és osztályozásával foglalkozik
- Sok klasszikus megközelítés csak szűk területen működik (pl. nem alkalmazható 3D objektumokra)
- A legtöbb olyan kutatás innen származik, amely kiváltotta a számítógépes látás fejlődését
- Sok mintafelismerés területén kidolgozott elvet a számítógépes látás esetében is használnak

1.18. Mesterséges intelligencia (AI)

- Intelligens rendszerek tervezésével és az intelligencia tanulmányozásával foglalkozó terület
- Miután a képek feldolgozásával a jellemzőket kinyertük, a jelenet **szimbolikus reprezentációjával** analizálhatjuk azt
- Sok AI technika jelentős szerepet játszik a számítógépes látás területén is
- A számítógépes látás az AI egyik gyakorlati része

2. Miért bonyolult a számítógépes látás?

- A nagyszámú felület különböző anyagokból, textúrázottsággal, geometriai jellemzőkkel és sokszor inhomogén, vagy eltérő megvilágítási körülmények között rendkívül eltérő képekhez vezet
- A 3D világ 2D kép transzformáció rengeteg információt elveszít – az ún. inverz térképezésnek nincs egyértelmű megoldása
- Számítástechnikailag “intenzív” (összetett, számításigényes megoldások)

- A felismerés menetét még nem értjük pontosan
- A valós esetekben a vizsgált célobjektumok mellett rengeteg irreleváns, vagy zavaró objektum, illetve zaj nehezíti az értelmezést. Információvesztést okozhatnak az által, hogy kitakarják a célobjektumot, vagy annak egy részét

2.1. Felismerés nehézségei

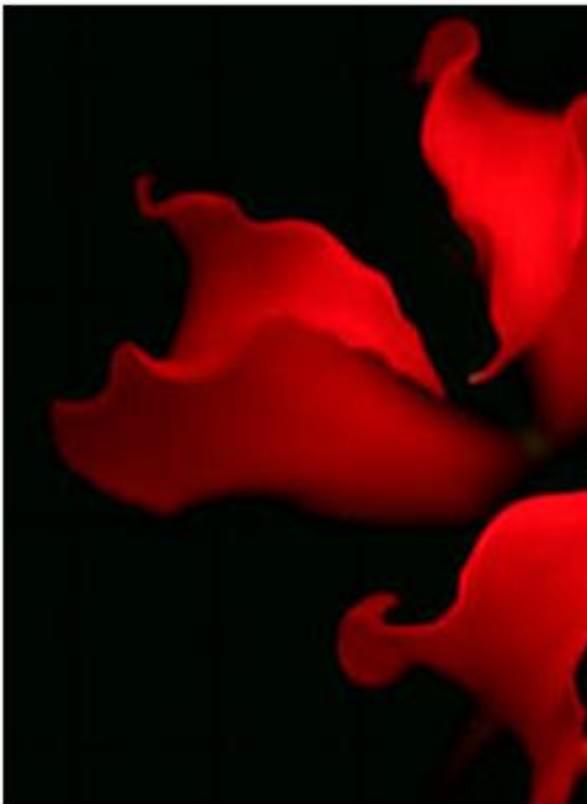
Jelenetek megértése, még komplex és rendezetlen kép esetében is egyszerű az ember számára



- Hogyan tudjuk megérteni, kivenni a valóságot, vagy a valóság képét?
- Mi a nyitja a képek megértésének?
- Milyen ismeretet használunk a képek megértéséhez?

2.2. A szín szerepe

- Mi az objektum?
- A színeknek van-e szerepe a felismerésben?
- Egyszerűbb-e felismerni a színeket különböző nézetekből?



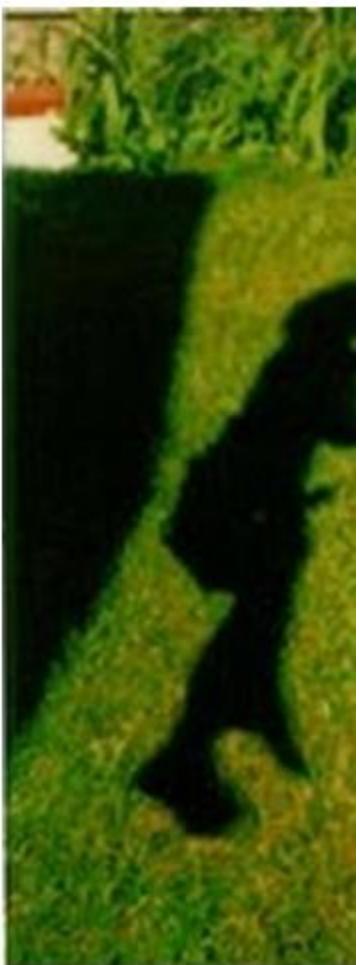
2.3. A textúra szerepe

Karakterisztikus képtextúrák segíthetnek az objektumok felismerésében



2.4. Az alak szerepe

Számos esetben a forma ad segítséget a jelenet megismeréséhez



2.5. A csoportosítás szerepe



2.6. Praktikus megfontolások

Vegyük figyelembe a jelenet körülményeit

- Gyűjtsünk minél több adatot (képet)
- Vegyük figyelembe a környező világ jellemzőit
- Számíthatóság és robosztusság

A számítógépes látórendszerknél, általában az iparban:

- A megvilágítási feltételeket mi szabályozzuk
- Az objektumot mi pozicionáljuk
- Az objektum jellemzőiben rejlő lehetőségeket használjuk ki



2.7. Gépi látás eléri-e, megelőzi-e az emberi látást?

Igen és nem (de általában nem!)

- emberek „összetett” dolgokban jobbak
- számítógép „egyszerű” dolgokban jobb



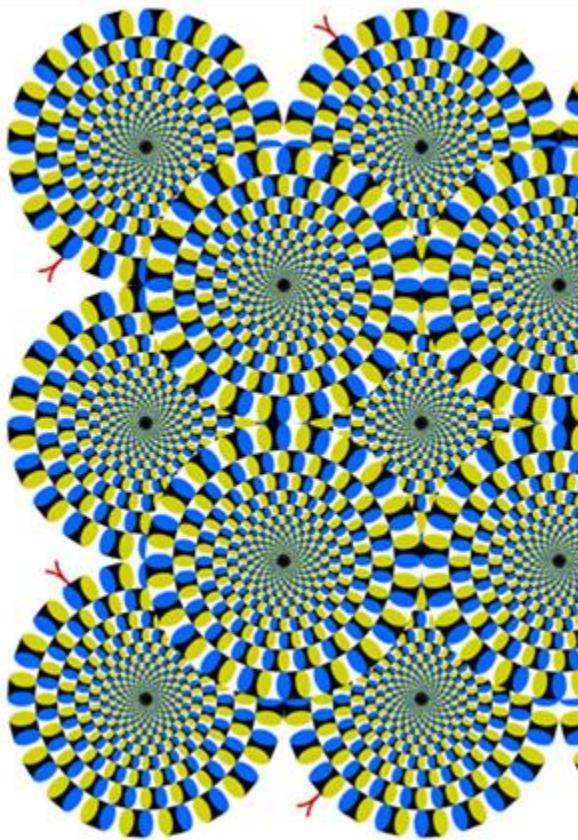
2.8. Emberi érzékelés korlátai

A számítógépes program nem látna különbséget (lásd az internetes hivatkozáson található cikket)



Sinha and Poggio, *Nature*, 1996

Illúzió: mozgónak látjuk a képrészeket



3. Hol tart a képfeldolgozás és a gépi látás?

3.1. Hol tart ma a gépi látás?

A következő diákok bemutatják, hogy a gépi látó rendszerek milyen problémákat képesek megoldani.

3.2. Föld megjelenítők (3D modell)



Microsoft: Virtual Earth (vagy: Google Earth)

3.3. Fotószintézis



Photo Tourism technology

3.4. Optikai karakterfelismerés (OCR)

- Digitális dokumentumok (szkennelt, fényképezett) szöveggé alakítása
- Manapság minden szkennerhez gyárilag adnak OCR programot



Számjegyek felismerése, AT&T labs
<http://yann.lecun.com/>

- Rendszámfelismerők - Automatic number plate recognition

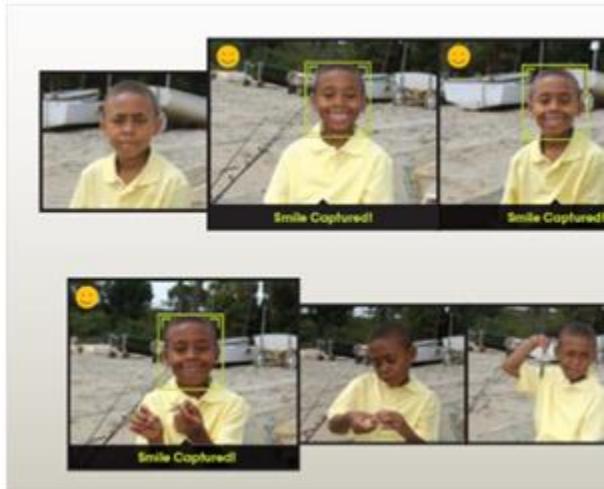
3.5. Arcdetektálás

Több digitális fényképezőgép esetén

- Canon, Sony, Fuji, ...



3.6. Mosoly detektálás



Sony Cyber-shot® T70 Digital Still Camera

3.7. Arcfelismerés



Kicsoda ő?

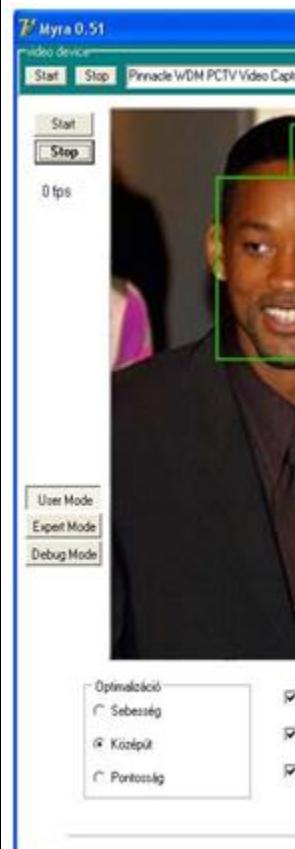
3.8. Biometria



“How the Afghan Girl was Identified by Her Iris Patterns” story



3.9. Biometrikus azonosítás



Ujjlenyomat szkennerek

Arcfelismerő
<http://www.sensiblevision.com/>

rendszerök

3.10. Objektum felismerés (mobil telefonokban)

- Microsoft Research
- Point & Find



3.11. Speciális effektusok



Matrix, ESC Entertainment

3.12. Speciális effektusok: motion capture technika



A Karib tenger kalózai - Click here for interactive demo

3.13. Sport



www.howstuffworks.com

3.14. Okos autók

Mobileye

- Látórendszer: BMW, GM, Volvo
- 2010 után: gyártók 70%-a



3.15. Google autó

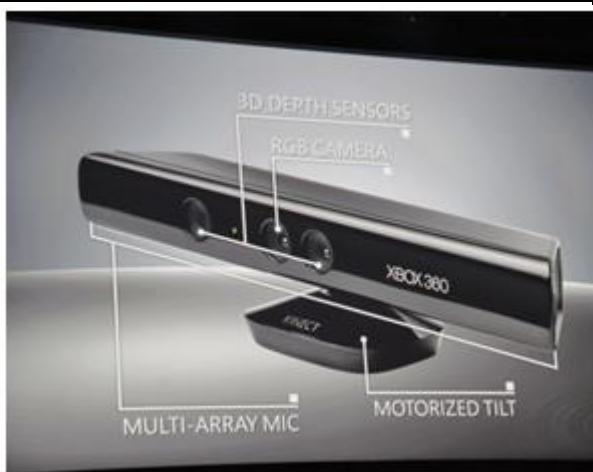


3.16. Látás alapú interaktivitás



Nintendo Wii infra-szenzor követés Lee munkája
CMU

Digimask: 3D avatar felhasználó arcával



Kinect szenzor

“Game turns moviegoers into Human Joysticks”,
CNET

3.17. Úralkalmazás



NASA'S Mars Exploration Rover Spirit 2007.

Látó rendszer feladatai (JPL)

- Panoramaképek összeillesztés (Panorama stitching)

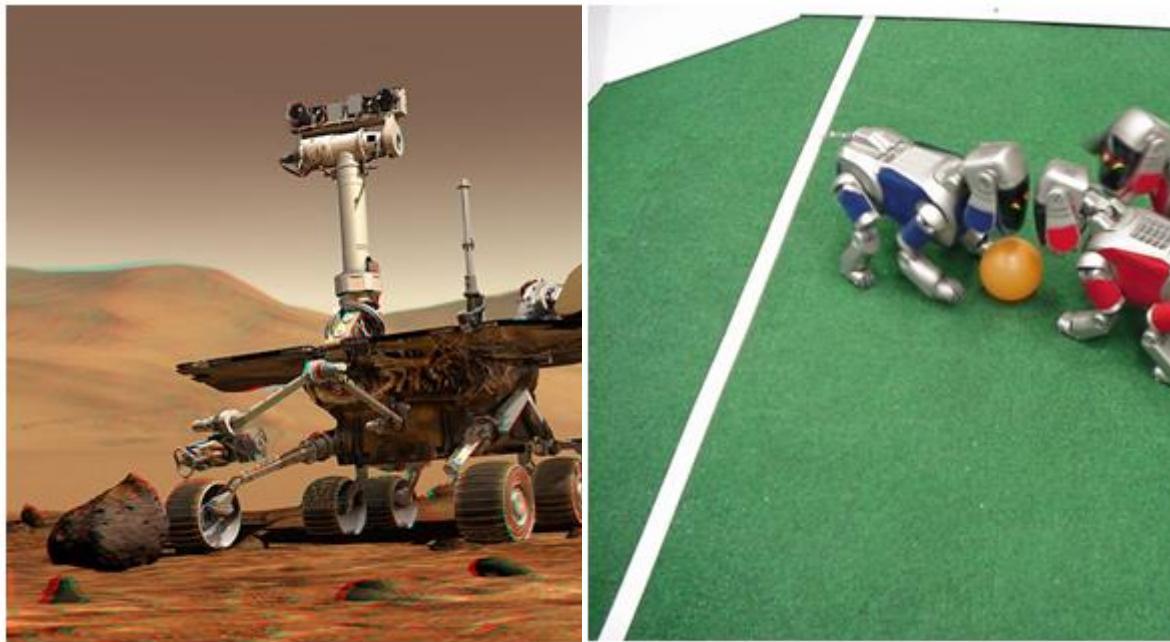
- 3D terepmodellezés
- Akadály detektálás, helyzet követés
- Computer Vision on Mars - Matthies et al.

3.18. Robotlátás (Robot Vision)

A számítógépes látás alkalmazása robotikában

Néhány fontos alkalmazás:

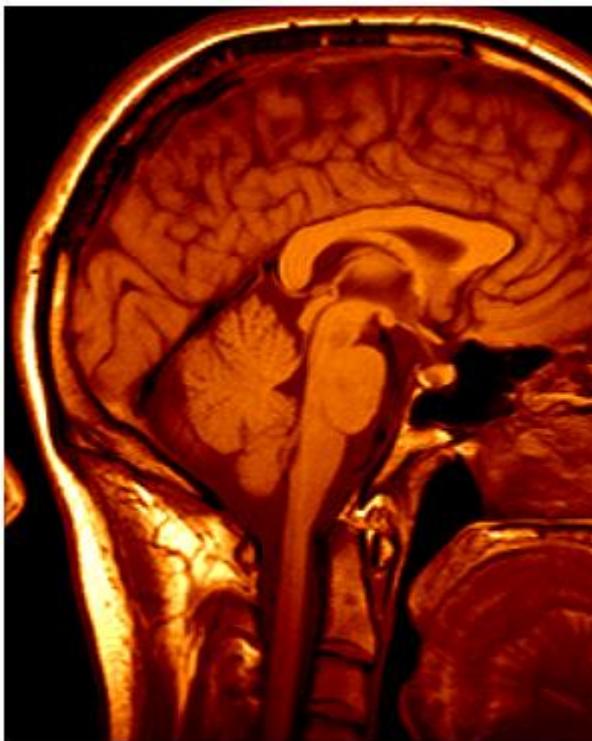
- Autonóm robotnavigáció (Autonomous robot navigation)
- Számítógépes felügyelet és összeszerelés (Inspection and assembly)



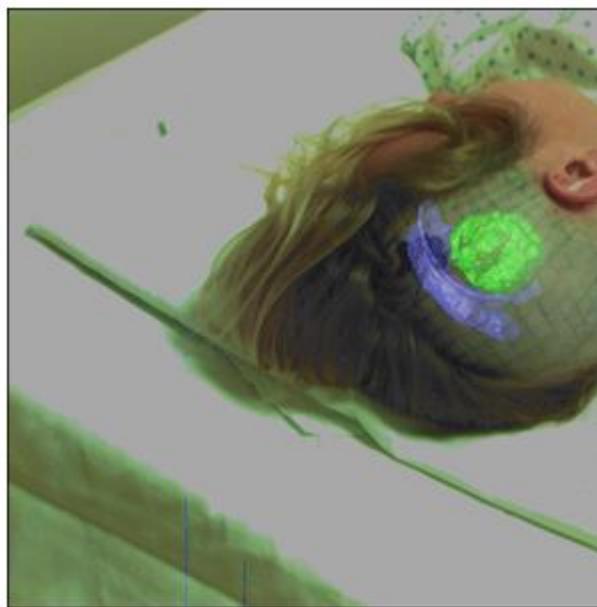
NASA' Mars Spirit Rover
http://en.wikipedia.org/wiki/Spirit_rover

<http://www.robocup.org/>

3.19. Orvosi alkalmazás



3D képalkotás - MRI, CT



Képvezérelt sebészet Grimson et al., MIT

3.20. „State of the art”

Utóbbi 5 évben jelentős változás

Érdemi gyűjtemény:

- David Lowe gépi látással foglalkozó lapja
<http://www.cs.ubc.ca/spider/lowe/vision.html>
- Computer Vision Online
<http://www.computervisiononline.com>

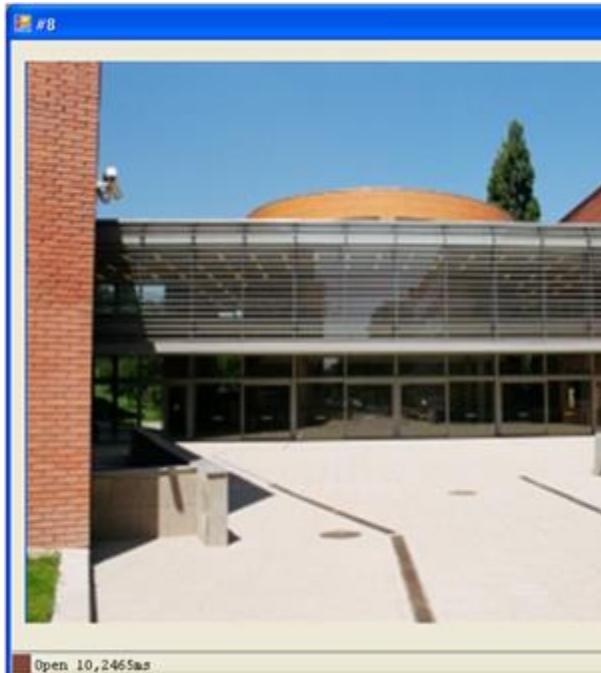
Felhasznált és javasolt irodalom

- [1] S. Seitz, R. Szeliski, *Computer Vision (CSE 576)*, University Washington, 2012.
- [2] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, ISBN: 978-1-84882-934-3 2011.
- [3] E. Trucco, A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, ISBN: 0-13-261108-2 1998.
- [4] R. C. Gonzales, R. E. Woods, *Digital Image Processing*, Pearson Education, Inc., 3rd ed., ISBN-13: 978-0-13-505267-9 2008.
- [5] D. H. Ballard, C. M. Brown, *Computer Vision*, Prentice Hall, ISBN: 0-13-155316-4 1982.

3. fejezet - Adatstruktúrák a képfeldolgozásban

Vámossy Zoltán

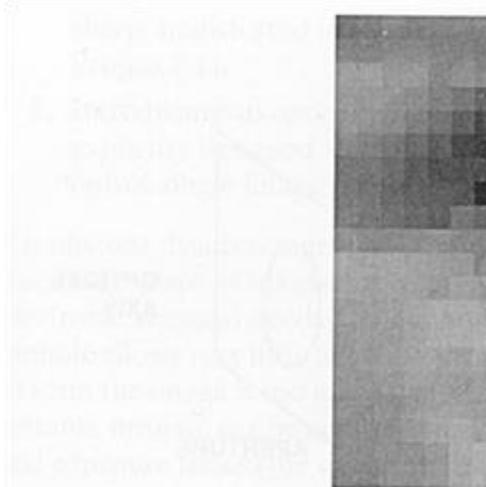
Ezt a fejezetet főleg Wilkinson és Allen párhuzamos programozás könyvének [1] képfeldolgozás része és Bradskiék prezentációja [2] alapján dolgoztuk fel. Az integrál képnek például [3]-ben lehet utána olvasni.



1. Kép

1.1. Képek ábrázolása

Az intenzitás képeket sokszor tömbként kezeljük

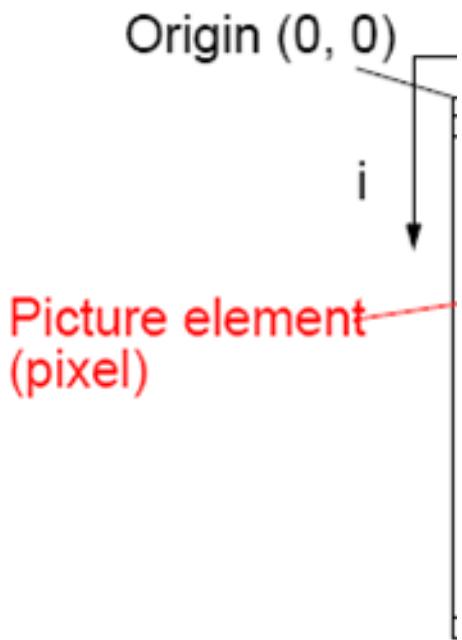


```
117 125 133 127 130 130 133 121 1  
134 133 138 138 132 134 130 133 1  
146 147 138 140 125 134 124 115 1  
144 141 136 130 120 108 88 74  
139 136 129 119 102 85 58 31  
132 127 117 102 87 57 49 77  
124 120 108 94 72 74 72 31  
125 115 102 93 88 82 42 79 1  
124 116 109 99 91 113 99 140 1  
136 133 133 135 138 133 132 144 1  
158 157 157 154 149 145 133 127 1  
155 154 156 155 146 155 154 154 1  
150 151 154 162 166 167 169 174 1  
145 149 151 157 165 169 173 179 1  
144 148 153 160 159 158 165 172 1  
144 141 147 155 154 149 156 151 1  
139 140 140 150 153 151 150 146 1  
136 134 138 146 156 164 153 146 1  
136 133 136 135 144 159 168 159 1  
133 129 140 142 146 159 167 165 1
```

1.2. Digitális kép

Alacsonyszintű képfeldolgozás

- A tárolt képen végez műveleteket, hogy javítsa/módosítsa azt
- A kép képelemek (pixel = picture element) kétdimenziós tömbje



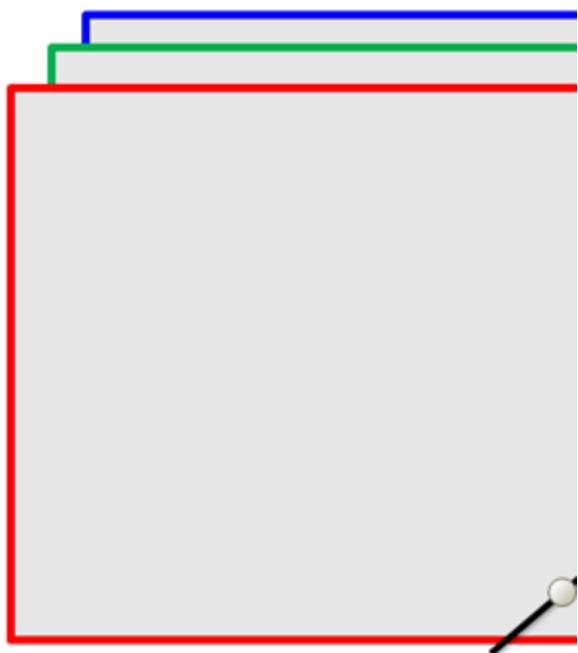
Számos alacsonyszintű képművelet az intenzitásokat (szürkeségi értékeket) manipulálja

1.3. Számítási igény

- Tételezzünk fel egy 1024×1024 pixelből álló képet, ahol az intenzitást 8-biten tároljuk.
- Tárolási igény 2^{20} byte (1 Mbytes)
- Tegyük fel, hogy minden pixelen csak egy műveletet végzünk, ekkor 2^{20} operációt kell végeznünk a képen. Ez kb. 10^{-8} mp/művelet, ami hozzávetőlegesen 10 ms-ot igényel.
- Valós idejű képfeldolgozás esetén tipikusan 25-30 képet kell másodpercenként feldolgozni (fps).
- Tipikusan nem csak egyetlen műveletet kell pixelenként elvégezni, hanem több és összetettebb funkciókat.

1.4. Színes képek

- A színes képeket sokszor a három alapszínből kikevertként modellezünk. Az egyes színrétegek önmagukban intenzitás képekhez hasonlóak
- RGB képek esetén az egyes rétegek lehetséges intenzitás értékei ugyanabba a tartományba esnek



1.5. Hisztogram

Hisztogram:

- Az egyes intenzitásokból hány darab van a képen

Halmozott hisztogram:

- Az adott intenzitásnál nem nagyobb képpontok száma



Normalizált hisztogram:

- A hisztogram elemeit elosztjuk a képpontok számával

Hisztogram készítés:

```
for(i = 0; i < height_max; x++)
    for(j = 0; j < width_max; y++)
        hist[p[i][j]] = hist[p[i][j]] + 1;
```

- A pixeleket a p[][] tömb tárolja és a hist[k] vektor megmondja, hogy a k-ik intenzitásból hány darab van a képen
- Egyszerű összegző tömb
- A számítás könnyen párhuzamosítható adatdekompozícióval

1.6. Integrál kép

Téglalap alakú részben az intenzitások összege

- Gyorsan számolható
- Az (x, y) pontban az érték



Az integrál kép két lépésben számolható:

$$s(x, y) = s(x, y - 1) + i(x, y)$$

$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

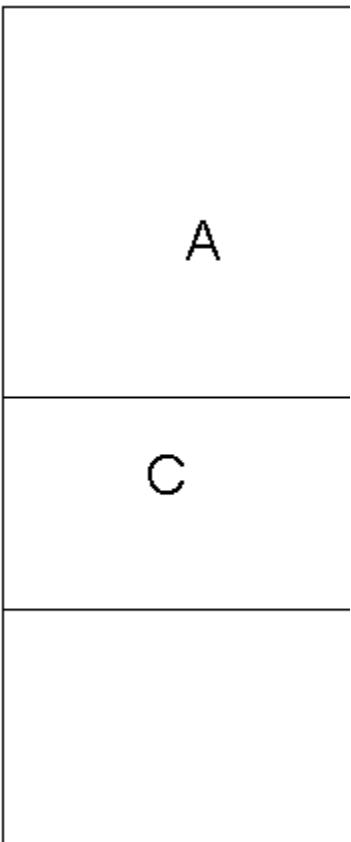
ahol $s(x, y)$: halmozott oszlopösszeg, és

$$s(x, -1) = 0$$

$$ii(-1, y) = 0$$

Egyszer kell végigmenni a képen

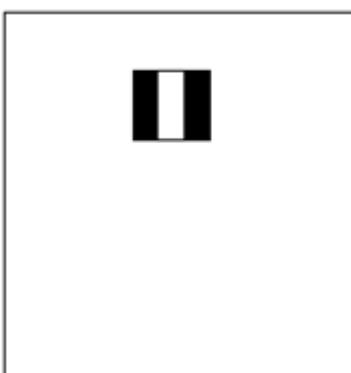
1.7. Adott téglalapra a számítás



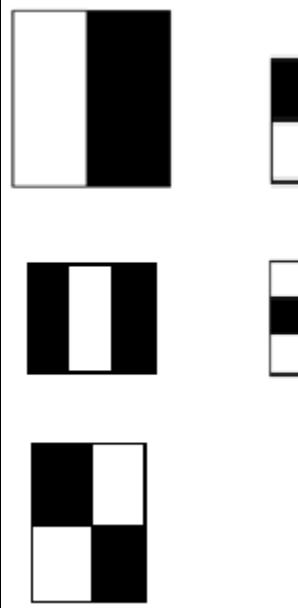
D téglalap számítása: $ii(4) - ii(3) - ii(2) + ii(1)$

1.8. Haar-szerű jellemzők számítása integrál képből

- Haar-szerű jellemzők: (Képpont intenzitások összege a fekete területen) - (Képpont intenzitások összege a fehér területen)
- Egyszerű jellemzők → gyors számítás
- Hogyan? A maszkokat végigfuttatjuk a képen, és számoljuk a jellemzőket



- Téglalap számítása: négy tömbhivatkozás
- Két téglalap: hat tömbhivatkozás
- Három téglalap: 8 tömbhivatkozás
- Négy téglalap: 9 tömbhivatkozás



1.9. Jellemzők felhasználása

- Durva, de érzékeny
- Hatékony

Viola és Jones valós idejű arcdetektáló rendszere:

- Robusztus, valós idejű arcdetektáló
- Részablakokban keres arcot



- Más felhasználások: képtömörítés, jellemző pontok meghatározása (SURF)

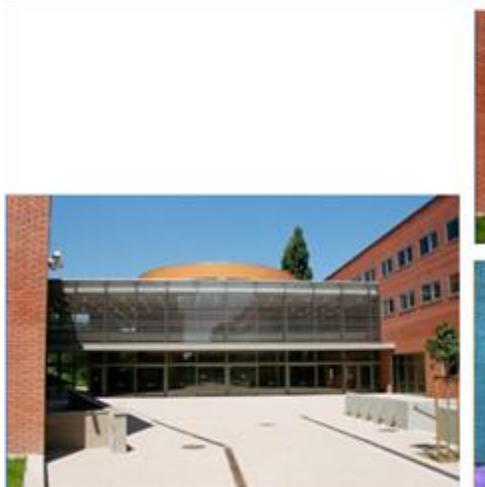
Felhasznált és javasolt irodalom

- [1] , B. Wilkinson, M. Allen: *Parallel Programming, Techniques and Applications Using Networked Workstations and Parallel Computers*, Pearson Education, Inc., 2nd ed. ISBN: 0-13-140563-2, p. 467. 2005.
- [2] S. Thrun, G. Bradski, D. Russakoff:, *Computer Vision, CS223B*, Stanford University, <http://robots.stanford.edu/cs223b>
- [3] E. Trucco, A. Verri:, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, ISBN: 0-13-261108-2, p. 343. 1998.
- [4] R. Szeliski:, *Computer Vision: Algorithms and Applications*, Springer, ISBN: 978-1-84882-934-3, p. 812. 2011.

4. fejezet - Intenzitás transzformációk

Vámossy Zoltán

A fejezet nagyrészt Gonzales-Woods [1] általánosan használt könyve alapján került feldolgozásra. A hibadiffúziós algoritmus bemutatásánál Akhter-Roberts [2] Multi-Core Programming könyvét használtuk forrásként.



1. Pont alapú műveletek

1.1. Képműveletek osztályozása

Képtérben történő műveletek

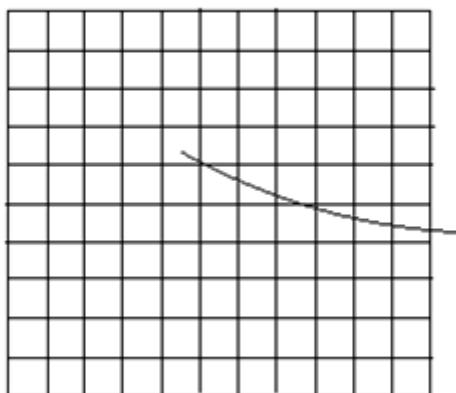
- Pont alapú műveletek
- Teljes képre vonatkozó transzformációk
- Geometriai transzformációk
- Maszk, vagy ablak alapú transzformációk

Frekvencia tartományban végzett műveletek

- Diszkrét Fourier transzformáció

1.2. Pont alapú műveletek

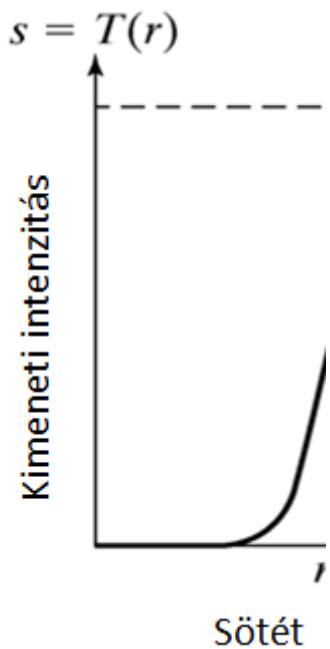
- $g(x,y) = T[f(x,y)]$, T egy pixelen operál: $s = T(r)$, ahol r a forrás, s a cél intenzitása
- Legegyszerűbb, elemi képfeldolgozási műveletek
- Intenzitás transzformáció, ami egy régi értéket egy újra cserél valamilyen függvény alapján
- Általában a kontraszt fokozására használják
- Mivel az adott pixel transzformációja független másik képponttól, nyilvánvaló az adatpárhuzamos megvalósítás lehetősége

Bemeneti kép

1.3. Pont alapú: identitás, vagy egység

$$g(x,y) = f(x,y)$$

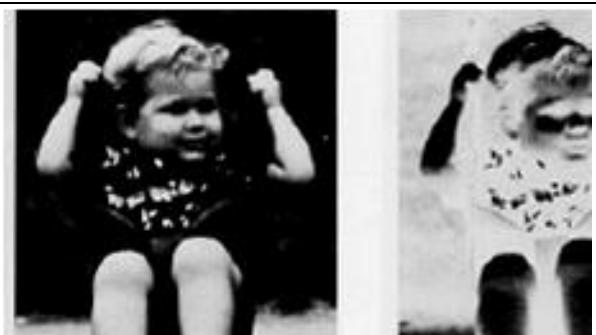
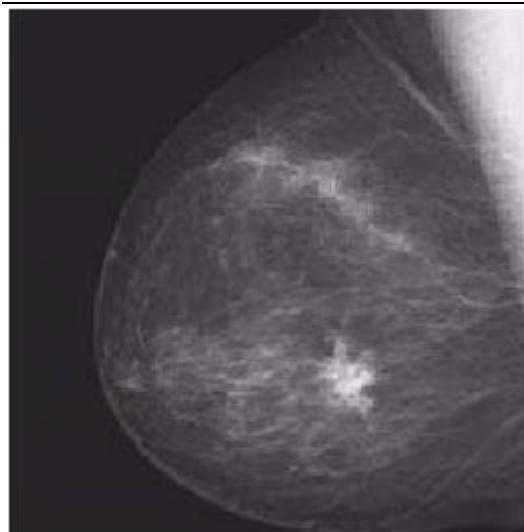
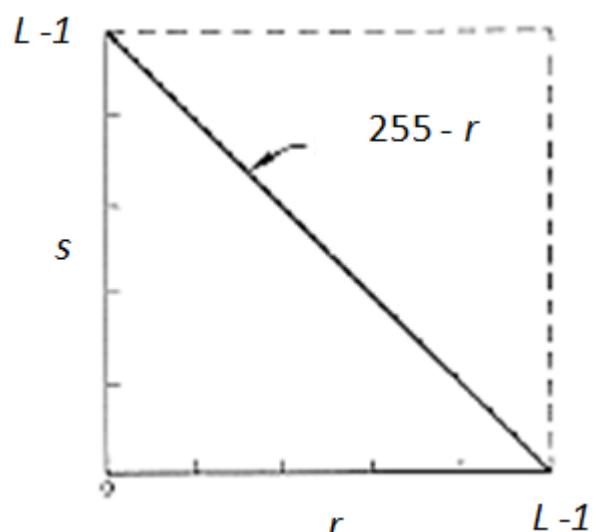
A bemenő $r = f(x,y)$ intenzitást nem változtatja meg a T transzformáció, azaz a kimeneti intenzitás $s = g(x,y) = r$



1.4. Pont alapú: negálás

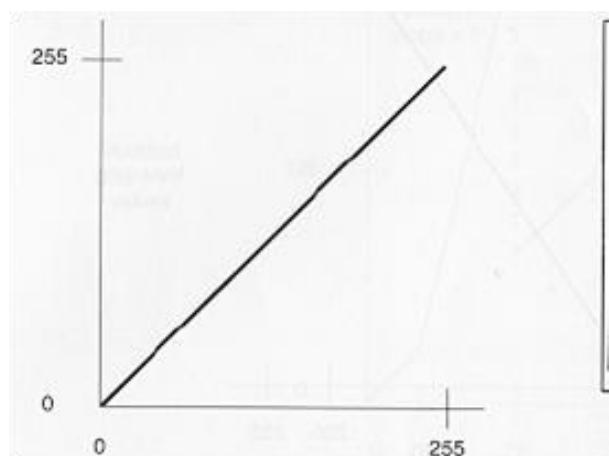
$$g(x, y) = 255 - f(x, y)$$

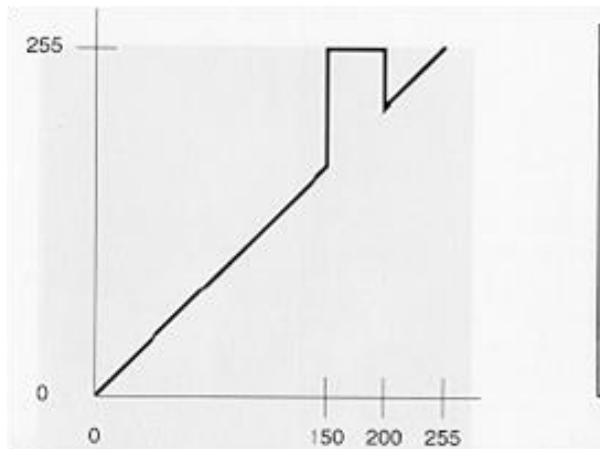
Sötét háttérben szürke, vagy fehér elemek kiemelésére



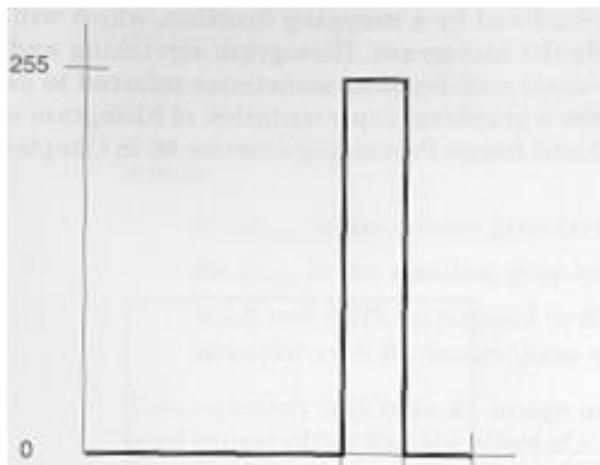
1.5. Pont alapú: intenzitás szintre vágás

Az intenzitás értékek egy tartományát emeli ki (Eredeti kép és alatta a kiemelt)



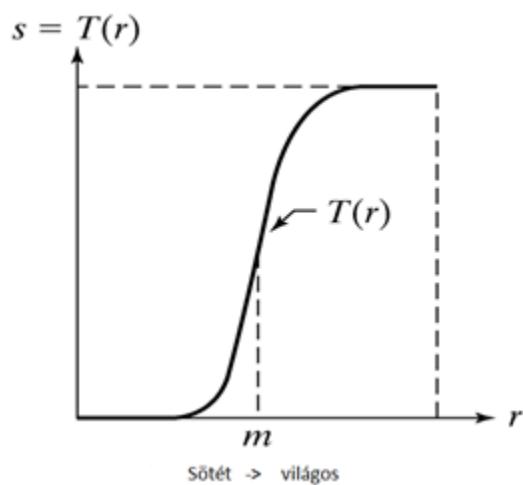


Dupla küszöbölés



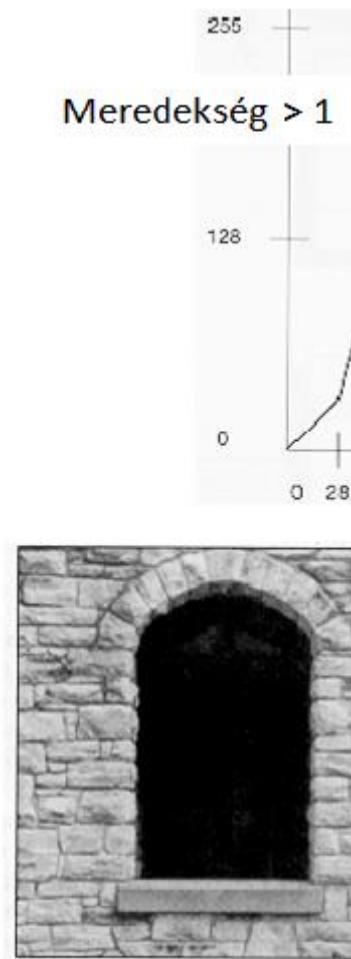
1.6. Pont alapú: kontraszt nyújtás

Nagyobb kontraszt érdekében: sötétítés m alatt, világosítás m fölött



1.7. Pont alapú: kontraszt növelés/csökkentés

Szürkeségi értékek nyújtása úgy, hogy több információt lássunk



1.8. Pont alapú műveletek

Kontraszt nyújtás:

- Az intenzitás értékek tartományát széthúzzuk, hogy a részletek jobban érzékelhetőek legyenek.
- Adott egy pixel intenzitása x_i az $[x_l \dots x_h]$ tartományból, a kontraszt nyújtás során az $[x_L \dots x_H]$ tartományba transzformáljuk az intenzitást

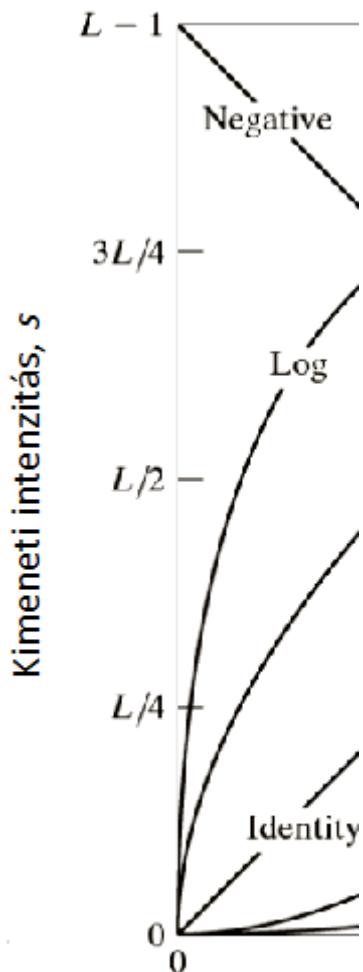
$$x_I = \frac{(x_H - x_L)}{(x_h - x_l)} x_i + x_L$$

Szürkeségi szint csökkentés:

A kevésbé szignifikáns biteket elhagyjuk.

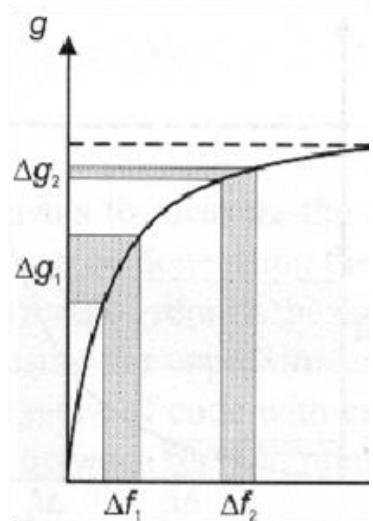
1.9. Pont alapú: nem lineáris transzformációk

Logaritmus, hatvány és gyök függvényeket is alkalmazhatunk transzformálásra



Tetszőleges függvényt használhatunk, ami $1 \rightarrow 1$, vagy sok $\rightarrow 1$ leképzést hajt végre

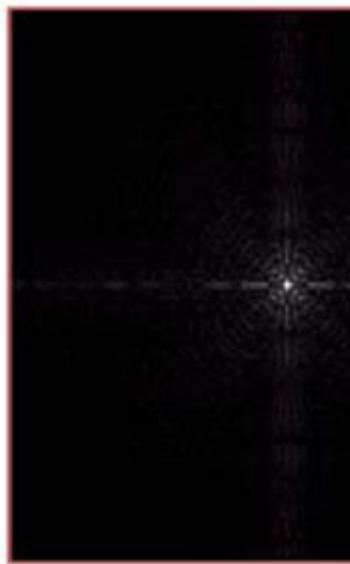
Logaritmikus skálázás: sötét régiókat jobban széthúzza, világosakat tömöríti $g(x, y) = c * \log [1.0 + f(x, y)]$, ahol c konstans



Az eredményt gyakran normalizálni kell

1.10. Logaritmikus skálázás példa

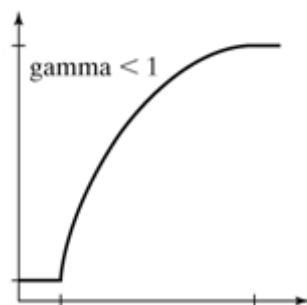
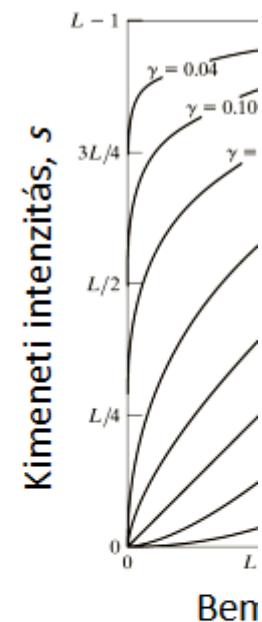
- Fourier transzformáció esetében
- Széles intenzitás spektrumot le lehet fedni



1.11. Hatvány és gyök függvények

$s = c * r^\gamma$, ahol c és γ pozitív konstans

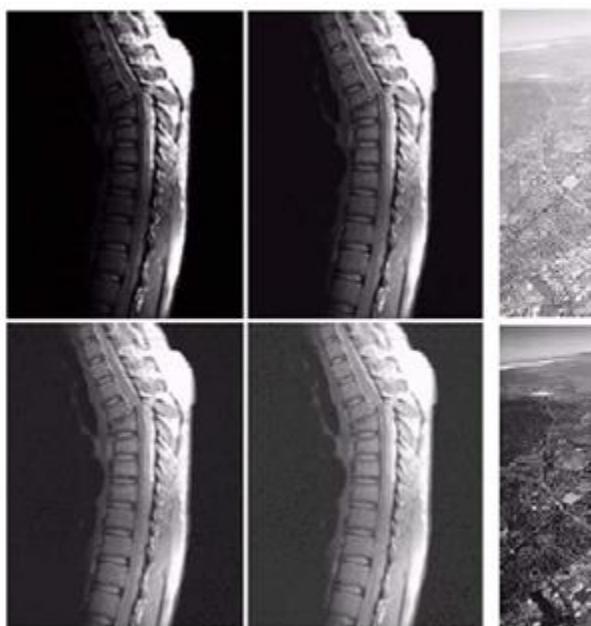
Gamma értékétől függően vagy a sötét értékeket transzformálja világosabbá, vagy fordítva



1.12. Gamma korrekció

Gamma különböző értékeinél ugyanaz az MRI kép ($\gamma < 1, 0.6, 0.4, 0.3$), illetve ($\gamma > 1, 3, 4, 5$)

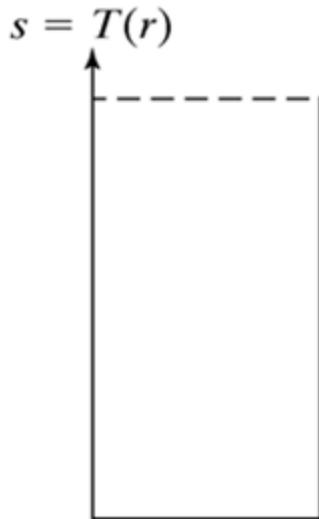
Különböző megjelenítőkön a színhelyes megjelenítést állítja be.



1.13. Pont alapú műveletek: küszöbölés

- Egy bizonyos határnál (threshold) nagyobb intenzitású képpont értékéhez az egyik szélső értéket, a többi képpont értékéhez pedig a másik szélsőértéket rendeljük hozzá
- Ha egy pixel intenzitása r , akkor minden pixelre:

if ($r < \text{threshold}$) $s = 0$; else $s = 255$;



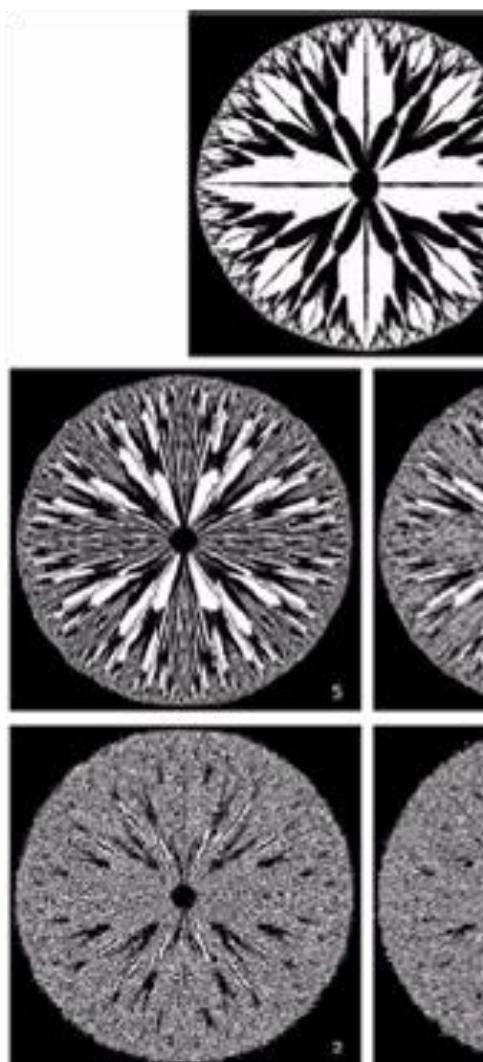
1.14. Bit-síkonkénti vágás

A bitsíkok közül a magasabbak tartalmazzák a lényegi információ javarészét

A szteganográfia alapja: alacsonyabb bitekben idegen adatot tárolunk



1.15. Bit-síkonkénti vágás (példa)



1.16. Hibadiffúziós algoritmus (Floyd 1975)

- Folytonos tónusú kép megjelenítése korlátozott színtartományú eszközön (pl. 8 bites szürke kép fekete-fehér nyomtatón)
- Közelítő technikával lehet szimulálni az árnyalatokat

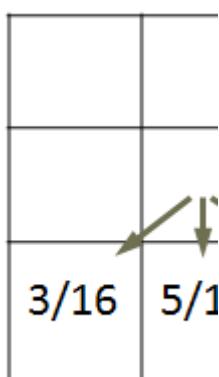
- Nyomtatók esetében „dithering”, vagy „halftone” technikának nevezik



- Bal oldali 8 bit felbontású, jobb oldali 2 bit felbontású kép

1.17. Algoritmus

1. Az aktuális képpont értéke alapján az output meghatározása: ha $[0, 127] \Rightarrow 0$; ha $[128, 255] \Rightarrow 1$
2. Hiba meghatározás: pl. ha az input 168 volt, akkor az output: 1, a hiba pedig az 1 reprezentációjának megfelelő 255 és az eredeti érték különbsége, azaz $168 - 255 = -87$
3. A hibaérték terjesztése a szomszédos képpontokra:



Pszeudódókód:

```

1 for each y fentről lefelé
2 for each x balról jobbra
3   korábbi_érték := pixel[x][y]
4   új_érték := A_legközelebbi_érték_a_cél_színskáláján (korábbi_érték)
5   pixel[x][y] := új_érték
6   hiba := korábbi_érték - új_érték

```

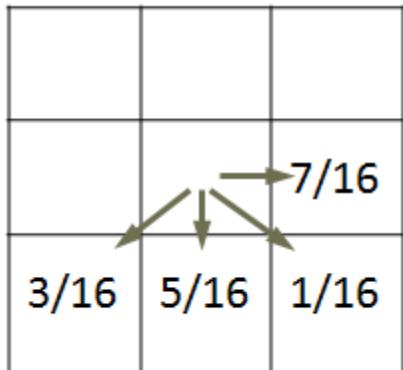
```

7     pixel[x+1][y] := pixel[x+1][y] + 7/16 * hiba
8     pixel[x-1][y+1] := pixel[x-1][y+1] + 3/16 * hiba
9     pixel[x][y+1] := pixel[x][y+1] + 5/16 * hiba
10    pixel[x+1][y+1] := pixel[x+1][y+1] + 1/16 * hiba

```

1.18. Párhuzamosított hibadiffúzió

Fordított jellegű megfogalmazás, mivel az előző pixelek intenzitásából generált hibát (a, b, c, d) kell ismerni:



A hibaterjesztéshez három elem kell az előző sorból és egy a megelőző oszlopból => két képpontnyi késleltetés (szinkronizáció)!



2. Hisztogram transzformációk

2.1. Hisztogram műveletek

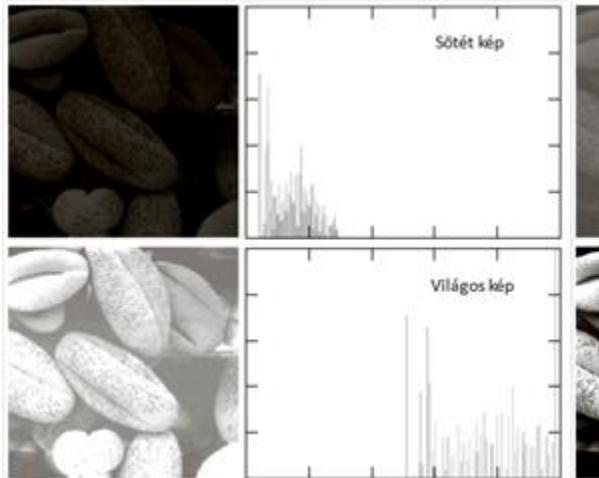
Alacsony kontrasztú képek, vagy túl sötétek, vagy túl világosak, vagy csak egy szűk intenzitás tartományban jellemzőek a szürkeségi értékek

Magas kontrasztú képek nagy sötét és nagy világos foltokat tartalmaznak

Jó kontrasztú képek

- Intenzitások széles tartománya
- Általában nincs domináns intenzitás érték
- A hisztogram – az intenzitások gyakorisága – a kép kontrasztosságáról ad információt

- Halmozott hisztogram: az adott szintig az intenzitások hány százalékát tartalmazza a kép
- A hisztogram kiszámítása adatpárhuzamos technikával – a kép különböző részleteire elkészítve, majd ezek összegzésével – megvalósítható
- A hisztogram ismeretében a következő műveletek szintén párhuzamosíthatóak



2.2. Hisztogram széthúzás

Lineáris skálázás

- A képen előforduló $[min, max]$ intervallumot lineárisan skálázza a $[0, L - 1]$ teljes intervallumra
- Tudjuk, hogy

$$0 = a * min + b \text{ és } L - 1 = a * max + b$$

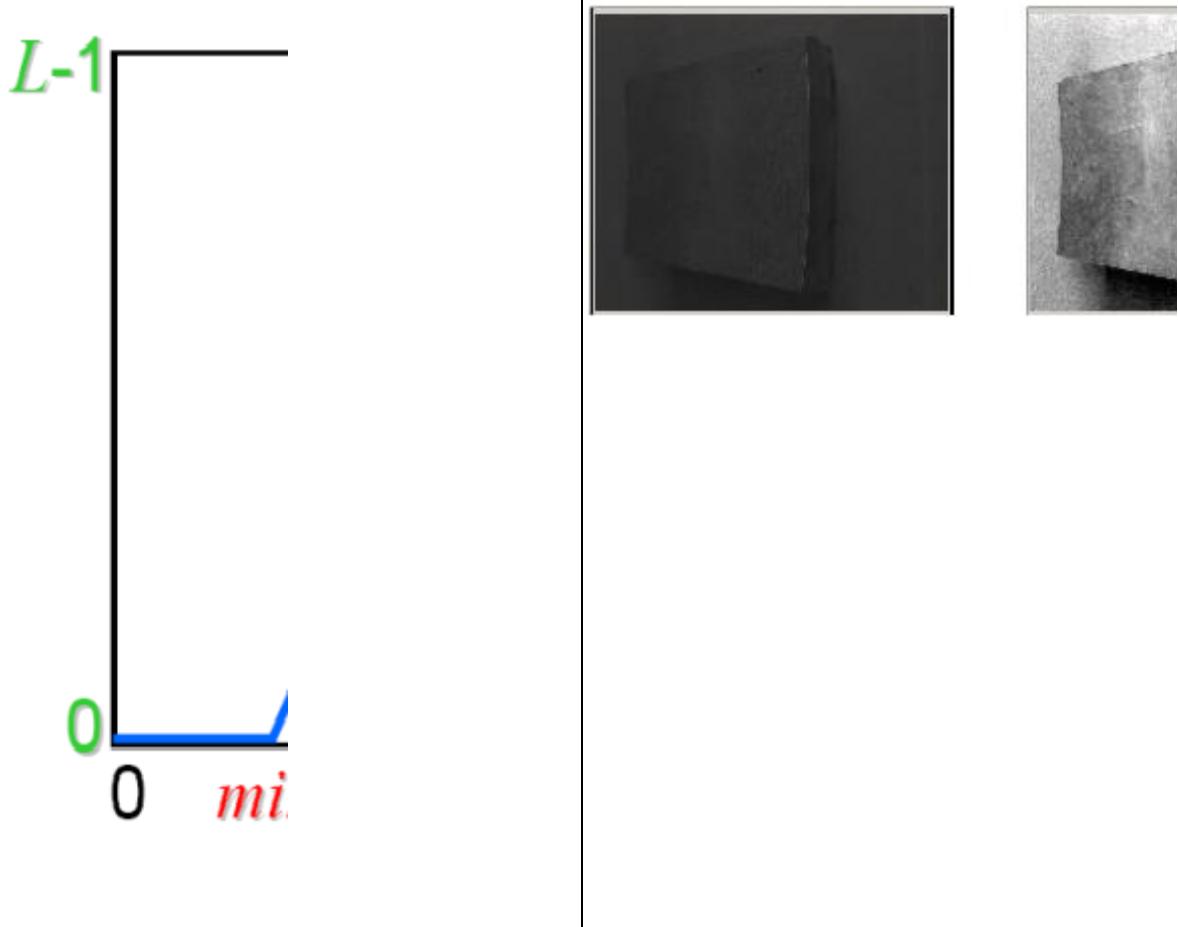
- Ezért

$$a = (L - 1) / (max - min)$$

$$b = -min * a = -min * (L - 1) / (max - min)$$

- Végül

$$s = T(r) = (L - 1)(r - min) / (max - min)$$



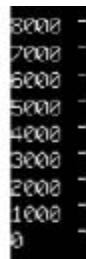
2.3. Kontraszt széthúzás

Lineáris skálázás, hasonlít a hisztorogram széthúzáshoz, de

- a képen előforduló $[min, max]$ intervallum helyett, egy $[low, high]$ intervallumot skáláz a $[0, L - 1]$ teljes intervallumra

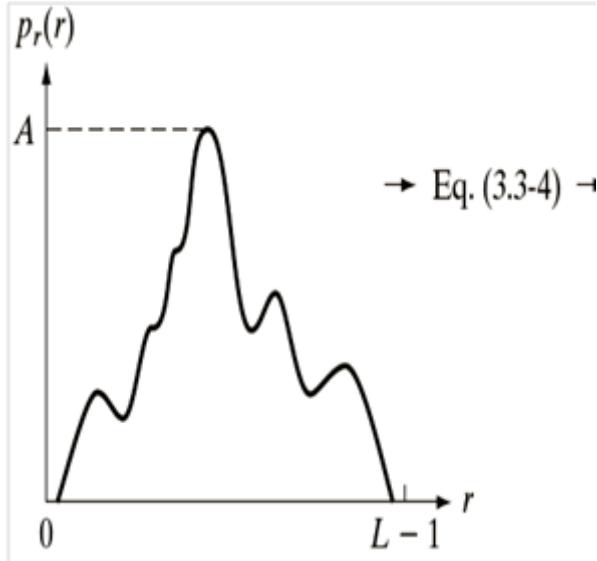


2.4. Példa: hisztogram és kontraszt széthúzás



2.5. Hisztogram kiegyenlítés

- A hisztogram kiegyenlítés olyan transzformáció, amely az intenzitás szinteket úgy transzformálja, hogy az egyenletes legyen
- A normalizált hisztogram értéke minden bemenetre $1/(L - 1)$ értéket vegyen fel
- Gyakorlatban ez csak közelíthető, mivel diszkrét értékeink vannak



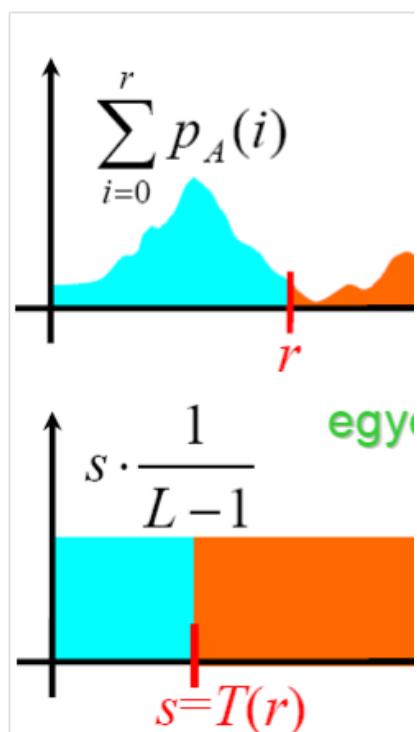
Eredeti kép és célkép normalizált hisztogramja

Cél:

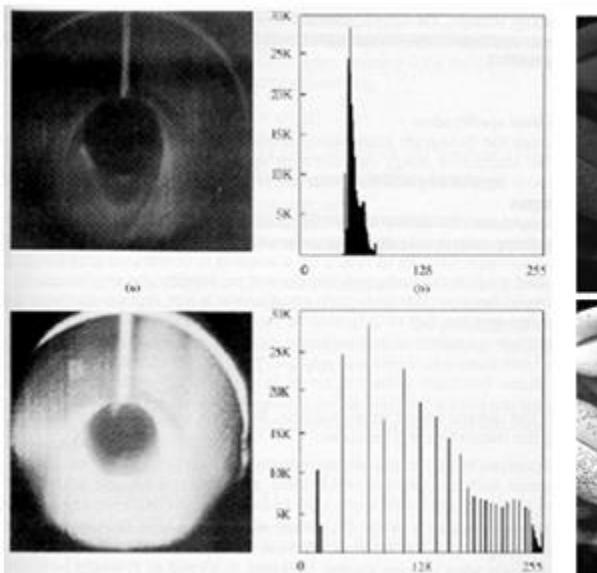
- Az új kép normalizált hisztogramja egyenletes legyen
- Ehhez keressük az $s = T(r)$ transzformációt:

$$s \cdot \frac{1}{L-1} = \sum_{i=0}^r p_A(i)$$

$$s = T(r) = (L-1) \cdot \sum_{i=0}^r p_A(i)$$



2.6. Hisztogram kiegyenlítés példa



3. Teljes képes és geometriai transzformációk

3.1. Teljes képre vonatkozó transzformációk

- Az eredmény pixelérték olyan műveleten alapszik, amely két vagy több képet használ
 - Általában minden output pixel ugyanabban a helyzetben marad
 - Az adatpárhuzamosítás lehetősége nyilvánvaló
- Összeadás (súlyozott)
- Két képen lévő információ kombinálásakor hasznos
 - $O(r, c) = a * I_1(r, c) + (1-a) * I_2(r, c)$



3.2. Teljes képre: átlagolás

- Képminőség javítható több kép átlagával
- $$O(r, c) = I(r, c) + n(r, c),$$
- $n(r, c)$ additív zaj
- Sok zajos kép esetén a zaj átlaga 0, a várható érték maga a kép

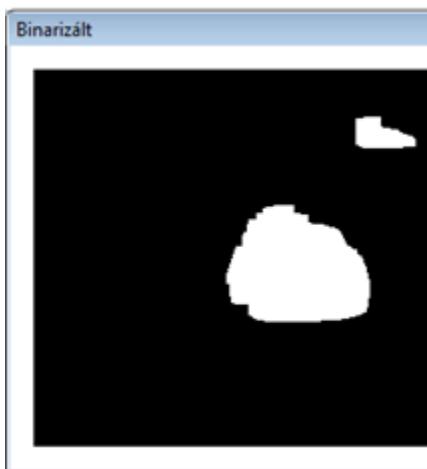


3.3. Teljes képre: kivonás

Változásdetektálás: $O(r, c) = |I_1(r, c) - I_2(r, c)|$



Háttérmodell és aktuális jelenet (a vizsgált térrész maszkjával)



A különbsékkép küszöbölés után és a változás megjelenítése

3.4. Teljes képre: AND/OR

AND: Kép adott részének kimaszkolása

OR: másik kép adott részének hozzáadása

AND



OR



3.5. Geometriai transzformációk

Az r sor- és c oszlopkoordináták megváltoztatása

- Eltolás (t_r, t_c) értékkal:

$$r' = r + t_r$$

$$c' = c + t_c$$

- Skálázás (s_r, s_c) faktorral:

$$r' = s_r r$$

$$c' = s_c c$$

- Elforgatás β szöggel (r_0, c_0) körül:

$$r' = r_0 + (r - r_0)\cos(\beta) - (c - c_0)\sin(\beta)$$

$$c' = c_0 + (r - r_0)\sin(\beta) + (c - c_0)\cos(\beta)$$

- Affín transzformáció:

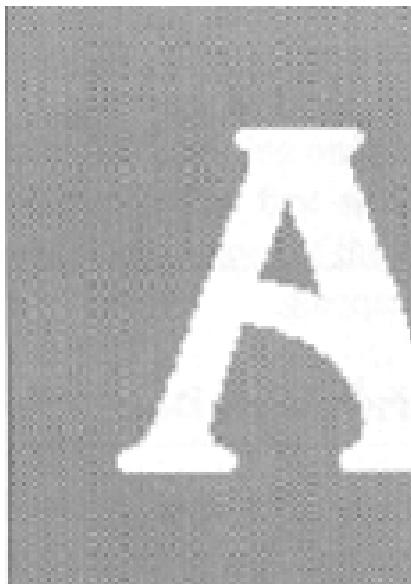
$$r' = a_{11}r + a_{12}c + b_1$$

$$c' = a_{21}r + a_{22}c + b_2$$

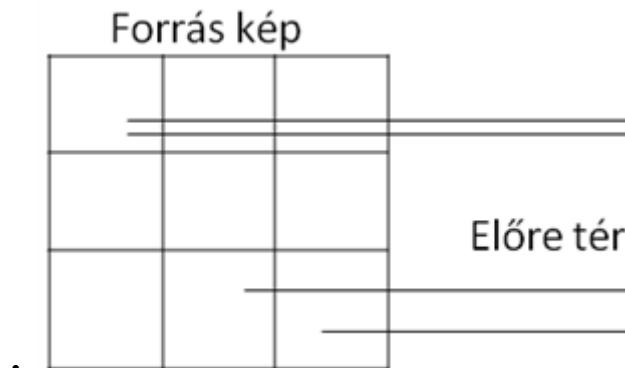


3.6. Geometriai transzformáció: problémák

- A transzformált pixelkoordináták nem a képen belülre esnek (1. probléma)
- Nem felel meg kölcsönösen egymásnak az input kép minden pixele az output minden pixelének (2. probléma)
- A transzformált pixelkoordináták nem egészek (3. probléma)



3.7. Geometriai transzformáció: 2. probléma



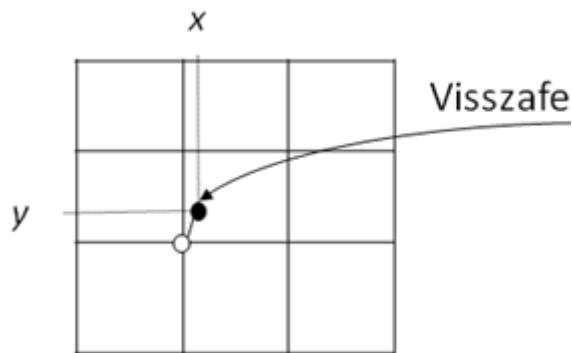
Az output koordinátákból kiindulva, az inverz transzformációval határozzuk meg, hogy melyik forráskoordinátából kell kiolvasni az értéket

- Ezt az értéket írjuk be a célképbe



3.8. Geometriai transzformáció: 3. probléma

- x' és y' egész, azonban x és y valós lehet, ezért a környező pixelekből interpolálhatunk
 - 0-ad rendű interpoláció, legközelebbi szomszéd (nearest neighbor)
 - Magasabb rendű interpoláció (bicubic, spline stb.)



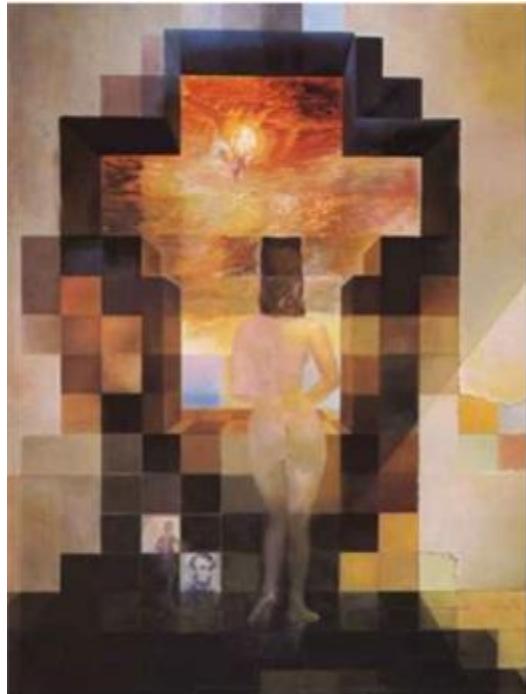
Felhasznált és javasolt irodalom

- [1] R. C. Gonzales, R. E. Woods;: *Digital Image Processing*, Pearson Education, Inc., 3rd ed., ISBN-13: 978-0-13-505267-9, p. 954. 2008.
- [2] S. Akhter, J. Roberts;: *Multi-Core Programming, Increasing Performance through Software Multithreading*, Intel Press, ISBN: 0-9764832-4-6, p. 336. 2006.

5. fejezet - Szűrés képtérben

Vámossy Zoltán

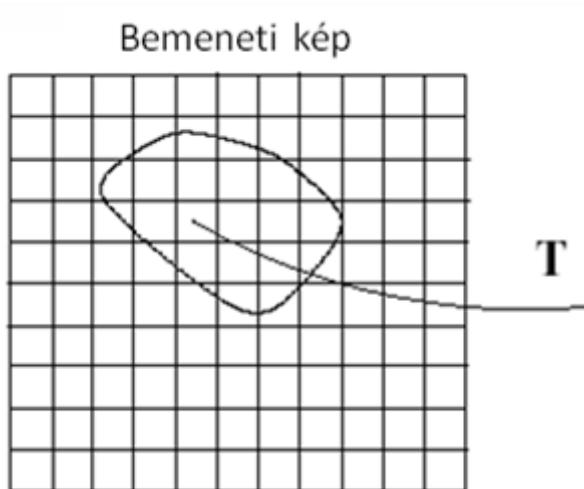
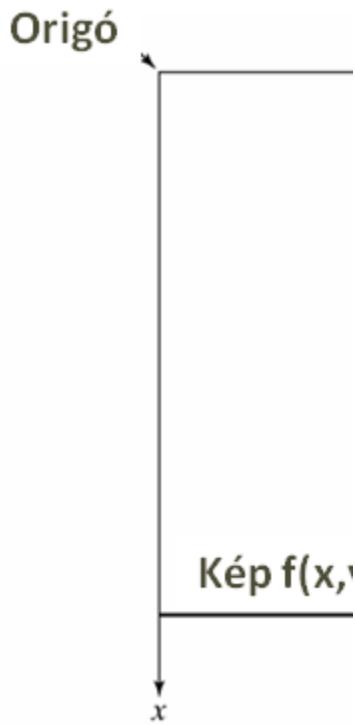
A fejezet nagyrészt a Gonzalez-Woods [1] és Forsyth-Ponce [2] széles körben használt könyvek alapján került feldolgozásra, valamint egyes részeknél Wilkinson-Allen Parallel Programming [3] és Trucco-Verri [4] műveit vettük alapul.



1. A szűrés elve, konvolúció, korreláció, lineáris, nem lineáris szűrők

1.1. Maszk, vagy ablak alapú műveletek

- $g(x, y) = T[f(x, y)]$, T a szomszédos pixeleken operál (lokális művelet)
- Nem önmagába írjuk az eredményt!



1.2. Mit jelent a képszűrés (filtering)?

- Maszk, vagy ablak alapú transzformációk
- A kép adott pixelét a környezetében lévő pixelek függvényében módosítjuk
- Az input és az output kép azonos méretű
- Pixelről pixelre haladunk
- A pont operációknál számítástechnikailag időigényesebbek, de hatékonyságuk jelentősebb

10	5	3
4	5	1
1	1	7

Valamelyei

1.3. Maszkok

- A maszk (kernel) egy mátrix, elemei a súlyok
- minden maszknak van origója
- A szimmetrikus maszkok origója rendszerint a középső elem
- Lineáris eset a legegyszerűbb és leghasznosabb
 - minden pixelt a környezetének lineáris kombinációjával helyettesítjük
- A lineáris kombinációval történő leírást – nemileg tévesen – konvolúciós maszknak, vagy konvolúciónak is nevezik

10	5	3
4	5	1
1	1	7

⊗

0	0
0	0.5
0	1.0

Maszk -

1.4. Lineáris szűrők

Módszer:

- Új képet készítünk, ahol a pixelek intenzitása az eredeti képen ugyanazon a helyen lévő pixel és szomszédjainak súlyozott intenzitás összegéből kerül kiszámításra
- Példa: simítás átlagolással
 - A szomszédok átlagából származik a célpixel
- Példa: simítás Gauss szűrővel
 - A szomszédok súlyozott átlagából származik a célpixel

Tulajdonságok

- Az output **lineáris függvénye** az inputnak
- Az output **eltolás invariáns** (shift-invariant) függvénye az inputnak (pl. az input kép két pixellel balra tolva és szűrve, az output pixelei is két pixellel balra tolva jelennek meg)
- A lineáris konvolúció asszociatív

1.5. Megjegyzés: Nem lineáris szűrők

Ha az output nem lineáris kombinációja az inputoknak, nem lineáris szűrésről beszélünk

Későbbi példák: medián, (erózió, dilatáció, maximum, minimum, extrénum művelet)

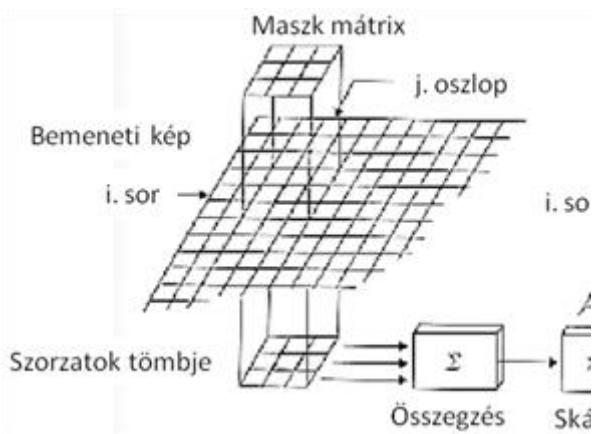
Ezeket nem maszkkal valósítjuk meg

1.6. Képtérben történő szűrések csoportosítása

Lineáris szűrők	Nem lineáris szűrők
Simítás:	Simítás:
<ul style="list-style-type: none"> • Átlagoló (alul-áteresztő) szűrők 	<ul style="list-style-type: none"> • Medián
Élesítés: <ul style="list-style-type: none"> • Felül áteresztő szűrők • Felül erősítő szűrők • Sávszűrők • Derivált szűrők 	

1.7. Maszk használata

- Az input minden pixelére egy maszkot helyezünk úgy, hogy annak origója az adott pixelre essék
- Az input kép maszk alatti pixeleit megszorozzuk a maszkban szereplő súlyokkal
- Az eredmény: az input helyzetének megfelelő pixel értéke a súlyozott értékek összege – esetleg skálázva

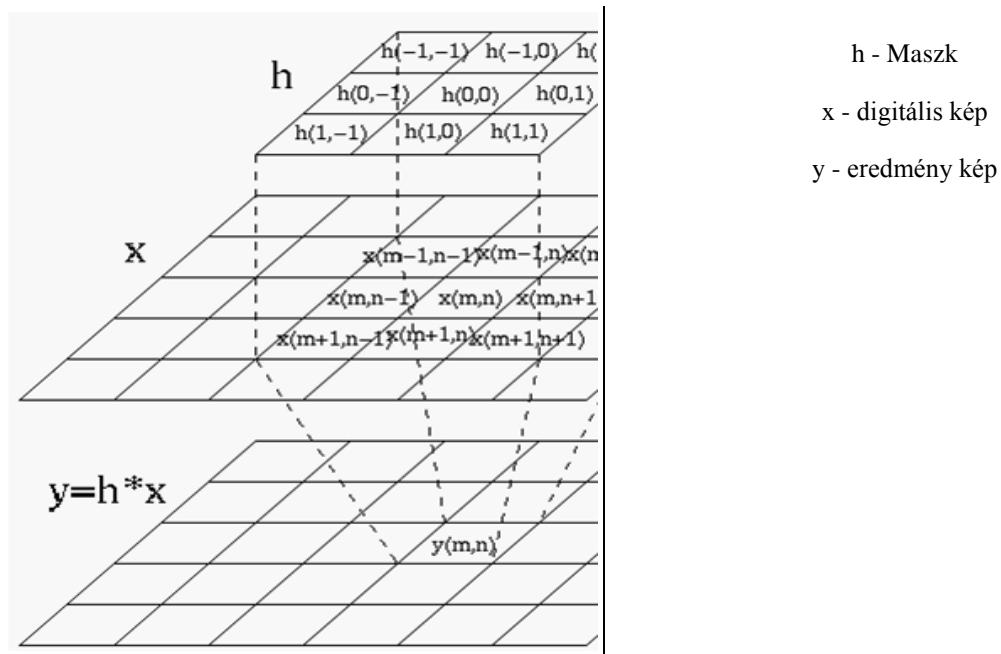


1.8. Maszk

- Ablak alapú műveletekhez gyakran alkalmazznak $n \times n$ méretű maszkot, ahol páratlan ($n = 3, 5, 7, \dots$)
- A maszk méretét általában nem választjuk nagyra

W_{00}	V
W_{10}	V
W_{20}	V

1.9. Maszk használata és konvolúció



A diszkrét két dimenziós konvolúció definíciója (páratlan méretű maszkra):

Figyeljük meg az indexek sorrendjét!

$$y(i, j) = \sum_{k=-n/2}^{n/2} \sum_{l=-n/2}^{n/2} h(k, l) x(i - k, j - l)$$

1.10. Az 1D konvolúcióról

- Két függvény között értelmezett lineáris művelet
- Az egyik függvényből vett minta súlyozott mozgó átlagának számítása egy adott súlyfüggvény alapján
 $f(t)$ függvényből vett minta f_1, f_2, \dots, f_{10}
 súlyfüggvény értékei $w_{-2}, w_{-1}, w_0, w_1, w_2$

$$f(t): f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 f_9 f_{10}$$

$$w(t): w_2 w_1 w_0 w_{-1} w_{-2}$$

a mozgó átlag: a_6

Az a_6 mennyiséget a következőképpen számítjuk ki:

$$a_6 = w_2 f_4 + w_1 f_5 + w_0 f_6 + w_{-1} f_7 + w_{-2} f_8$$

Általában $w_0 \cdot t f_n$ -nel szorozzuk:

$$a_n = \sum_{i=-2}^{n+2} w_{n-i} f_i$$

- Előző súlyfüggvény 0 a (-2, 2) intervallumon kívül
- Az a_n sorozatot nevezük diszkrét konvolúciót: az összegzést kiterjeszhetjük a $(-\infty \infty)$ intervallumra:

$$a_n = \sum_{i=-\infty}^{+\infty} w_{n-i} f_i$$

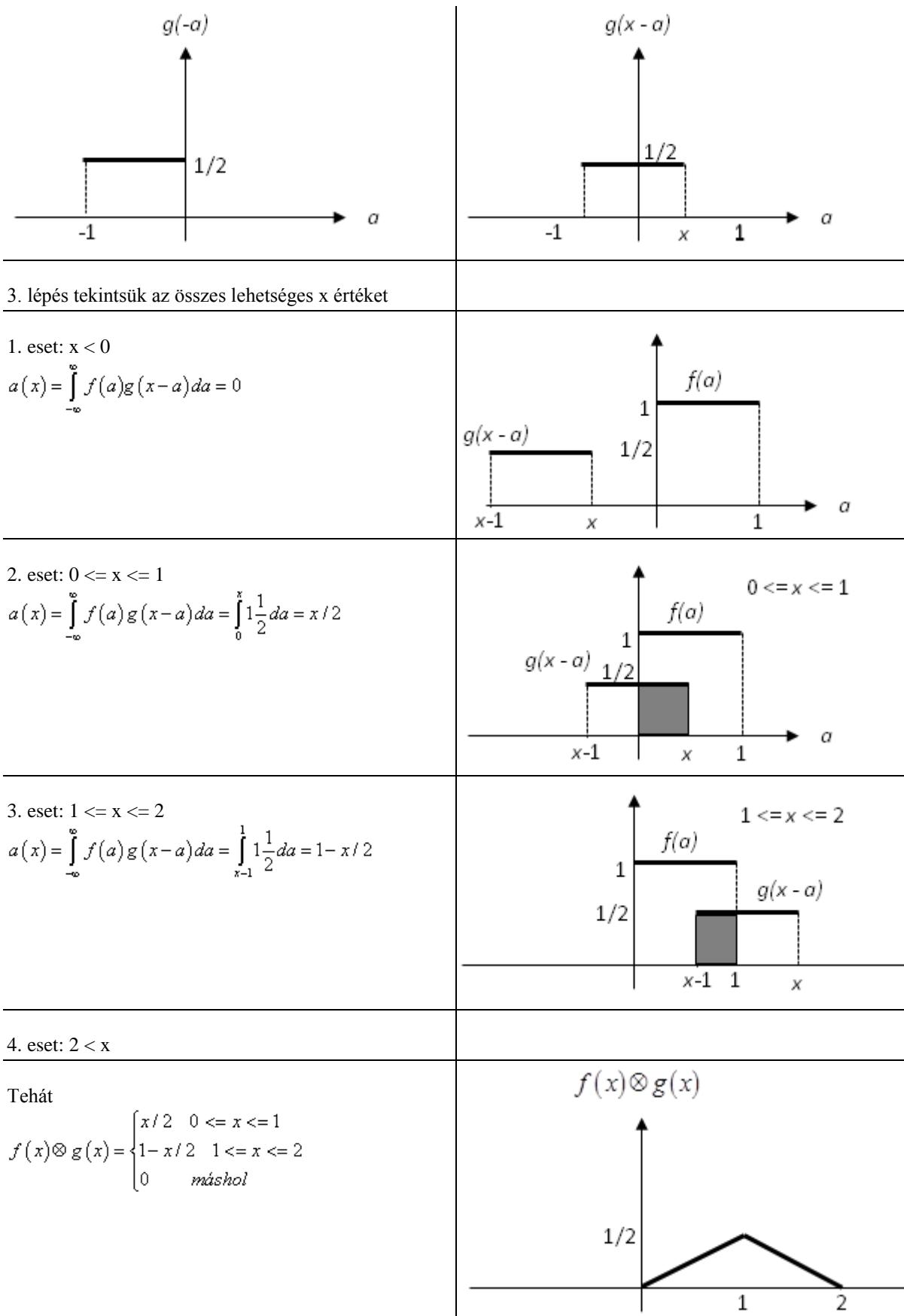
- Ha a függvényből és súlyból vett minták $f_i \cdot k$ és $w_i \cdot k$ közötti részt végtelen kicsinek vesszük, akkor folytonos függvények konvolúciós integrálját kapjuk:

$$a(t) = \int_{-\infty}^{\infty} w(t-t') f(t') dt'$$

- A konvolúcióra igaz: $a = w \otimes f = f \otimes w$
- A konvolúció műveletét \otimes vagy ritkábban * műveleti jellel szokták jelölni.

1.11. 1D konvolúció példa

A következő két függvény konvolúciója	
$f(a)$ 	$g(a)$
1. lépés $g(-a)$: tükrözés!	
2. lépés $g(x-a)$: eltolás	



Megjegyzés:

x értékre a bal oldali határ a számítás során:

$\max(f(x) \text{ bal határa}, g(x - a) \text{ bal határa})$

x értékre a jobb oldali határ a számítás során:

$\min(f(x) \text{ jobb határa}, g(x - a) \text{ jobb határa})$

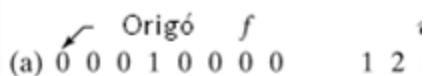
1.12. Megjegyzés: korreláció

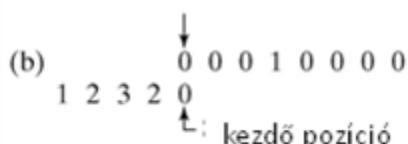
- A korreláció maszktükrözés nélkül transzformálja (szűri) az eredeti képet
- Gyakran használják olyan alkalmazásokban, ahol a hasonlóságot kell megmérni képek, vagy képrészek között
- Ha a maszk szimmetrikus (a tükrözött ugyanaz, mint az eredeti), akkor a konvolúció és a korreláció eredménye ugyanaz
- A diszkrét korreláció definíciója (páratlan méretű w maszkra):

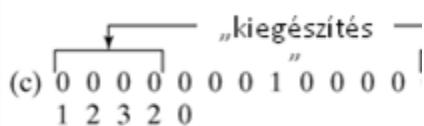
$$g(i, j) = \sum_{k=-n/2}^{n/2} \sum_{l=-n/2}^{n/2} w(k, l) f(i+k, j+l)$$

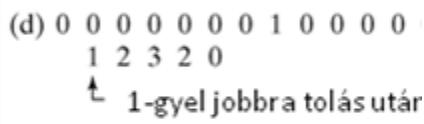
1.13. Konvolúció és korreláció 1D – példa

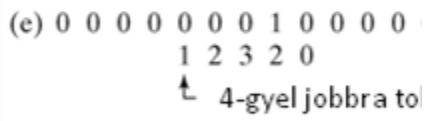
Konrreláció

(a)  Origó f

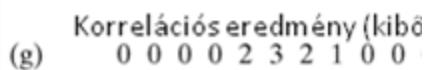
(b) 

(c) 

(d) 

(e) 

(f) 

(g) 

(h) 

1.14. Konvolúció és korreláció 2D – példa

Kiegés		
0 0 0		
0 0 0		
0 0 0		
✓ Eredeti f	0 0 0	0 0 0
0 0 0 0 0	0 0 0	0 0 0
0 0 0 0 0	$w(x, y)$	0 0 0
0 0 1 0 0	1 2 3	0 0 0
0 0 0 0 0	4 5 6	0 0 0
0 0 0 0 0	7 8 9	0 0 0

(a)

kezdő pozíció w			Kibőv		
1	2	3	0	0	0
4	5	6	0	0	0
7	8	9	0	0	0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

(c)

tükörzött w			Kibőv		
9	8	7	0	0	0
6	5	4	0	0	0
3	2	1	0	0	0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

(f)

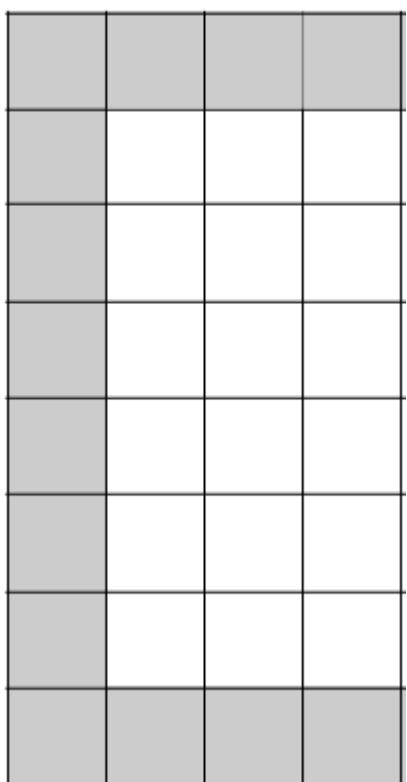
1.15. Maszksúlyok normalizálása

- A maszk elemek összege meghatározza az output kép egészének intenzitását
- Sok konvolúciós maszk esetében az összeg 1 (az eredmény képnek ugyanaz marad az átlagos intenzitása)
- Néhány maszkban negatív súlyok is vannak és az összegük 0
- Ha negatív értékek vannak a maszkban, akkor negatív pixeleredmény is lehet
- Negatív eredmény esetén lineáris normalizálást hajtunk végre az eredményképen

1.16. Gyakorlati problémák

Kép szélénnek kezelése

- A futási idő a maszk méretének exponenciális hatványával arányos
- Kiegészítés
- Szélső sorok, oszlopok elhagyása



1.17. Szűrő példák



Bemeneti kép



Bemeneti kép



Bemeneti kép

1.18. Példa: simítás (blurring) és kivonás



Bemeneti kép



2. Simító szűrés

2.1. Maszk alapú műveletek

Simítás (zajszűrés):

Az intenzitás nagy változásait simítjuk el, a magas-frekvenciás tartalom csökkentése (élek és hirtelen átmenetek elhomályosítása)



Élesítés:

A részletek kiemelése

- Adatpárhuzamos megvalósítás

2.2. Átlagoló szűrő

Az alkalmazott maszk az un. box-filter

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Egyszerű simítási technika, ahol az ablakban lévő intenzitások átlaga az új intenzitás érték:

$$x'_4 = \frac{x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8}{9}$$

Soros kód:

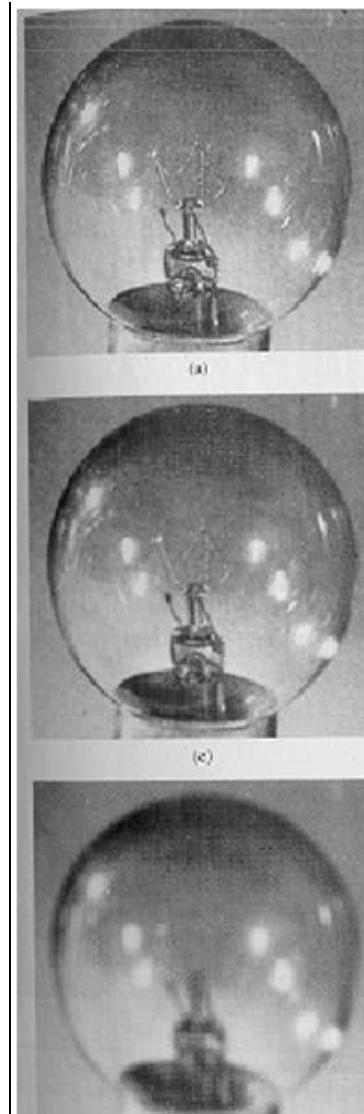
Kilenc lépés kell az átlag kiszámításához, n pixelre $9n$. Komplexitás: $O(n)$.

- A maszk elemei pozitívak
- A maszk mérete határozza meg a simítás mértékét

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(a)

$$\frac{1}{49} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



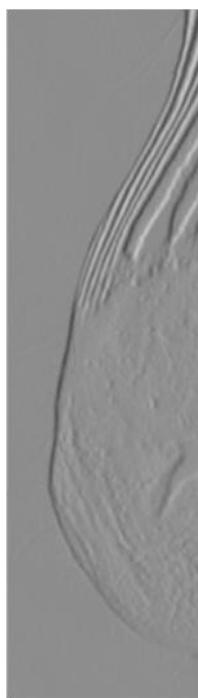
Szeparálható

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \frac{1}{3} [1 \ 1 \ 1] * \frac{1}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = h_x * h_y$$

Rekurzívan számolható, pixelenként 4 operációval



2.3. Simító ablak méretének hatása



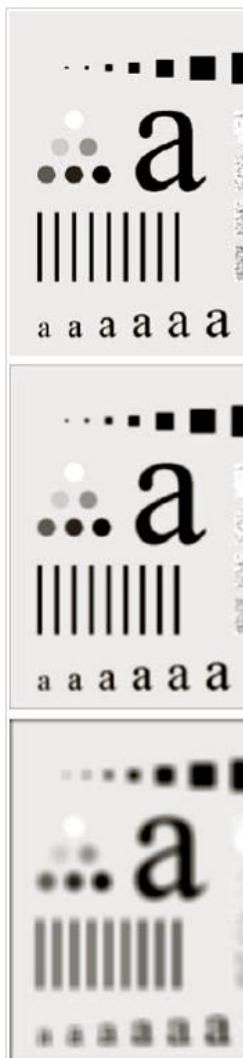
1 pixel



3 pixel



7 pixel



2.4. Példa: simítás átlagolással

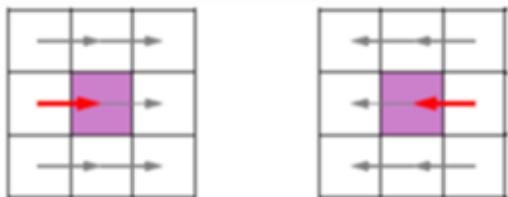
Begyűrűzés (ringing) effektus: élek mentén szétmosás. A begyűrűzés oka: a maszk szélénél hirtelen változás



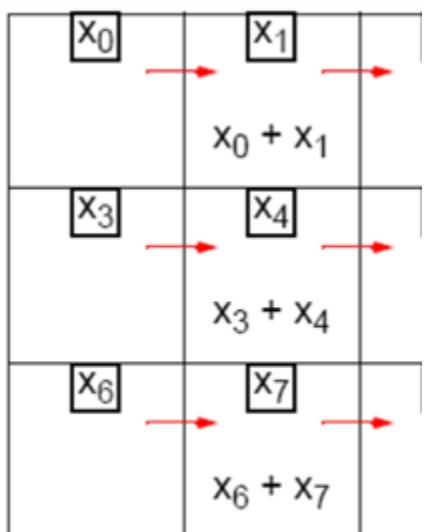
2.5. Párhuzamos átlagoló szűrő

A műveletek száma redukálható négyre

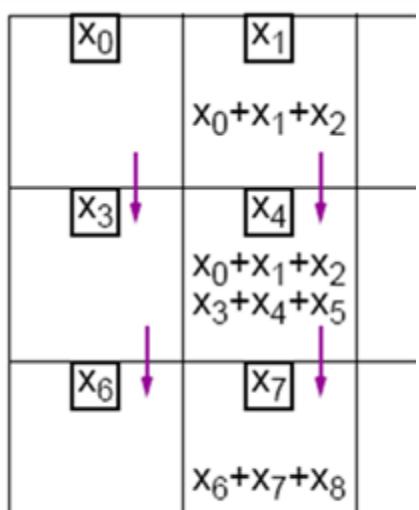
- Elv, ami pontosításra kerül mindenjárt



- minden pixelt összeadunk balról, jobbról, felülről és alulról



(a) Első lépés



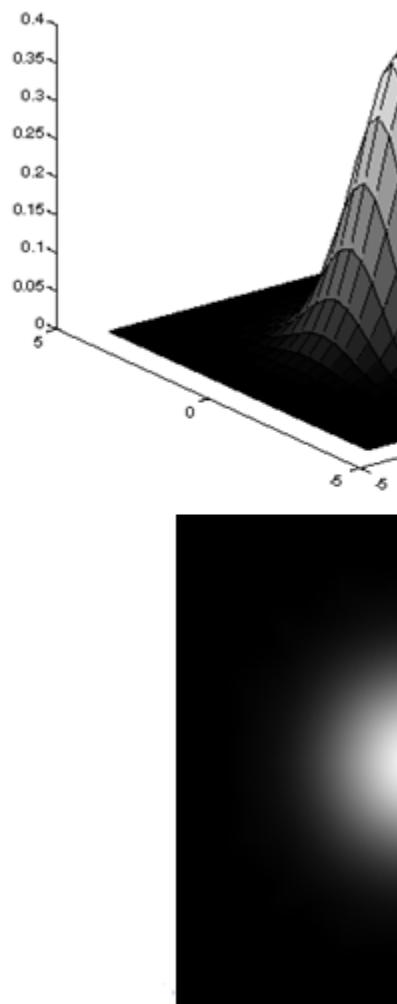
(c) Harmadik lépés

2.6. Simítás Gauss szűrővel

- Az átlagoló simítás nem azonos a defókuszált lencsével készített képpel
- A leginkább tapasztalható differencia: egy pont képe a defókuszált lencse esetében egy életlen folt; az átlagolás ezzel szemben téglalapot készít
- A simító maszk, mely arányos:

$$\exp\left(-\left(\frac{x^2+y^2}{2\sigma^2}\right)\right)$$

- A Gauss maszk a szélen közel 0
- σ a simítás mértéket határozza meg
- Körkörösen szimmetrikus életlen folt képének modellje
- Izotróp (nem irányérzékeny)



Nincs begyűrűzés, mert a maszk szélénél kis értékek vannak

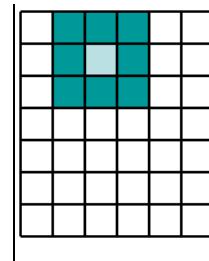


2.7. Gauss szűrő

- A súlyok a Gauss függvényből származnak

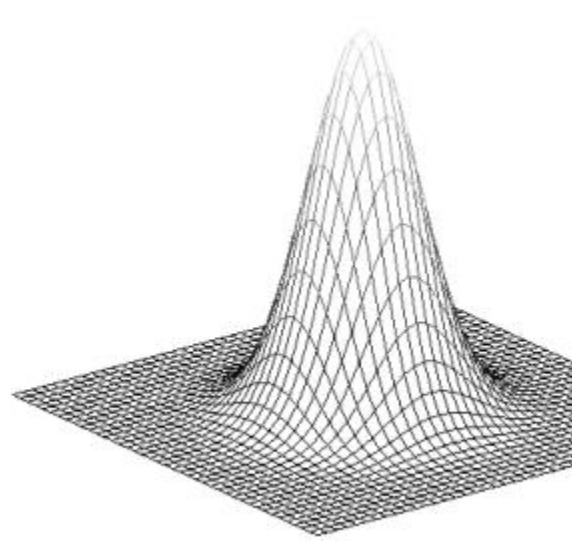
Maszk (kernel):

$$\begin{bmatrix} 0.0113 & 0.0838 & 0.0113 \\ 0.0838 & 0.6193 & 0.0838 \\ 0.0113 & 0.0838 & 0.0113 \end{bmatrix}$$



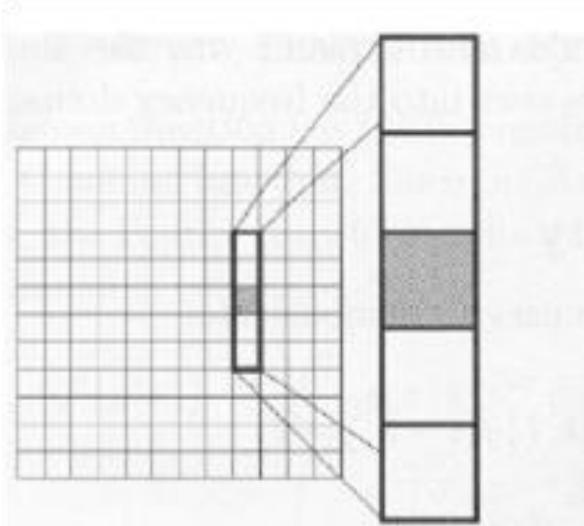
- Gauss konvolúció magja $\sigma = 0.5$
- A Gauss szűrőt hatékonyan lehet implementálni sor és oszlop műveletre, mert szeparálható:

$$\begin{aligned} g(i, j) &= f \otimes h = f \otimes h_y \otimes h_x \\ g(i, j) &= \sum_{k=-n/2}^{n/2} \sum_{l=-n/2}^{n/2} h(k, l) f(i-k, j-l) = \\ &\sum_{k=-n/2}^{n/2} \sum_{l=-n/2}^{n/2} e^{\left[\frac{-(k^2+l^2)}{2\sigma^2} \right]} f(i-k, j-l) = \\ &\sum_{k=-n/2}^{n/2} e^{\left[\frac{-k^2}{2\sigma^2} \right]} \sum_{l=-n/2}^{n/2} e^{\left[\frac{-l^2}{2\sigma^2} \right]} f(i-k, j-l) \end{aligned}$$



2.8. Gauss szűrő szeparált megvalósítása

- Az I kép Gauss szűrése $n \times n$ -es méretű, $\sigma = \sigma_g$ paraméterű g maszkkal
- Készítsünk egy 1D Gauss n szélességű maszkot (g), $\sigma_g = \sigma_G$
- Hajtsunk végre konvolúciót az I kép oszlopain g -vel, az új kép I_c
- Hajtsunk végre konvolúciót az I_c kép sorain g -vel



2.9. Gauss szűrő

- A σ (simítás mértékének) növelésével a maszk méretének is nőnie kell
- Magasság = szélesség = 5σ (a terület 98.76%-át fedи le)

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

2	2	3
2	3	4
3	4	6
4	5	7
5	7	9
5	7	10
6	8	10
6	8	11
6	8	10
5	7	10
5	7	9
4	5	7
3	4	6
2	3	4
2	2	3

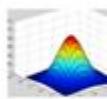
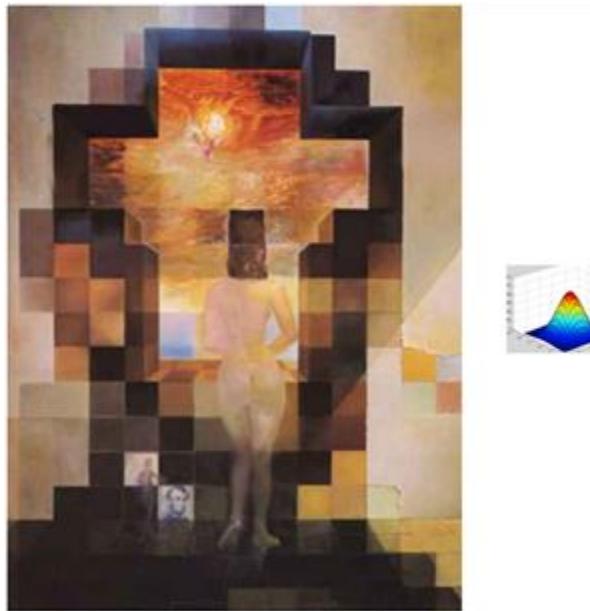
1	2	1
2	4	2
1	2	1

$\frac{1}{16} \times$

- A $\sigma=2$, 13x13-as maszk, 255-re felszorozva

0	0	0	0	1	2
0	0	1	3	6	9
0	1	4	11	20	30
0	3	11	26	50	73
1	6	20	50	93	136
2	9	30	73	136	198
2	11	34	82	154	225
2	9	30	73	136	198
1	6	20	50	93	136
0	3	11	26	50	73
0	1	4	11	20	30
0	0	1	3	6	9
0	0	0	0	1	2

Salvador Dali: Lincoln in Dalivision



2.10. Binomiális szűrő

- 3x3-as szimmetrikus Gauss szűrő
- $$h_3(x,z) = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$
- A binomiális sorozatok a Gauss függvény diszkrét közelítései:

aluláteresztő szűröként viselkednek

- Az 1/16-os osztás általános esetben $1/2^{2p}$
- Az $n=3$ elemű első sor elemei $[1, 2, 1]$ a $p=n-1 \Rightarrow 2$ -od rendű binomiális együtthatók
- A binomiális együtthatók a Pascal háromszögből is származtathatók

$$\binom{p}{i}$$

- Konvolúció segítségével is megkaphatjuk azokat:
- $[1, 1] \otimes [1, 1] = [1, 2, 1]$, $[1, 1] \otimes [1, 2, 1] = [1, 3, 3, 1]$

$$\mathbf{b}^1 = \frac{1}{2} [1 \ 1]$$

$$\mathbf{b}^2 = \frac{1}{4} [1 \ 2 \ 1] = \mathbf{b}^1 \otimes \mathbf{b}^1$$

$$\mathbf{b}^4 = \frac{1}{16} [1 \ 4 \ 6 \ 4 \ 1] = \mathbf{b}^1 \otimes \mathbf{b}^1 \otimes \mathbf{b}^1 \otimes \mathbf{b}^1$$

$$\mathbf{b}^6 = \frac{1}{64} [1 \ 6 \ 15 \ 20 \ 15 \ 6 \ 1] = \mathbf{b}^1 \otimes \mathbf{b}^1 \otimes \mathbf{b}^1 \otimes \mathbf{b}^1 \otimes \mathbf{b}^1 \otimes \mathbf{b}^1$$

A méret növekedésével a szűrő egyre jobban közelíti a Gauss szűrőt

- Szeparálhatók a binomiális szűrők

$$(1/4) * [1 \ 2 \ 1] \otimes (1/4) \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = h_3(x) \otimes h_3(y) = h_3(x, y) = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- A 2D konvolúció két 1D-s konvolúcióval számolható

- Példa $n = 3$

2D konvolúció: 9 szorzás, 8 összeadás

1D konvolúciókkal: 6 szorzás, 4 összeadás

- Példa általános eset n

2D konvolúció: n^2 szorzás, $n^2 - 1$ összeadás

1D konvolúciókkal: $2 * n$ szorzás, $2 * (n - 1)$ összeadás és $2 * (n - 1)$ shift jobbra

- Az 1D $h_n(x)$ maszk helyettesíthető n db $[1 \ 1]$ maszk konvolúciójával
 - ez 1 összeadás és egy shiftelés;
- 2D $h_n(x, y)$ ez $2 * (n - 1)$ összeadás, nincs szorzás és $2 * (n - 1)$ shiftelés



2.11. Medián szűrő (nem lineáris)

- A pixelt a környező pixelek mediánjával helyettesíti
- Sorbarendezést feltételező (rank order, vagy röviden RO) szűrő
- Nem lineáris:

$$\text{median}\{f_1 + f_2\} \neq \text{median}\{f_1\} + \text{median}\{f_2\}.$$

- De:

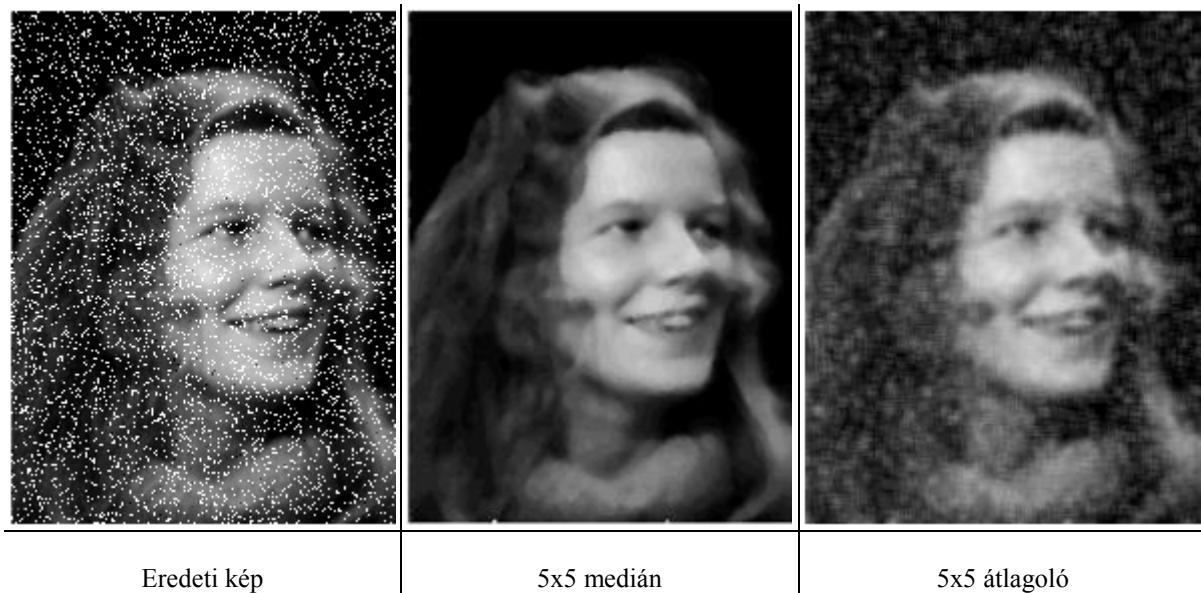
$$\text{median}\{cf\} = c \text{ median}\{f\},$$

$$\text{median}\{c + f\} = c + \text{median}\{f\}.$$

- Véletlenszerűen elhelyezkedő, impulzus szerű zaj kiszűrésére hatékony
- Az éleket megtartja
- Nagy zajnál nem hatékony
- Gauss-féle zaj esetén nem hatékony

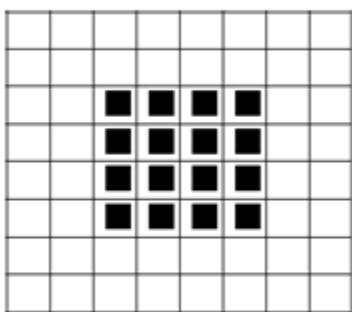
2.12. Medián szűrő

Medián és átlagoló összehasonlítása



2.13. Medián szűrő megjegyzés

Eredeti kép

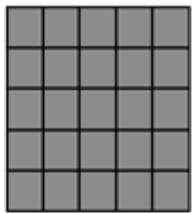


szűrők

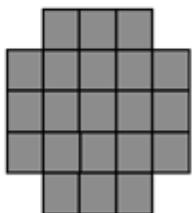


2.14. Medián szűrő negatív hatásai

Pontokat, 1 pixel széles vonalakat, sarkokat törli a medián szűrő



Teszt képek



2.15. Medián szűrő

Soros megvalósítás:

A medián meghatározása érdekében rendezni kell a pixelértékeket és a középsöt kell kiválasztani

- Például 3 x 3-as esetben a rendezett értékek: $y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7$, és y_8 .
A medián y_4
- Az ötödik elemet kell kivenni a rendezés után
- Pl. buborékos rendezésnél a műveletek (összehasonlítás és ha kell csere) száma: $8 + 7 + 6 + 5 + 4 = 30$ lépés, azaz n pixelre $30n$ művelet
- Mivel a medián szűrő nagyon hatékony eszköz, de futása a rendezés miatt viszonylag lassú, ezért számos továbbfejlesztett, vagy közelítő megoldást fejlesztettek ki a gyorsaság növelése érdekében

2.16. Közelítő medián szűrő – párhuzamosítás

Párhuzamos megvalósítás:

- Elsőként a soron belül hajtsunk végre három összehasonlítást és cserét:

$$p_{i,j-1} \leftrightarrow p_{i,j}$$

$$p_{i,j} \leftrightarrow p_{i,j+1}$$

$$p_{i,j-1} \leftrightarrow p_{i,j}$$

ahol \leftrightarrow jelenti, hogy hasonlítsd össze és cseréld fel, ha a baloldali érték nagyobb, mint a jobboldali.

- Ezután oszlopokra vonatkozóan három lépés:

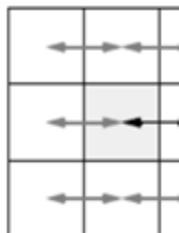
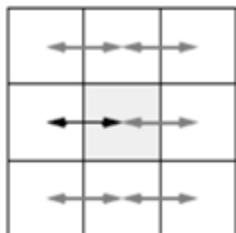
$$p_{i-1,j} \leftrightarrow p_{i,j}$$

$$p_{i,j} \leftrightarrow p_{i+1,j}$$

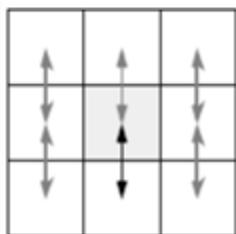
$$p_{i-1,j} \leftrightarrow p_{i,j}$$

- Összesen hat lépés

- Mikor közelít, mikor nem pontos?



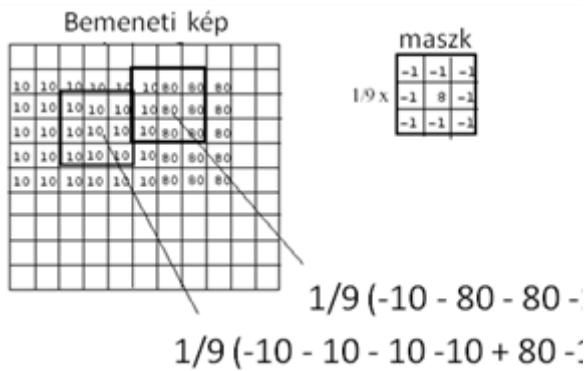
Alegnagyo
sorban



3. Élesítő szűrés

3.1. Élesítés (sharpening)

- A kép finom részleteinek kiemelésére szolgál
- A magas kontrasztú részeket a lokális környezetben számított intenzitás differenciák segítségével kaphatjuk meg
- A maszk súlyai pozitív és negatív értékek
- Közel konstans intenzitású rész felett a maszk eredményeként nulla közeli értéket kapunk
- Hirtelen változó intenzitásoknál a konvolúció eredménye nagyobb érték
- Ilyen pontok tipikusan az objektumok, vagy képrészek határain jelennek meg



3.2. Élesítés deriváltak használatával

- Az élesítéshez tébeli deriváltakat használunk fel
 - A gradiens és x irányú összetevőjének meghatározása

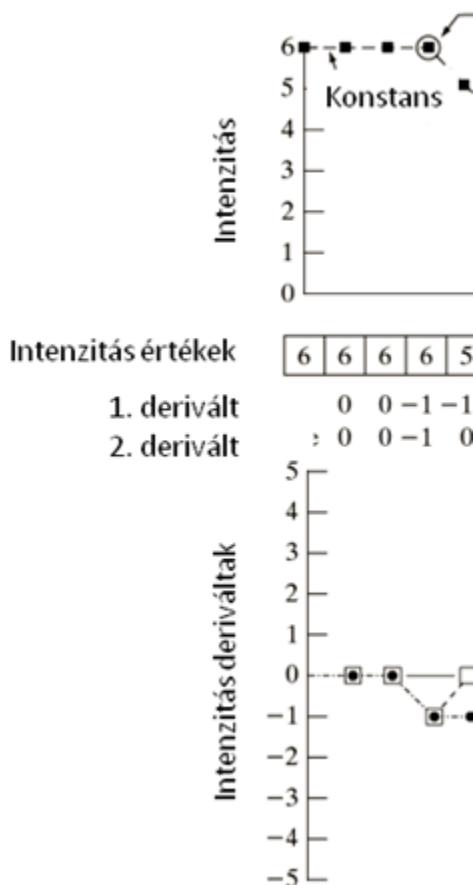
$$grad(f) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad \frac{\partial f}{\partial x} = \lim_{\varepsilon \rightarrow 0} \left(\frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon} \right)$$

- A gradienst véges differenciákkal közelítjük, amik maszkokkal is számíthatók

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

- Lineáris és eltolás invariáns művelet => konvolúció

3.3. Első és másodrendű differenciák példa



3.4. Élesítés

- Első és másodrendű differenciák összehasonlítása
- Elsőrendű differenciák vastagabb éleket generálnak
- Másodrendű differenciáknak erőteljesebb a válasza olyan finom részletekre, mint vékony vonalak, vagy izolált pontok
- Másodrendű differenciák dupla választ adnak az intenzitás lépcsős változásánál
- Élesítéshez gyakrabban alkalmaznak másodrendű deriváltakat

3.5. Laplace szűrő

- Izotrópikus szűrő: nem irányfüggő
- Legegyszerűbb másodrendű differenciákat tartalmazó szűrő a Laplace

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4 * f(x,y)$$

0	1	0
1	-4	1
0	1	0

0	-1	0
-1	4	-1
0	-1	0

- Folytonossági hiányok kiemelésére szolgál
- A háttért eltünteti
- A háttér visszakapható, ha az eredeti képet hozzáadjuk

$$g(x, y) = \begin{cases} f(x, y) - \nabla^2 f(x, y), & \text{ha a középső elem negatív} \\ f(x, y) + \nabla^2 f(x, y), & \text{ha a középső elem pozitív} \end{cases}$$

- Eredeti, szűrt kép, transzformált Laplace, élesített kép

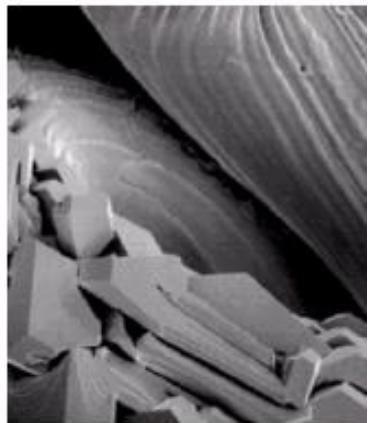


Laplace élesítés

$$g(x, y) = -(f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)) + 5 * f(x, y)$$

0	-1	0
-1	5	-1
0	-1	0

-1	-1
-1	9
-1	-1



3.6. Életlenítés (unsharpening) és felül erősítés

Élesített kép = eredeti – elmosott (blurred)

Felül erősített kép = eredeti – alul szűrt kép
 $g_s(x,y) = f(x,y) - \bar{f}(x,y)$

Felül erősítő szűrő, $A >= 1$

$$g_{sh}(x,y) = A * f(x,y) - \bar{f}(x,y)$$

$$g_{sh}(x,y) = (A-1) * f(x,y) + f(x,y) - \bar{f}(x,y)$$

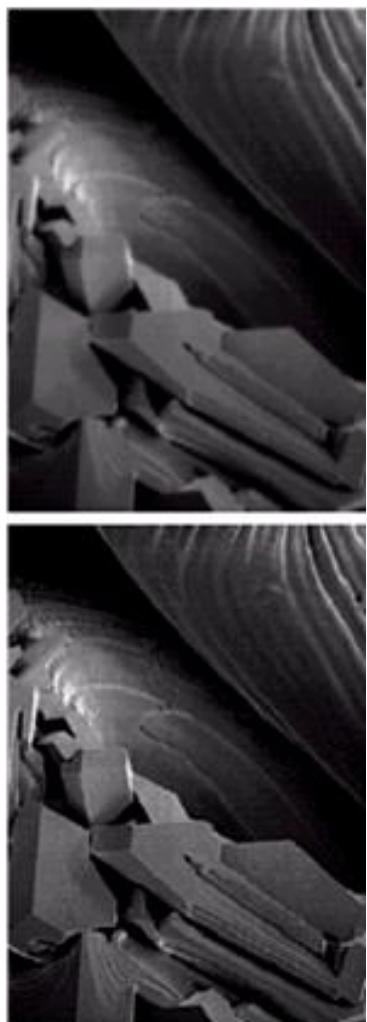
$$g_{sh}(x,y) = (A-1) * f(x,y) - g_s(x,y)$$

0	-1
-1	A + 4
0	-1

-1	-1
-1	$A + B$
-1	-1

3.7. Felül erősítés

Laplace, $A=1$, $B=1.7$



Felhasznált és javasolt irodalom

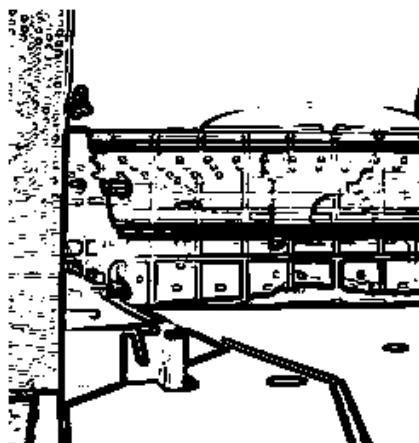
- [1] R. C. Gonzales, R. E. Woods:; *Digital Image Processing*, Pearson Education, Inc., 3rd ed., ISBN-13: 978-0-13-505267-9, p. 954. 2008.

- [2] D. A. Forsyth, J. Ponce:, *Computer Vision: A Modern Approach*, Prentice Hall, p. 792. 2003.
- [3] B. Wilkinson, M. Allen:, *Parallel Programming, Techniques and Applications Using Networked Workstations and Parallel Computers*, Pearson Education, Inc., 2nd ed. ISBN: 0-13-140563-2, p. 467. 2005.
- [4] E. Trucco, A. Verri:, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, ISBN: 0-13-261108-2, p. 343. 1998.

6. fejezet - Élek, sarokpontok, speciális szakaszok

Vámossy Zoltán

A fejezet Gonzalez-Woods [1], Forsyth-Ponce [2] és Trucco-Verri [3] művek alapján kerül bemutatásra. A SUSAN módszerhez Smith és Brady [4] cikkét használtuk



1. Éldetektálás elve és éldetektorok

1.1. Élek (edges)

Mit értünk él alatt?

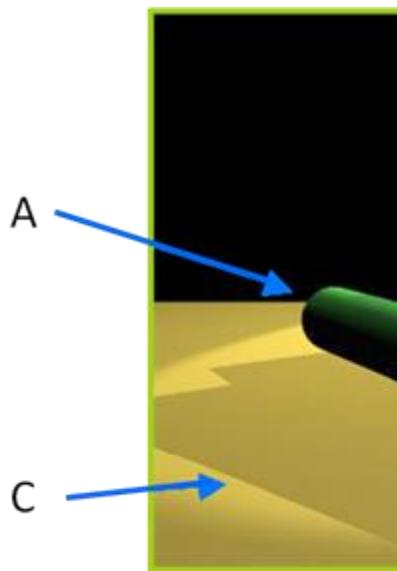
- Élek olyan pixelek ahol, vagy ami körül a kép intenzitás-értékei erőteljesen megváltoznak



Miért fontosak az élek?

- A legtöbb elem, objektum, vagy azok árnyékai éleket generálnak
- Az élek megtalálásával általában az objektum alakját és helyét is meg tudjuk határozni

1.2. Élek: Mi okoz hirtelen változást?



- A: Hirtelen mélységi változás
- B: Felület normálisának változása
- C: Megvilágítás változása: árnyékok, világítás változás
- D: Visszaverődésben változás: felület tulajdonság, jelek

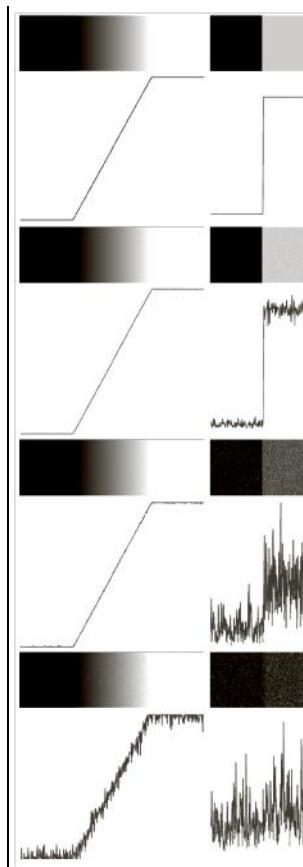
1.3. Tipikus élprofilok

- Ugrás
- Rámpa
- Gerinc
- Tető
- Vonal

Élintenzitások szintetikus képen



Élintenzitások zajjal terhelt képeken



1.4. Hogyan találhatunk éleket?

Éldetektálás lépései

- Zajcsökkentés (Noise reduction)
- Élkiemelés (Edge enhancement)
- Éldetektálás (Edge detection)
- Éllokalizálás (Edge localisation)



1.5. Definíciók

- Él-normális = merőleges az élre, a maximális intenzitás-változás iránya,
 $N(i, j) \perp f(i, j)$
- Él-irány = az él irányába, merőleges a normálisra
- Él-pozíció = ahol a képen elhelyezkedik az él
- Él-erősség = mutatja mennyire „jó” egy él. Nagy változás \rightarrow nagy erősség

Él-irány



1.6. Éldetektálás deriválással



1.7. Változás detektálás 1D-ben

Derivált

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \rightarrow 0} \left(\frac{f(x + \varepsilon) - f(x)}{\varepsilon} \right) \Rightarrow \frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}) - f(x_n)}{\Delta x}$$

Differenciáló szűrők

- Hátrafele differencia: [-1 1]
- Előrehaladó differencia: [1 -1]
- Központi differencia: [-1 0 1]

1.8. Deriváltak, differenciák 2D-ben

Definíció

$$\frac{\partial f(x, y)}{\partial x} = \lim_{\varepsilon \rightarrow 0} \left(\frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon} \right) \quad \frac{\partial f(x, y)}{\partial y} = \lim_{\varepsilon \rightarrow 0} \left(\frac{f(x, y + \varepsilon) - f(x, y)}{\varepsilon} \right)$$

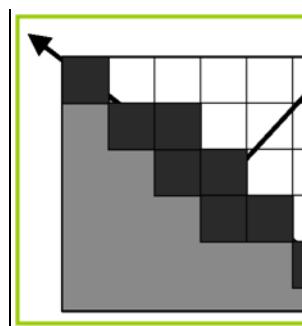
Közelítés

$$\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x_{n+1}, y_m) - f(x_n, y_m)}{\Delta x} \quad \frac{\partial f(x, y)}{\partial y} \approx \frac{f(x_n, y_{m+1}) - f(x_n, y_m)}{\Delta x} = f_y$$

$$f_x = [1 \quad -1] \quad f_y = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Konvolúciós magok

$$grad(f) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$



1.9. Gradiens nagyság és irány

Gradiens

$$\text{grad } (f) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

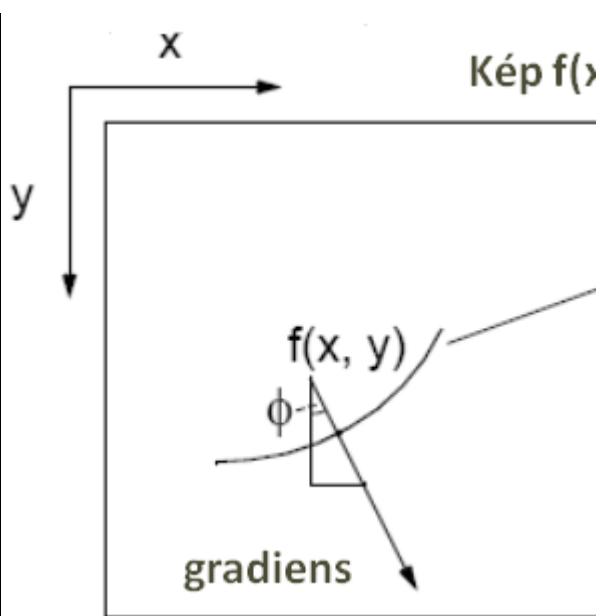
Gradiens nagyság

(Manhattan, illetve Euklideszi távolsággal)

$$|\nabla f| = \left| \frac{\partial f}{\partial x} \right| + \left| \frac{\partial f}{\partial y} \right| \quad |\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2}$$

Gradiens irány

$$\varphi(x, y) = \arctan \begin{bmatrix} \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial x} \end{bmatrix}$$



1.10. Differenciák



1.11. Éldetektáló maszkok

A gradiens összetevőinek közelítő számolása (központi differenciával)

$$\left| \frac{\partial f}{\partial x} \right| = (x_5 - x_3) \quad \left| \frac{\partial f}{\partial y} \right| = (x_7 - x_1)$$

A gradiens nagyság

$$|\nabla f| = \left| \frac{\partial f}{\partial x} \right| + \left| \frac{\partial f}{\partial y} \right| = |x_7 - x_1| + |x_5 - x_3|$$

Maszkok

0	-1	0
0	0	0
0	1	0

1.12. Prewitt maszk

A gradiens összetevőinek közelítő számolása (központi differenciával)

$$\left| \frac{\partial f}{\partial x} \right| = (x_2 - x_0) + (x_5 - x_3) + (x_8 - x_6) \quad \left| \frac{\partial f}{\partial y} \right| = (x_6 - x_0) + (x_7 - x_1) + (x_8 - x_2)$$

A gradiens nagyság

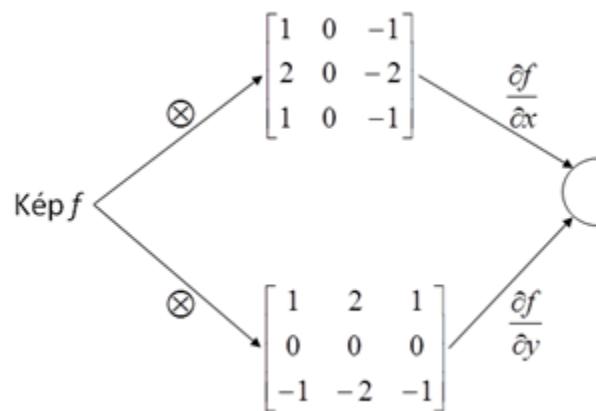
$$|\nabla f| = \left| \frac{\partial f}{\partial x} \right| + \left| \frac{\partial f}{\partial y} \right| = |(x_2 - x_0) + (x_5 - x_3) + (x_8 - x_6)| + |(x_6 - x_0) + (x_7 - x_1) + (x_8 - x_2)|$$

Maszkok

-1	-1	-1
0	0	0
1	1	1

1.13. Sobel éldetektáló

A Sobel éldetektálás során meghatározzuk a gradiens összetevőket, majd a gradiens nagyságát. Ha az így kapott érték nagyobb mint a küszöb, ott van él





1.14. Prewitt és Sobel éldetektálás

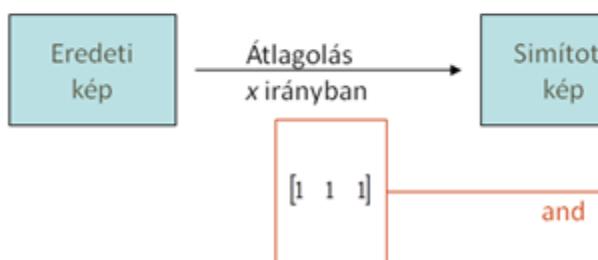
Kevéssé zajérzékeny (3x3 maszk jobban eltünteti a zajokat)

A nagyobb maszkméret miatt a meredek élek több pixel szélesen jelentkeznek

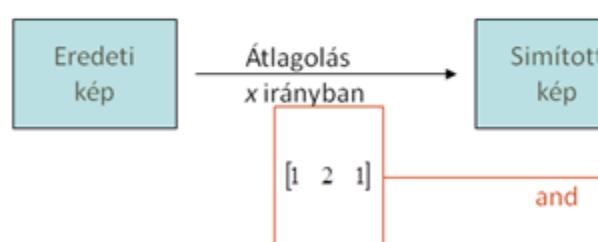
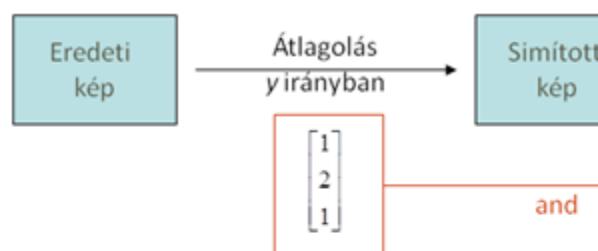
Főbb lépések

- Input: kép és küszöb
- Képszűrés
- Gradiens nagyság számolása
- Küszöbölés

1.15. Prewitt éldetektor párhuzamosítás



1.16. Sobel éldetektáló párhuzamosítás



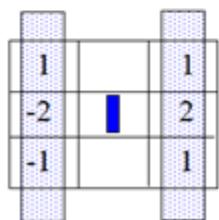
1.17. Sobel maszk: összefoglalás

-1	0	1
-2	0	2
-1	0	1

$$= 1/4 * [-1 \ 0 \ -1] \otimes \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

1	2	1
0	0	0
-1	-2	-1

$$= 1/4 * [1 \ 2 \ 1] \otimes \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$



Az éellel párhuzamosan

1.18. Robinson iránytű maszk

-1	0	1
-2	0	2
-1	0	1

0	1
-1	0
-2	-1



1	0	-1
2	0	-2
1	1	-1

0	-1
-1	0
2	1



1.19. További maszkok

1	1	1
1	-2	1
-1	-1	-1

Prewitt 1

5
-3
-3

1	1	1
0	0	0
-1	-1	-1

Prewitt 2

1
0
-1

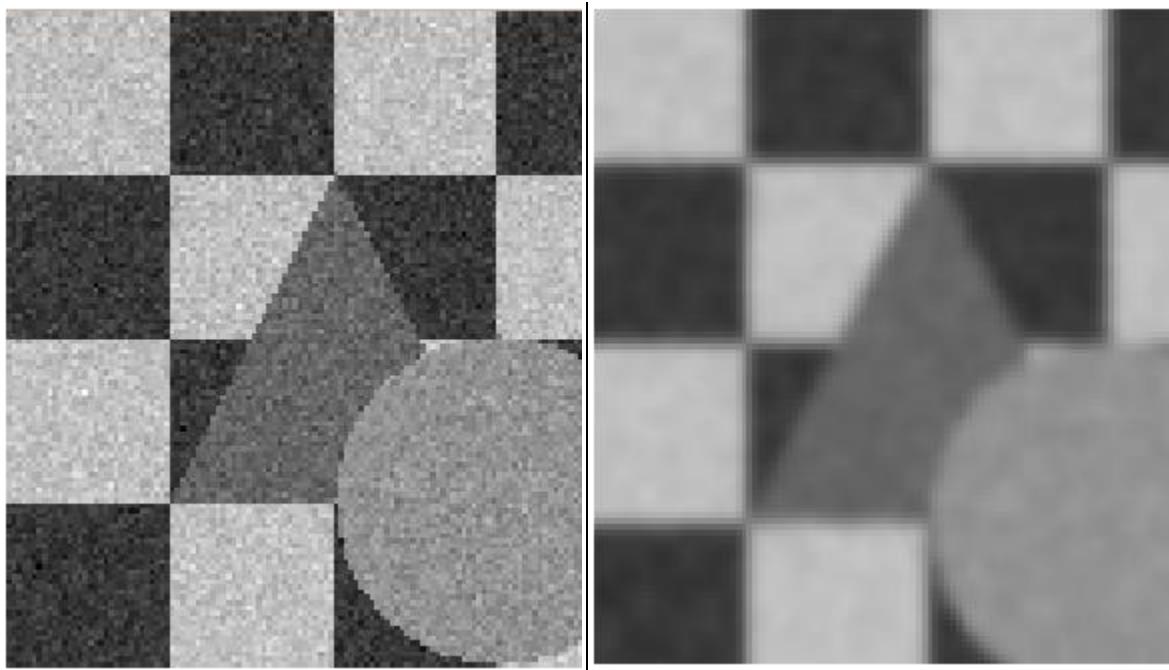
1.20. Laplace éldetektálás

- Ahol a gradiens maximális, ott a második derivált előjelet vált (0)
- Elmosódott élek esetén pontosabb lokalizálás
- Ebben az esetben csak az élek helyét tudjuk meghatározni, az irányát nem
- Az operátor nem érzékeny az elforgatásra, izotrópikus
- Zajérzékeny -> előtte simítás szükséges

1.21. Gauss simítás + Laplace (LoG)

- Zajra nagyon érzékeny éldetektálók esetében előbb simítást szoktak alkalmazni
- Például Gauss szűrőt
- Az irodalomban sokszor eltérő normalizáló szorzótagot használnak

$$g(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

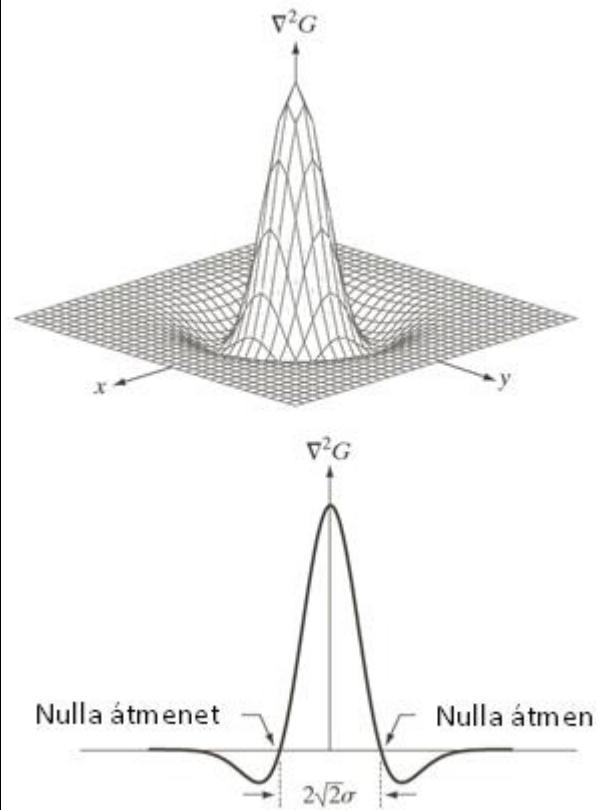


- Alkalmazhatjuk közvetlenül a Gauss szűrő Laplace-át – második derivált szerint (Laplacian of Gaussian) – LoG

$$\nabla^2(f \otimes g) = \nabla^2(g \otimes f) = \nabla^2g \otimes f$$

$$\nabla^2g = \frac{r^2 - \sigma^2}{\sigma^4} = e^{-\frac{r^2}{2\sigma^2}}, \quad r^2 = x^2 + y^2$$

- Elnevezés: Mexikói kalap



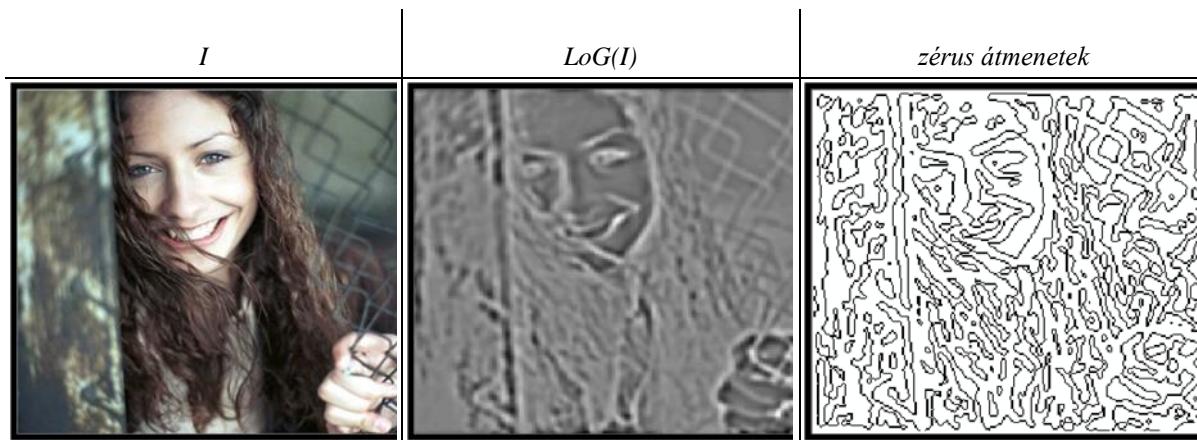
1.22. LoG – Marr-Hildreth éldetektáló

- Robusztus
- Simított kép deriváltját közelíti
- A zérus átmeneteket kell vizsgálni

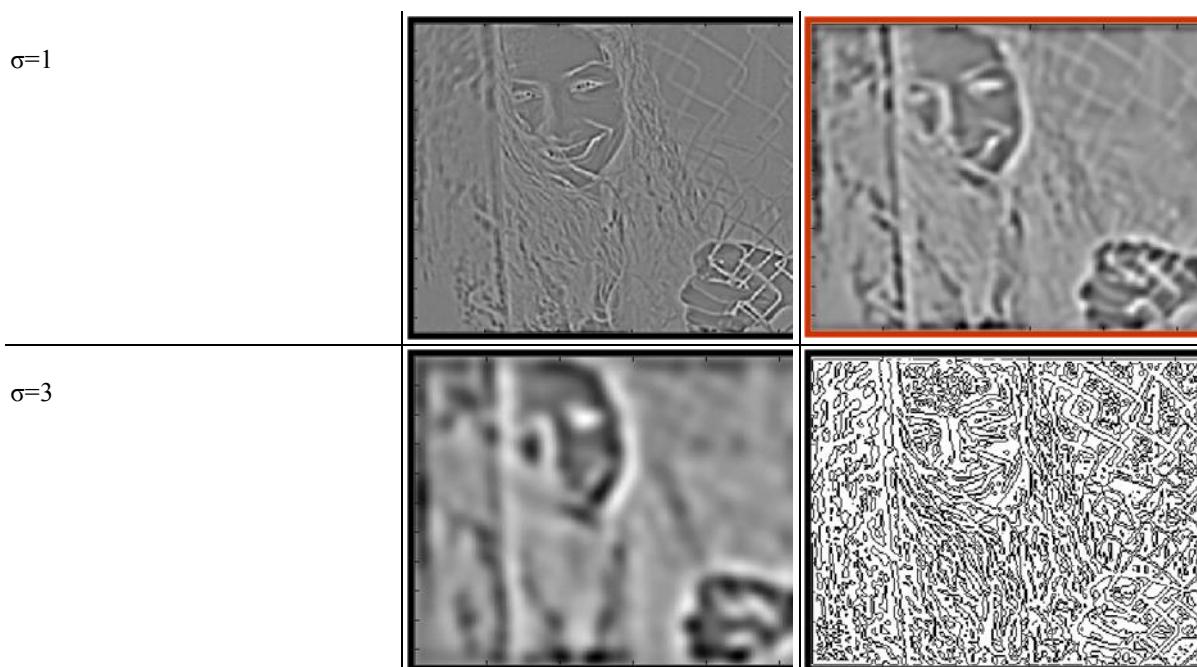
Négy eset:

- {+,-}
 - {+,0,-}
 - {-,+}
 - {-,0,+}
- Stabilabb zérushelyek jelölik az éleket, mert nem meredek éleknél is pontos

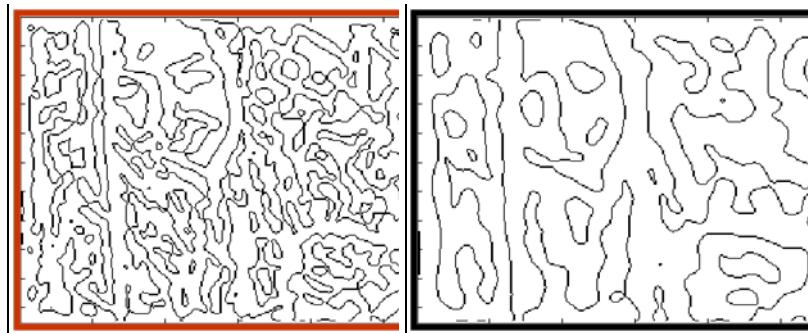
1.23. LoG példa



Különböző simítások hatása



$\sigma=6$



1.24. Canny éldetektor

John Canny, "Finding Edges and Lines in Images", Master's Thesis, MIT, June 1983.

- "Optimális" maszk – Gauss szűrő
- Élkiemelés
- Nem maximumok elnyomása (Non-maximum suppression) – eltávolítja a maximumra merőleges élgyanús pontokat
- Hiszterézises küszöbölés (Hysteresis thresholding) – hosszabb kontúrok készítése

Feltételezés

- Elsősorban lépcsős élek vannak a képen
- A kép Gauss-zajjal terhelt



1.25. Ideális éldetektáló

Milyen kritériumoknak kell megfelelnie egy ideális éldetektálónak?

- Megbízható
 - minden valódi élt detektál
 - nem detektál hibás éleket (zajos kép)
- Az éleket pontosan lokalizálja
- minden élt pontosan egyszer jelez

1.26. I. Canny éldetektor

Az élkiemelő elemei:

1. (Lineáris) konvolúció Gauss szűrővel

$$J = I \otimes G$$

I = eredeti kép (image)

G = Gauss szűrő magja (kernel)

Nagyobb szűrő jobban csökkenti a zajt, lassabb, de kevésbé lokalizálja jól az éleket!

2. Gradiens számolás

$$\forall I(i,j), \text{ kiszámoljuk } I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}$$

3. Normális és erősség (nagyság) számítás

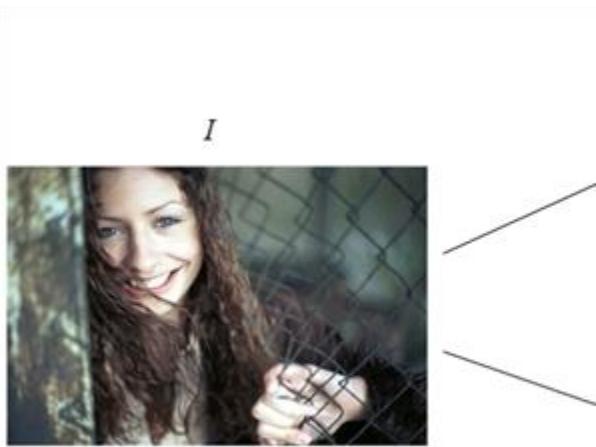
$$E_s(i,j) = \sqrt{I_x^2(i,j) + I_y^2(i,j)} \quad E_o(i,j) = \arctan\left(\frac{I_y}{I_x}\right)$$

Kimenet:

E_s = élerősség (milyen jó az él, a gradiens nagyságával arányos)

E_o = élorientáció (milyen irányba mutat)

1.27. I. Canny éldetektor – első két lépés példa



1.28. II. Non-maxima suppression

Élek ott vannak, ahol a gradiensnek lokális maximum van

A Non-maxima suppression (nem maximumok elnyomása) célja:

- Fals él pontok eltávolítása, amelyek az élre merőleges irányban vannak
- Egy vastagságú élekké zsugorítás



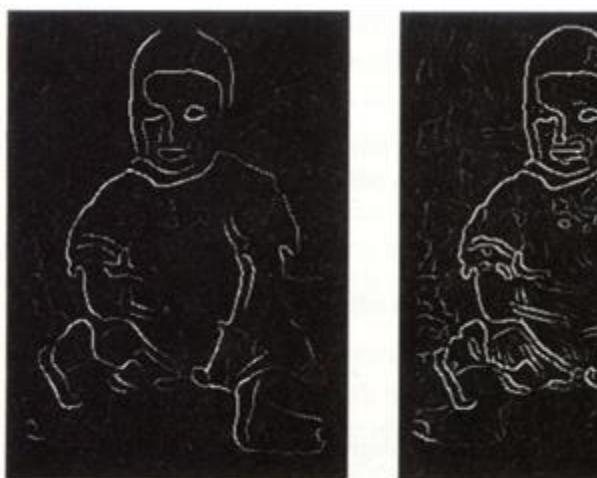
1.29. II. Non-maxima suppression algoritmus

1. minden (i, j) -re határozzuk meg azt a d_k ($0^\circ, 45^\circ, 90^\circ, 135^\circ$) irányt, ami legjobban közelíti az $E_s(i, j)$ élnormálist
2. Ha $E_s(i, j) <$ legalább egy szomszédjánál a d_k irányokban, akkor $I_N(i, j) = 0$ legyen (elnyomás), egyébként $I_N(i, j) = E_s(i, j)$

Eredmény: $I_N(i, j)$ vékonyított éleket tartalmazó kép a nem maximumk eltávolítása után

1.30. III. Canny – harmadik lépés oka

Élkiemelő - balról jobbra $\sigma=3, \sigma=2, \sigma=1$



1.31. III. Hysteresis thresholding

Miért szükséges a hiszterézises küszöbölés?

- Ha a küszöb túl alacsony, akkor fals él pontok maradnak
- A küszöb felett, illetve alatt is lehet maximum erősség
- Ha az élek értéke a küszöb körül ingadozik, akkor sok szakadás lehet



1.32. III. Canny alacsony, illetve magas küszöb



1.33. III. Hysteresis thresholding

Definiálunk két küszöböt τ_l és τ_h $\tau_l < \tau_h$

Minden $I_N(i, j)$ élpontra

1. Keressük meg a következő $I_N(i, j)$ él pontot, hogy

$$I_N(i, j) > \tau_h$$

2. $I_N(i, j)$ -től kiindulva kövessük a lokális maximumok láncát az élnormálisokra merőleges irányban mindaddig, amíg

$$I_N > \tau_h$$

Jelöljünk meg minden meglátogatott pontot (lista)

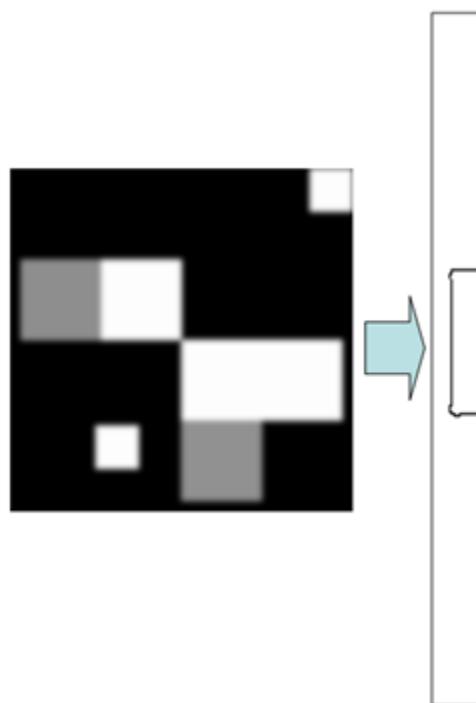
- Tehát ha a felső küszöbnél nagyobb, akkor vegyük fel élnek
- Ha az alsó küszöb alatt van, akkor nem él
- Ha a kettő között van, akkor vegyük fel élnek, ha egy szomszédos pixel élhez tartozik

1.34. Canny eredmények



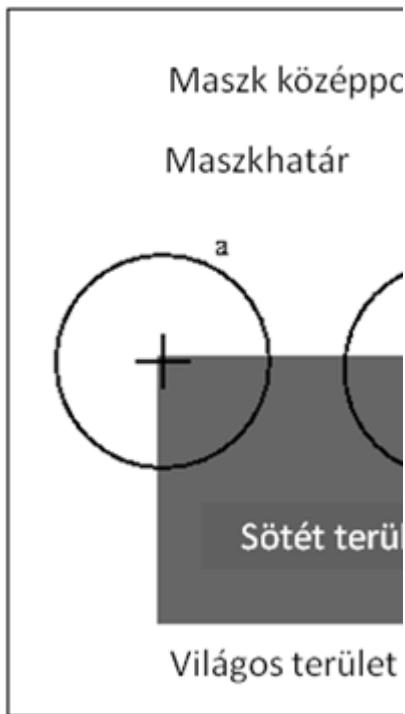
'Y' or 'T' csatlakozási

1.35. Canny: sarok effektus



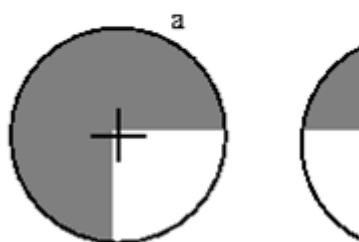
1.36. SUSAN algoritmus

- Konturkeresés nem differenciáló operátorral
- Smith és Brady ötlete
(<http://www.fmrib.ox.ac.uk/~steve/susan/susan/node1.html>)



- A maszk közepén levő pont a nucleus (középpont)
- Az USAN egy rövidítés, jelentése: a középponthoz hasonló intenzitásértékű szegmens (univalue segment assimilating nucleus)

A képpontok eltérő intenzitása



A képpontok hasonlósága

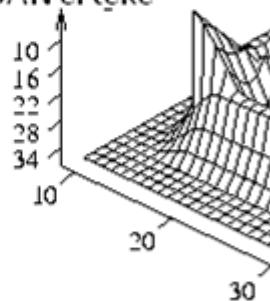
- A SUSAN jelentése:

smallest univalue segment assimilating nucleus

Eredeti kép



USAN értéke



- A középpont (nucleus) és a maszkban lévő pontok eltérése alapján megjelölés:

$$C(r, r_0) = \begin{cases} 1, & \text{ha } |I(r) - I(r_0)| \leq \Theta \\ 0, & \text{egyébként} \end{cases}$$

- A hasonló intenzitásúak száma

$$n(r_0) = \sum_r C(r, r_0)$$

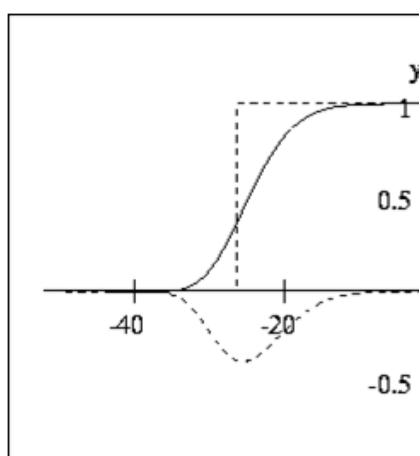
- A területen levő élek meghatározása az élválasz függvényel történik

(g geometriai küszöb)

$$R(r_0) = \begin{cases} g - n(r_0), & \text{ha } n(r_0) \leq g \\ 0 & \text{egyébként} \end{cases} \quad g = \frac{3}{4} n_{MAX}$$

- Az ugrásszerű átmenet elkerülése, stabilabb eredményt ad (LUT táblázat!)

$$C_2(r, r_0) = e^{-\left(\frac{|I(r) - I(r_0)|}{t}\right)^6}$$



2. Jellemző pontok keresése

2.1. Sarokpont detektálás

- Gyakran keresünk jellemző sarokpontokat a képen, ezekben a pontokban legalább két irányban erőteljes intenzitásváltozás van
- Alkalmazások: mozgás detektálás, sztereó illesztés, CBIR

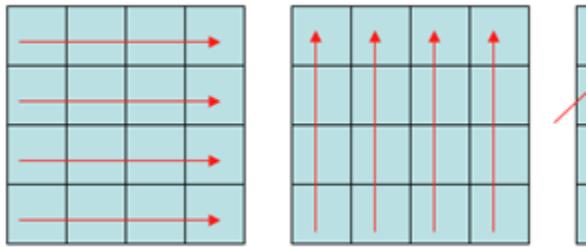
Módszerek:

- SUSAN algoritmus (lásd előbb, de más geometriai küszöbbel)
- Moravec operátor
- Harris sarokdetektáló



2.2. Moravec operátor

Számoljuk ki a az intenzitás változások varianciáját 4 irányban 4x4-es ablakokban



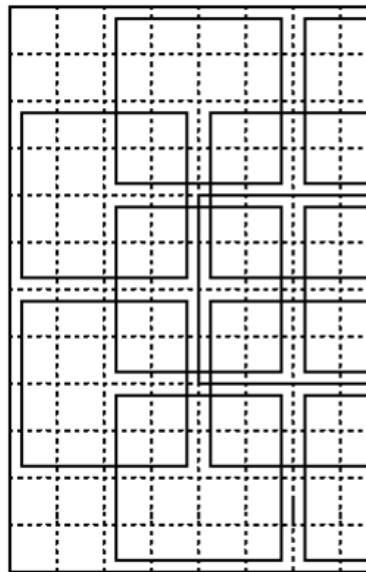
...

Válasszuk ki a minimumát a 4 irányban kiszámolt értékeknek

$$V(x, y) = \min(Vh(x, y), Vv(x, y), Vd(x, y), Va(x, y))$$

Egy 4 x 4-es, (x, y) középpontú ablak "érdekes", ha az alábbi 12 x 12-es szomszédságában, összesen 25 ablak közül lokális maximum

$$I(x, y) = \begin{cases} 1, & \text{ha } V(x, y) \geq V(p, q), \forall (p, q) \in N(x, y) \\ 0 & \text{egyébként} \end{cases}$$



2.3. Moravec - példa



2.4. Harris sarokdetektor

- Számítsuk ki a deriváltak közelítését, minden pontban (esetleg előtte simítsuk a képet): I_x, I_y
- Készítsük el a következő (gradiens momentum) mátrixot a pont valamely $(2n+1) \times (2n+1)$ ($1 < n < 10$) környezetében

$$\Delta I = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \quad M = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} \quad M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$
$$M_H = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

- Tulajdonképpen simítás egy környezetben – lehet más módon is megoldani

2.5. Harris sarokdetektáló

Számoljuk ki M_H sajátértékeit

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

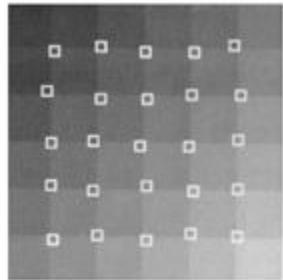
- Szimmetrikus mátrix: diagonalható, sajátértékek nem negatívak
- a sajátvektorok él irányt jelentenek, a sajátértékek él nagyságot
- Ha minden sajátérték elég nagy, akkor sarokpontot tároljuk el egy rendezett listában (a küszöb a hisztogramból származik: első völgy)

$$R = \min(\text{sajátértékek}) / (M_H) > Th$$

- Indulunk a legnagyobb értéktől (ez sarokpont), és töröljünk minden olyan tárolt pontot, ami már egy detektált sarokpont közelében van

2.6. Harris sarokdetektáló - példa

Küszöb a hisztogramból



2.7. Kanade-Lucas-Tomasi algoritmusa

Hasonló elv (Kanade-Lucas-Tomasi algoritmusa):

$$M_h = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

- Legyen $0 \leq k \leq 0.25$ skalár
 - Határozzuk meg a mátrix determinánsát (\det) és a főátlóban lévő elemek összegét (trace)
 - Küszöböljük az R kifejezést
- $$R = \det(M_h) + k \operatorname{trace}(M_h)^2 > Th$$
- k növekedésével érzéketlenebb a módszer



3. Elvárt helyzetű szakaszok detektálása

Adott futamra merőleges irányban keressük az élszakaszt

- Függőleges irányt tárgyalunk, de ez nem szükítés valójában

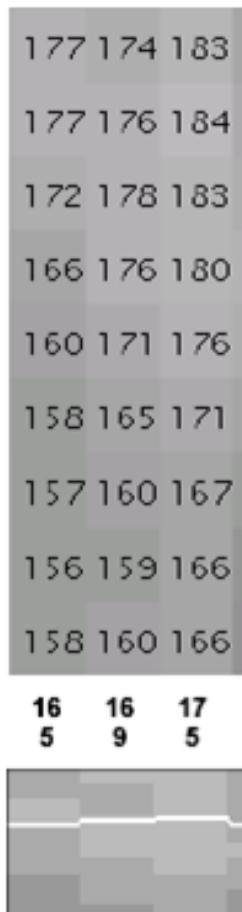
177	174	183
177	176	184
172	178	183
166	176	180
160	171	176
158	165	171
157	160	167
156	159	166
158	160	166

Lépések:

1. A futam irányára levetítés átlagos intenzitás számolásával (nem osztunk az oszlopan lévő pixelek számával egyenlő magas oszlopoknál)
2. Élerősség tömb elkészítése a futam mentén (differenciál szűrővel)

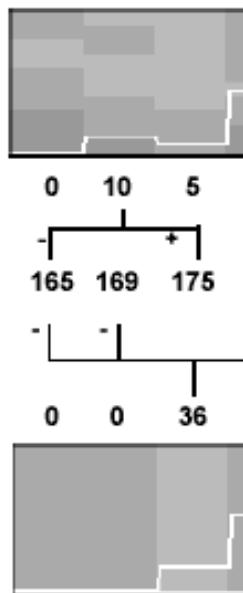
- a. Átlagos intenzitás tömbből számolunk
 - b. Intenzitásváltozások
 - c. Csúcsok: erős élek
3. Élszakasz meghatározása (lokális maximumok egy minimális küszöb felett)
4. Előre megadott feltételek vizsgálata az élekre vonatkozóan

3.1. Átlagos intenzitás számolása

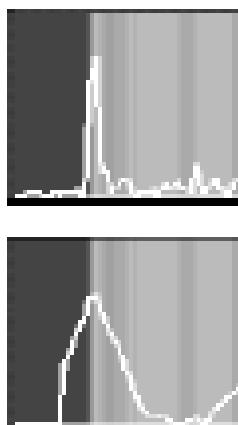


3.2. Élerősség tömb elkészítése

Nagyobb maszk: simít



Élszakasz meghatározása (lokális maximumok egy minimális küszöb felett)



3.3. Élfeltételek vizsgálata

- Minimális élerősség
- Elvárt élpozíció
- Élpár esetén az elvárt távolság
- Polaritás: világosból sötét, vagy fordítva
- Az előzetes feltételeket esetleg súlyozzuk: élkiértékelési függvény



Felhasznált és javasolt irodalom

- [1] R. C. Gonzales, R. E. Woods, *Digital Image Processing*, Pearson Education, Inc., 3rd ed., ISBN-13: 978-0-13-505267-9, p. 954. 2008.
- [2] D. A. Forsyth, J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, p. 792. 2003.
- [3] E. Trucco, A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, ISBN: 0-13-261108-2, p. 343. 1998.
- [4] S. M. Smith, J. M. Brady, *SUSAN – a new approach to low level image processing*, International Journal of Computer Vision, Vol. 23 (1) pp. 45–78. 1997.

7. fejezet - Képpiramisok

Vámossy Zoltán

A képpiramis módszer bemutatáshoz Gonzales-Woods [1], Forsyth-Ponce [2] és Trucco-Verri [3] könyveinek idevonatkozó fejezetét dolgoztuk fel, de egyes részeket Szeliski [4] műve alapján ismertetünk



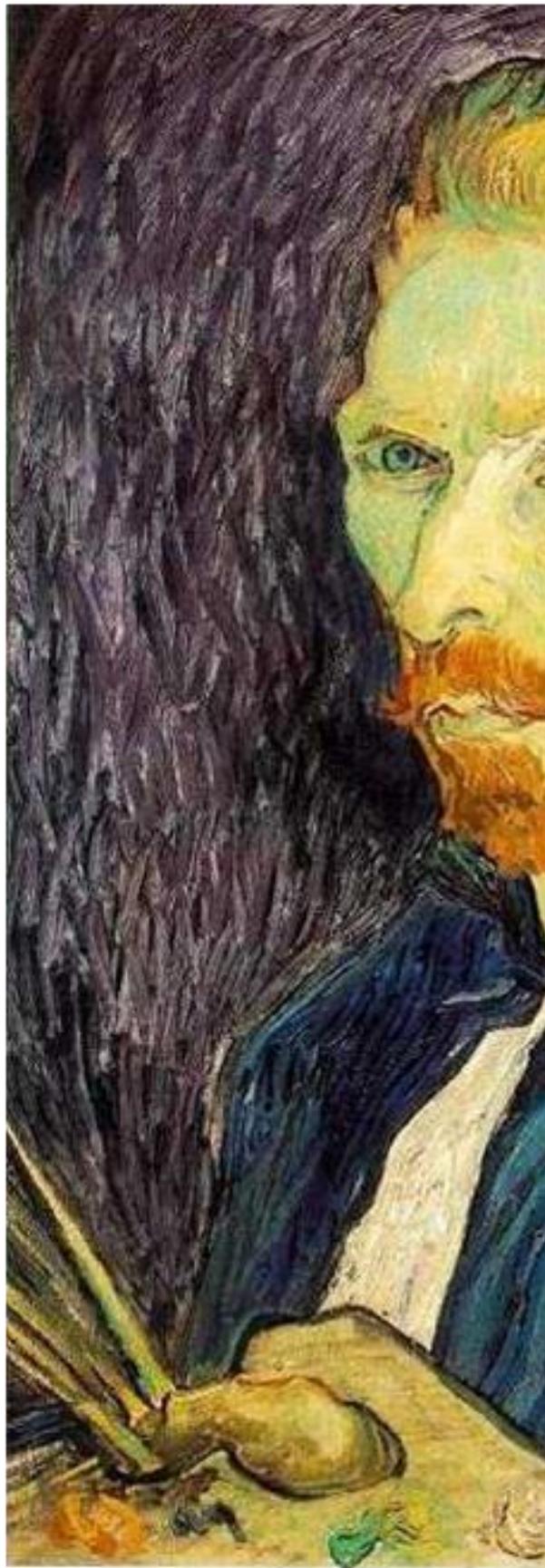
1. Képpiramisok bevezető

- Ha az objektumok képe túl kicsi, vagy nem elég kontrasztos, akkor általában nagyobb felbontással vizsgáljuk azokat
- Ha nagy méretűek, vagy kontrasztosak, akkor elegendő durva felbontás
- Ha mind kicsi, mind nagy, illetve alacsony és nagy kontrasztú objektumaink egyaránt vannak a képen, előnyös lehet különböző felbontással vizsgálni azokat
- A képpiramis olyan hatékony és egyszerű képrezentáció, aminek segítségével a kép több felbontását használjuk
- Más elnevezés: Felbontás hierarchiák (Resolution hierarchies)

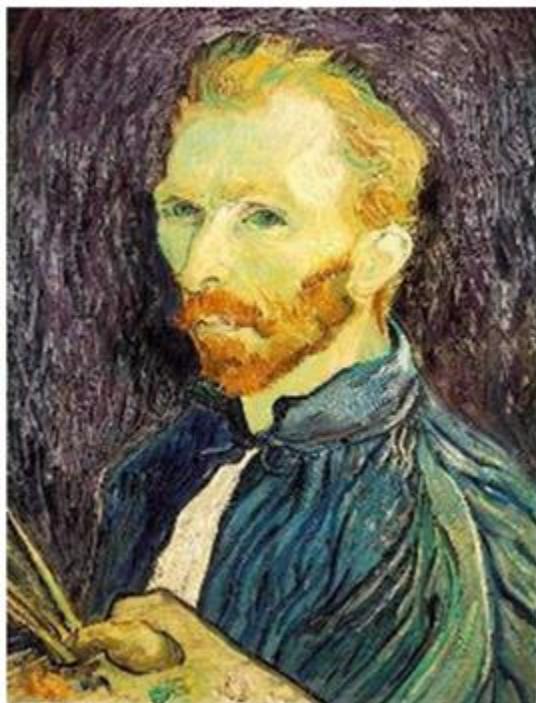
1.1. Skálázás

A kép túl nagy a megjelenítéshez. Hogyan csökkentsük?

Fele méret a cél.



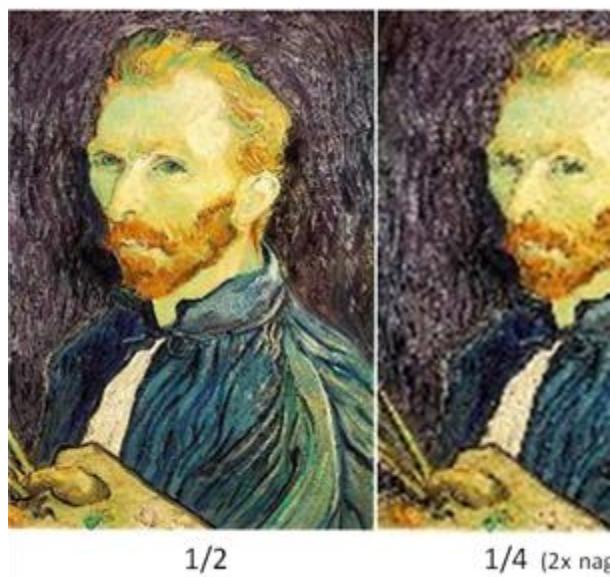
1.2. Sub-sampling (downsampling)



Minden második képpont elhagyásával 1/2 méretű kép

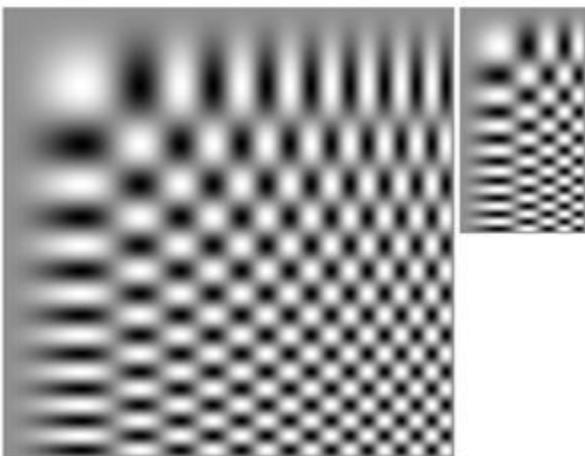
- Ezt nevezik *sub-sampling* műveletnek

1.3. Image sub-sampling - visszanagyítva



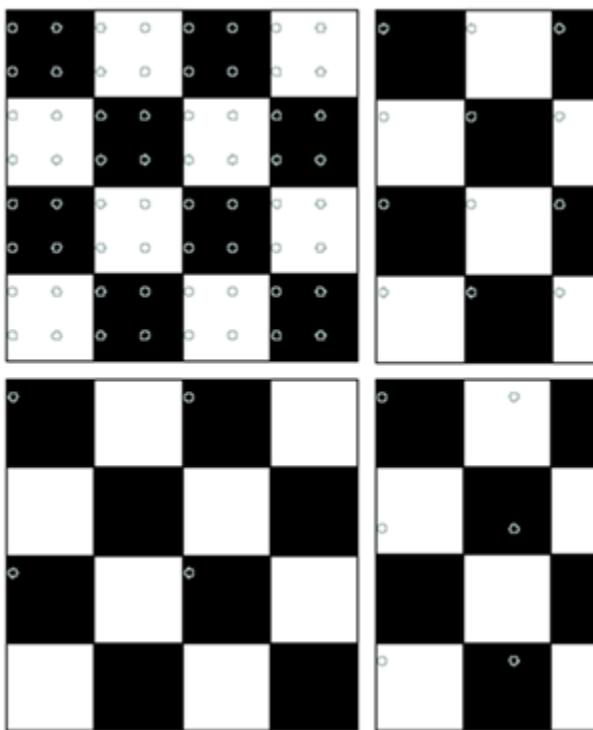
Miért néz ki olyan különösnek?

1.4. Sub-sampling minden második pixellel



Ha minden második pixellel készítjük a piramist, akkor az alacsonyabb szintek nem megfelelően reprezentálják a képet

1.5. Mintavételezés - 2D példa

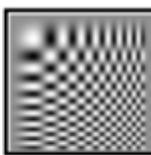


1.6. Simítás

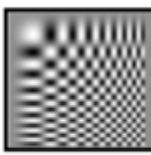
A mintavételezés során a magas frekvenciás jellemzők (élek) problémákhoz vezetnek

- Megoldás: mintavételezés előtt “élelnyomás”
- Alulátereszítő szűrőt kell használni: pl. átlagoló szűrő, vagy általános megoldás: Gauss szűrő használata

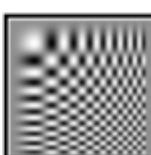
Simítás nélkül



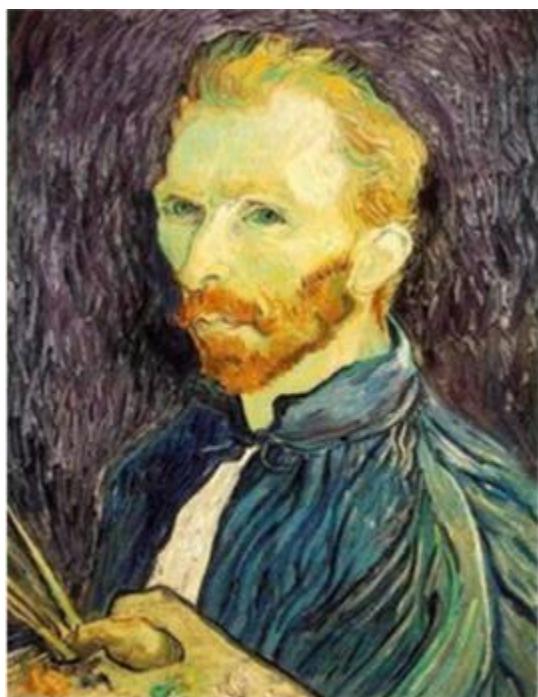
Simítás sigma 1 pixel



Simítás sigma 1.4 pixel



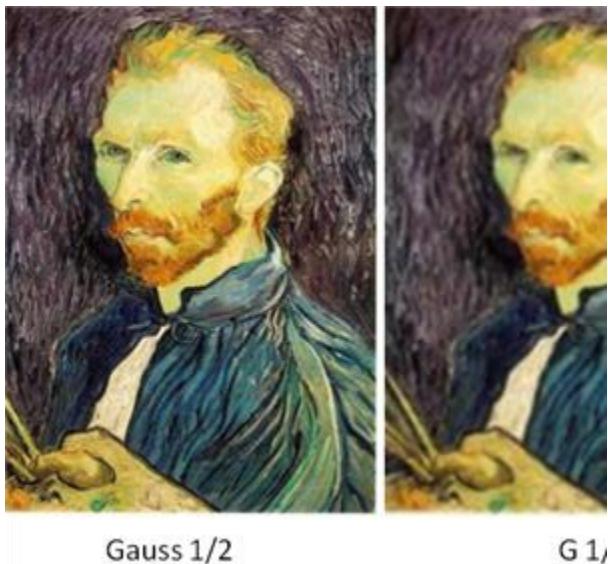
1.7. Sub-sampling Gauss szűrővel



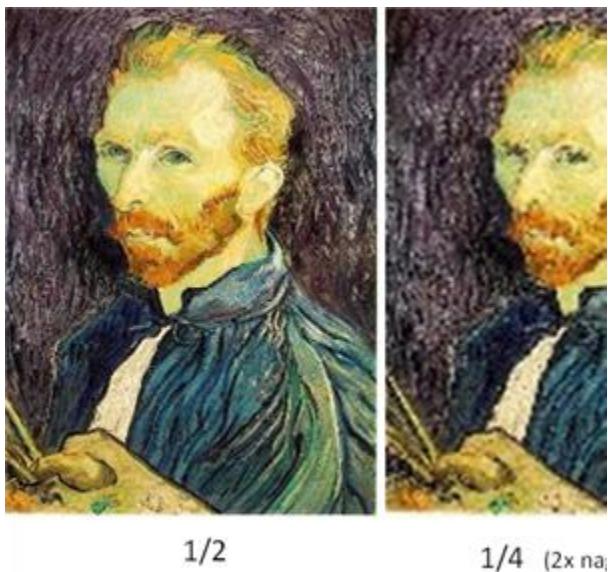
Gauss 1/2

Megoldás: szűrés, majd sub-sampling

1.8. Sub-sampling Gauss szűrővel - visszanagyítva



1.9. Csak sub-sampling...



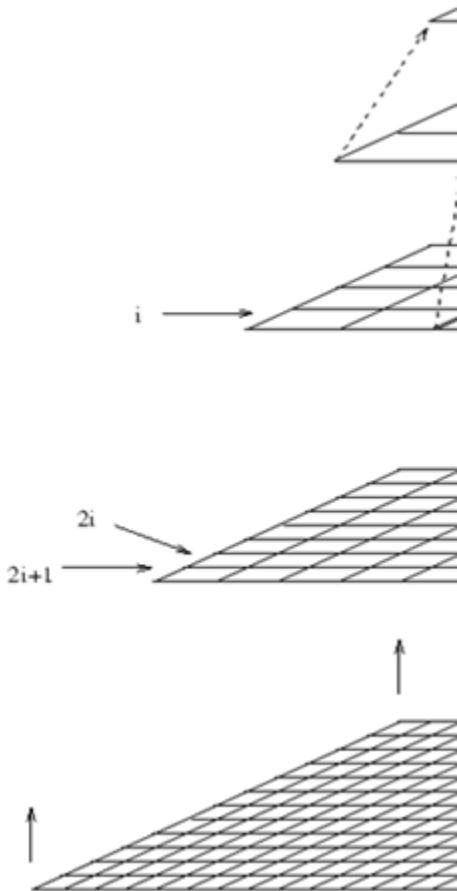
1.10. Képpiramisok

Cél: képek tömör reprezentációja, gyors algoritmusok készítése

- A képpiramisok (= felbontás hierarchiák) a kép különböző skálázású másolataiból épülnek fel
 - A piramis minden szintje az előző szint 1/4-e
 - A magasabb szint magasabb felbontást jelent
 - A legalacsonyabb szint a legkisebb felbontású
- (Megjegyzés: néha a szintek azonosítása éppen ellentétes e kijelentésekkel)
- A magasabb szint pixelértékeit “redukáló” (Reduce) függvény segítségével számoljuk

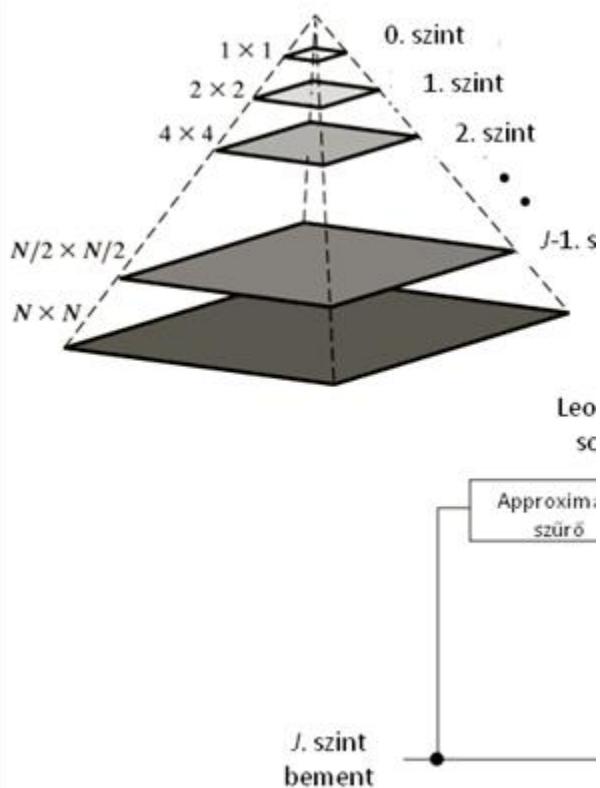
$$g_i = \text{REDUCE}[g_{i+1}]$$

1.11. Képpiramis



1.12. Piramisok készítése

- minden szinten van egy közelítő képünk és egy különbség (maradék) kép
- az eredeti kép (amely a piramis alapja) és az ö P közelítései a közelítő piramist építik fel
- a maradék outputok a “maradék piramist” építik fel
- mind a közelítő, mind a maradék piramisok iterációs módszerrel határozhatóak meg
- a $P+1$ szintű piramis a konstrukció algoritmusának P alkalommal történő futtatásakor keletkezik
- az első iterációban az eredeti $2J \times 2J$ méretű kép az input
- ebből készül a $J-1$ szintű approximációs és a J szintű maradék eredmény
- az iterációk során az előző iteráció eredményét használjuk az új lépés inputjaként



Minden iteráció három lépésből épül fel:

1. Számoljuk ki az input kép redukált felbontású közelítését. Ez szűréssel és pixellek leosztásával (downsampling by factor 2) történik
 - Szűrő: szomszédok átlagolása, v. Gauss szűrő, stb.
 - A közelítés pontossága függ a szűrőtől (lásd később)
2. A kapott output pixeleinek felszorzásával (upsampling by factor 2) és szűréssel készül a közelítő kép, aminek a felbontása megegyezik az inputéval.
 - A pixellek közötti interpolációs szűrő meghatározza, hogy mennyire jól közelítjük az inputot az 1. lépésben
3. Számoljuk ki a 2. lépében kapott közelítés és az 1. lépés inputjának különbségét (maradék). A különbség később az eredeti kép rekonstruálásához használható

1.13. Közelítő piramis és maradék piramis



1.14. Alkalmazási területek

Hasonló részek keresése

- Keressünk durva skálán, majd finomítsunk nagyobb felbontásnál

Élkövetés, mozgások vizsgálata

Minták keresése

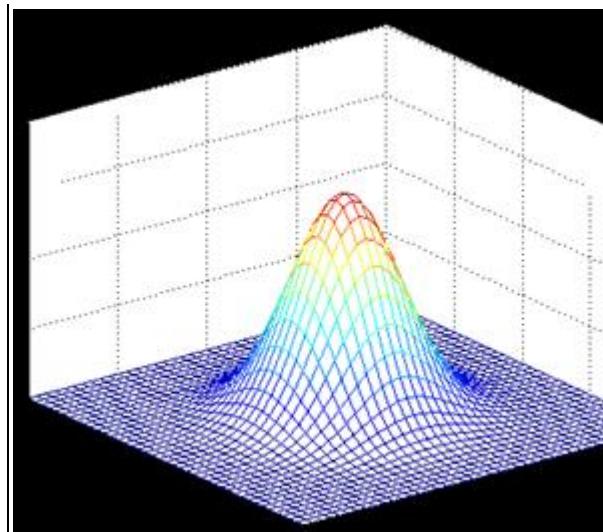
- Csíkok keresése
- Nagyon fontos textúrák vizsgálatánál

2. Gauss piramis

2.1. Gauss szűrő - emlékeztető

Tulajdonságok:

- Gauss*Gauss = másik Gauss
- Szimmetrikus
- Szeparálható
- Alul áteresztő
- Zajt elnyomja



$$G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x^2 + y^2)}{2\sigma^2}\right)$$

3. Gauss piramis

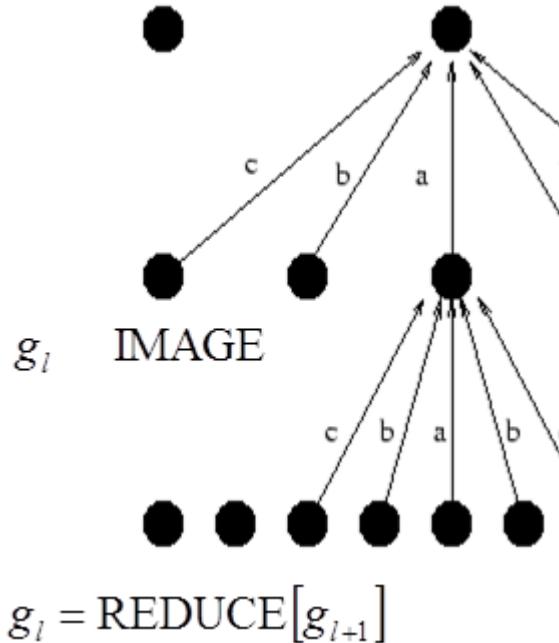
3.1. Gauss piramis 1D-ben

- Redukáló (Reduce) függvény meghatározása
- Legyen w Gauss szűrő

$$g_i(i) = \sum_{m=-2}^2 \hat{w}(m) g_{i+1}(2i+m)$$

$$g_i(2) = \hat{w}(-2) g_{i+1}(4-2) + \hat{w}(-1) g_{i+1}(4-1) + \\ + \hat{w}(0) g_{i+1}(4) + \hat{w}(1) g_{i+1}(4+1) + \hat{w}(2) g_{i+1}(4+2)$$

$$g_i(2) = \hat{w}(-2) g_{i+1}(2) + \hat{w}(-1) g_{i+1}(3) + \\ + \hat{w}(0) g_{i+1}(4) + \hat{w}(1) g_{i+1}(5) + \hat{w}(2) g_{i+1}(6)$$



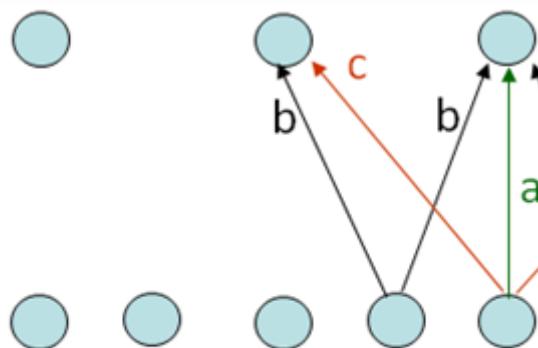
3.2. Redukáló függvény, konvolúciós maszk w

- Szimmetrikus konvolúciós maszk:

$$\begin{aligned} & [w(-2), w(-1), w(0), w(1), w(2)] \\ & w(i) = w(-i) \Rightarrow [c, b, a, b, c] \end{aligned}$$

- A maszk elemeinek összege 1: $a+2b+2c=1$

- Minden csomópont egy adott szinten ugyanannyi összsúllyal járul hozzá a következő szinthez:



$$a + 2b =$$

3.3. Konvolúciós maszkok (5×1)

$$w(0) = a$$

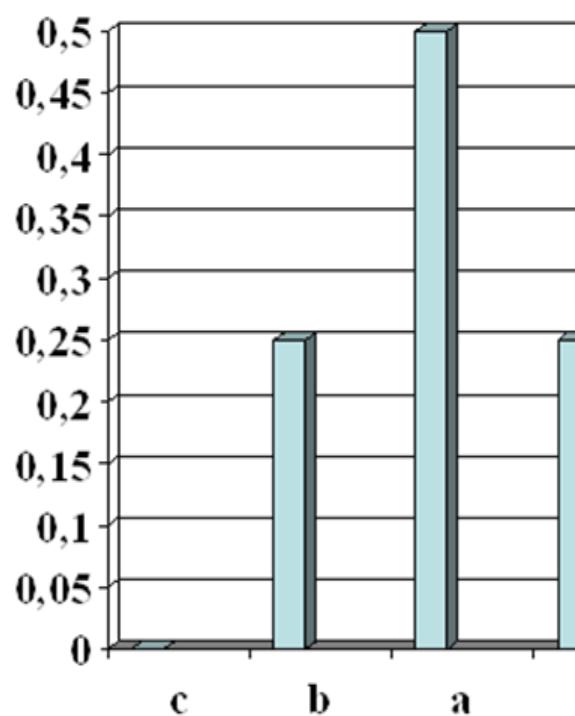
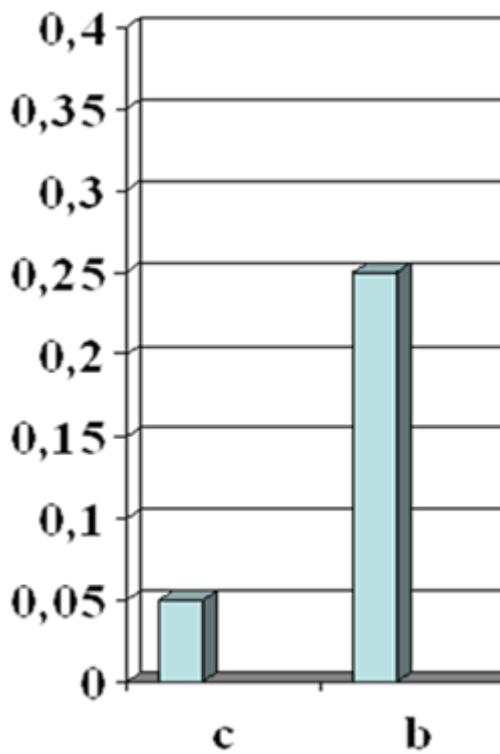
$$w(-1) = w(1) = \frac{1}{4}$$

$$w(-2) = w(2) = \frac{1}{4} - \frac{a}{2}$$

$a = 0.4$ - Gauss maszk

$a = 0.5$ - háromszög maszk

$a = 3/8$ - könnyen számolható maszk

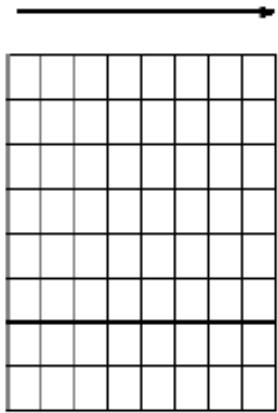


3.4. Gauss piramis megvalósítása képre

Gauss szeparálható:

$$\begin{aligned}\hat{I}(x, y) &= I(x, y) \otimes G(x, y) \\ \hat{I}(x, y) &= I(x, y) \otimes G(x) \otimes (y) \\ G(x) &= G^T(y) \text{ transponált}\end{aligned}$$

- Alkalmazzunk 1D maszkot a kép minden sorának módosítására
- Alkalmazzunk 1D maszkot az előzőleg kapott kép minden oszlopára



g_0

3.5. Gauss piramis példa



4. Laplace piramis

- Hasonló az élszűrt képekhez
- A legtöbb pixel 0
- Tömörítésre is használható
- Laplace piramis orientáció független

Laplace piramis készítése:

- Gauss piramis kiszámítása

$$g_k, g_{k-1}, g_{k-2}, \dots, g_2, g_1$$

- Laplace számítása: Gauss – „visszahízlalt (Expand) előző Gauss”

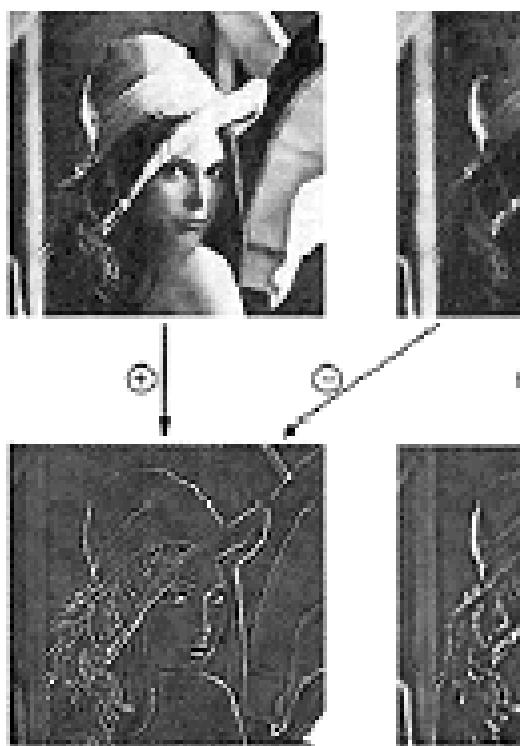
$$L_k = g_k - EXPAND(g_{k-1})$$

$$L_{k-1} = g_{k-1} - EXPAND(g_{k-2})$$

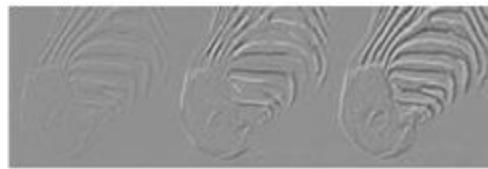
$$L_{k-2} = g_{k-2} - EXPAND(g_{k-3})$$

⋮

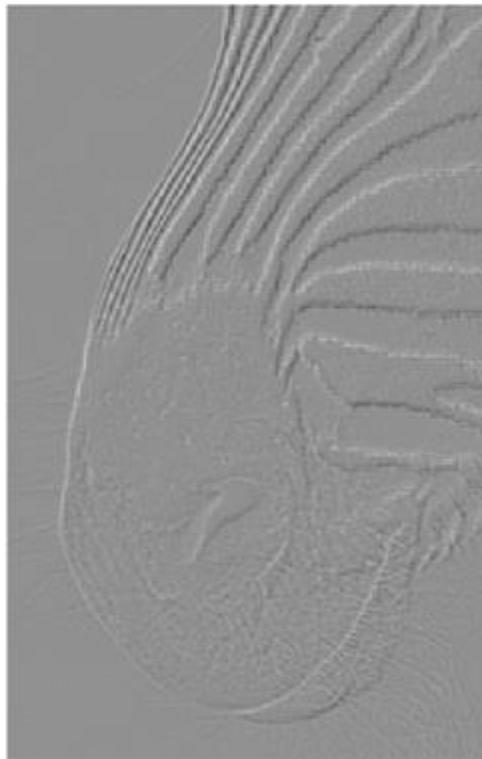
$$L_1 = g_1$$



4.1. Laplace piramis példa



512 256 128



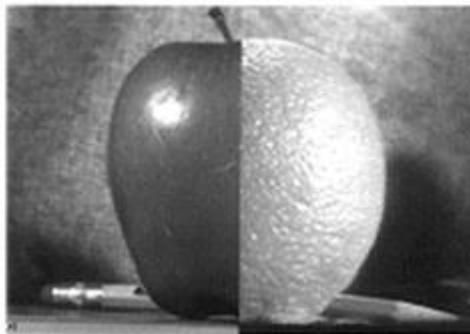
4.2. Képrekonstrukció piramisokból

Az eltárolt piramisokból az eredeti kép visszaállítható

- A Laplace piramis jól tömöríthető (a kép homogén részeinél)

$$\begin{aligned} g_1 &= L_1 \\ g_2 &= \text{EXPAND } (g_1) + L_2 \\ g_3 &= \text{EXPAND } (g_2) + L_3 \\ &\vdots \\ g_k &= \text{EXPAND } (g_{k-1}) + L_k \end{aligned}$$

4.3. Alma-narancs összeolvasztás

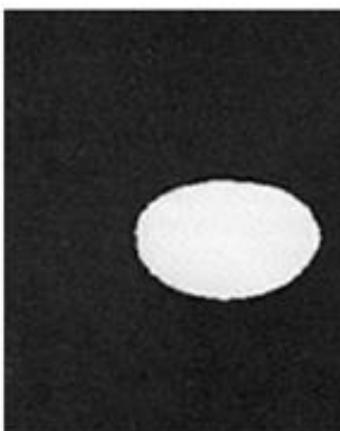


- Készítsük el a narancs kép Laplace piramisát (Ln)
- Készítsük el az alma kép Laplace piramisát (La)
- Készítsük el a következő összemásolt Lc piramist:
 - az alma La piramisának bal részét minden szinten és a narancs Ln piramis jobb oldalát minden szinten másoljuk egybe
 - Rekonstruáljuk a kombinált képet Lc-ből

4.4. Összeolvasztás maszkkal



(a)



(c)

Felhasznált és javasolt irodalom

- [1] R. C. Gonzales, R. E. Woods, *Digital Image Processing*, Pearson Education, Inc., 3rd ed., ISBN-13: 978-0-13-505267-9, p. 954. 2008.
- [2] D. A. Forsyth, J. Ponce, *Computer Vision: A Modern Approach*, Prentice Hall, p. 792. 2003.
- [3] E. Trucco, A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, ISBN: 0-13-261108-2, p. 343. 1998.
- [4] R. Szeliski:, *Computer Vision: Algorithms and Applications*, Springer, ISBN: 978-1-84882-934-3, p. 812. 2011.

8. fejezet - Képek szegmentálásának módszerei

Sergyán Szabolcs

A fejezet nagyrészt a Gonzalez-Woods [1] és Sonka-Hlavac-Boyle [6] széles körben használt könyvek alapján került feldolgozásra, valamint egyes részeknél merítettük Shah jegyzetéből [5] és a Trucco-Verri könyvból [8]. A Hough transzformáció ismertetésénél Nixon és Aguado könyvét [3], a binarizálásnál pedig Parker művét [4] vettük alapul. A színes képekkel kapcsolatos részek Matas PhD disszertációjából [2] származnak. A fejezetben közölt MATLAB kódokat a Svoboda-Kybík-Hlavac könyvből [7] vettük.



Képek szegmentálása

Szegmentálás során a képen olyan homogén régiókat határozunk meg, melyek pixelei egymással összefüggők.

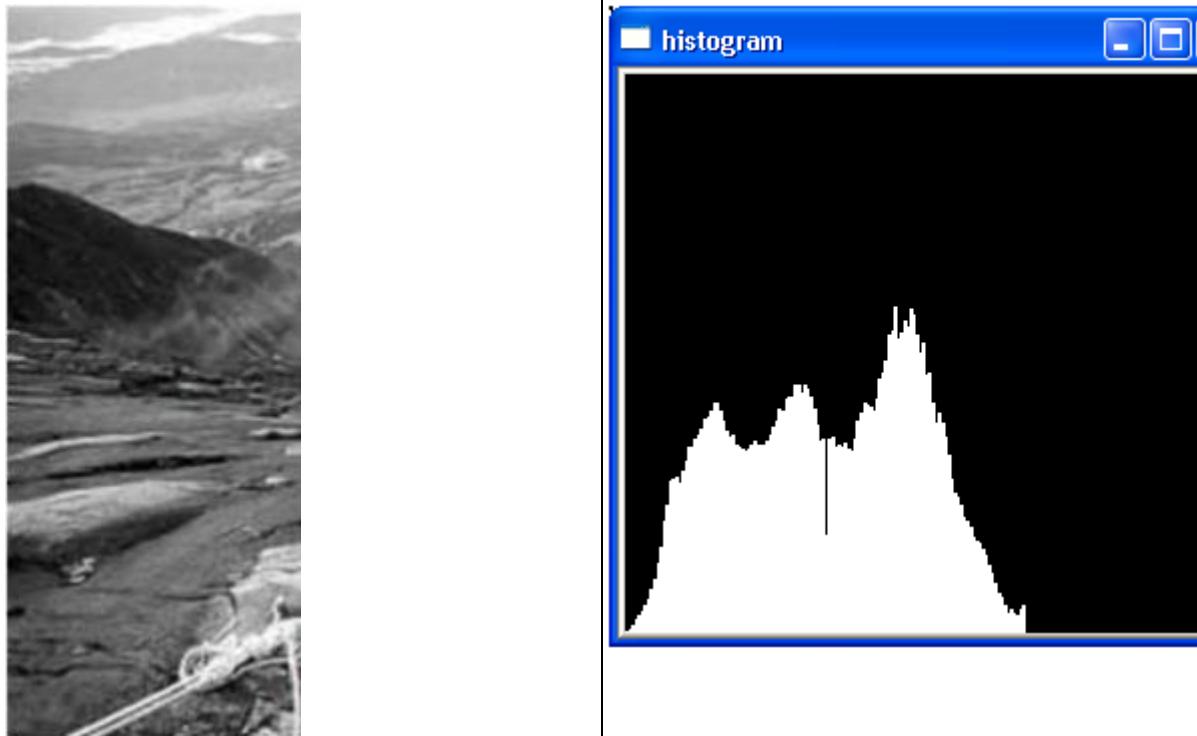
A fejezetet az alábbi részfejezetekre bontva tárgyaljuk:

- **Küszöbölés:** A kép hiszrogramjának küszöbölésével határozzuk meg, hogy mely intenzitású régiók tartoznak egy régióba. A módszer nem vizsgálja a pixelek összefüggőségét.
- **Hatórvonal alapú szegmentálás:** A régió határvonalának detektálásával határozzuk meg a régiót. Ebben a fejezetben tárgyaljuk a Hough transzformációt is, mellyel előre ismert alakú görbékkel (pl.\ egyenesek, körök, ellipszisek) lehet megtalálni a képen.
- **Összefüggő komponens analízis:** Homogén pixel térben összefüggő osztályokba sorolása a szomszédsági viszonyok figyelembe vételével.
- **Régió alapú szegmentálás:** Olyan technikák tárgyalása, melyek egyszerre képesek a régió homogenitását és összefüggőségét vizsgálni.

1. Küszöbölés

1.1. Hiszrogram

- A hiszrogram olyan függvény, amely minden lehetséges szürkeárnyalathoz (intenzitáshoz) hozzárendeli az adott árnyalatú pixelek számát a képen.
- Ha a hiszrogram értékeit elosztjuk a képen található pixelek számával, akkor az egyes intenzitások előfordulási valószínűségét adja meg a függvény. (Az így előállított normalizált hiszrogram megfelel a valószínűségi eloszlások sűrűség függvényének.)



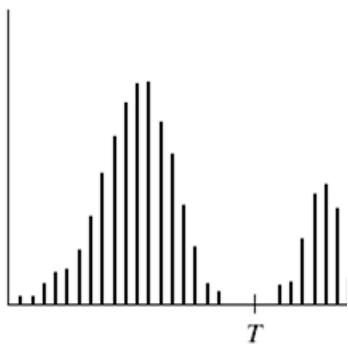
1.2. Hisztogram alapú technika

- A szürkeáryalatos képet (I) bináris képpé (B) konvertáljuk az alábbi módon:

$$B(i, j) = \begin{cases} 0, & \text{ha } I(i, j) < T \\ 1, & \text{ha } I(i, j) \geq T, \end{cases}$$

ahol i és j az adott pixel sor-, illetve oszlopkordinátája.

- Fő kérdés: Hogyan lehet meghatározni a küszöbértéket (T)?



1.3. Küszöb meghatározása

Intenzitás középérték mint küszöb

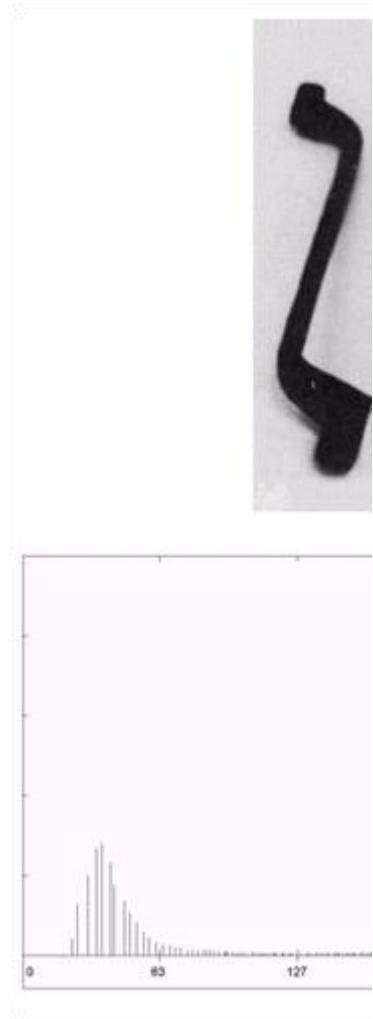
- Legyen a küszöb a teljes kép intenzitásainak középértéke, azaz

$$T = \frac{\sum_{i=1}^M \sum_{j=1}^N I(i, j)}{M * N},$$

ahol M és N az I kép sorainak, valamint oszlopainak a száma.

- A pixelek körülbelül fele lesz fekete, másik fele pedig fehér a binarizált képen. (Pontos felezés akkor következik be, ha az intenzitások eloszlása egyenletes.)

1.4. Küszöbölés eredménye

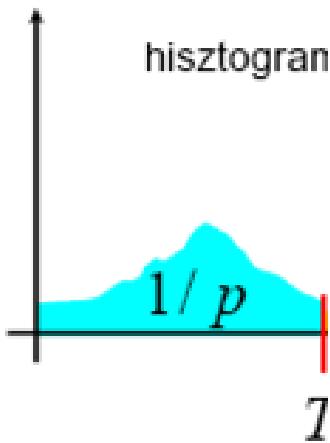


1.5. Küszöb meghatározása

p-csempe módszer

- Tudjuk (sejtjük) a világos pixelek arányát a képen (p).
- A kép hisztogramján a 0. vödörtől¹ addig lépegetünk tovább, amíg a már megvizsgált (sötét) pixelek aránya el nem éri $(1-p)$ -t. Ahol eléri, ott lesz a küszöb.

¹Vödörnek nevezünk egy intenzitás tartományt. Hisztogram esetén a hisztogram értéke egy vödörben megegyezik az adott intenzitású pixelek számával (gyakoriságával) a képen.



Hisztogram csúcsok kiválasztása

- Keressünk meg a hisztogramon két csúcsot. A köztük lévő legkisebb értékű vödörnél lesz a küszöb.
- Hogyan keressük meg a két szignifikáns csúcsot?
 - Az egyik csúcs egyszerűen megtalálható: a legnagyobb hisztogramérték.
 - A másikat a következő módon kaphatjuk:

$$\max \left\{ (k-j)^2 h[k] \right\}_{0 \leq k \leq M}$$

ahol h a hisztogram, j az első csúcs helye, M a vödrök száma a hisztogramban.

Iteratív küszöbölés

1. Válasszunk egy kiindulási küszöböt: T .
2. Küszöböljünk a választott T -vel: G_1 -be tartozik az összes T -nél kisebb intenzitású pixel, a többi pedig G_2 -be.
3. Számoljuk ki a G_1 és G_2 -beli intenzitások számtani közepét: μ_1, μ_2 .
4. Számítsunk ki egy új küszöböt:

$$T = \frac{\mu_1 + \mu_2}{2}$$

5. Folytassuk a 2. és 4. lépés közti részt addig, amíg az új és a régi küszöb közti különbség nem lesz kisebb, valamelyen T_0 értéknél.

1.6. Iteratív küszöbölés megvalósítása

MATLAB függvény: [out, threshold] = imthresh(im)

Bemenet:

```
im - m x n méretű szürkeárnyalatos kép 0 és 255
közötti intenzitásokkal
```

Kimenetek:

```
out - m x n méretű bináris kép, melyen 1 jelöli a
küszöbnél nagyobb intenzitású pixeleket, 0 pedig a többet
```

```
threshold - a küszöbérték
```

Először meg kell határozni a kép hisztogramját és kumulatív hisztogramját.

```
1 histogram = hist(im(:), 0:255);
hist_times_gray = cumsum(histogram.*[0:255]);
3 cumulative_histogram = cumsum(histogram);
```

Codes/imthresh.m

Első közelítésben háttérpixelként a négy sarokban található pixelt vesszük figyelembe, ezek középértéke lesz mean_1. Az összes többi pixel objektumpixel, melyek középértéke mean_2.

```
1 [m, n] = size (im);
sum_background = sum(im([1 m n*(m-1)+1 n*m]));
3 num_pix_background = 4;
mean_1 = sum_background / num_pix_background;
5 mean_2 = (sum(im(:)) - sum_background) / (n*m - num_pix_background);
threshold = ceil((mean_1 + mean_2) / 2);
```

Codes/imthresh.m

A küszöb alapján meghatározzuk az új háttér és objektum pixeleket, majd ezek segítségével az új küszöbértéket. Mindezt addig folytatjuk, amíg a küszöb nem stabilizálódik.

```
1 thresholdold = 0;
while threshold ~= threshold_old
3     threshold_old = threshold;
     mean_1 = hist_times_gray(threshold) / cumulative_histogram(threshold);
5     mean_2 = (hist_times_gray(end) - hist_times_gray (threshold + 1)) / ...
(cumulative_histogram(end) - cumulative_histogram(threshold + 1));
7     threshold = ceil((mean_1 + mean_2) / 2);
end
9 out = im >= threshold;
```

Codes/imthresh.m

1.7. Küszöb meghatározása

Iteratív küszöbölés hisztogramok használatával

- T_0 legyen a kiindulási küszöb, amelyet valamely korábbi módszer alkalmazásával határozunk meg.

$$T_k = \frac{\sum_{i=0}^{T_{k-1}} i \cdot h[i]}{2 \sum_{i=0}^{T_{k-1}} h[i]} + \frac{\sum_{j=T_{k-1}+1}^N j \cdot h[j]}{2 \sum_{j=T_{k-1}+1}^N h[j]},$$

ahol h a szürkeárnyalatos hisztogram.

- Az eljárás addig megy, amíg $T_k = T_{k+1}$ nem teljesül

1.8. Élpixelek használata

- Az élpixelek közel vannak egy objektum és a háttér határához, vagy két objektum határához.
- Az élpixelek néha a háttérhez, néha az objektumhoz tartoznak.

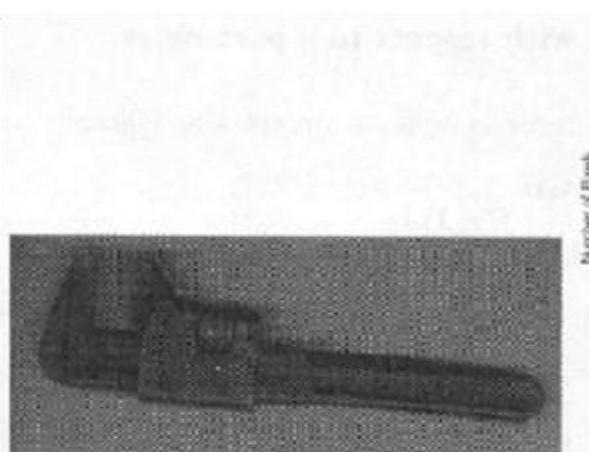
1. Végezzünk el egy konvolúciót valamelyen élszűrővel (pl.\ Laplace)

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

2. A szűrt képen válasszuk ki azon pixeleket, melyek intenzitása a maximális intenzitás 85-ánál nem kisebb.
3. A kiválasztott pixelek (eredeti) intenzitásait használva képezzünk egy hisztogramot, amelyen a korábban tárgyalt küszöbölések valamelyikét hajtsuk végre.

1.9. Hiszterézes küszöbölés

- Ha nincs "tiszta" völgy a hisztogramban, akkor a háttérben sok olyan pixel van, aminek az intenzitása megegyezik az objektum pixelek intenzitásával és fordítva.
- Két küszöböt definiálunk a völgy két szélénél.
- A nagyobb küszöb feletti pixelek objektumhoz, a kisebb alattiak a háttérhez tartoznak.
- A két küszöb közötti részben lévő pixelek akkor tartoznak objektumhoz, ha létezik az objektumhoz sorolt szomszédos pixele, egyéb esetben a háttérhez soroljuk azokat.



(a) Original image



1.10. Otsu algoritmus

- Keressük meg azt a T küszöbszámot, amely maximalizálja az objektum-háttér közötti varianciát (szórás négyzetet).
- Homogén régióknak a varianciája kicsi.
- Bimodális² hisztogramot tételez fel.
- p_i az i -edik szürkeségi érték gyakorisága a képen. Ez a normalizált hisztogram i -edik értéke is. L a lehetséges szürkeségi értékek száma.
- A háttér (B), valamint az előtér (O) pixelek gyakorisága adott T küszöb mellett:

$$q_B(T) = \sum_{i=1}^T p_i \quad q_O(T) = \sum_{i=T+1}^L p_i = 1 - q_B(T)$$

- A háttér, az előtér és a teljes kép pixeleinek középértéke:

$$\mu_B(T) = \frac{\sum_{i=1}^T i p_i}{q_B(T)} \quad \mu_O(T) = \frac{\sum_{i=T+1}^L i p_i}{q_O(T)} \quad \mu = \sum_{i=1}^L i p_i$$

Varianciák:

$$\sigma_B^2(T) = \frac{\sum_{i=1}^T (i - \mu_B(T))^2 p_i}{q_B(T)}$$

$$\sigma_O^2(T) = \frac{\sum_{i=T+1}^L (i - \mu_O(T))^2 p_i}{q_O(T)}$$

$$\sigma^2(T) = \sum_{i=1}^L (i - \mu)^2 p_i$$

- A súlyozott osztályon belüli variancia:

$$\sigma_w^2(T) = q_B(T) \sigma_B^2(T) + q_O(T) \sigma_O^2(T)$$

- Az osztályok közötti variancia:

$$\sigma_b^2(T) = q_B(T) q_O(T) [\mu_B(T) - \mu_O(T)]^2$$

- A teljes variancia (az osztályon belüli és az osztályok közötti varianciák összege):

$$\sigma^2 = \sigma_w^2(T) + \sigma_b^2(T)$$

- A teljes variancia nem függ a T küszöbtől, tehát a feladat vagy a $\sigma_w^2(T)$ minimalizálása, vagy a $\sigma_b^2(T)$ maximalizálása.

1.11. Entrópia használata

- Legyen p_i az i -edik hisztogramvödörbe esés valószínűsége, azaz $h[i]$ és a kép méretének hányadosa.
- Legyen a sötét pixelek entrópiája t -vel küszöbölve:

²A bimodális eloszlás két különböző várható értékű normális eloszlás szuperpozíciójaként áll elő.

$$H_t = -\sum_{i=0}^t p_i \log p_i$$

- A kép teljes entrópiája:

$$H_T = -\sum_{i=0}^{255} p_i \log p_i,$$

ha 256 vödröt használunk.

- A halmozódott valószínűség a t szürkeségi intenzitásig:

$$P_t = -\sum_{i=0}^t p_i$$

Meg kell találni azt a t küszöbértéket, mely az alábbi $f(t)$ függvényt maximalizálja:

$$\begin{aligned} f(t) &= \frac{H_t}{H_T} \frac{\log P_t}{\log(\max\{p_0, p_1, \dots, p_t\})} + \\ &+ \left[1 - \frac{H_t}{H_T}\right] \frac{\log(1 - P_t)}{\log(\max\{p_{t+1}, p_{t+2}, \dots, p_{255}\})} \end{aligned}$$

Legyen

$$H_b(t) = -\sum_{i=0}^t \frac{p_i}{P_t} \log\left(\frac{p_i}{P_t}\right)$$

és

$$H_w(t) = -\sum_{i=t+1}^{255} \frac{p_i}{1 - P_t} \log\left(\frac{p_i}{1 - P_t}\right).$$

Meg kell találni azt a t értéket, mely maximalizálja a

$$H = H_b(t) + H_w(t)$$

értéket.

Legyen

$$S_b(t) = \log\left(\sum_{i=0}^t p_i\right) + \frac{1}{\sum_{i=0}^t p_i} \left[E(p_t) + E\left(\sum_{i=0}^{t-1} p_i\right) \right]$$

és

$$S_w(t) = \log\left(\sum_{i=t}^{255} p_i\right) + \frac{1}{\sum_{i=t}^{255} p_i} \left[E(p_t) + E\left(\sum_{i=t+1}^{255} p_i\right) \right],$$

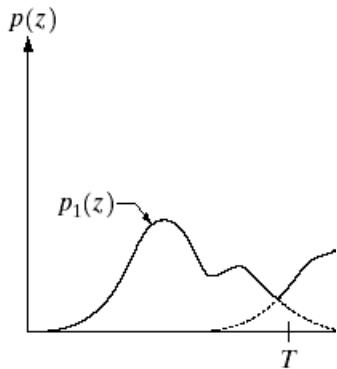
ahol

$$E(x) = -x \log(x).$$

A feladat az $S_b(t) + S_w(t)$ érték maximalizálása valamely t értékkel.

1.12. Minimális hibájú küszöbölés

- Legyen $p_1(x)$ és $p_2(x)$ a képen található objektum és háttér pixeleinek intenzitáseloszlása (sűrűségfüggvénye).



- Jelölje P_1 annak a valószínűségét, hogy egy véletlenszerűen kiválasztott pixel objektumpixel, P_2 pedig azt, hogy a pixel a háttérhez tartozik. Tegyük fel, hogy $P_1+P_2=1$, azaz minden pixel a két osztály közül valamelyikbe tartozik.
- T jelölje a küszöböt, az ennél kisebb intenzitású pixelek tartoznak a háttérhez, a többi pedig az objektumhoz.
- Annak a valószínűsége, hogy egy objektumpixelt tévesen osztályozunk:

$$E_1(T) = \int_{-\infty}^T p_2(x) dx$$

- Annak a valószínűsége, hogy egy háttérpixelt tévesen osztályozunk:

$$E_2(T) = \int_T^\infty p_1(x) dx$$

A teljes hiba valószínűsége pedig:

$$E(T) = P_2 E_1(T) + P_1 E_2(T)$$

- A hiba akkor lesz minimális, ha a deriváltja 0, azaz

$$P_2 \left(p_2(T) - \lim_{x \rightarrow -\infty} p_2(x) \right) + P_1 \left(\lim_{x \rightarrow +\infty} p_1(x) - p_1(T) \right) = 0$$

- Kihasználjuk, hogy p_1 és p_2 sűrűségfüggvények, az alábbi összefüggéseket kapjuk:

$$P_1 p_1(T) = P_2 p_2(T)$$

- Tételezzük fel, hogy a képen található objektum és a háttér pixelek intenzitása is normális eloszlást követ, azaz az objektumra

$$p_1(g) = \frac{1}{\sigma_1 \sqrt{2\pi}} e^{-\frac{(g-\mu_1)^2}{2\sigma_1^2}},$$

a háttérre pedig

$$p_2(g) = \frac{1}{\sigma_2 \sqrt{2\pi}} e^{-\frac{(g-\mu_2)^2}{2\sigma_2^2}}.$$

- A küszöb ott lesz, ahol a két függvény grafikonja metszi egymást, azaz

$$\frac{P_1}{\sigma_1 \sqrt{2\pi}} e^{-\frac{(T-\mu_1)^2}{2\sigma_1^2}} = \frac{P_2}{\sigma_2 \sqrt{2\pi}} e^{-\frac{(T-\mu_2)^2}{2\sigma_2^2}}.$$

- Az egyenlet megoldása érdekében vesszük minden két oldal logaritmusát:

$$\log P_1 - \log \sigma_1 - \frac{(T - \mu_1)^2}{2\sigma_1^2} = \log P_2 - \log \sigma_2 - \frac{(T - \mu_2)^2}{2\sigma_2^2}$$

- A kapott egyenlet T -re nézve másodfokú, tehát "könnyen" megoldható.

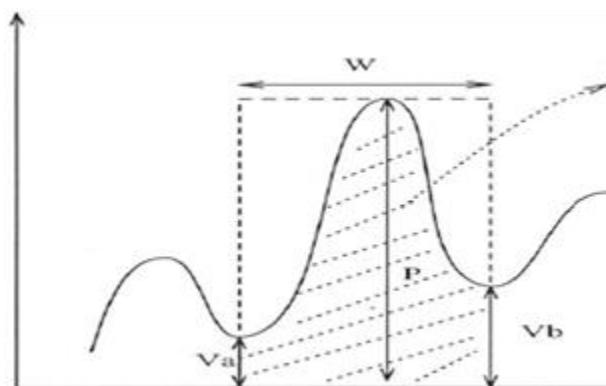
1.13. Több küszöb használata

$$B_1(x,y) = \begin{cases} 1 & \text{ha } f(x,y) < T_1 \\ 0 & \text{egyébként} \end{cases}$$

$$B_2(x,y) = \begin{cases} 1 & \text{ha } T \leq f(x,y) < T_2 \\ 0 & \text{egyébként} \end{cases}$$

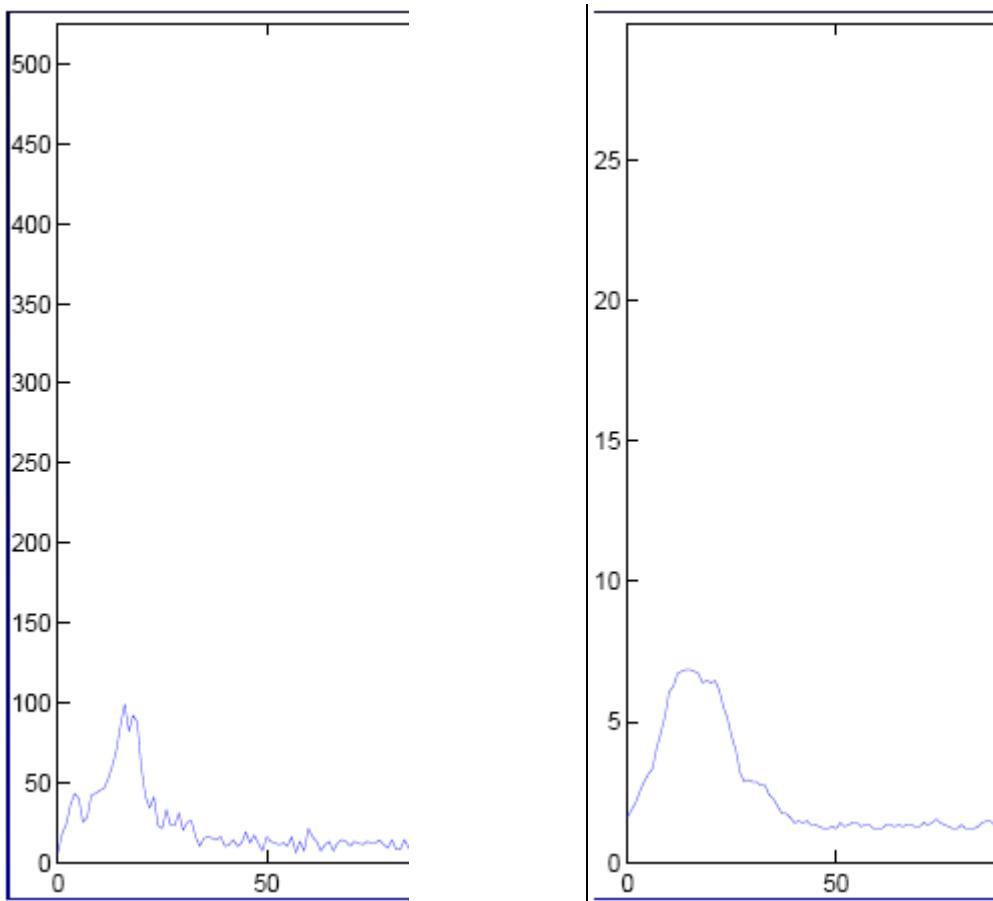
$$B_3(x,y) = \begin{cases} 1 & \text{ha } f(x,y) \geq T_2 \\ 0 & \text{egyébként} \end{cases}$$

1.14. Peakiness teszt



$$\text{Peakiness} = \left(1 - \frac{V_a + V_b}{2P}\right) \cdot \left(1 - \frac{N}{W \cdot P}\right)$$

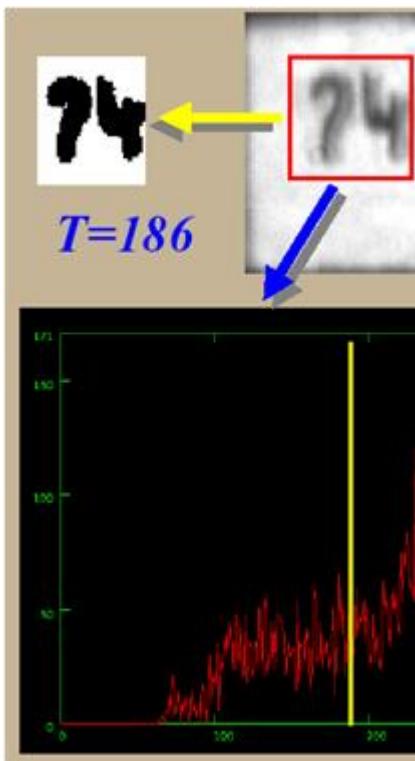
- Peakiness teszt alkalmazása előtt érdemes simítani a hisztogramot.



- A simításhoz pl. 1×15 méretű átlagoló maszkot használhatunk, akár többször egymás után alkalmazva.

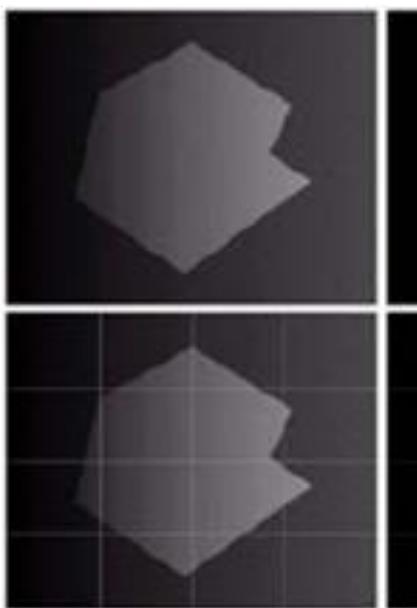
1.15. Adaptív (alkalmazkodó) küszöbölés

- A globális küszöbölési technikák a teljes kép (globális) hisztogramja alapján határoznak meg küszöböt, míg az adaptív eljárások a kép egyes helyein más-más küszöböt határoznak meg.



1.16. Adaptív küszöbölés

Globális és adaptív küszöbölés:



1. sor: Eredeti kép és a globális küszöbölés eredménye

2. sor: Amennyiben az eredeti képet részekre osztjuk, majd minden rész esetén külön-külön határozunk meg küszöb értékeket, akkor a jobb oldali bináris képet kapjuk.

- Meghatározzuk az (x,y) koordinátájú pixel valamely $k \times k$ méretű környezetében lévő pixelek intenzitásának átlagát: $\text{mean}(x,y)$
- Az (x,y) koordinátához tartozó küszöb:

$$T(x,y)=\text{mean}(x,y)-C,$$

ahol C egy előredefiniált konstans.

- Az átlag helyett lehet mediánt is alkalmazni.



1.17. Niblack algoritmus

- Az (x,y) koordinátájú pixelhez tartozó küszöb:

$$T(x,y)=\mu(x,y)+k\cdot\sigma(x,y),$$

ahol $\mu(x,y)$ az (x,y) koordinátájú pixel valamely környezetében lévő intenzitások átlaga, $\sigma(x,y)$ pedig az intenzitások szórása.

- k a szórás figyelembevételének mértéke:
 - Sötét objektum: $k<0$
 - Világos objektum: $k>0$
 - Általában $|k| \in \{0.2, 0.5\}$
- A környezet mérete általában 15×15

1.18. Hisztogram klaszterezés

A hisztogram klaszterezést használó algoritmusok esetén nem küszöbértéket határozunk meg a hisztogram figyelembe vételevel, hanem az egyes intenzitás értékeit előre nem definiált számú osztályokba (ún. klaszterekbe) soroljuk, majd ezt követően az azonos klaszterbe tartozó intenzitású pixelek teljesítik a homogenitási kritériumot.

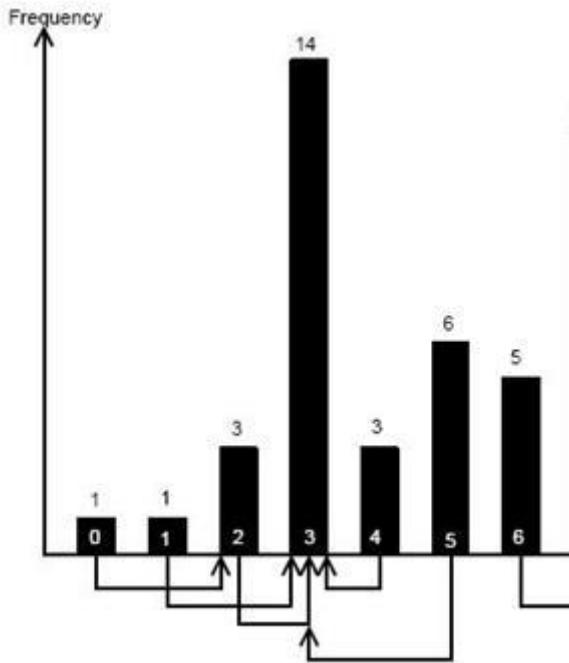
Az alábbi két konkrét algoritmust mutatjuk be:

- Csúcsok és természetes intervallumok
- Színes képek hisztogramjának klaszterezése

1.19. Csúcsok és természetes intervallumok

- Első lépésként meghatározzuk a kép $n \times n$ vödrű hisztogramját.

- Ezután minden vödörre meghatározzuk, hogy az §§ sugarú környezetében melyik vödörnek legnagyobb az értéke, és oda mutatunk egy pointerrel.
- Végül az ily módon összekapcsolt vödröket egy hisztogram klaszternek tekintjük.



1.20. Színes képek hisztogramjának klaszterezése

- RGB színtérből áttérünk az ún. kromaticitás síkra

$$ch_y = \frac{R}{R+G+B} \quad ch_g = \frac{G}{R+G+B}$$

- A kromaticitás sík fölött készítünk egy kétváltozós hisztogramot valamelyen rögzített vödörszámmal (pl. 6x6).

•

0	4
17	65
4	31
0	0
0	0
0	0

0	4
17	65
4	31
0	0
0	0
0	0

0	4
17	65
4	31
0	0
0	0
0	0

- Az azonos klaszterbe tartozó pixelek alapján már tudunk binarizálni.

2. Határvonal alapú szegmentálás

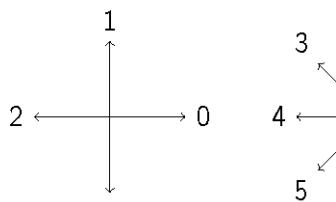
- Ebben az alfejezetben olyan algoritmusokat tárgyalunk, melyek egy régió határvonalának megtalálásával határozzák meg a régiót.
- Az egyes eljárások általában az eredeti kép élpixeleiire épülnek, tehát a módszereket minden esetben megelőzi valamilyen éldetektálási módszer alkalmazása.
- Az alábbi eljárásokat tárgyaljuk:
 - Hatóvonalak meghatározása
 - Hough transzformáció adott alakzatok detektálására

2.1. Belső határvonal bejárás

- A kép bal felső sarkából indulva keressük meg egy új régió bal felső pixelét. A megtalált P_0 pixel a régió legfelső pixelei közül a legbaloldalibb. (Minimális sorindexük között a minimális oszlopindexű.)

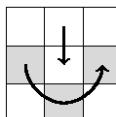
P_0 lesz a régió határ kiindulási pixele.

Jelölje a dir változó az elmozdulás irányát az előző pixeltől a jelenlegi pixelig. dir kiindulási értéke 4-es szomszédság esetén legyen 3, 8-as szomszédság esetén pedig 7. (Az egyes irányoknak megfelelő irányokat a következő ábra szemlélteti.)

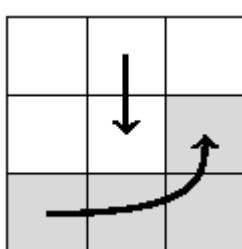


- Járjuk végig az aktuális pixel 3×3 -as szomszédságában lévő pixeleteket az óramutató járásával ellentétes irányban.

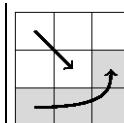
- A bejárást 4-es szomszédság esetében a $(dir + 3) \bmod 4$ iránynál kezdjük.



- 8-as szomszédság esetén, ha dir páros, akkor $(dir + 7) \bmod 8$, páratlan dir esetén pedig $(dir + 6) \bmod 8$ a kezdőirány.



dir páros



dir páratlan

Az első megtalált pixel lesz az új határpixel, jelölje ezt P_n . dir értékét aktualizáljuk.

3. Ha az aktuális P_n határpixel megegyezik P_1 -gyel, P_{n-1} pedig P_0 -lal, akkor véget ért a bejárás. Egyébként ismételjük a 2. lépést.
4. A detektált belső határvonal a P_0, P_1, \dots, P_{n-2} sorozat lesz.

2.2. Hough transzformáció

A feladat

Egy képen bizonyos alakzatokat (egyenes, kör, ellipszis, stb.) szeretnénk megkeresni.

- Éldetektálással előállítjuk a kép élpixeleinek a mátrixát.
- Hough transzformációval megkeressük a megadott alakzatot leginkább közelítő kontúrokhoz tartozó pixeleket.

2.3. Egyenes

- Az egy egyenesre illeszkedő (x, y) pontok halmaza

$$y = mx + c,$$

ahol m az egyenes meredeksége, $(0, c)$ pedig az y -tengellyel való metszéspontja.

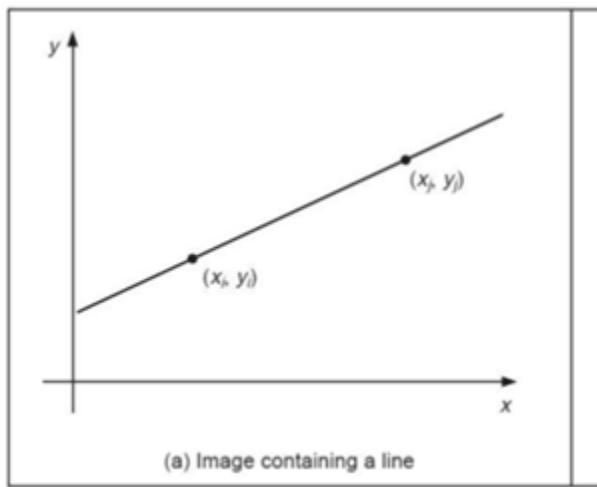
- Ezt az egyenletet átírhatjuk az

$$Ay + Bx + 1 = 0$$

alakba, ahol $A = -1/c$ és $B = m/c$. Ebben az alakban A és B határozza meg az egyenest.

- minden (A, B) számpár egy egyenest határoz meg az (x, y) koordináta-síkon, más megközelítésben minden (x, y) számpár egy egyenest határoz meg az (A, B) paraméter-síkon.

2.4. Koordináta-sík és paraméter-sík



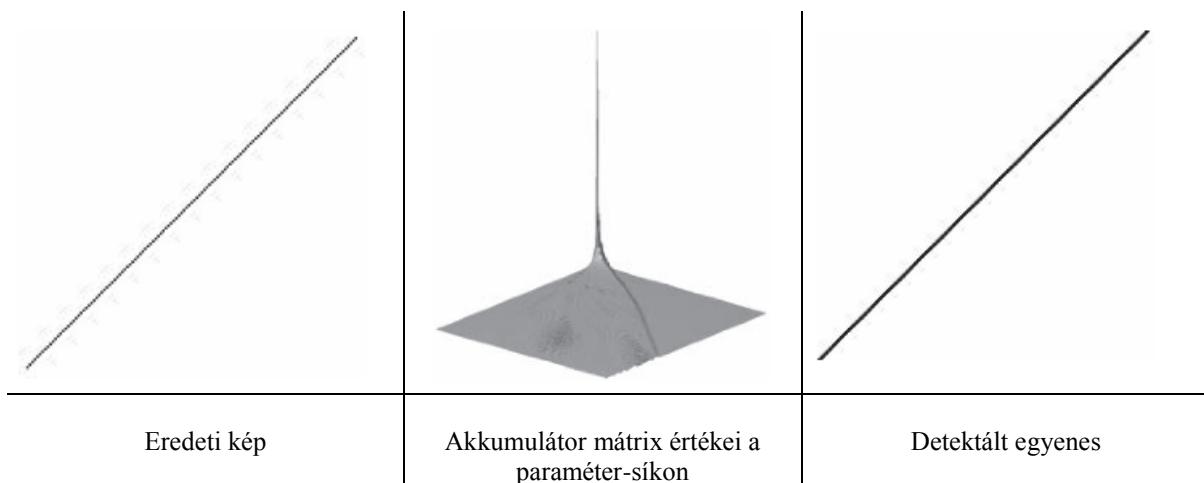
Forrás: M. S. Nixon, A. S. Aguado: Feature Extraction and Image Processing. Newnes, 2002

2.5. A Hough transzformáció alapötlete

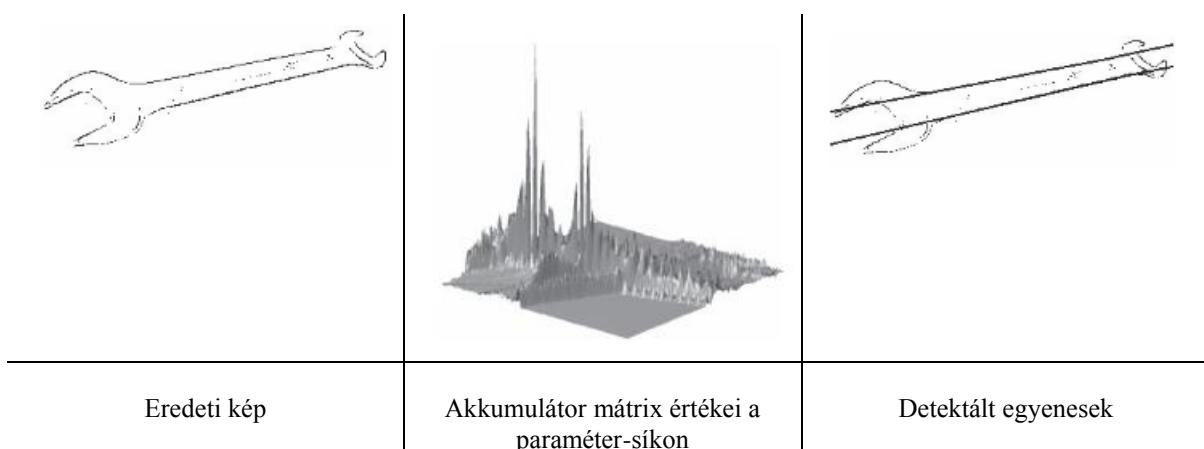
Minden a koordináta síkon elhelyezkedő egyenes megfeleltethető egy-egy, a paraméter síkon elhelyezkedő pontnak. Ezt kihasználva, az alábbi módszer segítségével megkereshetjük a koordináta síkon található egyeneseket:

1. Határozzuk meg az összes élpixelhez a hozzájuk tartozó összes lehetséges egyenest a paraméter-síkon.
 - a. Ennek érdekében létrehozunk egy ún. akkumulátor mátrixot.
 - b. A vizsgált pixelen áthaladó minden egyes egyenes paramétereinek megfelelően eggyel növeljük az akkumulátor mátrix adott paraméterekhez tartozó értékét.
2. Keressük meg az akkumulátor mátrix maximumát, azaz hogy hol van a legtöbb egyenesnek közös metszéspontja.
3. Az adott metszéspont megadja a keresett egyenes paramétereit a koordináta-síkon.

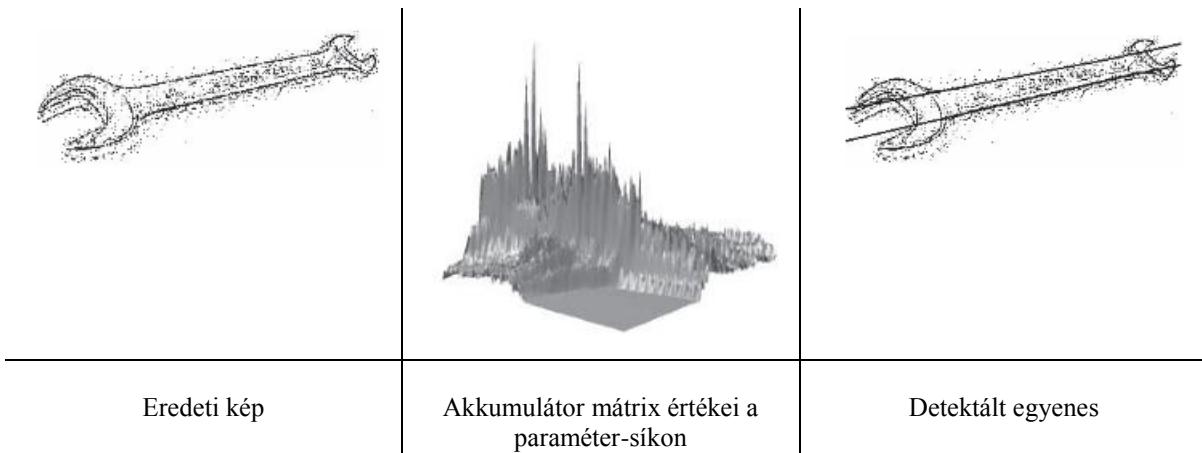
2.6. Egyenes szakaszok detektálása Hough transzformációval



Forrás: M. S. Nixon, A. S. Aguado: *Feature Extraction and Image Processing*. Newnes, 2002



Forrás: M. S. Nixon, A. S. Aguado: *Feature Extraction and Image Processing*. Newnes, 2002

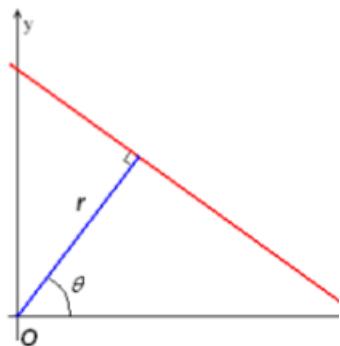


Forrás: M. S. Nixon, A. S. Aguardo: *Feature Extraction and Image Processing*. Newnes, 2002

2.7. Probléma

- Az $y = mx + c$ egyenletben m és c is a $]-\infty, +\infty[$ halmazból vehet fel értékeket.
- Ugyanez érvényes A -ra és B -re az $Ax + By - 1 = 0$ alakú egyenletnél.

2.8. Megoldás: Polár koordinátás reprezentáció



$$\rho = x \cos \theta + y \sin \theta,$$

ahol

$$c = \frac{\rho}{\sin \theta} \quad m = -\frac{1}{\tan \theta}$$

$$M \times N \text{ méretű kép esetén} \quad \rho \in [0, \sqrt{M^2 + N^2}], \quad \theta \in [0, \pi[$$

2.9. A végleges algoritmus

Bemenet: E egy $M \times N$ méretű kép elmxtrixa.

1. A (ρ, θ) paraméter-síkot osszuk fel δ_ρ , valamint δ_θ szélességű intervallumokra úgy, hogy a kialakuló intervallumot reprezentáló ρ_d (R hosszúságú) és θ_d (T hosszúságú) vektorok mérete megfelelő legyen.
2. Legyen $A(R, T)$ egy egész értékű mátrix, melynek kezdetben minden eleme 0. (Számlálóként fogjuk használni!)
3. minden $E(i,j)=1$ pixel esetén menjen egy ciklus $h=1$ -től T -ig:

- a. Legyen $\rho = i\cos\theta_d(h) + j\sin\theta_d(h)$
- b. Keressük meg ρ_d azon k indexű elemét, amely legközelebb van ρ -hoz
- c. Növeljük eggyel $A(k,h)$ értékét
4. Keressük meg az összes (k_p, h_p) lokális maximumot A-ban, amelyre $A(k_p, h_p) > \tau$, ahol τ egy előre definiált küszöbérték.

Kimenet: A $(\rho_d(k_p), \theta_d(h_p))$ párok halmaza, melyek egy-egy egyenest határoznak meg polár koordinátákkal.

2.10. Kör

- Az (x_0, y_0) középpontú r sugarú kör egyenlete:

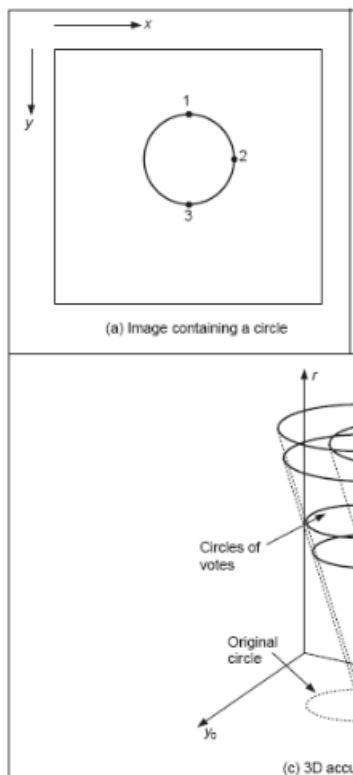
$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

- Vegyük észre, hogy ez az egyenlet szimmetriát mutat az (x, y) és az (x_0, y_0) párok között.
- Gyorsabb számolás érdekében érdemes a kör paraméteres egyenletrendszerét használni:

$$x = x_0 + r\cos\theta$$

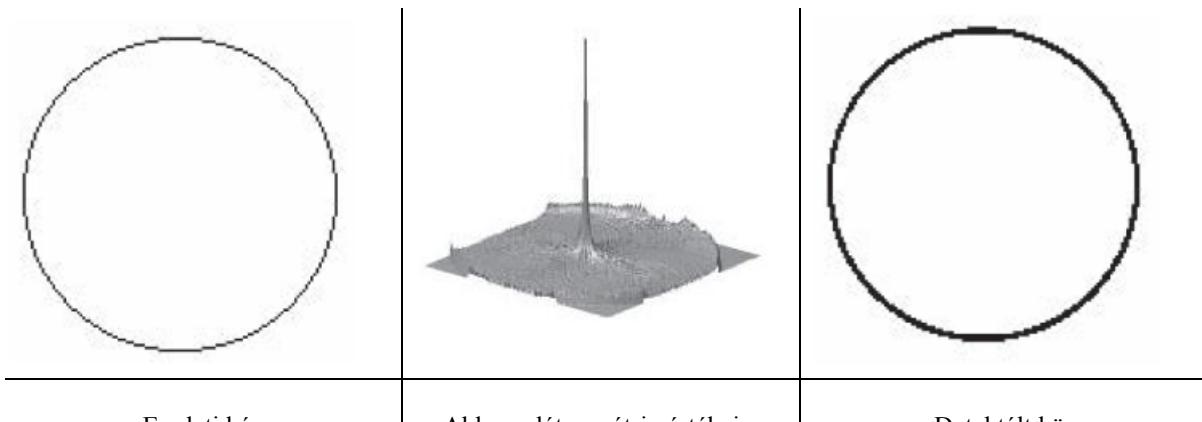
$$y = y_0 + r\sin\theta$$

ahol $\theta \in [0, 2\pi[$ nem független paraméter.



Forrás: M. S. Nixon, A. S. Aguado: *Feature Extraction and Image Processing*. Newnes, 2002

2.11. Körök detektálása Hough transzformációval

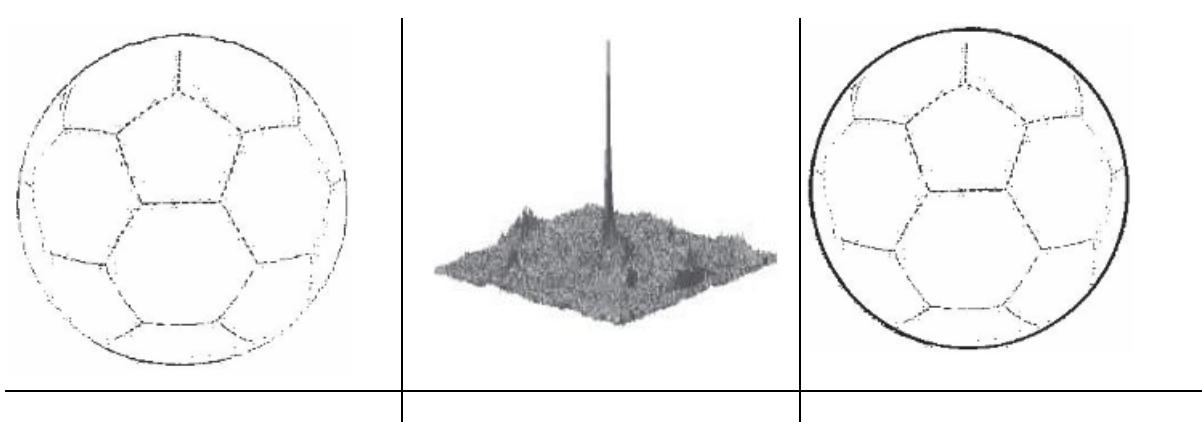


Eredeti kép

Akkumulátor mátrix értékei a paraméter-síkon

Detektált kör

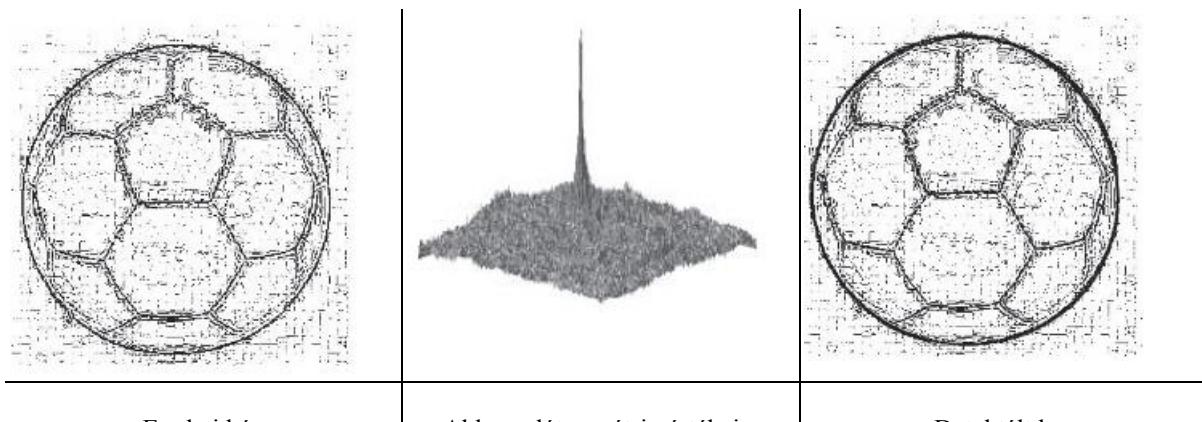
Forrás: M. S. Nixon, A. S. Aguado: *Feature Extraction and Image Processing*. Newnes, 2002



Eredeti kép

Akkumulátor mátrix értékei a paraméter-síkon

Detektált kör



Eredeti kép

Akkumulátor mátrix értékei a paraméter-síkon

Detektált kör

Forrás: M. S. Nixon, A. S. Aguado: *Feature Extraction and Image Processing*. Newnes, 2002

2.12. Ellipszis

- A kör alakú tárgyak a képeken gyakran ellipszisnek tűnnek a persepektív leképezés miatt.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \rho & \sin \rho \\ -\sin \rho & \cos \rho \end{bmatrix} \begin{bmatrix} S_x \\ S_y \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix},$$

ahol (x',y') a kör pontjainak koordinátái, ρ az orientáció, (S_x, S_y) a skálázási faktorok és (t_x, t_y) az eltolás vektora.

- Ha bevezetjük a következő jelöléseket:

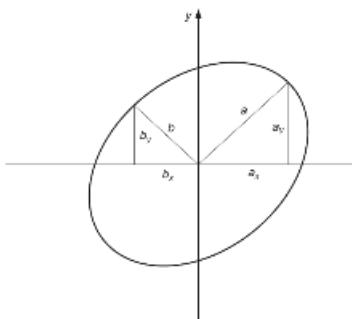
$$\begin{aligned} a_0 &= t_x & a_x &= S_x \cos \rho & b_x &= S_y \sin \rho \\ b_0 &= t_y & a_y &= -S_y \sin \rho & b_y &= S_x \cos \rho \end{aligned}$$

és áttérünk a kör paraméteres egyenletrendszerére, akkor a következő egyenletrendszeret kapjuk:

$$\begin{aligned} x &= a_0 + a_x \cos \theta + b_x \sin \theta \\ y &= b_0 + a_y \cos \theta + b_y \sin \theta \end{aligned}$$

- A kapott egyenlet egy ellipszis hat paraméterrel $(a_0, b_0, a_x, b_x, a_y, b_y)$ megadott egyenletrendszerre, melyben $\theta \in [0, 2\pi[$ nem független paraméter. A paraméterek közül viszont egy redundáns, mivel ellipszis esetében $a_x b_x + a_y b_y = 0$.
- Az orientáció, a nagytengely, valamint a kistengely az alábbi módon kapható a paraméterekből:

$$\tan \rho = \frac{a_y}{a_x} \quad a = \sqrt{a_x^2 + a_y^2} \quad b = \sqrt{b_x^2 + b_y^2}$$



Forrás: M. S. Nixon, A. S. Aguado: Feature Extraction and Image Processing. Newnes, 2002

2.13. Algoritmus általános paraméteres görbe esetén

Adott egy paraméteres görbe egyenlete $y = f(x, \mathbf{a})$ alakban.

1. Az $\mathbf{a}=(a_1, a_2, \dots, a_p)$ paraméter teret osszuk fel diszkrét részintervallumokra. s_i ($i = 1, \dots, n$) jelölje az egyes részintervallumok számát.
2. Legyen $A(s_1, s_2, \dots, s_p)$ p dimenziós tömb, melynek kezdetben minden eleme 0.
3. minden $E(i, j) = 1$ pixelre, növeljük az A tömb minden egyes elemét, amely esetében az $y = f(x, \mathbf{a})$ egyenlet teljesül.
4. Keressük meg az összes \mathbf{a}_m lokális maximumot, melyre $A(\mathbf{a}_m) > \tau$ teljesül, ahol τ egy előre definiált küszöbérték.

3. Összefüggő komponens analízis

3.1. Szegmentálás küszöböléssel

- Homogenitást küszöböléssel biztosítjuk.
- Előállítunk egy bináris képet.

- Összefüggő komponens analízissel megkeressük az összefüggő régiókat a bináris képen.
- Az egy régióba tartozó pixeleknek megfelelő pixelek az eredeti képen teljesítik az összefüggőségi és a homogenitási feltételt is, így egy szegmensbe tartoznak.

3.2. Rekurzív régió növelő algoritmus

Input:

- f : binarizált képmátrix

Feladat:

- f -ben megtalálni azon összefüggő komponenseket, melyek minden pixelére az f értéke 1

Átmeneti változók:

- L : f -fel azonos méretű mátrix, melynek kezdetben minden értéke 0
 - N : a már megtalált összefüggő komponensek száma
 - verem: ebben koordináta párokat fogunk tárolni
1. Keressünk olyan (x,y) pixelt, melyre $f(x,y)=1$ és $L(x,y)=0$ teljesül. Ha nem találunk ilyen pixelt, akkor vége az algoritmusnak.
 2. $N \leftarrow N + 1$
 3. $L(x,y) \leftarrow N$
 4. Ha $f(x - 1, y) = 1$ és $L(x - 1, y) = 0$, akkor $(x - 1, y)$ -t betesszük a verembe.
Ha $f(x + 1, y) = 1$ és $L(x + 1, y) = 0$, akkor $(x + 1, y)$ -t betesszük a verembe.
Ha $f(x, y - 1) = 1$ és $L(x, y - 1) = 0$, akkor $(x, y - 1)$ -t betesszük a verembe.
Ha $f(x, y + 1) = 1$ és $L(x, y + 1) = 0$, akkor $(x, y + 1)$ -t betesszük a verembe.
 5. Ha van elem a veremben, akkor kiveszük azt és ugrunk a harmadik lépéshöz. Ha nincs elem a veremben, akkor pedig az első lépéshoz ugrunk.

3.3. Rekurzív régió növelő algoritmus (példa)

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \mathbf{1} & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & \mathbf{1} & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

3.4. Szekvenciális algoritmus

1. Bejárjuk a bináris képet balról jobbra, illetve fentről lefelé.
2. Ha találunk egy $f(x,y)=1$ tulajdonságú pixelt, akkor az $(x - 1, y)$ és az $(x, y - 1)$ koordinátájú pixelek alapján a következő négy esettel találkozhatunk:

$$\begin{array}{ccc} 0 & & 0 \\ 0 & * & \rightarrow & 0 & a \\ a & & \rightarrow & a \\ 0 & * & \rightarrow & 0 & a \end{array}$$

3. Meghatározzuk a jelölők ekvivalenciaosztályait.

4. Átjelöljük a jelölőmátrixot.

3.5. Szekvenciális algoritmus (példa)

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & \mathbf{1} & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & \mathbf{1} & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & \mathbf{1} & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & \mathbf{1} & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ \mathbf{1} & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & \mathbf{1} & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & \mathbf{1} & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & \mathbf{1} & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & \mathbf{1} & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ \mathbf{1} & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & \mathbf{1} & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & \mathbf{1} & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & \mathbf{1} & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & \mathbf{1} & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & \mathbf{1} & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ \mathbf{1} & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & \mathbf{1} & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & \mathbf{1} & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & \mathbf{1} & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & \mathbf{1} & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ \mathbf{1} & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & \mathbf{1} & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$f = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

4. Régió alapú szegmentáló algoritmusok

4.1. Régió növeztés (Region Growing)

Bemenet: I képmátrix

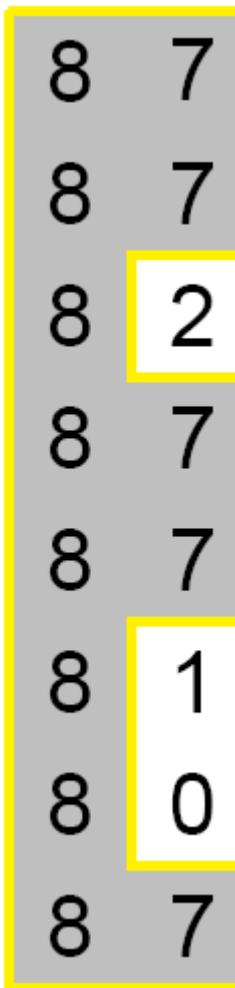
Kimenet: Szegmensek halmaza (S)

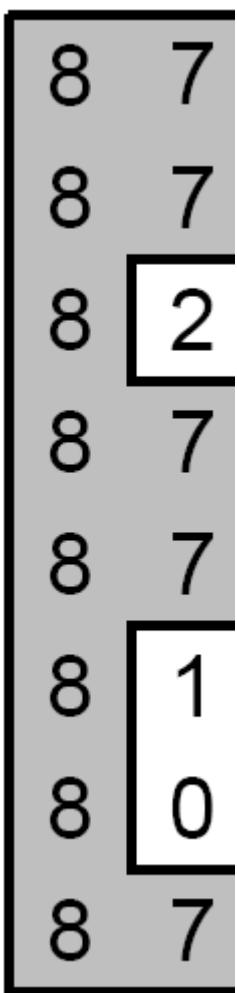
1. **elkészítjük** a szegmensek S (kezdetben üres) halmazát
2. **ciklus**
3. **eltávolítunk** egy P pixelt I -ből
4. **készítünk** egy új R_p szegmenst, mely P -t tartalmazza
5. **ciklus**
6. **eltávolítunk** egy P' pixelt I -ből, melyre P' a szomszédja
 valamely P'' -nek ($P'' \in R_p$) és $\{P'\} \cup R_p$ homogén
7. **hozzáadjuk** P' -t R_p -hez
8. **amíg** nem találunk több ilyen P' -t

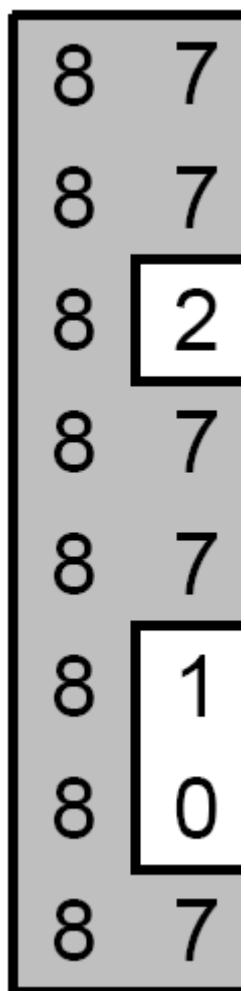
9. hozzáadjuk R_p -t S -hez
10. amíg I nem üres

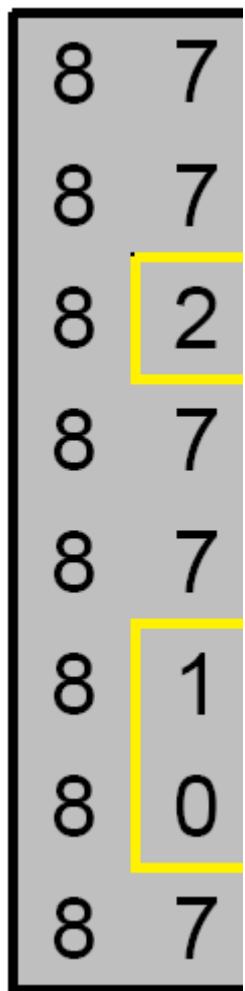
4.2. Régió növesztés (példa)

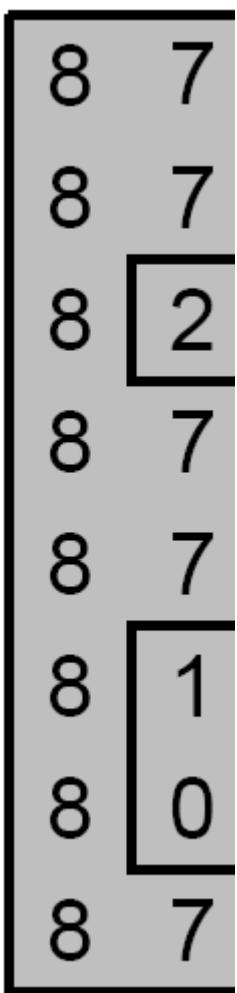
Homogenitás: Az R régió pontosan akkor homogén, ha $| \max(R) - \min(R) | \leq 2$.

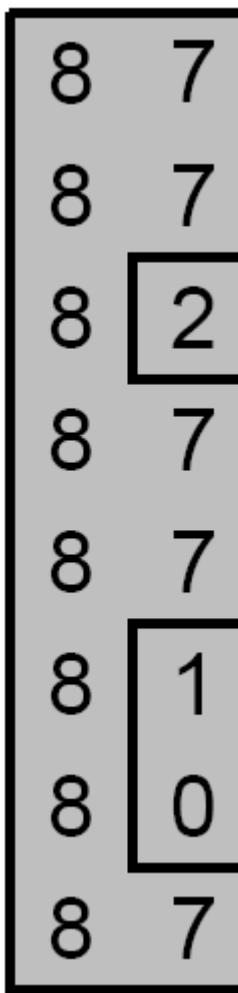












4.3. Split and Merge

A *Split and Merge* egy ún. quadtree struktúrát használó szegmentáló algoritmus.

Bemenet: $2^n \times 2^n$ méretű I képmátrix.

Split fázis:

1. Legyen $R(:=I)$ a kiindulási régió, mely a teljes képet reprezentálja.
2. Ha egy R régió nem elégül homogén ($H_s(R) = \text{false}$), akkor R -et felosztjuk négy darab (R_1, \dots, R_4) egyenlő méretű régióra.
3. A második lépést addig ismétljük, amíg vannak inhomogén régiók.

A Split fázis eredménye egy ún. quadtree-je lesz az I képnek.

Merge fázis:

1. Az R_1 és R_2 szomszédos régiókat összeolvastjuk R régiójává, ha R_1 hasonló R_2 -hez ($R_1 \phi_m R_2$).
2. Az 1. lépést addig ismétljük, amíg van összeolvasható szomszédos régió.

$$H_s(R) = \text{true} \Leftrightarrow \max(R) - \min(R) \leq 1$$

$$R_1 \phi_m R_2 \Leftrightarrow \max(R_2) - \min(R_1) \leq 1$$

0	1	:
1	0	:
0	2	:
4	4	:
0	0	:
1	1	:
2	4	:
2	3	:

0	1	:
1	0	:
0	2	:
4	4	:
0	0	:
1	1	:
2	4	:
2	3	:

0	1	:
1	0	:
0	2	:
4	4	:
0	0	:
1	1	:
2	4	:
2	3	:

0	1	:
1	0	:
0	2	:
4	4	:
0	0	:
1	1	:
2	4	:
2	3	:

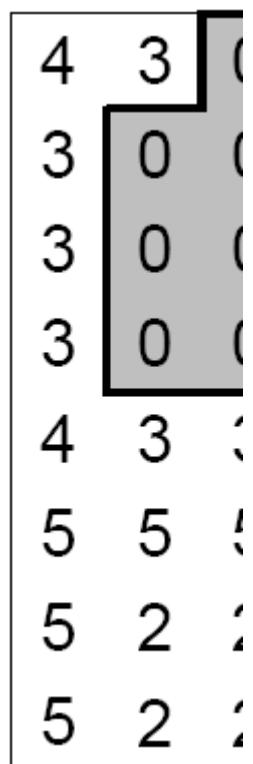
0	1	:
1	0	:
0	2	:
4	4	:
0	0	:
1	1	:
2	4	:
2	3	:

4.4. Watershed algoritmus

- A Watershed algoritmus egy képből készített gradiens képet használ, amit egy topografikus felületként kezel.
- Képzeletben vízzel árasztjuk el a topografikus felület völgyeit.
- Ahol a víz az egyik völgyből átfolyhatna egy másik völgybe, ott vízválasztót (watershed) építünk.
- A vízválasztóval körülhatárolt völgyeket tekintjük szegmenseknek.

4	3	(
3	0	(
3	0	(
3	0	(
4	3	:
5	5	:
5	2	:
5	2	:

Gradiens kép



Vízszint = 0 (Új medence)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	:
5	5	:
5	2	:
5	2	:

Vízszint = 0 (Új medence)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	:
5	5	:
5	2	:
5	2	:

Vízszint = 1 (Medence növelése)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	:
5	5	:
5	2	:
5	2	:

Vízszint = 1 (Medence növelése)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	:
5	5	:
5	2	:
5	2	:

Vízszint = 1 (Medence növelése)

4	3	
3	0	
3	0	
3	0	
4	3	:
5	5	:
5	2	:
5	2	:

Vízsint = 1 (Új medence)

4	3	
3	0	
3	0	
3	0	
4	3	:
5	5	:
5	2	:
5	2	:

Vízsint = 2 (Medence növelése)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	:
5	5	:
5	2	:
5	2	:

Vízszint = 2 (Medence növelése)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	:
5	5	:
5	2	:
5	2	:

Vízszint = 2 (Medence növelése)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	0
5	5	0
5	2	0
5	2	0

Vízsint = 2 (Új medence)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	0
5	5	0
5	2	0
5	2	0

Vízsint = 3 (Medence növelése)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	0
5	5	5
5	2	2
5	2	2

Vízsint = 3 (Medence növelése)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	0
5	5	5
5	2	2
5	2	2

Vízsint = 3 (Medence növelése)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	0
5	5	5
5	2	2
5	2	2

Vízsint = 4 (Medence növelése)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	0
5	5	5
5	2	2
5	2	2

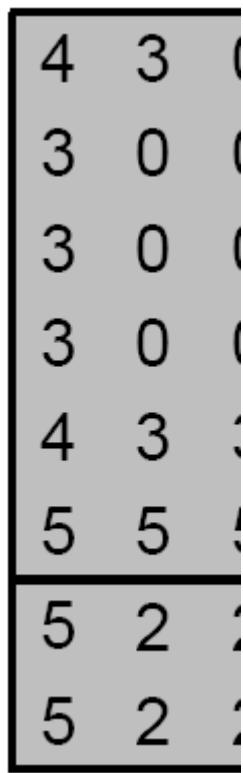
Vízsint = 4 (Medence növelése)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	0
5	5	5
5	2	2
5	2	2

Vízsint = 5 (Medence növelése)

4	3	0
3	0	0
3	0	0
3	0	0
4	3	0
5	5	5
5	2	2
5	2	2

Vízsint = 5 (Medence növelése)



Vízszint = 5 (Medence növelése)

Irodalomjegyzék

- [1] R. C. Gonzalez, R. E. Woods, *Digital Image Processing (2nd Edition)*. Prentice-Hall, 2002.
- [2] J. Matas, *Colour-based Object Recognition*. PhD dissertation, University of Surrey, 1996.
- [3] M. S. Nixon, A. S. Aguado, *Feature Extraction and Image Processing*, Newnes, 2002.
- [4] J. R. Parker, *Algorithms for Image Processing and Computer Vision*, John Wiley & Sons., 1997.
- [5] M. Shah, *Fundamentals of Computer Vision*, Computer Science Department, University of Central Florida, 1997.
- [6] M. Sonka, V. Hlavac, R. Boyle, *Image Processing, Analysis, and Machine Vision*, Thomson, 2008.
- [7] T. Svoboda, J. Kybic, V. Hlavac, *Image Processing, Analysis, and Machine Vision -- A MATLAB Companion*, Thomson, 2008.
- [8] E. Trucco, A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, 1998.

9. fejezet - Alakleírók, alakfelismerés módszerei

Sergyán Szabolcs

A fejezetet a Sonka-Hlavac-Boyle könyv [1] alapján dolgoztuk fel. A fejezetben található MATLAB kódok az említett könyv kiegészítéséből [2] származnak.



Alakleírók

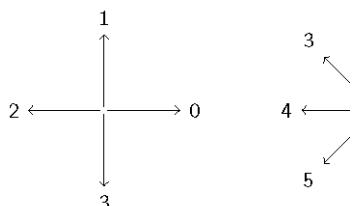
Az alak-, illetve objektum leírás módszerei két csoportra oszthatók:

- Az alakzat külső határvonalának reprezentálása.
- Az alakzat belső jellemzőinek reprezentálása:
 - Az alakzat szürkeségének, illetve színének jellemzői.
 - Az alakzat textúrájának leírása.

1. Határvonal alapú leírók

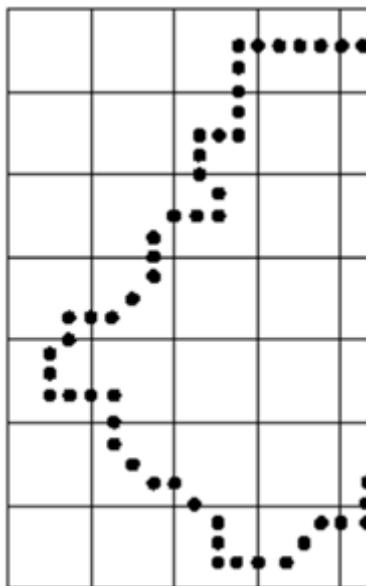
1.1. Lánckód

- A lánckód a régió határvonalának leírását teszi lehetővé. A leírás során két pont között adott irányú egyenes szakaszokat használunk.
 - A szakaszok irányánál a 4-es vagy 8-as szomszédságnak megfelelő kitüntetett irányokat használunk

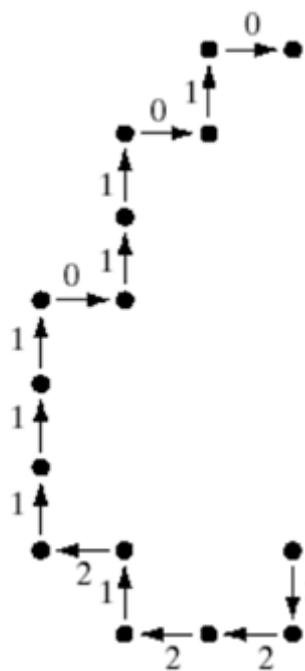


- Minden egyes szakasz "rácspontról" között halad, így hosszuk előre meghatározott

A lánckód nagyon érzékeny a határvonalon található zajokra. Ennek csökkentése érdekében nem minden határpontot veszünk figyelembe, hanem a határpontokból mintavételezünk, vagy valamilyen simító eljárás (pl. átlagolás, medián) alkalmazása után a simított határpontokat használjuk.



- A lánckód irása az óramutató járásának irása. Az alábbi ábra 4-es, illetve 8-as szomszédságon alapuló lánckódot szemléltet.



- A 4-es szomszédságú példa lánckódja, ha kiindulási pontnak a felső pontok legbaloldalibbét tekintjük:

0033333323221211101101

- Problémák a lánckóddal:

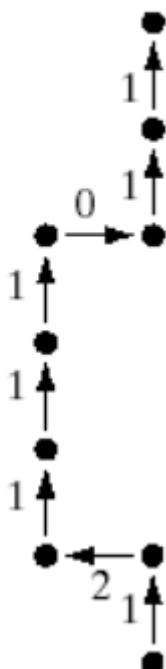
- A lánckód függ a kiindulási ponttól.

- A régió elforgatásával változik, tehát nem invariáns a forgatásra.
- Erősen függ a mintavételezéstől.
- Módosítás a forgatási invariancia érdekében:
- Az egymást követő elemek különbségét képezzük (modulo 4 vagy 8), így az egymást követő irányok változását tárolhatjuk el.
- Az így módosított kód a *differenciált lánckód*.

A kiindulási ponttól való függés megszüntetése érdekében:

- A differenciák körkörös sorozatában indulunk el a legkisebb értéktől.
- Az így kapott szám az *alakszám*.

Lánckód:	0033333323221211101101
Differenciált lánckód:	030000031303130031013
Alakszám:	000003130313003101303

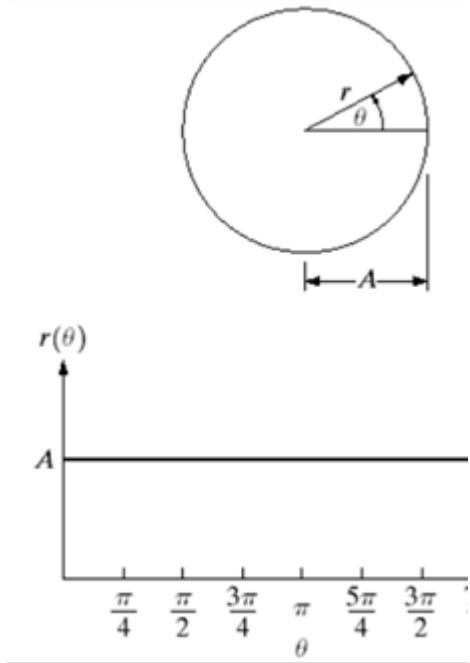


1.2. Szigntúra

- A régió határvonalát polárkoordinátás alakban írja le.
- Általában a régió tömegközéppontjától mért távolságot tárolja el az egyes kitüntetett irányokban.
 - A tömegközéppont meghatározása:

$$x_c = \frac{\sum_{i=1}^n x_i}{n}, \quad y_c = \frac{\sum_{i=1}^n y_i}{n},$$

ahol az (x_i, y_i) a régió pontjainak koordinátái, n pedig a régió pontjainak száma.



- Eltolásra invariáns.
- Forgatásra invariánssá tehető, ha kitüntetett pontot tekintünk kezdőpontnak (pl. a tömegközépponttól legtávolabbi határpont).
- Érzékeny a határzajokra, ez a határ simításával csökkenhető.
- Skálázási invariancia elérhető, ha egy adott tartományra normalizáljuk:

$$r' = \frac{r - r_{\min}}{r_{\max} - r_{\min}}, \text{ ahol}$$

ahol r és r' egy távolság eredeti, valamint normalizált értékei, r_{\min} a legkisebb, r_{\max} pedig a legnagyobb tömegközépponttól vett távolság.

1.3. Határvonalak Fourier transzformáltja

- Tegyük fel, hogy C egy zárt görbe (határvonal) a komplex számsíkon. Az óramutató járásával ellentétesen "állandó sebességgel" végigjárva a görbét kapunk egy $z(t)$ komplex függvényt, ahol t az idő változó. A sebességet úgy érdemes megválasztani, hogy a teljes körüljárás ideje 2π legyen.
- $z(t)$ Fourier reprezentációja:

$$z(i) = \sum_n T_n e^{inx},$$

ahol i a képzetes egységgöök.

- T_n a C görbe Fourier leírója.
- Az idő helyett érdemes a görbén végigjárt utat (s) használni a leírásra:

$$t = \frac{2\pi s}{L},$$

ahol L a görbe hossza.

- A Fourier leíró a következő módon adható meg:

$$T_n = \frac{1}{L} \int_0^L z(s) e^{-i(2\pi/L)ns} ds$$

- A Fourier leíró invariáns az eltolásra és forgatásra, ha megfelelően választottuk meg a koordináta-rendszert.
- Amennyiben a határvonal pontjai az $(x_1, y_1), (x_2, y_2), \dots, (x_L, y_L)$ koordináta párokkal adott (4-es szomszédságot figyelembe véve), ahol $(x_1, y_1) = (x_L, y_L)$, akkor értelmezhetők az alábbi Fourier leírók:

$$\alpha_n = \frac{1}{L-1} \sum_{m=1}^{L-1} x_m e^{-i(2\pi/(L-1))nm}$$

$$\beta_n = \frac{1}{L-1} \sum_{m=1}^{L-1} y_m e^{-i(2\pi/(L-1))nm}$$

- Ezekből képezhető az r_n leíró, mely eltolás és forgatás invariáns:

$$r_n = \left(|\alpha_n|^2 + |\beta_n|^2 \right)^{1/2}$$

- Amennyiben átméretezésre is invariáns leírót szeretnénk kapni, használjuk az alábbi w_n -t:

$$w_n = \frac{r_n}{r_1}$$

- Az eredeti pontok számánál sokkal kevesebb Fourier leíróval ($n << L$) jól visszaállítható a görbe.

1.4. Fourier leírók (MATLAB példa)

Az alábbi MATLAB függvény előállítja a Fourier leírókat.

MATLAB függvény: `function w = boundarydescr(xy,n)`

Bemenetek:

`xy` - $2 \times M$ méretű mátrix, melynek oszlopai az egyes határpontok koordinátáit tartalmazzák.
`n` - az w_n leíróból figyelembe vett elemek darabszáma.

Kimenet:

`w` - a w_n leíró.

```

1 function w=boundarydescr(xy,n)
3 x = xy(1,:); y = xy(2,:);
5 dx = x(2:end) - x(1:end-1);
5 dy = y(2:end) - y(1:end-1);
7 d = sqrt(dx.*dx+dy.*dy);
7 d = [0 d];
9 maxd = d(end);

```

```

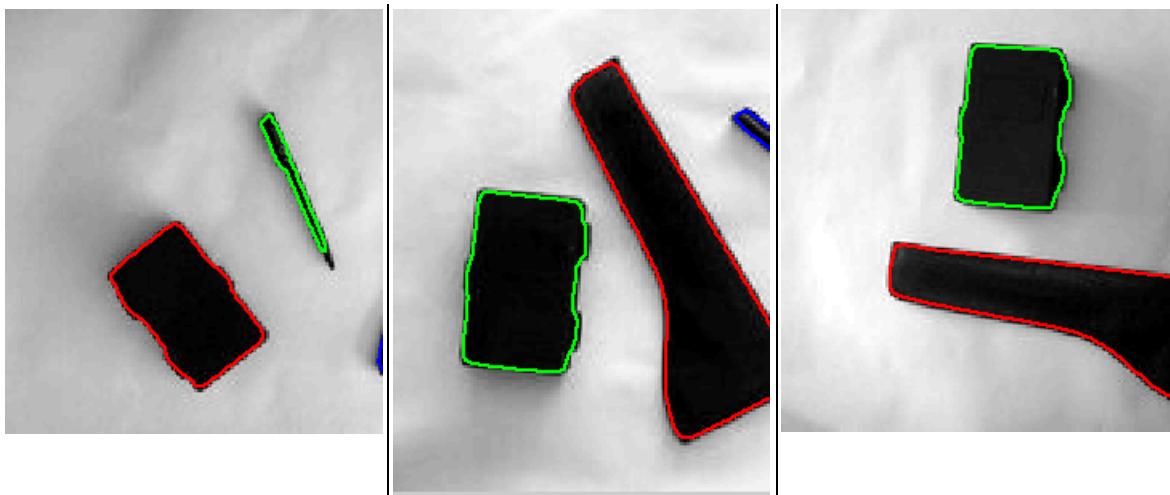
11 N = 256;
  step = maxd/N;
13 si = (0:step:maxd-step)';
  xi = interp1( d, x, si );
15 yi = interp1( d, y, si );

17 xf = fft(xi);  yf = fft(yi);
  r = sqrt( abs(xf).^2 + abs(yf).^2 );
19 w = r(3:n+2) / r(2);

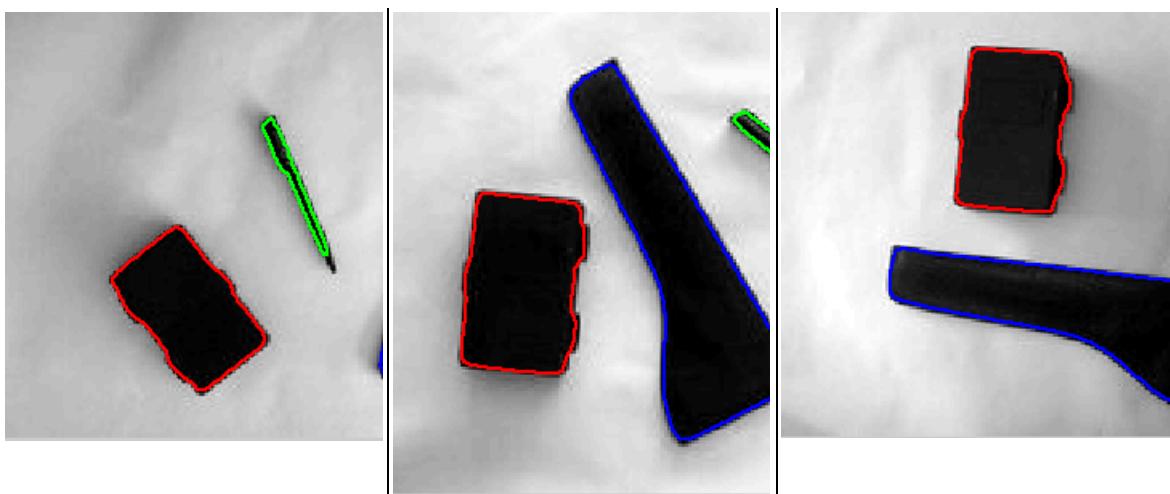
```

Codes/boundarydescr.m

- Szegmentálás eredményei három különböző képen:



- Fourier leíró használatával meghatározott összetartozó régiók:



2. Régió alapú alakleírás

2.1. Terület

- Adott régióra jellemző, egyszerű jellemző a terület.
- Meghatározása: a régiót alkotó pixelek száma.
- Eltolásra invariáns.

- Kis mértékben függ az elforgatástól.

Amennyiben a kép quadtree (négyesfa) struktúrával leírt, akkor a terület kiszámítása speciális:

1. minden régióhoz egy terület-változót rendelünk, amely kezdeti értéke nulla. H jelöli a quadtree mélységét (pl. 256×256 méretű kép esetén $H = 8$).
2. Járjuk be a fát szisztematikus módon. Ha egy h mélységű levél már megjelölt, akkor lépjünk a 3. pontra.
3. Számítsuk ki a megjelölt régió (`region_label`) új terület értékét:

$$\text{area}(\text{region_label}) = \text{area}(\text{region_label}) + 4^{(H-h)}$$

4. A régió területe az algoritmus végén az `area(region_label)` változóban tárolt.

Ha a régió egy n csúcsú síkidommal leírható, ahol ismertek a csúcspontok koordinátái $((x_0, y_0), (x_1, y_1), \dots, (x_n, y_n))$, ahol $(x_0, y_0) = (x_n, y_n)$, akkor a terület ezektől egyszerűen számítható:

$$\text{area} = \frac{1}{2} \sum_{k=0}^{n-1} |x_k y_{k+1} - x_{k+1} y_k|$$

Ha ismert a régió határvonalát leíró lánckód, 4-es szomszédságot feltételezve a régió területe az alábbi módon számítható ki:

1. A régió területét nullára állítjuk: `area` $\leftarrow 0$. A kiindulási pont x koordinátáját eltároljuk a `vertical_position` változóba.
2. minden elemre a lánckódban tegyük az alábbit:
 - 0 érték esetén: `area` $\leftarrow \text{area} - \text{vertical_position}$
 - 1 érték esetén: `vertical_position` $\leftarrow \text{vertical_position} + 1$
 - 2 érték esetén: `area` $\leftarrow \text{area} + \text{vertical_position}$
 - 3 érték esetén: `vertical_position` $\leftarrow \text{vertical_position} - 1$
3. A lánckód összes elemének a 2. pont szerinti feldolgozása után az `area` változó tartalmazza a régió területét.

2.2. Momentumok

- A régió momentum alapú reprezentáció úgy tekint egy normalizált¹ szürkeárnyalatos képet, mint egy kétdimenziós valószínűségi változó sűrűség függvénye.
- A $(p+q)$ rendű momentum függ a skálázástól, eltolástól, elforgatástól és a szürkeségi transzformációktól. Definíciója:

$$m_{pq} = \iint_{-\infty}^{\infty} x^p y^q f(x, y) dx dy;$$

digitális képek esetén pedig:

$$m_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} i^p j^q f(i, j),$$

ahol x, y, i, j a régióból pontok megfelelő koordinátái.

Centrális momentumok alkalmazása esetén eltolásra invariáns momentumokat kapunk:

¹a szürkeségi intenzitások összege egyet ad

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - x_c)^p (y - y_c)^q f(x, y) dx dy,$$

vagy digitális esetben:

$$\mu_{pq} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} (i - x_c)^p (j - y_c)^q f(i, j),$$

ahol x_c és y_c a régió tömegközéppontok koordinátái:

$$x_c = \frac{m_{10}}{m_{00}}, \quad y_c = \frac{m_{01}}{m_{00}}$$

- Skálázott centrális momentumok használatakor skálázás invariáns jellemzőket kapunk:

$$\eta_{pq} = \frac{\mu'_{pq}}{(\mu'_{00})^\gamma}, \quad \gamma = \frac{p+q}{2} + 1, \quad \mu'_{pq} = \frac{\mu_{pq}}{\alpha^{p+q+2}},$$

ahol $x' = \alpha x$, $y' = \alpha y$, a normalizált centrális momentum pedig:

$$\vartheta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^\gamma}$$

- Ha a nyomatéki főtengelyeket választjuk koordináta rendszernek, akkor elforgatás invariáns momentumokat kapunk, ekkor, hogy $\mu_{11} = 0$ is teljesül.

Általában az alábbi 7, Hu-féle forgatásra, eltolásra és skálázásra invariáns nyomatéki invariánsokat használják:

$$\begin{aligned} \phi_1 &= \vartheta_{20} + \vartheta_{02} \\ \phi_2 &= (\vartheta_{20} - \vartheta_{02})^2 + 4\vartheta_{11}^2 \\ \phi_3 &= (\vartheta_{30} - 3\vartheta_{12})^2 + (3\vartheta_{21} - \vartheta_{03})^2 \\ \phi_4 &= (\vartheta_{30} + \vartheta_{12})^2 + (\vartheta_{21} + \vartheta_{03})^2 \\ \phi_5 &= (\vartheta_{30} - 3\vartheta_{12})(\vartheta_{30} + \vartheta_{12}) \left((\vartheta_{30} + \vartheta_{12})^2 - 3(\vartheta_{21} + \vartheta_{03})^2 \right) + \\ &\quad + (3\vartheta_{21} - \vartheta_{03})(\vartheta_{21} + \vartheta_{03}) \left(3(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2 \right) \\ \phi_6 &= (\vartheta_{20} - \vartheta_{02}) \left((\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2 \right) + 4\vartheta_{11}(\vartheta_{30} + \vartheta_{12})(\vartheta_{21} + \vartheta_{03}) \\ \phi_7 &= (3\vartheta_{21} - \vartheta_{03})(\vartheta_{30} + \vartheta_{12}) \left((\vartheta_{30} + \vartheta_{12})^2 - 3(\vartheta_{21} + \vartheta_{03})^2 \right) - \\ &\quad - (\vartheta_{30} - 3\vartheta_{12})(\vartheta_{21} + \vartheta_{03}) \left(3(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2 \right) \end{aligned}$$

Öt különböző kép, amelyek térfelvétel transzformációval átvihetők egymásba, valamint az előzőekben kiemelt momentumok értékei az egyes képekre:



Invariant (Log)	O
ϕ_1	
ϕ_2	1
ϕ_3	2
ϕ_4	2
ϕ_5	4
ϕ_6	?
ϕ_7	4

Az alábbi 4 momentum invariáns az általános affin transzformációkra is:

$$\begin{aligned}
 I_1 &= \frac{\mu_{20}\mu_{02} - \mu_{11}^2}{\mu_{00}^4} \\
 I_2 &= \frac{\mu_{30}^2\mu_{03}^2 - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} + 4\mu_{30}^2\mu_{12}^2 + 4\mu_{21}^2\mu_{03}^2 - 3\mu_{21}^2\mu_{12}^2}{\mu_{00}^{10}} \\
 I_3 &= \frac{\mu_{20}(\mu_{21}\mu_{03} - \mu_{12}^2) - \mu_{11}(\mu_{30}\mu_{03} - \mu_{21}\mu_{12}) + \mu_{02}(\mu_{30}\mu_{12} - \mu_{21}^2)}{\mu_{00}^7} \\
 I_4 &= \left(\mu_{20}^3\mu_{03}^2 - 6\mu_{20}^2\mu_{11}\mu_{12}\mu_{03} - 6\mu_{20}^2\mu_{02}\mu_{21}\mu_{03} + 9\mu_{20}^2\mu_{02}\mu_{21}^2 \right. \\
 &\quad + 12\mu_{20}\mu_{11}^2\mu_{21}\mu_{03} + 6\mu_{20}\mu_{11}\mu_{02}\mu_{30}\mu_{03} - 18\mu_{20}\mu_{11}\mu_{02}\mu_{21}\mu_{12} \\
 &\quad - 8\mu_{11}^3\mu_{30}\mu_{03} - 6\mu_{20}\mu_{02}^2\mu_{30}\mu_{12} + 9\mu_{20}\mu_{02}^2\mu_{21}^2 \\
 &\quad \left. + 12\mu_{11}^2\mu_{02}\mu_{30}\mu_{12} - 6\mu_{11}\mu_{02}^2\mu_{30}\mu_{21} + \mu_{02}^3\mu_{30}^2 \right) / \mu_{00}^{11}
 \end{aligned}$$

- A momentumok alkalmasak az alak leírására abban az esetben is, ha a régiónak csak a határvonala ismert.
- Tegyük fel, hogy a régió zárt határvonala ismert, amit a $z(i)$ sorozat reprezentál, amely N darab határpixelnek a régió középpontjától számított euklideszi távolságát tartalmazza.
- Az r -edik kontúr sorozat momentum:

$$m_r = \frac{1}{N} \sum_{i=1}^N (z(i))^r$$

- Az r -edik centrális kontúr sorozat momentum:

$$\mu_r = \frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^r$$

- Az r -edik normalizált kontúr sorozat momentum:

$$\bar{m}_r = \frac{m_r}{(\mu_2)^{r/2}} = \frac{\frac{1}{N} \sum_{i=1}^N (z(i))^r}{\left(\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^2 \right)^{r/2}}$$

Az r -edik normalizált centrális kontúr sorozat momentum:

$$\bar{\mu}_r = \frac{m_r}{(\mu_2)^{r/2}} = \frac{\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^r}{\left(\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^2 \right)^{r/2}}$$

- Ez a két momentum invariáns az eltolásra, fogatásra és skálázásra.

Az alábbi momentumok zajokra kevésbé érzékenyek:

$$F_1 = \frac{(\mu_2)^{1/2}}{m_1} = \frac{\left(\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^2 \right)^{1/2}}{\frac{1}{N} \sum_{i=1}^N z(i)}$$

$$F_2 = \frac{\mu_3}{(\mu_2)^{3/2}} = \frac{\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^3}{\left(\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^2 \right)^{3/2}}$$

$$F_3 = \frac{\mu_4}{(\mu_2)^2} = \frac{\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^4}{\left(\frac{1}{N} \sum_{i=1}^N (z(i) - m_1)^2 \right)^2}$$

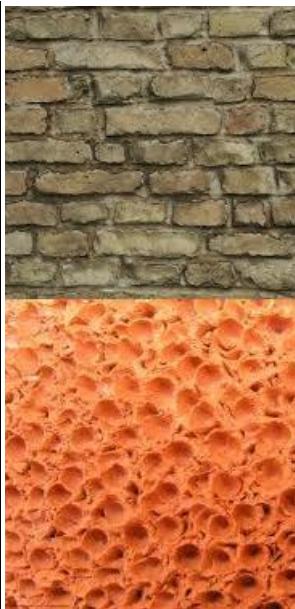
$$F_4 = \bar{\mu}_5$$

Irodalomjegyzék

- [1] M. Sonka, V. Hlavac, R. Boyle, *Image Processing, Analysis, and Machine Vision*. Thomson, 2008.
- [2] T. Svoboda, J. Kybic, V. Hlavac, *Image Processing, Analysis, and Machine Vision -- A MATLAB Companion*. Thomson, 2008.

10. fejezet - Textúra

A fejezetet a Sonka-Hlavac-Boyle könyv [2] és Palágyi Kálmán jegyzete [1] alapján dolgoztuk fel.



- Textúrán az objektumok felszínére, szerkezetére, mintázatára jellemző tulajdonságokat értjük.
- A textúra egymással kölcsönhatásban álló ismétlődő elemekből áll, melyeket textúra primitíveknek vagy textúra elemeknek (*texeleknek*) nevezünk.

1. Statisztikai textúra leírók

1.1. Frekvencián alapuló módszerek

- A durvább textúrák nagyobb texeleket tartalmaznak, így alacsonyabb térbeli frekvenciákkal jellemzhetők.
- A finomabb textúrák vonzata a nagyobb térbeli frekvencia.
- A térbeli frekvenciák például az $M \times N$ méretű f kép autokorrelációjával jellemzhetők.

1.2. Autokorrelációs textúra leíró

- Az autokorrelációs együttható a p és q paraméterek különböző értékei esetén:

$$C_{ff}(p, q) = \frac{MN}{(M-p)(N-q)} \frac{\sum_{i=1}^{M-p} \sum_{j=1}^{N-q} f(i, j)f(i+p, j+q)}{\sum_{i=1}^M \sum_{j=1}^N f(i, j)},$$

ahol p és q a pixel koordináták közötti különbség az x , illetve y irányban, M és N pedig a kép sor- és oszlopszáma.

- A C_{ff} mátrix elemei a korrelációs együtthatók.
- Durva textúrák esetén a korrelációs együttható értéke lassan csökken, a finomak esetén viszont gyorsan.

1.3. Autokorrelációs függvény a frekvencia tartományban

Az autokorrelációs függvény a frekvencia tartományban is értelmezett:

$$C_{ff} = F^{-1}\{|\mathbf{F}|^2\}$$

1.4. Együttes előfordulási (co-occurrence) mátrixok

Vizsgáljuk a kép egy $M \times N$ méretű tartományát. A szürkeségi értékek előfordulása leírható relatív frekvenciák $P_{\phi,d}(a,b)$ mátrixával, amely megmutatja, hogy milyen gyakorisággal fordulnak elő pixelek az a és b intenzitásokkal egymástól d távolságra a ϕ irány mentén.

$$\begin{aligned} P_{0^*,d}(a,b) &= \left| \left[(k,l), (m,n) \right] \in D : \right. \\ &\quad \left. k-m=0, |l-n|=d, f(k,l)=a, f(m,n)=b \right\} \\ P_{45^*,d}(a,b) &= \left| \left[(k,l), (m,n) \right] \in D : \right. \\ &\quad \left. (k-m=d, l-n=-d) \vee (k-m=-d, l-n=d), f(k,l)=a, f(m,n)=b \right\} \\ P_{90^*,d}(a,b) &= \left| \left[(k,l), (m,n) \right] \in D : \right. \\ &\quad \left. |k-m|=d, l-n=0, f(k,l)=a, f(m,n)=b \right\} \\ P_{135^*,d}(a,b) &= \left| \left[(k,l), (m,n) \right] \in D : \right. \\ &\quad \left. (k-m=d, l-n=d) \vee (k-m=-d, l-n=-d), f(k,l)=a, f(m,n)=b \right\} \end{aligned}$$

ahol $D = (M \times N) \times (M \times N)$

1.5. Együttes előfordulási mátrixok

Ha a képünk például az alábbi 4×4 -es mátrixszal reprezentálható

$$\begin{array}{c} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 \end{bmatrix}, \\ \text{akkor } P_{0^*,1} = \begin{bmatrix} 4 & 2 & 1 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 0 & 6 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix} \text{ és } P_{135^*,1} = \begin{bmatrix} 2 & 1 & 3 & 0 \\ 1 & 2 & 1 & 0 \\ 3 & 1 & 0 & 2 \\ 0 & 0 & 2 & 0 \end{bmatrix}. \end{array}$$

Az együttes előfordulási mátrixokból több jellemző is származtatható.

Energia (más néven iránymenti második momentum)

$$energy = \sum_{a,b} P_{\phi,d}^2(a,b)$$

Entrópia

$$entropy = \sum_{a,b} P_{\phi,d}^2(a,b) \log_2 P_{\phi,d}(a,b)$$

Maximális valószínűség

$$maximal_probability = \max_{a,b} P_{\phi,d}(a,b)$$

Kontraszt

$$contrast = \sum_{a,b} |a-b|^{\kappa} P_{\phi,d}^{\lambda}(a,b)$$

általában $\kappa = 2$ és $\lambda = 1$

Inverz eltérési momentum

$$\text{inverse_difference_moment} = \sum_{a,b;a \neq b} \frac{P_{g,d}^4(a,b)}{|a-b|^k}$$

Korreláció

$$\text{correlation} = \frac{\sum_{a,b} [(ab) P_{g,d}(a,b)] - \mu_x \mu_y}{\sigma_x \sigma_y},$$

ahol μ_x , μ_y a középértékek, σ_x és σ_y pedig a szórás, azaz

$$\begin{aligned}\mu_x &= \sum_a a \sum_b P_{g,d}(a,b), & \sigma_x &= \sum_a (a - \mu_x)^2 \sum_b P_{g,d}(a,b), \\ \mu_y &= \sum_b b \sum_a P_{g,d}(a,b), & \sigma_y &= \sum_b (b - \mu_y)^2 \sum_a P_{g,d}(a,b).\end{aligned}$$

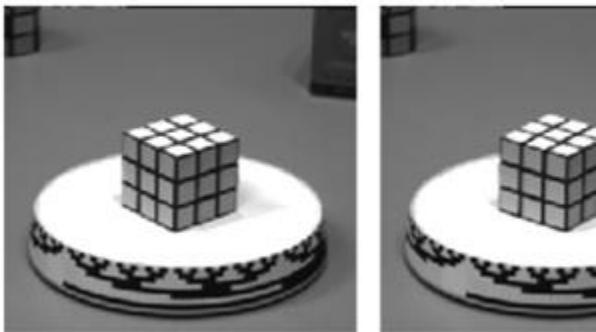
Irodalomjegyzék

- [1] Palágyi K., *Képfeldolgozás haladóknak*, Typotex, 2011.
- [2] M. Sonka, V. Hlavac, R. Boyle, *Image Processing, Analysis, and Machine Vision*, Thomson, 2008.

11. fejezet - Optikai áramlás

Vámossy Zoltán

A fejezetet Bradskiék [1] anyaga, valamint a Forsyth-Ponce [2] és Trucco-Verri [3] könyvekből merítve állítottuk össze. Szeliski [4] munkájának egyes része is megjelenik a bemutatóban.



1. Optikai folyamok alkalmazása

1.1. Mozgás

- A valós világban (3D) történő mozgás leképzése általában mozgást eredményez a képsíkon (2D) – de az információ redukálódik
- A 2D-s mozgás képek sorozatán jelenik meg: legalább két kép kell a meghatározásához
- Általában a mozgó objektum konstans intenzitását tételezik fel a meghatározáshoz
- A mozgó pixel fényessége (intenzitása) nem változik az időben (brightness constancy)
$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$
- minden egyes pixel mozgását mérjük

1.2. minden egyes pixel mozgását mérjük

Pixelek áramlása

- Optikai folyam, vagy optikai áramlás kifejezéseket szinonimaként használjuk



1.3. Hol használjuk a mozgást a gépi látás területein?

Alkalmazási területek

- Mozgásdetektálás
- Objektum követés
- Kamera mozgások korrekciója (stabilizáció)
- Képek egymáshoz igazítása (mozaikozás)
- 3D alak rekonstrukció
- Videó tömörítés

1.4. Mozgásdetektálás

Két frame különbsége

- Közel 0, ha nincs mozgás
- Nem 0, ha elmozdulás történt



1.5. Mozaikozás

Több frame felhasználásával panoráma kép, egybefüggő környezet



Képsorozatból alkotunk panorámaképet

- Azonosan elmozdult képpontokat is használhatunk az illesztésre



1.6. Képszegmentálás

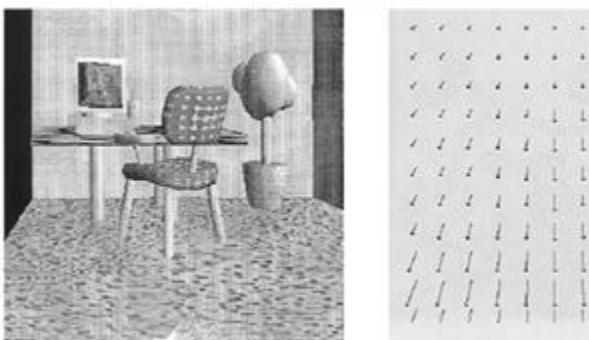
Mozgó objektumok könnyebben szegmentálhatók

- Az elmozdulás-vektorok homogének

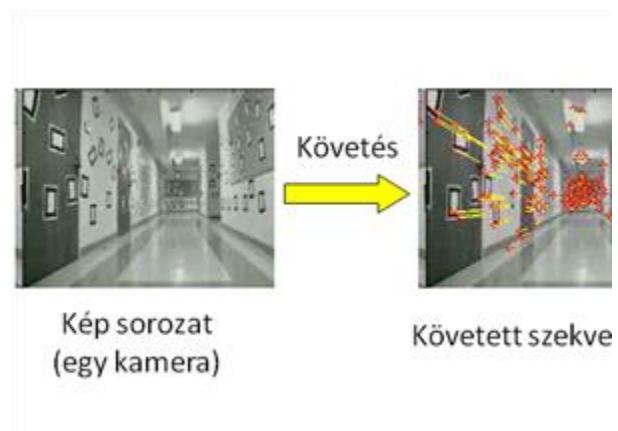


1.7. Struktúra meghatározás mozgás alapján

Mozgó kamera esetén a pixeláramlásból következtetni lehet az objektumrészek távolságára

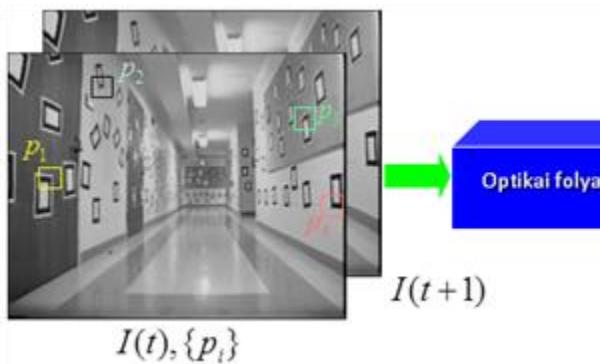


1.8. Optikai áramlás (Optical flow)



2. Jellemzők és egyenletrendszerek

2.1. Mi az optikai folyam?



Az optikai folyam a mozgásmezőhöz történő megfeleltetés:

- De a pontok fizikai mozgásának 2D leképezése függ a megfigyelő és a kép pixel csoportjainak egymáshoz viszonyított helyzetétől

Általános feltétel:

- Intenzitás állandóság (brightness constancy)

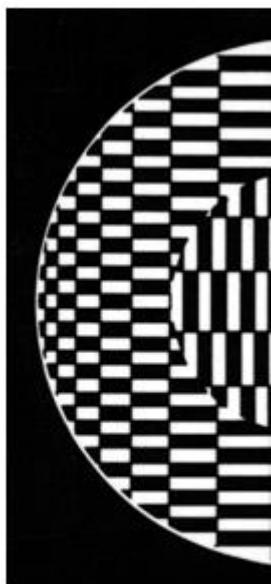
$$I(p_i, t) = I(p_i + \vec{v}_i, t+1)$$

2.2. Az optikai folyam speciális esetei

Speciális esetek:

Minden olyan eset, amikor a pixel elmozdulások nem jelentik a térbeli pontok fizikai mozgását

1. TV illuzórikus mozgáson alapszik
2. Egyenletesen forgó gömb – semmi sem tűnik mozgónak
3. A fény intenzitásának vagy irányának változása miatt a dolgok mozgónak tűnhetnek



2.3. Apertúra probléma

Illúzió

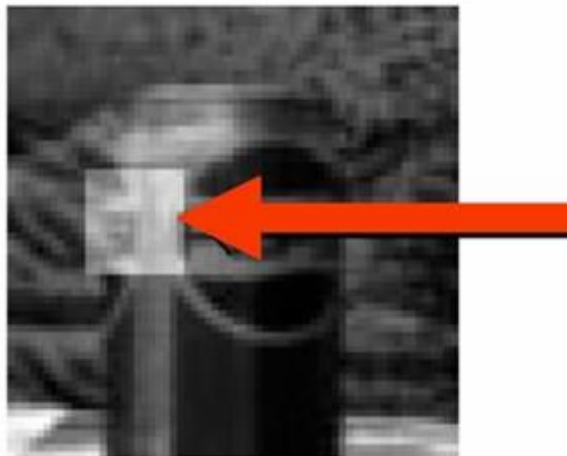


Forgó rúd

2.4. Kiinduló feltevések (1): konstans fényesség

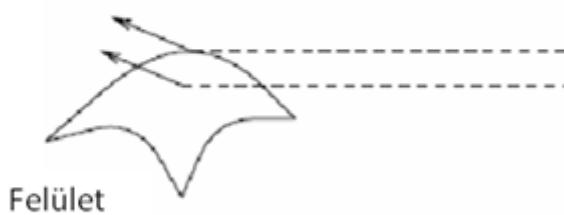
Kis régióban a képértékek (fényesség) nem változik

- A helyzetre ez természetesen nem igaz



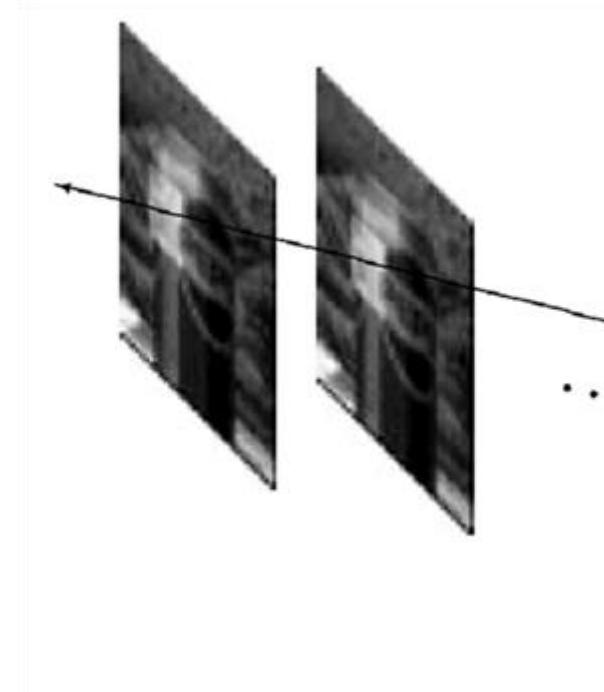
2.5. Kiinduló feltevések (2): térbeli összetartozás

- A kép szomszédos pontjai tipikusan ugyanahhoz az objektumhoz tartoznak és hasonló a mozgásuk
- Elvárjuk, hogy a képen is közelí pontokként jelenjenek meg



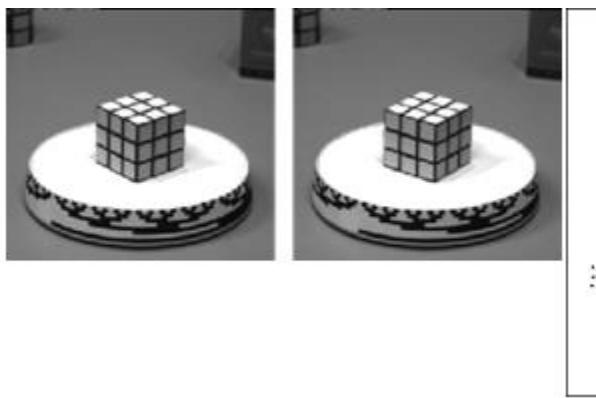
2.6. Kiinduló feltevések (3): időbeli összetartozás

- A foltok mozgása a képen fokozatos az idő függvényében



2.7. Optikai folyam egyenletek

A (2D) képtérben a mozgás vektor számítása



- A mozgó pixel intenzitása **nem változik az időben**
- $I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$
- A jobb oldal Taylor sorba fejtése: **kis elmozdulást feltételezve**

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \Delta x \frac{\partial I}{\partial x} + \Delta y \frac{\partial I}{\partial y} + \Delta t \frac{\partial I}{\partial t}$$

$$I(x, y, t) = I(x, y, t) + \Delta x I_x + \Delta y I_y + \Delta t I_t$$

$$0 = \Delta x I_x + \Delta y I_y + \Delta t I_t$$

$$0 = \frac{\Delta x}{\Delta t} I_x + \frac{\Delta y}{\Delta t} I_y + I_t$$

Bevezetve u és v változókat

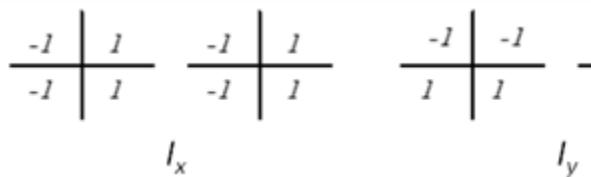
$$u = \frac{\Delta x}{\Delta t} \quad v = \frac{\Delta y}{\Delta t}$$

Konstans intenzitás feltétel

$$uI_x + vI_y + I_t = 0$$

Az (u, v) térben ez egy egyenes egyenlete

$$v = -u \frac{I_x}{I_y} - \frac{I_t}{I_y}$$

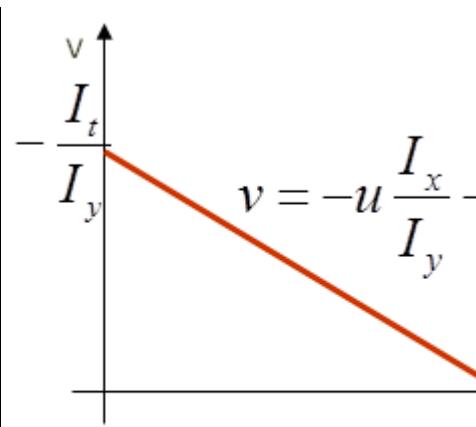


I_x , I_y és I_t számítása képből

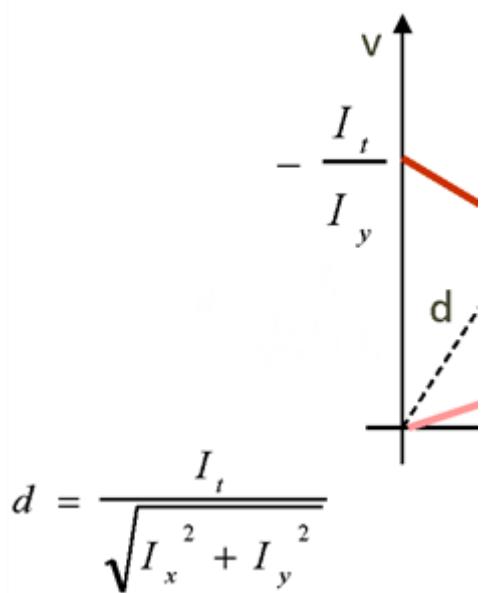
- a maszkok origója a jobb alsó sarkuk
- I_x és I_y gradiens összetevők
- I_t két egymás utáni kép különbsége

Minden pontra egy egyenletet eredményez ez

- 2 ismeretlenünk van: u , v
- A megoldás valahol az egyenesen van

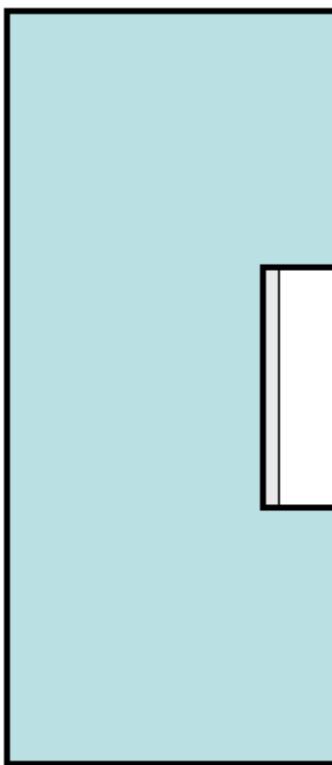


- Legyen (u', v') a valódi folyam
- A valódi optikai folyamnak két komponense van:
 - Normál irányban: d irány
 - Párhuzamos irányban: p irány
- A normál folyam meghatározható (gradiens irányú komponens)
- A párhuzamos (p) NEM

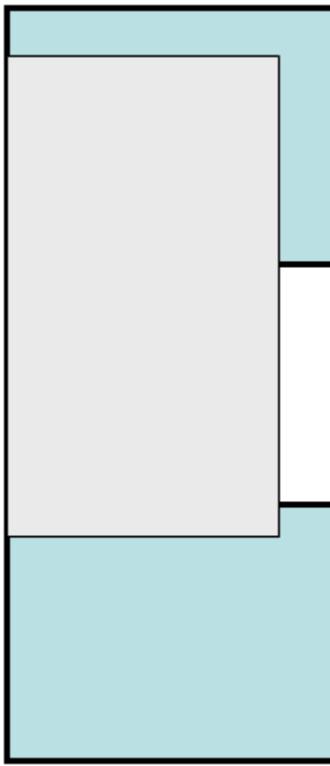


2.8. Apertúra probléma

Miként jelenik meg a képen, hogy csak normális irányú mozgást érzékelünk?



Félreérthetőség



3. Megoldási módszerek

3.1. Megoldási technikák

Horn & Schunck

- Konstans intenzitás + simasági feltétel

Schunck

- Általános folyam

Lucas & Kanade

- Konstans intenzitás

...

3.2. Horn & Schunck megoldása

$$uI_x + vI_y + I_t = 0$$

Definiáljuk a következő energia függvényt és minimalizáljuk

(fényesség konstans + kis mozgások)

$$E(x, y) = (uI_x + vI_y + I_t)^2 + \lambda \overbrace{(u_x^2 + u_y^2 + v_x^2 + v_y^2)}^f$$

Az ismeretlen u és v szerint deriválva

$$\frac{\partial E}{\partial u} = 2I_x(uI_x + vI_y + I_t) + \lambda \frac{\partial f}{\partial u}$$

$$\frac{\partial E}{\partial v} = 2I_y(uI_x + vI_y + I_t) + \lambda 2(v_{xx} + v_{yy})$$

$$I_x(uI_x + vI_y + I_t) + \lambda \Delta^2 u = 0 \quad I_y(uI_x + vI_y + I_t) + \lambda \Delta^2 v = 0$$

Az optikai folyam simasági feltételének Laplace kifejezése

- Egy megoldást kereshetünk a következő alakban

$$\Delta^2 u = u - u_{avg}, \Delta^2 v = v - v_{avg}.$$

ahol u_{avg} négy szomszédos pixelen számolt átlag

- Átrendezve az egyenleteket

$$u(\lambda + I_x^2) + vI_xI_y + I_xI_t - \lambda u_{avg} = 0$$

$$v(\lambda + I_y^2) + uI_xI_y + I_yI_t - \lambda v_{avg} = 0$$

- 2 egyenlet 2 ismeretlennel
- Fejezzük ki v -t u -val
- Helyettesítsük be a másik egyenletbe

$$u = u_{avg} - I_x \left(\frac{I_x u_{avg} + I_y v_{avg} + I_t}{I_x^2 + I_y^2 + \lambda} \right) \quad v = v_{avg} - I_y \left(\frac{I_x u_{avg} + I_y v_{avg} + I_t}{I_x^2 + I_y^2 + \lambda} \right)$$

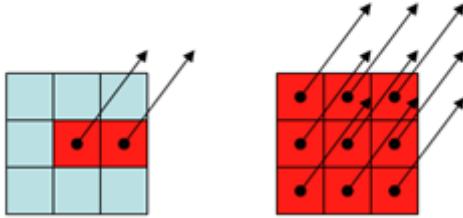
Iterációs módszerrel határozzuk meg u és v értékét

- Kiszámoljuk a deriváltakat előre
- Kezdetben tegyük fel, hogy u és $v = 0$
- Számoljuk u_{avg} és v_{avg} értékét a szomszédokból az iteráció során

3.3. Schunck módszere

Ha két szomszédos pixel ugyanazzal a sebességgel mozog

- A hozzá tartozó folyamegyenlet megoldásai egy pontban metszik egymást az (u, v) térsben
- Hatózzuk meg a metszéspontot
- A több szomszéd (Schunck 8-at használ) általában nem egy pontot, hanem egy halmazt (klasztert) határoz meg
- A legnagyobb ilyen halmaz meghatározza a sebességet



3.4. Lucas & Kanade módszere

Hasonló az egyenes illesztéses módszerhez

- Ez is egy energia-kifejezést definiál és a minimumát keresi

$$E = \sum (uI_x + vI_y + I_t)^2$$

A deriváltja tehát 0

$$\frac{\partial E}{\partial u} = \sum 2I_x(uI_x + vI_y + I_t) = 0 \quad \frac{\partial E}{\partial v} = \sum 2I_y(uI_x + vI_y + I_t) = 0$$

$$\begin{aligned} \sum uI_x^2 + \sum vI_xI_y + \sum I_xI_t &= 0 \\ u\sum I_x^2 + v\sum I_xI_y &= -\sum I_xI_t \end{aligned} \quad \begin{aligned} \sum uI_xI_y + \sum vI_y^2 + \sum I_yI_t &= 0 \\ u\sum I_xI_y + v\sum I_y^2 &= -\sum I_yI_t \end{aligned}$$

$$\begin{bmatrix} \sum I_x^2 & \sum vI_xI_y \\ \sum I_xI_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\sum I_xI_t \quad \begin{bmatrix} \sum I_xI_y & \sum I_y^2 \\ \sum I_yI_t & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\sum I_yI_t$$

$$\begin{bmatrix} \sum I_x^2 & \sum I_xI_y \\ \sum I_xI_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum I_xI_t \\ -\sum I_yI_t \end{bmatrix}$$

$$\begin{bmatrix} \sum I_x^2 & \sum I_xI_y \\ \sum I_xI_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum I_xI_t \\ -\sum I_yI_t \end{bmatrix}$$

$$Au = B \quad A^{-1}Au = A^{-1}B \quad IU = A^{-1}B \quad u = A^{-1}B$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_x^2 & \sum I_xI_y \\ \sum I_xI_y & \sum I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_xI_t \\ -\sum I_yI_t \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\sum I_x^2 \sum I_y^2 - (\sum I_xI_y)^2} \begin{bmatrix} \sum I_y^2 & -\sum I_xI_y \\ -\sum I_xI_y & \sum I_x^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_xI_t \\ -\sum I_yI_t \end{bmatrix}$$

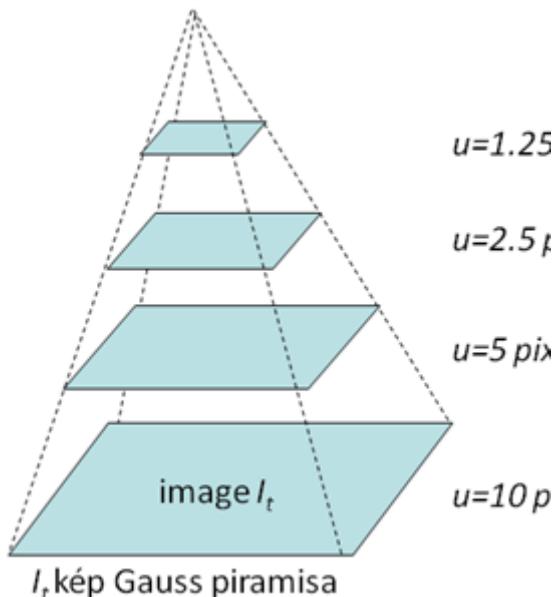
3.5. Fontos megjegyzések

- A Horn-Schunck és a Lucas-Kanade optical flow módszer csak kis mozgásokra működik
- Ha az objektum gyorsan mozog, akkor a fényessége (intenzitása) gyorsan változik, és a derivált értékek nem jól közelítik a térbeli és az időbeli deriváltakat

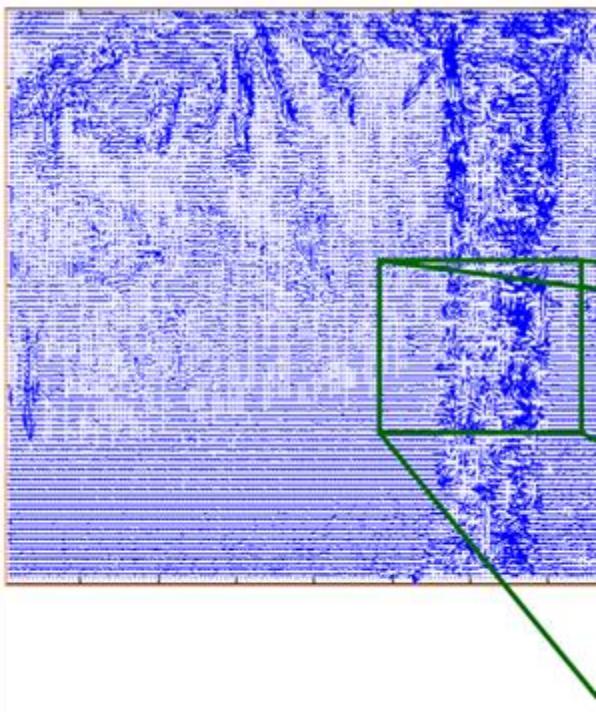
- Homogén részknél nem alkalmazható (gradiens!), textúra is gond lehet
- A piramis technika alkalmazható nagyobb mozgások esetén

3.6. Optikai folyam számítása piramis módszerrel

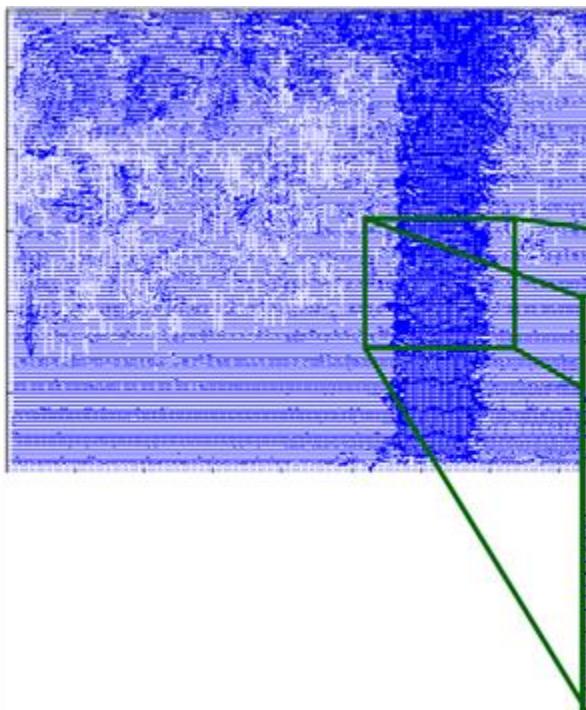
- A kis méretű képeken kis mozgást feltételezünk
- Az eredményeket visszaterjesztjük nagyobb képre



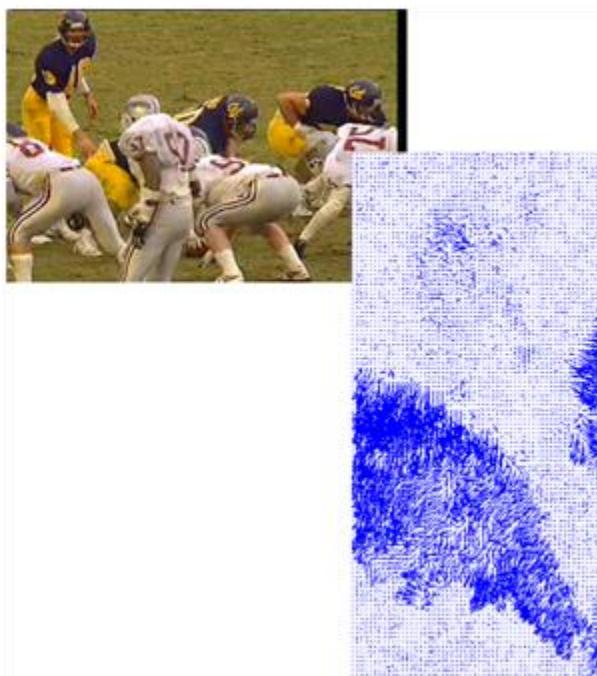
3.7. Optikai folyam meghatározása nagy mozgás esetén – hiba

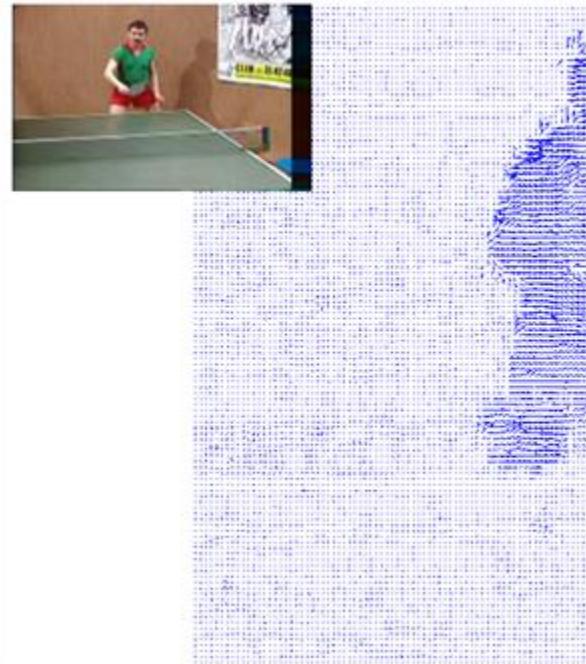


3.8. Optikai folyam meghatározása piramis módszerrel



3.9. Optikai folyam példa





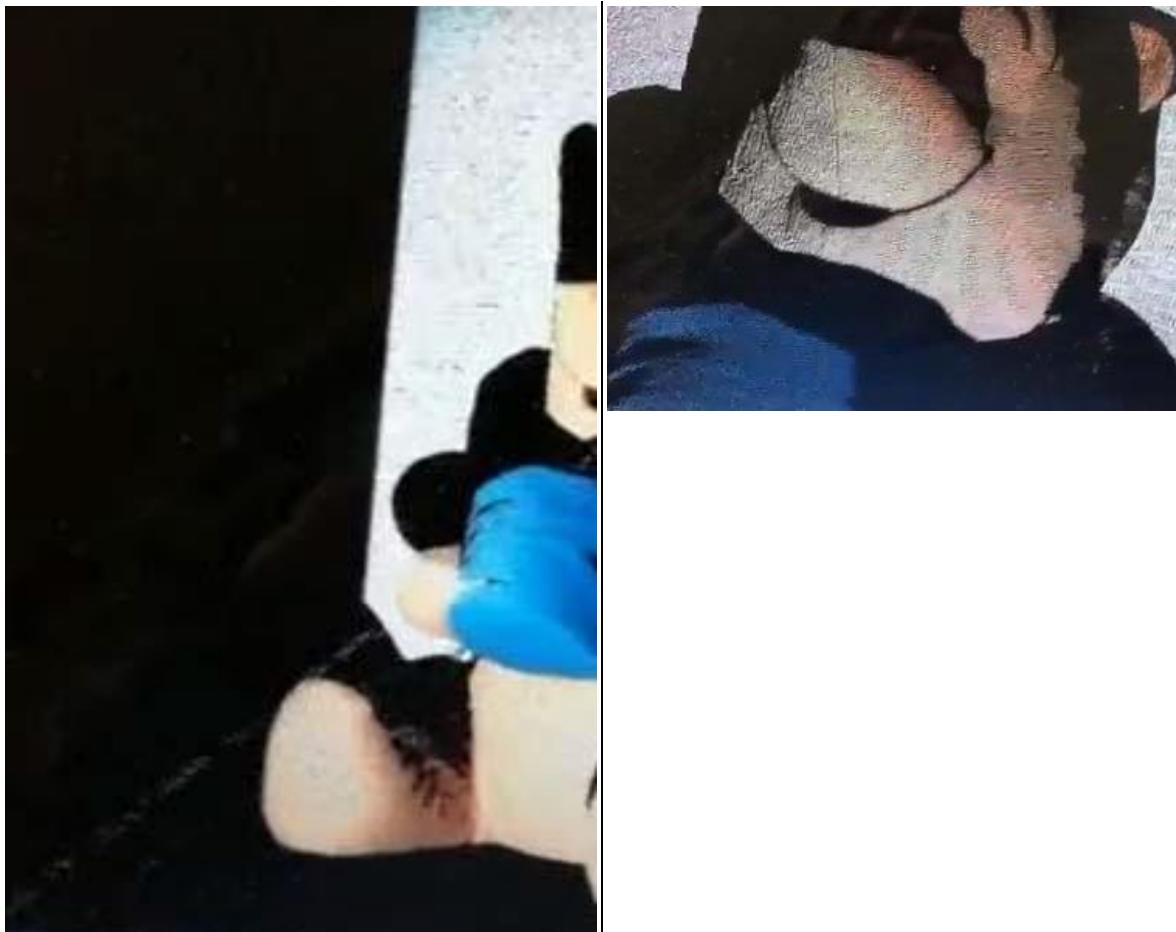
4. Irodalomjegyzék

- [5] M. Sonka, V. Hlavac, R. Boyle: *Image Processing, Analysis and Machine Vision*. Thomson, 2008
Sebastian Thrun (Carnegie Mellon University), Gary Bradski, Daniel Russakoff, Stanford CS223B Computer Vision, <http://robots.stanford.edu/cs223b>

Felhasznált és javasolt irodalom

- [1] S. Thrun, G. Bradski, D. Russakoff , *Computer Vision, CS223B*, Stanford University, <http://robots.stanford.edu/cs223b>
- [2] D. A. Forsyth, J. Ponce , *Computer Vision: A Modern Approach*, Prentice Hall, p. 792. 2003.
- [3] E. Trucco, A. Verri , *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, ISBN: 0-13-261108-2, p. 343. 1998.
- [4] R. Szeliski , *Computer Vision: Algorithms and Applications*, Springer, ISBN: 978-1-84882-934-3, p. 812. 2011.
- [5] S. Seitz, R. Szeliski , *Computer Vision (CSE 576)*, University Washington, 2012.

12. fejezet - Sztereó látás, 3D rekonstrukció



Az élővilágban a sztereó látás elsősorban a mélység érzékeltetése szempontjából nélkülözhetetlen, azaz két különböző nézőpontból készített képből meghatározhatók az azokon megfigyelhető pontok térbeli koordinátái egy előre megválasztott koordináta rendszerben. Ehhez ismernünk kell a kamerák belső és külső paramétereit (kamera kalibráció), és be kell tudnunk azonosítani az egymásnak megfelelő pontokat a különböző nézőpontkból készített képeken.

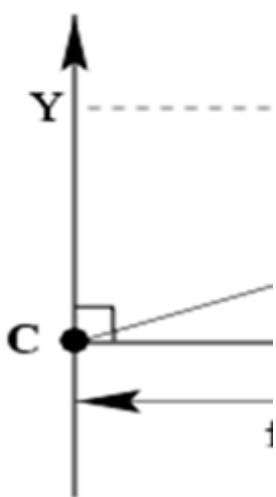
A következőkben tekintsük át néhány a 3D rekonstrukció során alkalmazott módszert és eljárást.

1. Perspektív vetítés és a kamera paraméterei

1.1. "Pinhole" kamera modell

A "pinhole" kamera modell a térbeli pont koordinátái és annak egy ideális ún. "pinhole" kamera képsíkján alkotott vetülete között fennálló matematikai összefüggést írja le. A pinhole kamera esetében a rekesz pontszerű és a fény fókusztávolsához nem alkalmaz lencséköt. A modell nem veszi figyelembe a lencsékből adódó torzulásokat, a képi felbontást, stb.

Kamera középpont **C**, képsík és a főtengely metszéspontja **P**, **M** - tetszőleges térbeli pont, az **M** pont képsíkon alkotott vetülete **m**



1.2. Perspektív vetítés és a kamera paraméterei

Az M térbeli pont képsíkon alkotott vetületének koordinátaira érvényes:

$$x = \frac{Xf}{Z}$$

$$y = \frac{Yf}{Z}$$

Felírhatjuk az alábbi egyenletet:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

1.3. Kamera belső paraméterei

Kép koordinátákból pixel koordinátákra való áttérés:

$$x_s = s_x x$$

$$y_s = s_y y$$

Mátrixos alakban kifejezve:

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Origó eltolása:

$$x' = x_s + o_x$$

$$y' = y_s + o_y$$

Homogén koordinátákkal kifejezve mátrixos alakban:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Teljes transzformáció a kamera koordináta-rendszerből (3D) pixel koordináta-rendszerbe (2D):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} fs_x & 0 & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}_e} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Legyen \mathbf{M} egy térbeli pont a kamera koordináta-rendszerben $\mathbf{M}_c = [X_c \ Y_c \ Z_c]$, ill. a világ koordináta-rendszerben, $M_w = [X_w \ Y_w \ Z_w]^T$ koordinátákkal kifejezve. Legyen továbbá

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

egy forgató mátrix és $\mathbf{T} = [t_x \ t_y \ t_z]^T$ egy eltolás vektor.

A kamera koordináta-rendszerbeli koordinátákat az alábbi módon fejezhetjük ki:

$$\mathbf{M}_c = \mathbf{R} (\mathbf{M}_w - \mathbf{T}),$$

azaz

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_w - t_x \\ Y_w - t_y \\ Z_w - t_z \end{bmatrix}$$

$$\mathbf{K}_e = \begin{bmatrix} r_{11} & r_{12} & r_{13} - \mathbf{R}_1 \mathbf{T} \\ r_{21} & r_{22} & r_{23} - \mathbf{R}_2 \mathbf{T} \\ r_{31} & r_{32} & r_{33} - \mathbf{R}_3 \mathbf{T} \end{bmatrix},$$

ahol $\mathbf{R}_i = [r_{i1} \ r_{i2} \ r_{i3}]$, $i = 1 \dots 3$. A projekciós mátrix az alábbi módon adódik:

$$\begin{bmatrix} x_k \\ y_k \\ w \end{bmatrix} = \mathbf{K}_{\text{in}} \mathbf{K}_{\text{ex}} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

2. Kamera kalibráció

2.1. Kamera kalibráció feladata

A kamera kalibráció feladata meghatározni a kamera külső és belső paramétereit. A külső paraméterek a kamera koordináta-rendszer és a világ koordináta-rendszer közti viszonyt (forgatás és eltolás) írják le, míg a belső paraméterek a kamera jellemzőire vonatkoznak, mint pl. a fókusztávolság, a pixel koordináták és a kép koordináták közti kapcsolat, lencsetorzítás. Ezek ismeretében egy tetszőleges világ koordináta-rendszerbeli pont kamera képsíkon alkotott vetületének pixel koordinátái meghatározhatók (lásd előző fejezet).

2.2. Zhang-féle kamera kalibráció

Legyen adott egy \mathbf{M} pont és annak képsíkon alkotott vetülete \mathbf{m} . Az \mathbf{M} és \mathbf{m} közti kapcsolatot az előzőek során bemutatott modell írja le, azaz:

$$s\mathbf{m} = \underbrace{\begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}}_{[\mathbf{R} \mathbf{t}]} \mathbf{M}$$

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} [\mathbf{r}_1 \mathbf{r}_2 \mathbf{r}_3 \mathbf{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Ha a kalibrációs pontok a világ koordináta-rendszer XY síkjában helyezkednek el a fenti összefüggés átírható az alábbi módon:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} [\mathbf{r}_1 \mathbf{r}_2 \mathbf{r}_3 \mathbf{t}] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{A} [\mathbf{r}_1 \mathbf{r}_2 \mathbf{t}] \begin{bmatrix} X \\ Y \\ 1 \\ 1 \end{bmatrix}$$

Így a kalibrációs sík és a kamera képsíkja között homográfia áll fenn:

$$s\mathbf{m} = \mathbf{HM}$$

$$\mathbf{H} = \lambda \mathbf{A} [\mathbf{r}_1 \mathbf{r}_2 \mathbf{t}],$$

ahol λ egy tetszőleges skalár.

Mivel a forgató mátrix ortonormált, felírhatjuk az alábbiakat:

$$\mathbf{r}_1 \perp \mathbf{r}_2 \rightarrow \mathbf{r}_1^T \mathbf{r}_2 = 0$$

$$\|\mathbf{r}_i\| = 1 \rightarrow \mathbf{r}_i^T \mathbf{r}_i = 1$$

$$[\mathbf{h}_1 \mathbf{h}_2 \mathbf{h}_3] = \lambda \mathbf{A} [\mathbf{r}_1 \mathbf{r}_2 \mathbf{t}]$$

Ebből következik, hogy

$$\mathbf{r}_1 = \frac{1}{\lambda} \mathbf{A}^{-1} \mathbf{h}_1; \mathbf{r}_2 = \frac{1}{\lambda} \mathbf{A}^{-1} \mathbf{h}_2$$

Behelyettesítve a fentieket a következőket kapjuk:

$$\begin{aligned} \mathbf{r}_1^T \mathbf{r}_2 &= 0 \Rightarrow \left(\frac{1}{\lambda} \mathbf{A}^{-1} \mathbf{h}_1 \right)^T \left(\frac{1}{\lambda} \mathbf{A}^{-1} \mathbf{h}_2 \right) = 0 \\ &\Rightarrow \frac{1}{\lambda^2} (\mathbf{A}^{-1} \mathbf{h}_1)^T (\mathbf{A}^{-1} \mathbf{h}_2) = 0 \\ &\Rightarrow \frac{1}{\lambda^2} \mathbf{h}_1^T \underbrace{\mathbf{A}^{-T} \mathbf{A}^{-1}}_{\mathbf{B}} \mathbf{h}_2 = 0 \end{aligned}$$

illetve

$$\begin{aligned} \mathbf{r}_1^T \mathbf{r}_2 &= 1 \Rightarrow \left(\frac{1}{\lambda} \mathbf{A}^{-1} \mathbf{h}_1 \right)^T \left(\frac{1}{\lambda} \mathbf{A}^{-1} \mathbf{h}_1 \right) = 1 \\ &\Rightarrow \frac{1}{\lambda^2} \mathbf{h}_1^T \underbrace{\mathbf{A}^{-T} \mathbf{A}^{-1}}_{\mathbf{B}} \mathbf{h}_1 = 1 \\ &\Rightarrow \mathbf{h}_1^T \underbrace{\mathbf{A}^{-T} \mathbf{A}^{-1}}_{\mathbf{B}} \mathbf{h}_1 = \mathbf{h}_2^T \underbrace{\mathbf{A}^{-T} \mathbf{A}^{-1}}_{\mathbf{B}} \mathbf{h}_2 \end{aligned}$$

Szorzás után \mathbf{B} elemeire az alábbiak adódnak:

$$\begin{aligned} \mathbf{B} &= \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \\ \mathbf{B} &= \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2 \beta} & \frac{\nu_0 \gamma - u_0 \beta}{\alpha^2 \beta} \\ -\frac{\gamma}{\alpha^2 \beta} & \frac{\gamma^2 + 1}{\alpha^2 \beta + \beta^2} & -\frac{\gamma(\nu_0 \gamma - u_0 \beta)}{\alpha^2 \beta^2} - \frac{\nu_0}{\beta^2} \\ \frac{\nu_0 \gamma - u_0 \beta}{\alpha^2 \beta} & -\frac{\gamma(\nu_0 \gamma - u_0 \beta)}{\alpha^2 \beta^2} - \frac{\nu_0}{\beta^2} & \frac{(\nu_0 \gamma - u_0 \beta)^2}{\alpha^2 \beta^2} - \frac{\nu_0^2}{\beta^2} + 1 \end{bmatrix} \end{aligned}$$

Jelöljük a \mathbf{H} homográfia mátrix oszlopait az alábbi formában:

$$\mathbf{h}_i = [h_{i1} \ h_{i2} \ h_{i3}]^T$$

Az előzőek során kapott $\mathbf{h}_i^T \underbrace{\mathbf{A}^{-T} \mathbf{A}^{-1}}_{\mathbf{B}} \mathbf{h}_j$ összefüggésbe helyettesítve a fentieket a következőt írhatjuk fel:

$$\begin{aligned} \mathbf{h}_i^T \mathbf{B} \mathbf{h}_j &= [h_{i1} \ h_{i2} \ h_{i3}] \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} h_{j1} \\ h_{j2} \\ h_{j3} \end{bmatrix} = \\ &= h_{i1} (b_{11} h_{j1} + b_{12} h_{j2} + b_{13} h_{j3}) + h_{i2} (b_{21} h_{j1} + b_{22} h_{j2} + b_{23} h_{j3}) + h_{i3} (b_{31} h_{j1} + b_{32} h_{j2} + b_{33} h_{j3}) \end{aligned}$$

Figyelembe véve, hogy \mathbf{B} szimmetrikus mátrix a fentiek alapján az alábbi egyenletrendszer írhatjuk fel:

$$\underbrace{\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix}}_{\mathbf{v}} \mathbf{b} = \mathbf{0},$$

ahol

$$\mathbf{v}_y = [h_{11}h_{j1} \ h_{11}h_{j2} + h_{12}h_{j1} \ h_{12}h_{j2} \ h_{11}h_{j3} + h_{13}h_{j1} \ h_{12}h_{j3} + h_{13}h_{j2} \ h_{13}]^T$$

$$\mathbf{b} = [b_{11} \ b_{12} \ b_{22} \ b_{13} \ b_{23} \ b_{33}]$$

A kalibrációs elem (pl. sakktábla) különböző elhelyezései különböző homográfiákat eredményeznek. minden homográfia két sorral bővíti a \mathbf{V} mátrixot.

Végül a fenti egyenletrendszer megoldása után a kamera belső paramétereit \mathbf{B} mátrix elemeinek ismeretében az alábbi módon fejezhetjük ki.

$$\nu_0 = (b_{12}b_{13} - b_{11}b_{23}) / (b_{11}b_{22} - b_{12}^2)$$

$$\lambda = b_{33} - [b_{13}^2 + \nu_0(b_{12}b_{13} - b_{11}b_{23})] / b_{11}$$

$$\alpha = \sqrt{\lambda / b_{11}}$$

$$\beta = \sqrt{\lambda b_{11} / (b_{11}b_{22} - b_{12}^2)}$$

$$\gamma = -b_{12}\alpha^2\beta / \lambda$$

$$u_0 = \gamma\nu_0 / \alpha - b_{13}\alpha^2 / \lambda$$

A kamera külső paramétereire a következőket írhatjuk fel:

$$\mathbf{r}_1 = \frac{\mathbf{A}^{-1}\mathbf{h}_1}{\|\mathbf{A}^{-1}\mathbf{h}_1\|}$$

$$\mathbf{r}_2 = \frac{\mathbf{A}^{-1}\mathbf{h}_2}{\|\mathbf{A}^{-1}\mathbf{h}_2\|}$$

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$$

$$\mathbf{t} = \frac{\mathbf{A}^{-1}\mathbf{h}_3}{\|\mathbf{A}^{-1}\mathbf{h}_3\|}$$

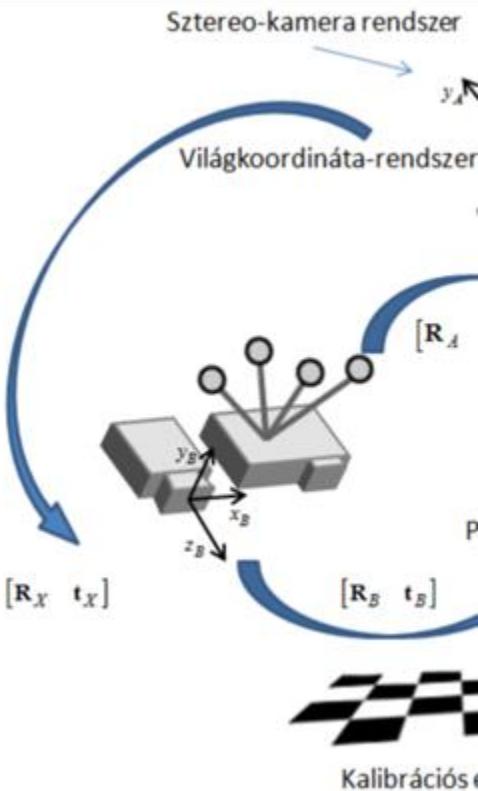
2.3. Felhasznált és javasolt irodalom

[1] Zhengyou Zhang, *A Flexible New Technique for Camera Calibration*. Microsoft Research Technical Report MSR-TR-98-71, Microsoft Corporation, Redmond, WA 98052, 2 December, 1998.

[2] Michael Greenspan, *Elec 824 Course Notes, Calibration II: Flexible Calibration, Chapter 14*. Kingston, Canada, pp. 1-10, 2008.

2.4. Hand-eye kalibráció

Tekintsük az alábbi "hand-eye" típusú rendszer kalibrációját:



2.5. "Hand-eye" kalibráció

Az ábrán látható \mathbf{R}_x forgató mátrixra és \mathbf{t}_x vektorra az alábbiakat írhatjuk fel, amely a "hand-eye" eszköz egyik elmozdulására vonatkozik.

$$\begin{aligned} \mathbf{R}_A \mathbf{R}_x &= \mathbf{R}_X \mathbf{R}_B \\ (\mathbf{R}_A - \mathbf{I}) \mathbf{t}_x &= \mathbf{R}_X \mathbf{t}_B - \mathbf{t}_A \end{aligned} \quad (12.1)$$

ahol \mathbf{I} 3x3-as egységmátrixot jelöl. A fenti egyenlet megoldásához használjuk fel a rotációs mátrixok algebrai és geometriai tulajdonságait.

$$\mathbf{R}_A = \mathbf{R}_X \mathbf{R}_B \mathbf{R}_X^T \quad (12.2)$$

Az (12.2) egyenlet hasonlósági transzformáció, mivel \mathbf{R}_x ortogonalis, így \mathbf{R}_A és \mathbf{R}_B sajátértékei megegyeznek. A forgató mátrixok egyik tulajdonsága, hogy egyik sajátértéke minden egységnyi. Legyen \mathbf{n}_B az ehhez tartozó sajátvektor. Mindezkből kiindulva (12.2) az alábbi módon is kifejezhető:

$$\mathbf{R}_A \mathbf{R}_X \mathbf{n}_B = \mathbf{R}_X \mathbf{R}_B \mathbf{n}_B = \mathbf{R}_X \mathbf{n}_B$$

$$\mathbf{n}_A = \mathbf{R}_X \mathbf{n}_B$$

Az eszköz minden elmozdulására felírható (12.1). n elmozdulás esetén az alábbiakat írhatjuk fel:

$$\begin{aligned} f_1(\mathbf{R}_X) &= \sum_{i=1}^n \|\mathbf{n}_{Ai} - \mathbf{R}_X \mathbf{n}_{Bi}\|^2 \\ f_2(\mathbf{R}_X, \mathbf{t}) &= \sum_{i=1}^n \|\mathbf{R}_X \mathbf{t}_B - (\mathbf{R}_A - \mathbf{I}) \mathbf{t}_x - \mathbf{t}_A\|^2 \end{aligned}$$

Két megközelítés:

- Először \mathbf{R}_x -et határozzuk meg f_1 -et minimalizálva utána \mathbf{t}_x -et f_2 minimalizálásával.

- \mathbf{R}_x és \mathbf{t}_x egyidejű meghatározása $f_1 + f_2$ minimalizálásával.

f_1 minimalizálását egységnyi quaterniók segítségével végezzük el, így f_1 felírható az alábbi formában:

Quaterniók áttekintése

$$f_1(\mathbf{R}_X) = f_1(\mathbf{q}) = \mathbf{q}^T \mathbf{A} \mathbf{q}$$

Bizonyítás

$$\begin{aligned}\mathbf{R}_X \mathbf{n}_B &= \mathbf{q} * \mathbf{n}_B \bar{\mathbf{q}} \\ \|\mathbf{n}_A - \mathbf{q} * \mathbf{n}_B * \bar{\mathbf{q}}\|^2 &= \|\mathbf{n}_A - \mathbf{q} * \mathbf{n}_B * \bar{\mathbf{q}}\|^2 \|\mathbf{q}\|^2 = \|\mathbf{n}_A * \mathbf{q} - \mathbf{q} * \mathbf{n}_B\|^2 \\ &= (\mathbf{Q}(\mathbf{n}_A) \mathbf{q} - \mathbf{W}(\mathbf{n}_B) \mathbf{q})^T (\mathbf{Q}(\mathbf{n}_A) \mathbf{q} - \mathbf{W}(\mathbf{n}_B) \mathbf{q}) = \mathbf{q}^T \mathbf{A}_i \mathbf{q} \\ \mathbf{A}_i &= (\mathbf{Q}(\mathbf{n}_A) - \mathbf{W}(\mathbf{n}_B))^T (\mathbf{Q}(\mathbf{n}_A) - \mathbf{W}(\mathbf{n}_B)) \\ f_1(\mathbf{R}_X) = f_1(\mathbf{q}) &= \sum_{i=1}^n \|\mathbf{n}_A - \mathbf{q} * \mathbf{n}_B * \bar{\mathbf{q}}\|^2 = \sum_{i=1}^n \mathbf{q}^T \mathbf{A}_i \mathbf{q} = \mathbf{q}^T \mathbf{A} \mathbf{q} \\ \mathbf{A} &= \sum_{i=1}^n \mathbf{A}_i\end{aligned}$$

Az előzőek alapján

$$\min_{\mathbf{R}_x} f_1(\mathbf{R}_X)$$

feladatot f_1 feltételes minimalizálásaként is értelmezhetjük. A feltétel esetünkben a q quaternióra vonatkozik, azaz q egységnyi kell legyen. A feladatot Lagrange-szorzók segítségével fogjuk megoldani:

$$\min_{\mathbf{q}} f_1 = \min_{\mathbf{q}} (\mathbf{q}^T \mathbf{A} \mathbf{q} + \lambda (1 - \mathbf{q}^T \mathbf{q}))$$

A hibafüggvény \mathbf{q} szerinti deriváltja: $\mathbf{A} \mathbf{q} = \lambda \mathbf{q}$

A megoldás az \mathbf{A} mátrix legkisebb (pozitív) sajátértékéhez tartozó sajátvektora.

- \mathbf{R}_x ismeretében $\mathbf{t}_x f_2$ minimalizálásával meghatározható, azaz:

$$\min_{\mathbf{t}_x} \sum_{i=1}^n \|\mathbf{R}_x \mathbf{t}_B - (\mathbf{R}_A - \mathbf{I}) \mathbf{t}_X - \mathbf{t}_A\|^2$$

- Legkisebb négyzetek módszere

2.6. Forgató mátrix és a quaterniók kapcsolata

A quaterniókat a komplex számok speciális eseteiként foghatjuk fel, melyeknek egy valós és három képzetes részük van, azaz

$$\mathbf{q} = q_0 + i q_x + j q_y + k q_z,$$

ahol

$$i^2 = j^2 = k^2 = ijk = -1$$

Quaterniók szorzása:

$$\mathbf{r} * \mathbf{q} = (r_0 + i r_x + j r_y + k r_z) (q_0 + i q_x + j q_y + k q_z)$$

$$\mathbf{r} * \mathbf{q} = \mathbf{Q}(\mathbf{r}) \mathbf{q} = \mathbf{W}(\mathbf{q}) \mathbf{r}$$

$$\mathbf{Q}(\mathbf{r}) = \begin{bmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & -r_z & r_y \\ r_y & r_z & r_0 & -r_x \\ r_z & -r_y & r_x & r_0 \end{bmatrix}, \mathbf{W}(\mathbf{r}) = \begin{bmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & r_z & -r_y \\ r_y & -r_z & r_0 & r_x \\ r_z & r_y & -r_x & r_0 \end{bmatrix}$$

A quaterniók néhány tulajdonsága

1. skaláris szorzat: $\mathbf{r} \cdot \mathbf{q} = r_0 q_0 + r_x q_x + r_y q_y + r_z q_z$

2. \mathbf{q} konjugált alakja: $\bar{\mathbf{q}} = q_0 - i q_x - j q_y - k q_z$

3. $\mathbf{q} * \bar{\mathbf{q}} = \mathbf{q} \cdot \mathbf{q} = \|\mathbf{q}\|^2$

4. $\|\mathbf{r} * \mathbf{q}\|^2 = \|\mathbf{r}\|^2 \|\mathbf{q}\|^2$

3-dimenziós vektor zérus valós részt tartalmazó quaternióként is értelmezhető. Ebben az esetben $\mathbf{W}(\mathbf{v})$ és $\mathbf{Q}(\mathbf{v})$ ferdeszimmetrikus mátrixok. Legyen \mathbf{q} egységnyi quaternió és legyen \mathbf{v} olyan quaternió melynek valós része zérus. Felírható az alábbi összefüggés:

$$\begin{aligned} \mathbf{v}' &= \mathbf{q} * \mathbf{v} * \bar{\mathbf{q}} \\ &= (\mathbf{Q}(\mathbf{q}) \mathbf{v}) * \bar{\mathbf{q}} \\ &= (\mathbf{W}(\mathbf{q})^T \mathbf{Q}(\mathbf{q})) \mathbf{v} \end{aligned}$$

$$\mathbf{W}(\mathbf{q})^T \mathbf{Q}(\mathbf{q}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z - q_0 q_y) \\ 0 & 2(q_x q_y + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 0 & 2(q_x q_z - q_0 q_y) & 2(q_y q_z + q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}$$

2.7. Felhasznált és javasolt irodalom

[1] Radu Horaud and Fadi Dornaika: *Hand-Eye Calibration*, International Journal of Robotics Research, Vol. 14, No. 3, pp. 195-210, 1995.

[2] Park, F.C.; Martin, B.J.: *Robot sensor calibration: solving AX=XB on the Euclidean group* IEEE Transactions on Robotics and Automation, Vol. 10, No. 5, pp. 717-721, Oct 1994. An Approach to Improve Online Hand-Eye Calibration

3. Epipoláris geometria

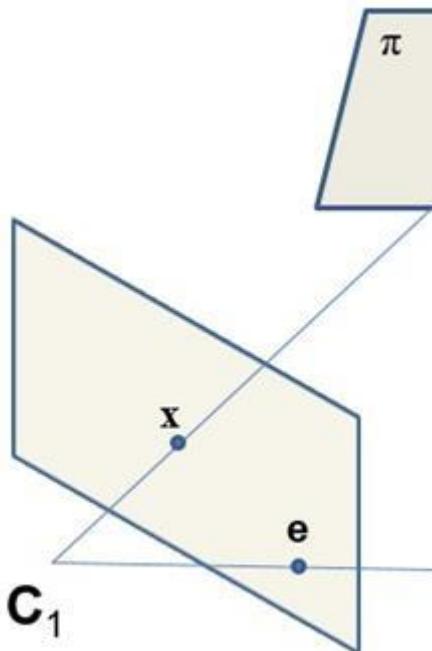
- Két nézet közti geometria
- Az egymásnak megfelelő pixelek meghatározása
- A megfeleltetés problémáját egyenes menti keresésre redukálja az ún. epipoláris egyenes mentén
- A kamerák belső paramétereitől és a kamerák relatív helyzetétől függ
- A kapcsolat az ún. fundamentális mátrix segítségével írható le az alábbi módon:

$$\mathbf{x}_i^T \mathbf{F} \mathbf{x}_i = 0,$$

ahol \mathbf{x}_i' és \mathbf{x}_i az egymásnak megfelelő pontokat reprezentálják a sztereó képpáron.

Sztereó párosítás (megfeleltetés)

- Kamera középpont \mathbf{C}_1 és \mathbf{C}_2
- Egy térbeli pont \mathbf{X} és annak vetületei \mathbf{x}'_i és \mathbf{x} , az ún. epipoláris síkon fekszenek
- Epipoláris egyenesek \mathbf{l} és \mathbf{l}'



3.1. Epipoláris geometria - Fundamentális mátrix

- \mathbf{F} meghatározásához nincs szükség a kamerák belső paramétereire, ill. a kamerák helyzetének ismeretére
- \mathbf{F} egymásnak megfelelő pontok ismeretében meghatározható

Legyen x' a kamera képsík egy pontja. Az x' ponton áthaladó l' epipoláris egyenesre érvényes a következő:

$$l' = e' \times x' = [e']_{\times} x'$$

A két képsík közti homográfián keresztül x -re felírható: $x' = H_x x$, azaz a fundamentális mátrix az alábbi módon adódik:

$$I' = \underbrace{\begin{bmatrix} e' \\ 1 \end{bmatrix}}_{\mathbf{f}} \underbrace{H_x}_{\mathbf{H}} x$$

A fundamentális mátrix a kamerák projekciós mátrixainak ismeretében is meghatározható. Jelölje \mathbf{C}_1 , ill. \mathbf{C}_2 a sztereó kamera páros középpontjait.

Legyen \mathbf{x} a \mathbf{C}_1 középponttal rendelkező kamera képsíkjának egyik pontja. Keressünk egy olyan pontot, amely a \mathbf{C}_2 pontok által meghatározott sugáron fekszik. Az $\mathbf{X} = \mathbf{P}^+ \mathbf{x}$ pont kielégíti az $\mathbf{x} = \mathbf{P} \mathbf{X}$ egyenletet, ahol \mathbf{P} a kamera projekciós mátrixát jelöli. $\mathbf{P}^+ = \mathbf{P}^T (\mathbf{P} \mathbf{P}^T)^{-1}$

$$\mathbf{P} \mathbf{X} = \mathbf{P} \mathbf{P}^T (\mathbf{P} \mathbf{P}^T)^{-1} \mathbf{x} = \mathbf{x}$$

Az \mathbf{X} pont képe a másik kamera esetében

$$\mathbf{x}' = \mathbf{P}' \mathbf{X} = \mathbf{P}' \mathbf{P}^+ \mathbf{x}.$$

\mathbf{C}_1 képe a \mathbf{C}_2 középpontú kamera esetében az $\mathbf{e}' = \mathbf{P} \mathbf{C}_1$ epipólus.

Az \mathbf{x} -hez tartozó epipoláris egyenesre felírható:

$\mathbf{l}' = (\mathbf{e}') \times (\mathbf{P}'\mathbf{P}'^+\mathbf{x}) = [\mathbf{e}'] \cdot \mathbf{P}'\mathbf{P}'^+\mathbf{x}$, $\mathbf{l}' = \mathbf{F}\mathbf{x}$. Mivel \mathbf{x}' az \mathbf{l}' -en fekszik érvényes $\mathbf{x}'\mathbf{l}' = 0$, azaz $\mathbf{x}'^T\mathbf{F}\mathbf{x} = 0$.

A fundamentális mátrix meghatározása egymásnak megfeleltethető pontok alapján: minden párosra felírható $\mathbf{x}'_i \mathbf{F} \mathbf{x}_i = 0$, azaz

$$\mathbf{x}'_1 f_{11} + \mathbf{x}'_1 y_{12} + \mathbf{x}'_1 f_{13} + \mathbf{y}'_1 x_{21} + \mathbf{y}'_1 y_{22} + \mathbf{y}'_1 f_{23} + \mathbf{x}'_1 f_{31} + \mathbf{y}'_1 f_{32} + \mathbf{y}'_1 f_{33} = 0$$

Az \mathbf{f} vektort az alábbi egyenlet megoldásaként kapjuk:

$$\begin{bmatrix} \mathbf{x}'_1 x_1 & \mathbf{x}'_1 y_1 & \mathbf{x}'_1 & \mathbf{y}'_1 x_1 & \mathbf{y}'_1 y_1 & \mathbf{y}'_1 & x_1 & y_1 & 1 \\ \mathbf{x}'_2 x_2 & \mathbf{x}'_2 y_2 & \mathbf{x}'_2 & \mathbf{y}'_2 x_2 & \mathbf{y}'_2 y_2 & \mathbf{y}'_2 & x_2 & y_2 & 1 \\ \vdots & \vdots \\ \mathbf{x}'_N x_N & \mathbf{x}'_N y_N & \mathbf{x}'_N & \mathbf{y}'_N x_N & \mathbf{y}'_N y_N & \mathbf{y}'_N & x_N & y_N & 1 \end{bmatrix} \mathbf{A} \mathbf{f} = \mathbf{0}$$

3.2. Felhasznált és javasolt irodalom

[1] R. Hartley and A. Zisserman: *Multiple View Geometry in Computer Vision* Cambridge University Press, 2000.

4. 3D rekonstrukció

- Megfelelő pontok problémája:
 - Amennyiben a célfelület mintázatokat tartalmaz, az egymásnak megfelelő pontok azonosíthatók (pl. az epipoláris geometria segítségével és a képpontok környezetének vizsgálatával)
 - Homogén felületek esetében a képpontok környezetének vizsgálata nem jelent megfelelő alternatívát. Ebben az esetben ugyanis nincsenek mintázatok, a képpontok környezetében nem tapasztalható jelentősebb eltérés (ha a fényviszonyok hatását is figyelembe vesszük)
- A problémát legegyszerűbben úgy tudjuk kiküszöbölni, ha mintázatokat vetítünk a célfelületre. Erre számos módszert találhatunk az irodalomban [1]-[5].

4.1. Kamera-lézer alapú 3D rekonstrukció

A következőkben a kamera-lézer alapú, ill. a kamera-projektor alapú megközelítéseket ismertetjük. Tekintsük az alábbi ábrát:

12.1. ábra - Kamera-lézer alapú 3D mérés



célobjekt

- Az objektum felületére lézercsíkot vetítve a megfelelő pontok az alábbi módon egyértelműen azonosíthatók
 - Legyen a világ koordináta-rendszer a kamera koordináta-rendszere. Miután a kamerát kalibráltuk (pl. a korábban ismertetett Zhang-féle eljárással) a kamera koordináta-rendszereben a lézercsík által generált Γ sík egyenlete meghatározható.
 - Az objektum felületére vetített lézercsíkon található 3D pont koordinátáit a Γ sík és a kamera középpontjából indított, a képsík megfelelő pixelén áthaladó sugár metszete adja (lásd 6. ábra).

Lézer kalibrálása

1. Legyen Π' egy tetszőleges sík. Jelölje továbbá Π a kamera képsíkját
2. Vezessünk be egy Π' síkbeli $u'v'$ Descartes koordináta-rendszert
 - Ha kalibrációs elemként pl. sakktáblát használunk, akkor Π' tekinthetjük a sakktábla síkjának, a sakktábla két egymásra merőleges külső élét pedig az u' , v' koordináta tengelyeknek. Ebben az esetben a sakktábla csomópontjainak $[u',v']$ koordinátái adottak.
3. Ismert koordinátájú P_i , $i = 1\dots N$ kontrollpontok megadása az $u'v'$ koordináta-rendzszerben (pl. az előző pontban említett sakktábla csomópontjai)
4. A kontrollpontok és azok képsíkon alkotott vetületeinek segítségével az $u'v'$ (tetszőlegesen megválasztott) koordináta-rendzszer és a pixelkoordináták közötti \mathbf{H} homográfia meghatározása.

$$\begin{bmatrix} wu'_a \\ wv'_a \\ w \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u_a \\ v_a \\ 1 \end{bmatrix},$$

ahol u'_a , v'_a a Π' síkban elhelyezett pont koordinátája, az u_a , v_a pedig a képsíkon alkotott vetületének pixelkoordinátái.

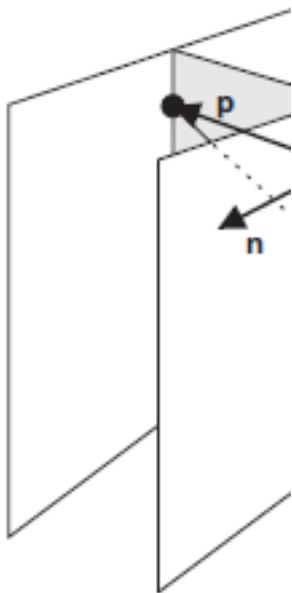
5. A Γ sík egyenletének a kamera koordináta-rendszerében (a választott világ koordináta-rendszerünk) való meghatározására, a Γ síkon fekvő, legalább három pont világkoordinátáira van szükség. Ezeket pl. az alábbi módszerrel határozhatjuk meg:

- Lézercsík vetítése a Γ síkra
- A kameraképből, a lézercsíkon fekvő pontokhoz tartozó képsíkbeli pixelkoordináták kinyerése
- A lézercsíkon fekvő pontok u', v' koordinátáinak meghatározása a \mathbf{H} homográfia segítségével.
- Az $[u', v', 0]$ koordináták világ koordináta-rendszerbe való transzformálása

6. Új Π' sík választása és az 1-5 lépések megismétlése

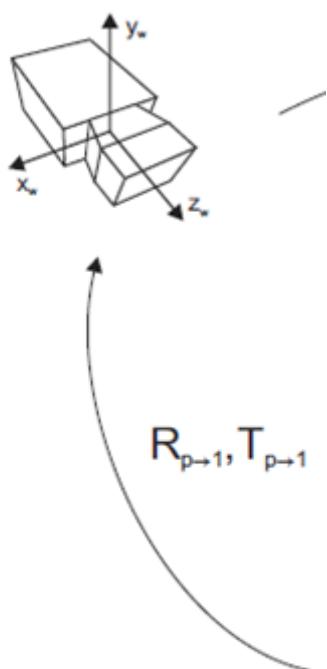
7. A Γ sík egyenletének meghatározása

12.2. ábra - Lézersík meghatározása



- Ahhoz, hogy egy felületet a KL (kamera-lézer) alapú rendszer segítségével rekonstruálhassunk az egység pozíciójának és orientációjának időbeni követése szükséges.
- Erre több alternatívával is találkozhatunk az irodalomban, ilyen pl. KL egység robotkarral való pozicionálása vagy kamerákkal való nyomon követése. Utóbbi esetben a KL egység pillanatnyi pozíciójának és orientációjának meghatározásához pl. az ahhoz rögzített markereket használhatjuk fel.
 - Csupán a markerek pozíciójának ismerete nem elegendő, azok egyértelmű azonosítása is nélkülözhetetlen ahhoz, hogy a relatív elmozdulás vektort és elforgatás mátrixot meg tudjuk határozni. Erre passzív és aktív módszereket is találunk az irodalomban (pl. modellillesztés alapú megoldások (passzív), aktív LED alapú módszerek (aktív)).

12.3. ábra - Kamera-lézer alapú rendszer sztereó kamerákkal való követése



Az egyes kamera képek függetlenül feldolgozhatók. Mind a lézer detektálás mind pedig a markerek detektálása az egyes képeken párhuzamosítható.

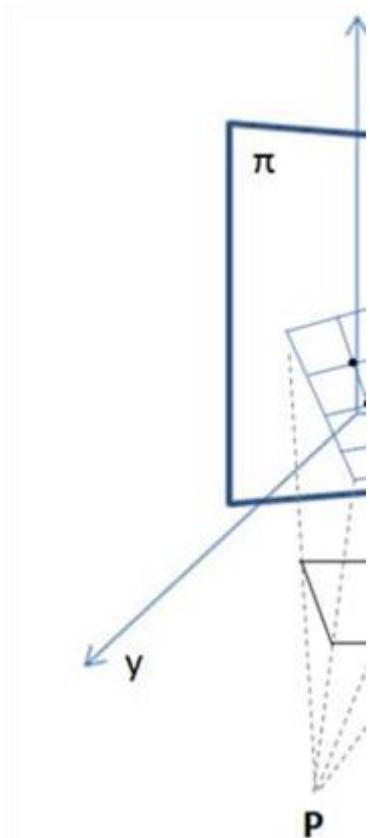
12.4. ábra - Kamera-lézer alapú 3D rekonstrukció párhuzamos megvalósításának folyamatábrája



4.2. Kamera-projektor alapú 3D rekonstrukció

A projektor kalibrálásánál a kivetített pl. sakktábla csúcsPontjait kell detektálni majd az uv és $u'v'$ koordináta-rendszerök közti homográfia segítségével a $[u_i, v_i, 0]$ koordinátákat meghatározni.

12.5. ábra - A projektor inverz kamera modellként való kalibrálása



Ezt követően alkalmazható a Zhang-féle kalibrációs eljárás.

- Ha egyszerre több mintát (párhuzamos csíkot) vetítünk a rekonstruálendő felületre, felmerül a síkok azonosításának problémája, ugyanis a kamera képen látható csíkok pontjairól nem tudjuk egyértelműen eldönteni, hogy azok melyik csík által generált síkon fekszenek.
- Az elsődleges probléma tehát a szóban forgó síkok azonosítása különféle kódolási technikák alkalmazásával
 - Bináris szekvencia
 - "Gray code" szekvencia
 - stb.
- Tekintsük a "Gray code" szekvencia alapú kódolást.
 - Ebben az esetben olyan bináris szekvenciával kódoljuk a projektor egyes oszlopait/sorait, melyben a szomszédos kódok csak egy bitben térnek el.
 - Ez elsősorban a hibásan azonosított sorok/oszlopok (pl. zajos kép esetén hibás detektálás) korrekciójára ad hatékony megoldást.

12.6. ábra - 4 bites Gray code



- A nehezen mérhető (takarásban lévő) felületelemek nagy pontosságú rekonstrukcióját a kamera-lézer alapú megoldás esetében alkalmazott többkamerás rendszerrel valósíthatjuk meg.
- A rendszer architektúráját a 8. ábra szemlélteti.
- Az aktív optikai alrendszer egymáshoz képest fix helyzetű komponensekből, azaz ebben az esetben egy projektorból és egy vagy több kamerából áll, melyek aktuális pozíciójának és orientációjának meghatározásánál pl. az alrendszerhez rögzített markerek detektálásából indulhatunk ki.

12.7. ábra - A 3D rekonstrukciós rendszer architektúrája



12.8. ábra - Szkennelés eredménye



Felhasznált és javasolt irodalom

- [1] Jason Geng., *Structured-light 3D surface imaging: a tutorial*. Advances in Optics and Photonics, Vol. 3, Issue 2, pp. 128-160, 2011.
- [2] E. Lilienblum, B. Michaelis., *Optical 3D Surface Reconstruction by a Multi-Period Phase Shift Method*. Journal of Computers, Vol. 2, No. 2, pp. 73-83, 2007.
- [3] N. Karpinsky, S. Zhang., *High-Resolution, Real-Time 3D Imaging with Fringe Analysis*. Journal of Real-Time Image Processing, Springer-Verlag, Vol. 7, Issue 1, pp. 55-66, 2010.
- [4] Yan Cui, Schuon, S., Chan, D., Thrun, S., Theobalt, C., *3D Shape Scanning with a Time-of-Flight Camera*. 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1173-1180, 13-18 June 2010.
- [5] Rövid, A.; Szeidl, L.; Yamage, Y.; Takeshi, M.; Hashimoto, T., *Indoor Real-Time 3D Reconstruction of Crash-Tested Car Cabins*. 8th IEEE International Symposium on Intelligent Systems and Informatics (SISY), pp. 257-261, 10-11 Sept. 2010.

13. fejezet - Fourier transzformáció és alkalmazásai

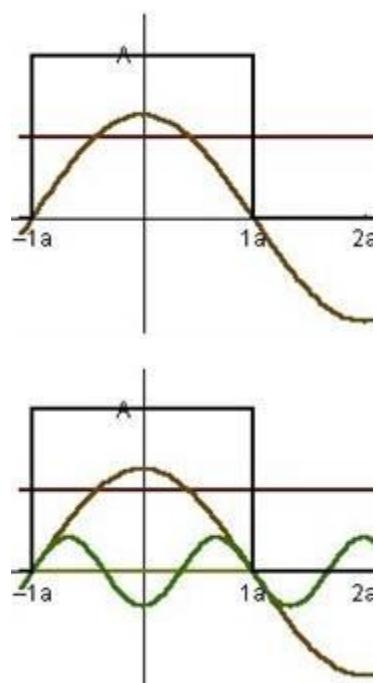


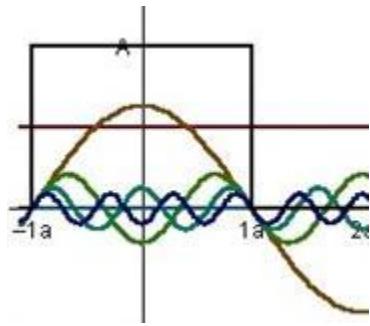
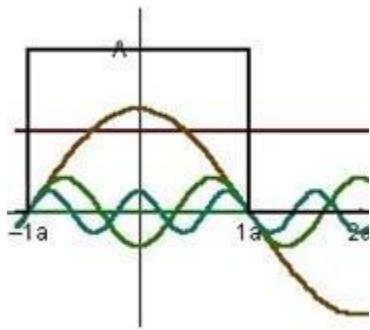
1. Fourier sorok és a Fourier transzformáció

- Jean Baptiste Joseph Fourier Auxerre (Franciaország) városában született 1768-ban
- Leghíresebb munkáját 1822-ben publikálta "La Théorie Analytique de la Chaleur" (angol fordításban 1878-ban jelent meg: The Analytic Theory of Heat) címmel.

Fourier sorok: minden periodikus függvény kifejezhető trigonometrikus függvényekből képzett függvényssoroként. Forrás - 2012.12.29.

13.1. ábra - Minél több a közelítésben résztvevő trigonometrikus függvények száma, annál pontosabb a közelítés





$$f(x) = a_0 + \left(\sum_{n=1}^{\infty} a_n \cos nx + b_n \sin nx \right)$$

$$\int_{-x}^x f(x) dx = \int_{-x}^x a_0 dx + \int_{-x}^x \left(\sum_{n=1}^{\infty} a_n \cos nx + b_n \sin nx \right) dx =$$

$$2\pi a_0 + \sum_{n=1}^{\infty} a_n \int_{-x}^x \cos(nx) dx + b_n \int_{-x}^x \sin(nx) dx$$

$$\int_{-x}^x \cos(nx) dx = \left[\frac{1}{n} \sin nx \right]_{-x}^x = \frac{1}{n} [\sin n\pi - \sin(-n\pi)] = 0$$

$$a_0 = \frac{1}{2\pi} \int_{-x}^x f(x) dx$$

$$\begin{aligned} \int_{-x}^x f(x) \cos(mx) dx &= \\ &= \int_{-x}^x \left[a_0 + \left(\sum_{n=1}^{\infty} a_n \cos nx + b_n \sin nx \right) \right] \cos(mx) dx \\ &= a_0 \int_{-x}^x \cos(mx) dx + \sum_{n=1}^{\infty} a_n \int_{-x}^x \cos(nx) \cos(mx) dx + \\ &+ \sum_{n=1}^{\infty} a_n \int_{-x}^x \sin(nx) \cos(mx) dx \end{aligned}$$

$$\int_{-x}^x \sin(nx) \cos(mx) dx = 0, \quad \forall m, n$$

$$\int_{-x}^x \cos(nx) \cos(mx) dx = \begin{cases} 0, & n \neq m \\ \pi, & n = m \end{cases}$$

$$a_n = \frac{1}{\pi} \int_{-x}^x f(x) \cos(nx) dx$$

$$b_n = \frac{1}{\pi} \int_{-x}^x f(x) \sin(nx) dx$$

$$\begin{aligned}
 f(x) &= \frac{a_0}{2} + \left(\sum_{n=1}^{\infty} a_n \cos(n\omega_0 x) + b_n \sin(n\omega_0 x) \right) = \\
 &= \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \frac{e^{inx} + e^{-inx}}{2} + b_n \frac{e^{inx} - e^{-inx}}{2i} = \\
 &= \frac{a_0}{2} + \sum_{n=1}^{\infty} \frac{a_n - ib_n}{2} e^{inx} + \sum_{n=1}^{\infty} \frac{a_n + ib_n}{2} e^{-inx} = \sum_{n=-\infty}^{\infty} \frac{a_n + ib_n}{2} e^{-inx}
 \end{aligned}$$

$$\begin{aligned}
 \int_{-T_0/2}^{T_0/2} f(x) e^{-ik\omega_0 x} dx &= \int_{-T_0/2}^{T_0/2} c_0 e^{-ik\omega_0 x} dx + \\
 &+ \int_{-T_0/2}^{T_0/2} c_1 e^{ik\omega_0 x} e^{-ik\omega_0 x} dx + \dots + \int_{-T_0/2}^{T_0/2} c_k e^{ik\omega_0 x} e^{-ik\omega_0 x} dx + \dots
 \end{aligned}$$

A nem periodikus jel felfogható egy a végtelenben ismétlődő periodikus jelként. Így bizonyos korlátokkal alkalmazni lehet a periodikus jelekre vonatkozó Fourier-sorfejtést. minden folytonos véges "energiával" rendelkező $x(t)$ jelre alkalmazható, azaz az alábbi feltételek teljesülnie kell:

$$\int_{-\infty}^{\infty} |f(x)|^2 dx < \infty$$

$$\begin{aligned}
 c_k &= \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} f(x) e^{-ik\omega_0 x} dx \\
 &= \sum_{n=-\infty}^{\infty} e^{in\omega_0 x} \frac{2\pi}{T_0} \frac{1}{2\pi} \int_{-T_0/2}^{T_0/2} f(x) e^{-inx} dx
 \end{aligned}$$

Ha T_0 végtelenhez tart $\lim_{T_0 \rightarrow \infty} \frac{2\pi}{T_0} = d\omega$, $f(x)$ -re az alábbiakat írhatjuk fel:

$$\begin{aligned}
 f(x) &= \int_{-\infty}^{\infty} e^{i\omega x} d\omega \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{-inx} dx = \\
 &= \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{i\omega x} \underbrace{\left[\int_{-\infty}^{\infty} f(x) e^{-inx} dx \right]}_{F(\omega)} d\omega
 \end{aligned}$$

Azaz az $f(x)$ függvény Fourier transzformáltjára az alábbi integrál adódik:

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-ix\omega} dx$$

A Fourier transzformáció inverze:

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{ix\omega} d\omega$$

1.1. Fourier transzformáció - kétváltozós eset

Kétváltozós függvények Fourier transzformáltja az alábbi módon fejezhető ki:

$$F(\omega_1, \omega_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-2\pi i(\omega_1 x + \omega_2 y)} dx dy$$

illetve az inverze:

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\omega_1, \omega_2) e^{j2\pi(\omega_1 x + \omega_2 y)} d\omega_1 d\omega_2$$

2. Diszkrét Fourier transzformáció (DFT)

- A vizsgált jelről csak mintavételi pontokban van információink.
 - Legyen $N = (2n + 1)$ pontban vett mintánk Δt mintavételezési frekvenciával, azaz $T = N\Delta t$.
- Ezeket a mért értékeket trigonometrikus bázisfüggvények lineáris kombinációjával kívánjuk előállítani.
- N pont esetében N darab bázisfüggvényre lesz szükségünk
- Milyen frekvenciákat válasszunk? Példa-1, Példa-2
- A Shannon-féle mintavételezési elv: Shannon mintavételi törvénye értelmében egy folytonos idejű jel elvileg tökéletesen visszaállítható mintából, ha a mintavételi frekvencia legalább kétszer akkora, mint a jel legmagasabb frekvenciájú komponense.

$$x(m\Delta t) = \sum_{k=0}^{N-1} X[k] e^{-jk\frac{2\pi}{T}m\Delta t} = \sum_{k=0}^{N-1} X[k] e^{-jk\frac{2\pi}{T}m\frac{T}{N}} = \sum_{k=0}^{N-1} X[k] e^{-j2\pi m\frac{k}{N}}$$

$$x[m] = \sum_{k=0}^{N-1} X[k] e^{jk\frac{2\pi}{N}m}, \quad m = 0 \dots N-1$$

N mérési pont esetén N ismeretlen és N darab egyenlet adódik, így az $X[k]$ együtthatók meghatározhatók.

$$X[k] = \frac{1}{N} \sum_{m=0}^{N-1} x[m] e^{-jk\frac{2\pi}{N}m}, \quad k = 0 \dots N-1$$

2.1. 2D Diszkrét Fourier transzformáció (DFT)

Kétváltozós eset:

DFT

$$X[k, l] = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x[m, n] e^{-j2\pi \left(\frac{km}{M} + \frac{ln}{N} \right)}$$

$k = 1 \dots M - 1; l = 1 \dots N - 1$

Inverz DFT

$$x[m, n] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} X[k, l] e^{j2\pi \left(\frac{km}{M} + \frac{ln}{N} \right)},$$

$m = 1 \dots M - 1; n = 1 \dots N - 1$

3. Gyors Fourier transzformáció (FFT)

Ahhoz, hogy a DFT segítségével minden

$$X[k] = \frac{1}{N} \sum_{m=0}^{N-1} f[m] e^{-jk\frac{2\pi}{N}m}, \quad k = 0 \dots N-1$$

elemet kiszámítunk N darab komplex szorzásra és $N-1$ darab összeadásra van szükség. Mivel a komplex szorzás számításigénye sokkal nagyobb az összeadásnál, ezért vegyük csak a komplex szorzásokat figyelembe. Ahhoz, hogy az $X[0], X[1], \dots, X[N-1]$ értékeket meghatározzuk N^2 komplex szorzásra van szükség.

1965-ben James Cooley and John Tukey módszert dolgoztak ki a gyors Fourier transzformáció (FFT) számítására, amely $N \log_2 N$ szorzást igényel.

Tételezzük fel, hogy $N=2^r$, $r=1,2,\dots$. Az FFT algoritmus alapgondolata, hogy N mintavételi pont esetében a DFT két párhuzamosan végrehajtható lépéstre bontható. Ezek mindegyike egy $N/2$ mintavételi pontból álló DFT. A folyamat rekurzív módon tovább tördelhető mindaddig, amíg el nem érjük a lépésekénti egy mintavételi pontra végrehajtott DFT-t.

Legyen $W_N = e^{-i2\pi/N}$. Felírhatjuk a következőt:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi nk/N} = \sum_{n=0}^{N-1} x[n] W_N^{kn}$$

Az $x[n]$ mérési pontjainkból állítsunk elő két $a[n]$ és $b[n]$ jelet az alábbi módon:

$$a[n] = x[2n]$$

$$b[n] = x[2n+1], n = 0, 1, \dots, N/2-1$$

$a[n]$ -re és $b[n]$ -re számított DFT-k a következők:

$$A[k] = \sum_{n=0}^{N/2-1} a[n] e^{-i2\pi nk/(N/2)}$$

$$B[k] = \sum_{n=0}^{N/2-1} b[n] e^{-i2\pi nk/(N/2)}$$

$X[k]$ kifejezhető az alábbi módon:

$$X[k] = \begin{cases} A[k] + W_N^k B[k] & k = 0, 1, \dots, \frac{N}{2}-1 \\ A\left[k - \frac{N}{2}\right] - W_N^k B[k] & k = \frac{N}{2}, \frac{N}{2}+1, \dots, N-1 \end{cases}$$

Tehát, ahhoz hogy az $X[k]$, $k = 1\dots N-1$ elemeket meghatározzuk ki kell számítani az $a[n]$ és $b[n]$ jelek DFT-jét, majd a kapott részeredményeket a fenti formula segítségével össze kell vonni.

Bizonyítás

$$\begin{aligned} A[k] + W_N^k B[k] &= \sum_{n=0}^{N/2-1} a[n] e^{-i2\pi nk/(N/2)} + \\ &+ W_N^k \sum_{n=0}^{N/2-1} b[n] e^{-i2\pi nk/(N/2)} = \\ &= \sum_{n=0}^{N/2-1} a[n] W_{N/2}^{nk} + \sum_{n=0}^{N/2-1} b[n] W_{N/2}^{nk} W_N^k \end{aligned}$$

Jegyezzük meg továbbá:

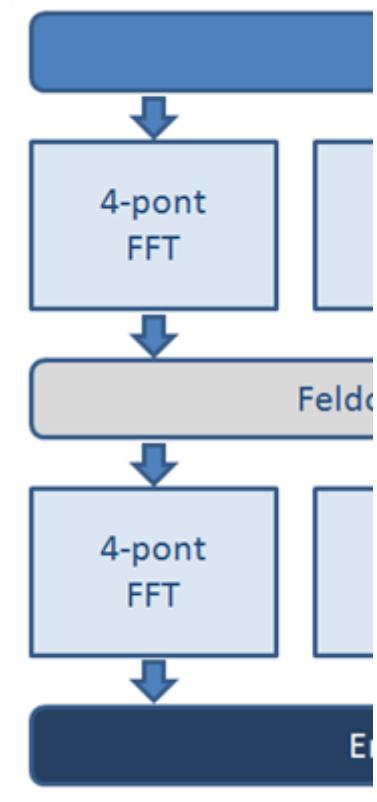
$$W_{N/2}^{nk} = \left(e^{-i2\pi n/(N/2)} \right)^{nk} = \left(e^{-i2\pi n/N} \right)^{2nk} = W_N^{2nk}$$

Behelyettesítve az alábbi adódik:

$$\begin{aligned}
 A[k] + W_N^k B[k] &= \sum_{n=0}^{N/2-1} a[n] W_N^{2nk} + \sum_{n=0}^{N/2-1} b[n] W_N^{(2n+1)k} = \\
 &= \sum_{n=0}^{N/2-1} x[2n] W_N^{2nk} + \sum_{n=0}^{N/2-1} x[2n+1] W_N^{(2n+1)k} = \\
 &= \sum_{n=0}^{N-1} x[\bar{n}] W_N^{nk} + \sum_{n=1}^{N-1} x[\bar{n}] W_N^{nk} = \sum_{n=0}^{N-1} x[n] W_N^{nk} = \\
 &= \sum_{n=0}^{N-1} x[n] e^{-i2\pi nk/N} = X[k]
 \end{aligned}$$

GPU-n való párhuzamos feldolgozás lehetőségének szemléltetése:

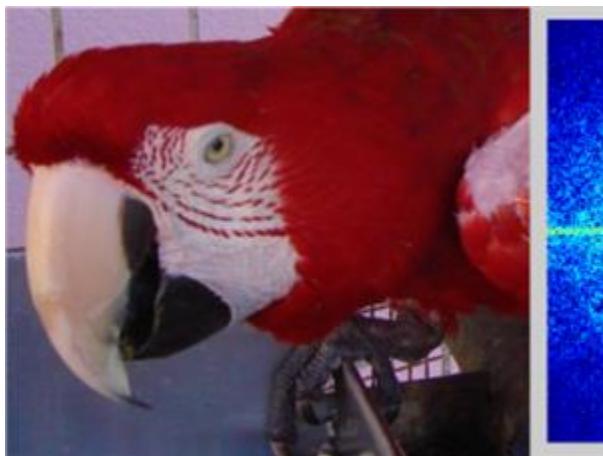
13.2. ábra - 16 pontos FFT 4 feldolgozó egység használatával



CUFFT - Az NVIDIA CUDA gyors Fourier transzformációt megvalósító könyvtára

3.1. Fourier transzformáció - példa

13.3. ábra - Eredeti kép (balra), A kép Fourier transzformáltja (jobbra)



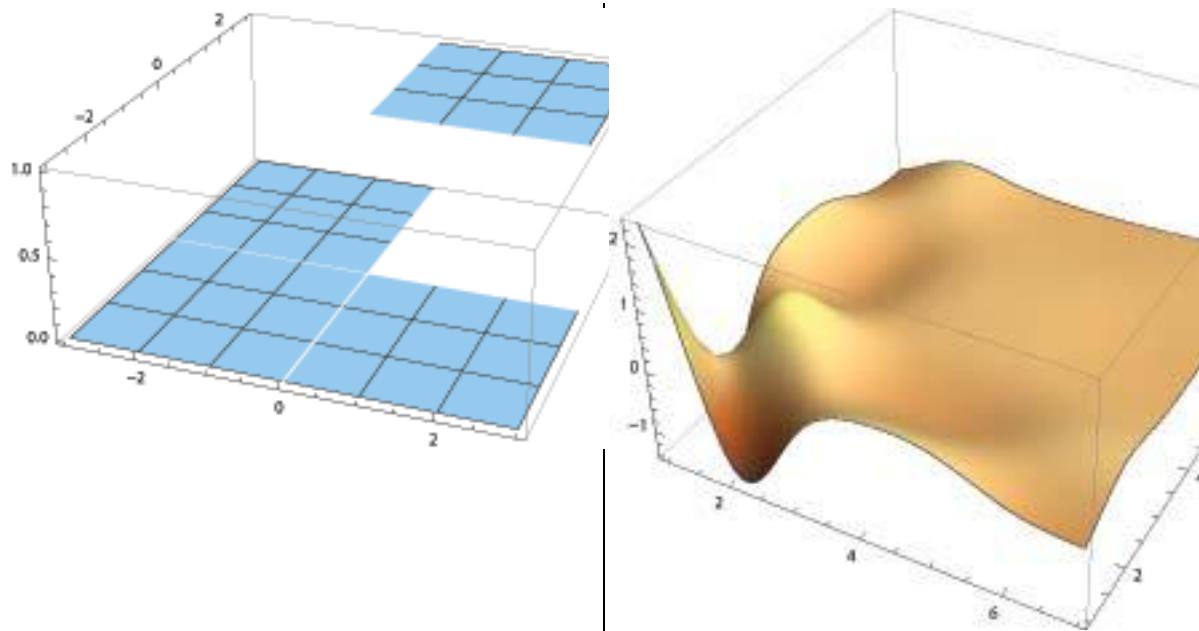
13.4. ábra - A nagyobb frekvenciák elhagyásának hatása



3.2. Felhasznált és javasolt irodalom

- [1] J. Fessler: *The Discrete Fourier Transform*, Chapter 5 (student version), p. 31. May 27, 2004.
- [2] *A diszkrét Fourier-transzformáció alkalmazása a jelfeldolgozásban*, előadás fóliák Hidrodinamikai rendszerek tanszék D.322, May 7, 2009. Forrás - 2013.02.01
- [3] Maxim Raginsky: *Lecture XI: The Fast Fourier Transform (FFT) algorithm*, BME 171: Signals and Systems, Duke University, October 17, 2008.
- [4] M. Sonka, V. Hlavac, R. Boyle: *Image Processing, Analysis, and Machine Vision, 3rd edition*, Thomson Learning, 2007.
- [5] R. C. Gonzalez and R. E. Woods: *Digital Image Processing (2nd Edition)*, Prentice Hall, January 2002.
- [6] Rövid, András and Szeidl, László and Várlaki, Péter: *Polylinear form of functions on HOSVD basis in relation to Fourier image processing*, Proceedings of the 14th WSEAS International Conference on Applied Mathematics, pp. 239-244, Canary Islands, 2009.

14. fejezet - Diszkrét koszinusz transzformáció



Tételezzük fel, hogy N mérési pontunk van. Hozunk létre egy új $2N$ pontot tartalmazó szekvenciát, az alábbi módon:

$$x'[m] = \begin{cases} x[m] & 0 \leq m \leq N-1 \\ x[-m-1] & -N \leq m \leq -1 \end{cases}$$

$$m' = m + 0.5$$

$$x'[m] = x'[m' - 0.5]$$

Az így kapott diszkrét jel koszinusz transzformáltját az alábbi módon írhatjuk fel:

$$\begin{aligned} X[n] &= \frac{1}{\sqrt{2N}} \sum_{m'=-N+0.5}^{N-0.5} x[m' - 0.5] e^{-j2\pi m' n / 2N} = \\ &= \frac{1}{\sqrt{2N}} \sum_{m'=-N+0.5}^{N-0.5} x[m' - 0.5] \cos\left(\frac{2\pi m' n}{2N}\right) = \\ &= \sqrt{\frac{2}{N}} \sum_{m'=-N+0.5}^{N-0.5} x[m' - 0.5] \cos\left(\frac{2\pi m' n}{2N}\right) \\ X[n] &= \sqrt{\frac{2}{N}} \sum_{m=0.5}^{N-0.5} x[m' - 0.5] \cos\left(\frac{2\pi m' n}{2N}\right) = \\ &= \sqrt{\frac{2}{N}} \sum_{m=0}^{N-1} x[m] \cos\left(\frac{n\pi(2m+1)}{2N}\right) = \sqrt{\sum_{m=0}^{N-1} k^2[n, m]} = \\ &= \sqrt{\frac{2}{N}} \sum_{m=0}^{N-1} \cos\left(\frac{n\pi(2m+1)}{2N}\right) = \begin{cases} \sqrt{2} & n = 0 \\ 1 & n = 1, 2, \dots, N-1 \end{cases} \\ x[n] &= \sum_{m=1}^{N-1} \alpha[n] X[n] \cos\left(\frac{n\pi(2m+1)}{2N}\right). \end{aligned}$$

Ahhoz, hogy a DCT ortogonális legyen, definiálni kell egy indextől függő $\alpha[n]$ koefficienset az alábbi módon:

$$x[n] = \sum_{m=1}^{N-1} \alpha[m] X[m] \cos\left(\frac{n\pi(2m+1)}{2N}\right),$$

$$\alpha[n] = \begin{cases} \sqrt{1/N} & n=0 \\ \sqrt{2/N} & n=1, 2, \dots, N-1 \end{cases}$$

1. Felhasznált és javasolt irodalom

[1] Ruye Wang: *Discrete Cosine Transform* -- E186 Handout",
<http://fourier.eng.hmc.edu/e161/lectures/dct/dct.html>, megtekintve 2013.04.05

15. fejezet - Waveletek és alkalmazásai



1. Rövid idejű Fourier transzformáció (STFT)

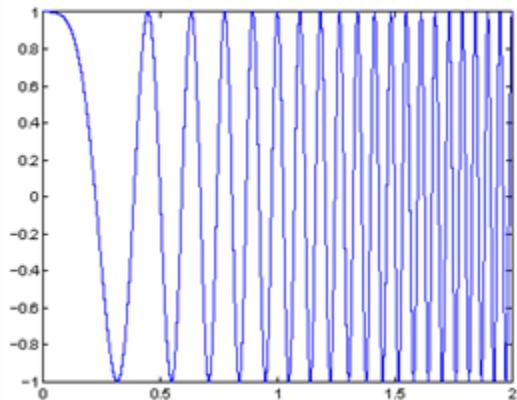
A rövid idejű Fourier transzformáció (STFT) esetében a jelet olyan szegmensekre bontjuk, melyeken belül az stacionáriusnak tekinthető. Erre a célra ablakfüggvényt használunk, amely egységnyi értéket vesz fel az ablakon belül és zérus az ablakon kívül. Először az ablakot az időtengely origójában helyezzük el. A jelet az ablakfüggvényvel beszorozva majd a Fourier transzformációt végrehajtva kapjuk a rövid idejű Fourier transzformációt. Ezt követően az ablakot tovább csúsztatjuk és a fenti lépéseket megismételjük mindaddig, amíg a jel végéhez nem érünk.

2. Wavelet transzformáció

- Az STFT segítségével a jel idő-frekvencia reprezentációját kapjuk
 - felbontási problémák
 - az ablak méretéből eredően az időbeliség pontosságának korlátai vannak.
- A wavelet transzformáció segítségével a jel idő-frekvencia reprezentációját kapjuk
 - nagyfrekvenciák esetében a wavelet transzformáció jó időbeni, de rossz frekvenciabeli felbontást eredményez
 - kisfrekvenciák esetében a wavelet transzformáció jó frekvenciabeli, de rossz időbeni felbontást eredményez

A előzőek során tárgyalt Fourier transzformáció segítségével meghatározható, hogy egy adott jelben milyen frekvenciák milyen mértékben vannak jelen. Arra azonban a Fourier transzformáció nem ad választ, hogy egy adott frekvencia mikor van jelen a jelben. Stacionárius jelek esetében a jel frekvencia összetevői időben nem változnak, azaz a spektrumban jelen lévő frekvenciák a jelben mindenkorban jelen vannak. Ebben az esetben nincs is szükségünk arra, hogy egy adott frekvencia melyik időben fordul elő, mivel minden frekvencia a jel teljes időtartama során jelen van. Nemstacionárius jelek esetében azonban a jel frekvenciatartalma időben változik.

15.1. ábra - Nemstacionárius jel (balra); Stacionárius jel (jobbra)



- Az előzőek során már megfigyelhettük, hogy Fourier transzformáció esetében az alapfüggvények komplex exponenciális függvények: $e^{i\omega t}$. A transzformáció eredménye pedig egy egyváltozós függvény a frekvenciatartományban: $F(\omega)$.
- Rövid idejű Fourier transzformáció esetében az alapfüggvények ún. ablakolt komplex exponenciális függvények: $w(t)e^{i\omega t}$, a transzformáció eredménye pedig egy kétváltozós függvény: $F(\omega\tau)$, amely a jel és az ω körfrekvenciával rendelkező szinuszoid közti hasonlóságot mutatja egy meghatározott intervallumon belül, melynek közepét az időtengelyen a τ időben helyeztük el.
- Folytonos wavelet transzformáció esetében az alapfüggvényünk az ún. wavelet "hullámcska". A transzformáció a jelet a wavelet függvény skálázott és eltolt változataival veti össze.
- Abban az esetben, ha a wavelet és a transzformált jel energiája egységes, a transzformáció eredményeképpen kapott kétváltozós függvényünk értékei korrelációs együtthatóknak felelnek meg.
- Ha a jelet különböző skálával és eltolással rendelkező waveletekkel vetjük össze (vizsgáljuk a jel és a különböző waveletek közti hasonlóságot) egy kétváltozós függvényt kapunk. Ha a wavelet komplex értékű függvény a transzformált is komplex értékű lesz.

Ha valós értékű jelről van szó akkor annak a transzformáltja is valós értékű függvénye lesz a skálának és eltolásnak.

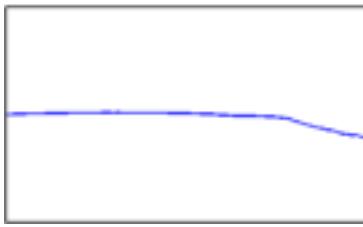
$$C(a, b, f(t), \psi(t)) = \int_{-\infty}^{\infty} f(t) \frac{1}{\sqrt{a}} \psi^*(\frac{t-b}{a}) dt,$$

ahol a a skálázást jelöli b pedig az eltolást. Ha folytonosan változtatjuk az a és b paramétereket, megkapjuk a transzformáció $C(a, b)$ koefficienseit.

2.1. Wavelet transzformáció - skála

Több különböző megengedett wavelettel találkozhatunk az irodalomban, melyeket a wavelet transzformáció esetében alkalmaznak. Attól függően, hogy milyen jellemzőket kívánunk detektálni a jelben, szabadon választhatunk a waveletek közül.

A skálázás hatása:



2.2. Wavelet transzformáció - skála és frekvencia

A skála és a frekvencia között az alábbi kapcsolat figyelhető meg:

- skála értéke kicsi: összenyomott wavelet, hirtelen változások figyelhetők meg, magas frekvencia
- skála értéke nagy: széthúzott wavelet, lassú változások, nincsenek részletek, alacsony frekvencia

Matlab Wavelet Toolbox

A skála alapján a frekvencia meghatározására az alábbi Matlab függvényeket használhatjuk:

```
scal2frq(A,'wname',DELTA)
```

```
centfrq('wname')
```

2.3. Wavelet transzformáció - felbontás

A wavelet analízis megengedi hosszú intervallumok alkalmazását ott, ahol szükségszerű az alacsonyfrekvenciájú komponensek precízebb feltérképezése és rövidebb intervallumot ott, ahol magasfrekvenciájú komponensek időbeni előfordulását szeretnénk pontosabban tudni, de ez a frekvenciafelbontás csökkenését vonja maga után (Heisenberg-féle határozatlansági elv alapján).

2.4. Wavelet transzformáció - inverz FT

A wavelet transzformációt frekvencia alapú szűrőként is értelmezhetjük, ha inverz Fourier transzformációként fejezzük ki, azaz

$$C(a, b, f(t), \psi(t)) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{f}(\omega) \sqrt{a} \psi(a\omega)^* e^{i\omega t} d\omega,$$

ahol $\hat{f}(\omega)$ és $\psi(\omega)$ a jel és a wavelet Fourier transzformáltjai.

2.5. Wavelet transzformáció - skála és wavelet

Tekintsük az alábbi folytonos függvényt,

$$\phi(t) = \begin{cases} 1 & 0 \leq t < 1 \\ 0 & \text{egyébként} \end{cases}$$

ill. annak különböző skálázott változatait

$$\{\phi_n(t)\} = \phi(nt)$$

Nézzük meg továbbá, hogy a $\{\phi_n(t)\} = \phi(nt)$ függvénycsalád ortonormált rendszert alkot-e. Könnyen észrevehető, hogy ezek a függvények nem ortonormáltak, mivel

$$\langle \phi_0(t), \phi_1(t) \rangle = \int_{-\infty}^{\infty} \phi_0(t) \phi_1(t) dt = \int_0^{1/2} dt = \frac{1}{2}$$

A Gram-Schmidt féle eljárással a $\{\phi_n(t)\}$ függvényrendszer ortogonalizálható az alábbi módon:

$$\begin{aligned} \phi'_0(t) &= \phi_0(t) = \phi(t) \\ \phi'_1(t) &= \phi'_1(t) - \frac{\langle \phi_1(t), \phi_0(t) \rangle}{\langle \phi_0(t), \phi_0(t) \rangle} \phi_0(t) = \begin{cases} 1/2 & 0 \leq t < 1/2 \\ -1/2 & 1/2 \leq t < 1 \\ 0 & \text{egyébként} \end{cases} \end{aligned}$$

Itt a $\phi(t)$ -t skálázó, $\psi(t)$ -t pedig wavelet függvénynek nevezzük.

A skálázó és wavelet függvények skálázásával és eltolásával ortonormált bázis állítható elő, azaz

$$\phi_{j,k}(t) = 2^{j/2} \phi(2^j t - k)$$

$$\psi_{j,k}(t) = 2^{j/2} \psi(2^j t - k)$$

Jelölje továbbá a $V_j = \text{span}\{\phi_{j,k}(t)\}$ skálázó függvények alterét, $W_j = \text{span}\{\psi_{j,k}(t)\}$ pedig a wavelet függvényekét. Érvényes az alábbi:

$$V_j \subset V_{j+1}$$

$$V_1 = V_0 \oplus W_0$$

$$V_2 = V_1 \oplus W_1$$

$$L^2(\mathbb{R}) = V_0 \oplus W_0 \oplus W_1 \oplus \dots$$

Az előzőekből kiindulva $\phi_{0,0}(t)$ -t az alábbi formában fejezhetjük ki:

$$\phi(t) = \phi_{0,0}(t) = \frac{1}{\sqrt{2}} \phi_0(t) + \frac{1}{\sqrt{2}} \phi_1(t),$$

$$\phi(t) = \frac{1}{\sqrt{2}} \sqrt{2} \phi(2t) + \frac{1}{\sqrt{2}} \sqrt{2} \phi(2t-1),$$

$$\phi(t) = \sum_n h_\phi[n] \sqrt{2} \phi(2t-n)$$

ahol

$$h_\phi[n] = \left\{ \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right\}$$

$\psi(t)$ esetében hasonló módon járhatunk el, azaz

$$h_\psi[n] = \left\{ \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \right\}$$

$$\text{ahol } \psi(t) = \sum_n h_\psi[n] \sqrt{2} \phi(2t-n),$$

3. Diszkrét Wavelet transzformáció

Az előzőek során ismertettük a skálázó és wavelet függvényeket, megmutattuk azok tulajdonságait. Tételezzük fel, hogy a skálázó és wavelet függvényeink Haar típusúak. Egy $f[n] \in L^2(\mathbb{Z})$ diszkrét jel a skálázó és wavelet diszkrét függvények segítségével az alábbi módon approximálható:

$$f[n] = \frac{1}{\sqrt{N}} \sum_k W_\phi[j_0, k] \phi_{j_0, k}[n] + \frac{1}{\sqrt{N}} \sum_{j=j_0}^{\infty} \sum_k W_\psi[j, k] \psi_{j, k}[n],$$

ahol $\phi_{j_0, k}[n]$, $\psi_{j_0, k}[n]$ és $f[n]$ a $[0, N-1]$ diszkrét intervallumon definiált diszkrét függvények. A $W_\phi[j_0, k]$ és $W_\psi[j, k]$ koefficienseket az alábbi módon határozhatjuk meg:

$$W_\phi[j_0, k] = \frac{1}{\sqrt{N}} \sum_n f[n] \phi_{j_0, k}[n], \quad W_\psi[j, k] = \frac{1}{\sqrt{N}} \sum_n f[n] \psi_{j, k}[n]$$

4. Gyors Wavelet transzformáció

Az előzőek alapján belátható, hogy

$$\phi_{j, k}[n] = 2^{j/2} \phi[2^j n - k] = 2^{j/2} \sum_n h_\phi[n] \sqrt{2} \phi(2(2^j n - k) - n')$$

Legyen $n' = m - 2k$, melynek segítségével $\phi_{j, k}[n]$ -t az alábbi módon is kifejezhetjük:

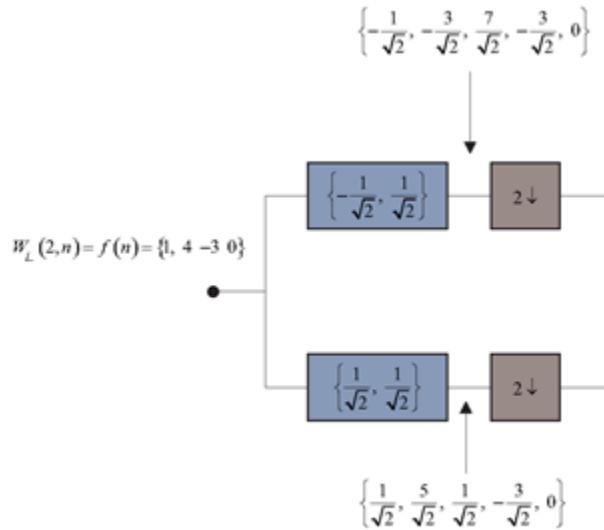
$$\phi_{j, k}[n] = 2^{j/2} \sum_n h_\phi[m - 2k] \sqrt{2} \phi(2^{j+1} n - m)$$

Az előzőet behelyettesítve a korábban ismertetett $W_\phi[j, k]$ koefficiensekre felírt formulába a következőket írhatjuk fel:

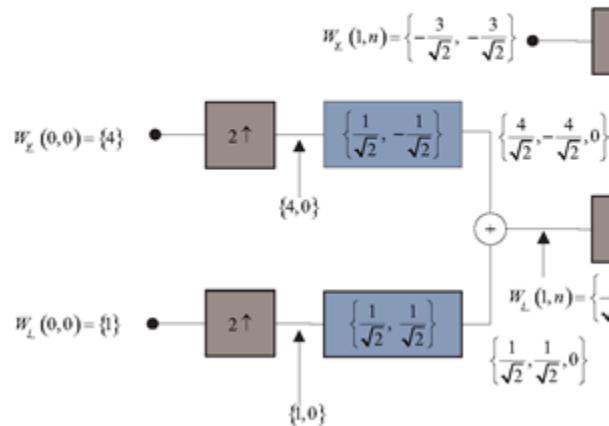
$$\begin{aligned} W_\phi[j, k] &= \frac{1}{\sqrt{N}} \sum_n f[n] \phi_{j, k}[n] = \\ &= \frac{1}{\sqrt{N}} \sum_n f[n] 2^{j/2} \phi(2^j n - m) = \\ &= \frac{1}{\sqrt{N}} \sum_n f[n] 2^{j/2} \sum_m h_\phi[m - 2k] \sqrt{2} \phi(2^{j+1} n - m) = \\ &= \sum_m h_\phi[m - 2k] \left(\frac{1}{\sqrt{N}} \sum_n f[n] 2^{(j+1)/2} \phi(2^{j+1} n - m) \right) \\ &= \sum_m h_\phi[m - 2k] W_\phi[j+1, m] = \\ &= h_\phi[-n] * W_\phi[j+1, n] \\ f[n] &= W_\phi[J, n] \end{aligned}$$

$W\psi$ -t hasonlóképpen fejezhetjük ki.

4.1. Wavelet transzformáció - példa



4.2. Inverz Wavelet transzformáció - példa



4.3. Diszkrét Wavelet transzformáció kétváltozós esetben

Kétváltozós esetben a skálázó és wavelet függvények az alábbi módon definiáltak:

$$\phi_{j,m,n}(u, v) = 2^{j/2} \phi(2^j u - m, 2^j v - n)$$

$$\psi_{j,m,n}(u, v) = 2^{j/2} \psi^{H,V,D}(2^j u - m, 2^j v - n)$$

A $\psi^{H,V,D}$ jelölés a három különböző wavelet függvényre utal.

Az egyváltozós esetből kiindulva a kétváltozós diszkrét wavelet transzformáció esetére az alábbit írhatjuk fel:

$$W_\phi(j_0, m, n) = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f(u, v) \phi_{j_0, m, n}(u, v)$$

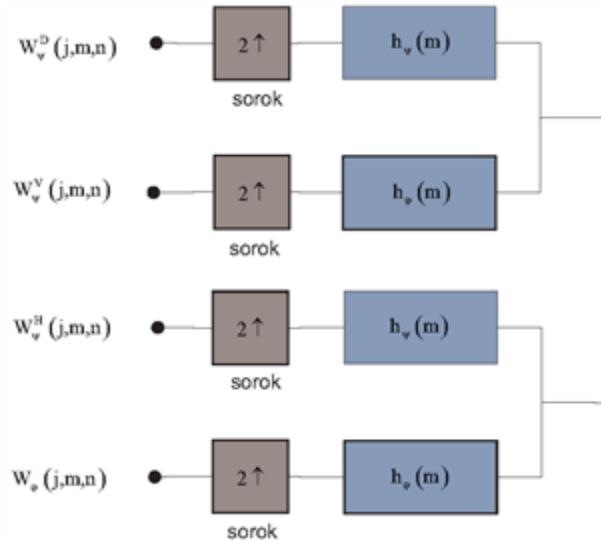
$$W_\psi^{H,V,D}(j, m, n) = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f(u, v) \psi_{j, m, n}^{H,V,D}(u, v)$$

Az $f(u, v)$ függvény skálázó és wavelet komponensekkal az alábbi módon fejezhető ki:

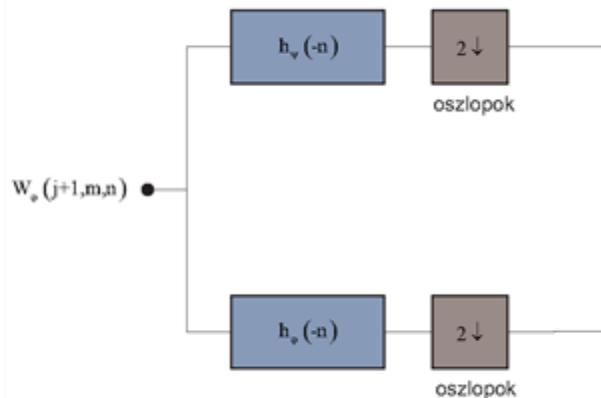
$$f(u, v) = \frac{1}{\sqrt{MN}} \sum_m \sum_n W_\phi(j_0, m, n) \phi_{j_0, m, n}(u, v) +$$

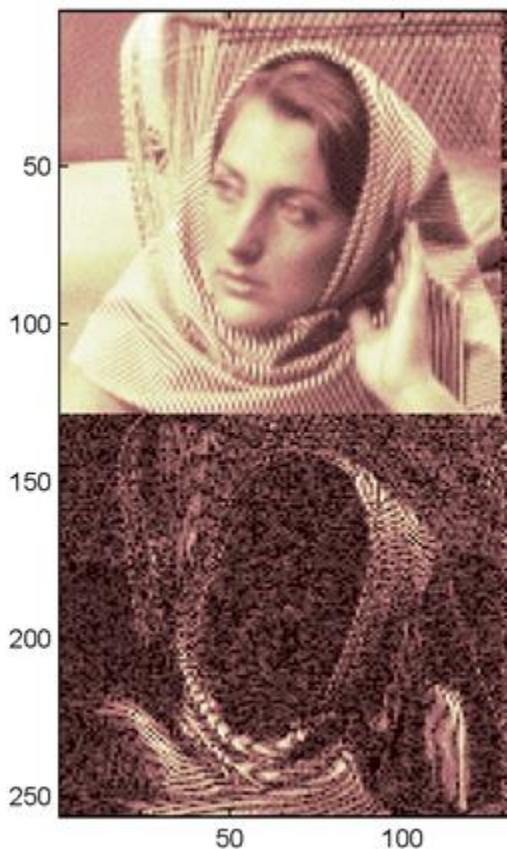
$$+ \frac{1}{\sqrt{MN}} \sum_{t=H,V,D} \sum_{j=j_0}^{\infty} \sum_m \sum_n W_\psi^t(j, m, n) \psi_{j, m, n}^t(u, v)$$

4.4. 2D Wavelet transzformáció - példa



4.5. 2D Inverz Wavelet transzformáció - példa





4.6. 2D Wavelet transzformáció - párhuzamos végrehajtás GPU-n

- A 2D FWT (Fast Wavelet Transform) párhuzamos végrehajtásának lépései
 1. A kép "page-locked" pufferbe való betöltése
 2. A puffer másolása a globális memóriába
 3. 1D-FWT végrehajtása a kép minden sorára
 - minden sorra párhuzamosan végrehajtható
 4. Az így kapott képmátrix transzponálása (párhuzamosan végrehajtható)
 5. 1D-FWT végrehajtása a kép minden oszlopára
 - minden oszlopra párhuzamosan végrehajtható
 6. Az így kapott képmátrix transzponálása (párhuzamosan végrehajtható)
 7. Az eredmény visszamásolása a globális memóriából a "page-locked" pufferbe.

4.7. Felhasznált és javasolt irodalom

[1] S. G. Mallat: *A theory for multiresolution signal decomposition: the wavelet representation* vol. 11, pp. 674-693, July 1989.

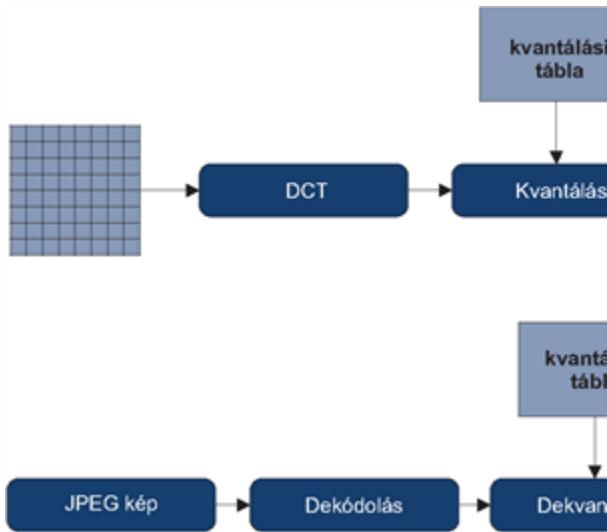
[2] Chun-Lin, Liu: *A Tutorial of the Wavelet Transform* p. 72, February 23, 2010.

- [3] R.X. Gao and R. Yan: *Wavelets: Theory and Applications for Manufacturing* DOI 10.1007/978-1-4419-1545-02, ISBN 978-1-4419-1545-0, Springer, XIV, p. 224, 2011.
- [4] C. Sidney Burrus: *A multiresolution formulation of Wavelet Systems* Connexions module: m45081, p. 34. Forrás - 2013.02.10.
- [5] Chun-Lin, Liu: *A Tutorial of the Wavelet Transform* February 23, p. 71, 2010. Forrás - 2013.02.10.
- [6] Robi Polikar: *The Wavelet Tutorial, Second Edition, Part I* Forrás - 2013.02.15.
- [7] Sheng, Y.: *Wavelet Transform The Transforms and Applications Handbook*: Second Edition. Ed. Alexander D. Poularikas Boca Raton: CRC Press LLC, 2000.
- [8] Franco, J. - Bernabe, G. - Fernandez, J. - Acacio, M.E.: *A Parallel Implementation of the 2D Wavelet Transform Using CUDA* 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp.111-118, 18-20 Feb. 2009.

16. fejezet - Képtömörítési eljárások párhuzamos környezetben



1. JPEG tömörítő eljárás



1.1. JPEG tömörítő eljárás - példa

- Tekintsük a digitális kép egy tetszőlegesen választott 8x8 pixel méretű blokkját
- Legyenek a blokkon belüli pixelek intenzitásai az alábbiak:

$$I_1 = \begin{pmatrix} 188 & 145 & 88 & 58 & 67 & 110 & 134 & 134 \\ 187 & 193 & 152 & 125 & 115 & 130 & 137 & 139 \\ 166 & 184 & 201 & 194 & 198 & 195 & 151 & 139 \\ 152 & 168 & 188 & 214 & 229 & 225 & 172 & 143 \\ 156 & 159 & 165 & 181 & 201 & 199 & 169 & 144 \\ 168 & 163 & 164 & 158 & 167 & 174 & 156 & 145 \\ 174 & 171 & 169 & 158 & 155 & 163 & 160 & 144 \\ 170 & 172 & 167 & 161 & 159 & 167 & 169 & 147 \end{pmatrix}$$

A tömörítés lépései:

- Az intenzitásértékek eltolása, azaz $I_2 = I_1 - 128$

$$I_2 = I_1 - 128 = \begin{pmatrix} 60 & 17 & -40 & -70 & -61 & -18 & 6 & 6 \\ 59 & 65 & 24 & -3 & -13 & 2 & 9 & 11 \\ 38 & 56 & 73 & 66 & 70 & 67 & 23 & 11 \\ 24 & 40 & 60 & 86 & 101 & 97 & 44 & 15 \\ 28 & 31 & 37 & 53 & 73 & 71 & 41 & 16 \\ 40 & 35 & 36 & 30 & 39 & 46 & 28 & 17 \\ 46 & 43 & 41 & 30 & 27 & 35 & 32 & 16 \\ 42 & 44 & 39 & 33 & 31 & 39 & 41 & 19 \end{pmatrix}$$

- Diszkrét koszinusz transzformáció (DCT) alkalmazása I_2 -re ($F_1 = DCT(I_2)$)

$$R_1 = \begin{pmatrix} 263 & 46.69 & -10.84 & 45.62 & -28 & -4.91 & 5.84 & -5.57 \\ -67.22 & 24.13 & 56.18 & 6.96 & -2.32 & -8.65 & 0.87 & -5.86 \\ -119.71 & 32.57 & 129.22 & -7.86 & -8.71 & 10.05 & -8.83 & 1.19 \\ -87.58 & -7.34 & 71.1 & 16.89 & 4.75 & 4.86 & -1.32 & 2.37 \\ -11.75 & -33.87 & -3.76 & 23.69 & 4.75 & 5.4 & -3.85 & -0.44 \\ 3.79 & -10.86 & -13.13 & 8.78 & 1.96 & 4.01 & 6.98 & 0.26 \\ -1.56 & -6.2 & -7.08 & 4.27 & 1.17 & -0.42 & 8.42 & 2.77 \\ -2.73 & -0.94 & -1.93 & 1.05 & 0.31 & 0.95 & 2.29 & 3.48 \end{pmatrix}$$

- Kvantálás

- A Q kvantálási tábla kisebb elemei annak bal felső, míg a nagyobb elemek annak jobb alsó sarka közelében helyezkednek el
- F_1 elemeit leosztva a Q kvantálási tábla megfelelő elemeivel a nagyobb frekvenciák kinullázhatók ($F_2(i,j) = F_1(i,j)/Q(i,j)$). A nagyobb táblaelemek nagyobb veszteséget fognak maguk után az osztás következtében. A kvantáló táblát nem szükséges kitalálni, a JPEG szabvány tartalmaz ajánlásokat, bár az adaptív eljárások során a kép alapján dinamikusan alkotják meg őket (pl. high-profile fényképezők).

$$Q = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

- Zig-Zag alapú szekvencia létrehozása F_2 elemeiből
- Figyeljük meg, hogy az így létrehozott szekvencia zérus elemei annak végén foglalnak helyet

$$F_2 = \begin{pmatrix} 16 & 4 & -1 & 3 & -1 & 0 & 0 & 0 \\ -6 & 2 & 4 & 0 & 0 & 0 & 0 & 0 \\ -9 & 3 & 8 & 0 & 0 & 0 & 0 & 0 \\ -6 & 0 & 3 & 1 & 0 & 0 & 0 & 0 \\ -1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$S_1 = \{16, 4, -6, -9, 2, -1, 3, 4, 3, -6, -1, 0, 8, 0, -1, 0, 0, 0, 3, -2, 0, 0, 0, 0, 1, 0, \dots, 0\}$$

ZigZag szekvencia								
16	→	4	→	-1	→	3	→	-1
-6	↓	2	→	4	→	0	→	0
-9	↓	3	→	8	→	0	→	0
-6	↓	0	→	3	→	1	→	0
-1	↓	-2	→	0	→	0	→	0
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	

- S_1 értékeinek reprezentálása (lásd táblázat)

- A kapott szekvencia (S_2) (i,j), k típusú elemeiben i az elem előtti zérusok számát, j az elem kódolásához szükséges bitek számát, k pedig az elem kódját jelölik.

Ertek	Kod	Hossz
-9	100	4
-6	1	3
-2	1	2
-1	0	1
1	1	1
2	10	2
3	11	2
4	100	3
8	1000	4
16	10000	5

$S_2 = \{(0, 5), 10000; (0, 3), 100; (0, 3), 001; (0, 4), 0110; (0, 2), 10; (0, 1), 0; (0, 2), 11; (0, 3), 100; (0, 2), 11; (0, 3), 001; (0, 1), 0; (1, 4), 1000; (1, 1), 0; (3, 2), 11; (0, 2), 01; (4, 1), 1; (0, 0)\}$

- Huffman kódolás (lásd táblázat)
- Tömörített 8x8-as blokk (S_2)

11001100000110001001110000110001010000
 $S_3 = 01101100001101001100011011100011010011$
 100110001111011101

1.2. Felhasznált és javasolt irodalom

[1] John W. O'Brien: *The JPEG Image Compression Algorithm* APPM-3310 FINAL PROJECT, DECEMBER 2, 2005.

2. HOSVD-alapú tömörítés

Legyen $f(\mathbf{x})$, $\mathbf{x} = (x_1, x_2, \dots, x_N)^T$, $x_n \in [a_n, b_n]$, $n = 1 \dots N$ egy n -változós sima függvény. Ekkor $f(\mathbf{x})$ az alábbi módon közelíthető egyváltozós ortonormált rendszert alkotó függvényekkel:

$$f(\mathbf{x}) = \sum_{k_1=1}^{J_1} \dots \sum_{k_N=1}^{J_N} \alpha_{k_1, \dots, k_N} p_{1, k_1}(x_1) \cdot \dots \cdot p_{N, k_N}(x_N)$$

ahol a $p_{n,k}(x_n)$ függvények megválaszthatók egyrészt klasszikus módon ortonormált polinomok vagy trigonometrikus függvények formájában, másrészt olyan ortonormált rendszert alkotó specifikus függvények segítségével, melyek jellege az approximálandó n -változós függvényre nézve specifikus.

Ebből az is következik, hogy sokkal kevesebb ily módon megválasztott függvényre van szükségünk ugyanannak az n -változós függvénynek egy előre meghatározott pontossággal történő approximációjához, mint amennyi trigonometrikus függvényre vagy ortonormált polinomra ugyanazon közelítési pontosság esetében.

A magasabb rendű szinguláris értékdekompozíció segítségével (HOSVD) ezek az egyváltozós sima függvények a súlyokkal együtt numerikusan meghatározhatók. [2][3][5].

Legyen $A \in \mathbb{R}^{I_1 \times \dots \times I_N}$ egy N dimenziós tenzor és \mathbf{U} egy $K_n \times I_n$ mátrix. $A \times_n \mathbf{U}$ az ún. n -módú tenzor mátrix szorzat az alábbi tulajdonságokkal:

- Eredménye egy $I_1 \times \dots \times I_{n-1} \times K_n \times I_{n+1} \times \dots \times I_N$ tenzor

$$\begin{aligned} & (A \times_n \mathbf{U})_{i_1, \dots, i_{n-1}, k_n, i_{n+1}, \dots, i_N} = \sum_{1 \leq k_n \leq I_n} a_{i_1, \dots, i_n, \dots, i_N} U_{k_n, i_n} \\ & A \times_{n-1}^N \mathbf{U}_n \text{ az ún. többszörös szorzat, amely a következő alakban is felírható [6]: } A \times_{n-1}^N \mathbf{U}_n \end{aligned}$$

A HOSVD-ből kiindulva belátható, hogy $f(\mathbf{x})$ felírható az alábbi formában [6]: $f(\mathbf{x}) = D \times_{n=1}^N \mathbf{w}_n(x_n)$, $\mathbf{w}_n(x_n) = (w_{n,1}(x_n), \dots, w_{n,r_n}(x_n))^T$, $1 \leq n \leq N$ ahol $D \in \mathbb{R}^{I_1 \times \dots \times I_N}$ egy speciális ún. magtenzor az alábbi tulajdonságokkal:

- $r_n = \text{rank}_n(A)$ az tenzor n -módú rangja
- D ortogonális, azaz tetszőleges n , α és β , $\alpha \neq \beta$ esetén $D_{i_n=\alpha}$ és $D_{i_n=\beta}$ altenzorok skaláris szorzata nulla, azaz $\langle D_{i_n=\alpha}, D_{i_n=\beta} \rangle = 0$
- $\|D_{i_n=1}\| \geq \|D_{i_n=2}\| \geq \dots \geq \|D_{i_n=r_n}\| > 0$,
- $\|D_{i_n=\alpha}\| = \langle D_{i_n=\alpha}, D_{i_n=\alpha} \rangle$
- $\mathbf{w}_n(x_n) = (w_{n,1}(x_n), \dots, w_{n,r_n}(x_n))^T$, $1 \leq n \leq N$ elemei ortonormáltak

Tételezzük fel, hogy $f(\mathbf{x})$ csak mérési pontjaiban ismert. Ekkor ezekben a pontokban ismert függvényértékeket egy N dimenziós tenzorral is megadhatjuk. Jelöljük az így létrehozott tenzort B -vel. Amennyiben a B tenzoron szinguláris értékfelbontást hajtunk végre, a kapott magtenzor és a dimenziónként adódó ortonormált mátrixok segítségével $\$B\$$ tenzor-szorzat alakban az alábbi formában is kifejezhető [1][2]:

$$B = D \times_{n=1}^N \mathbf{W}_n$$

ahol a W_n mátrixok oszlopai az említett egy változós specifikus függvények diszkretizált változatainak felelnek meg, D pedig a már szintén említett magtenzort jelöli.

RGB digitális képek esetében három változós esetről van szó, így a fentiek alapján az $f(\mathbf{x})$, $\mathbf{x} = (x_1, x_2, x_3)^T$ függvény felírható az alábbi formában:

$$f(\mathbf{x}) = \sum_{k_1=1}^{I_1} \sum_{k_2=1}^{I_2} \sum_{k_3=1}^{I_3} a_{k_1, k_2, k_3} w_{1, k_1}(x_1) \cdot w_{2, k_2}(x_2) \cdot w_{3, k_3}(x_3)$$

Hasonlóképpen az n -változós esethez, a képet tenzor formában is kifejezhetjük az

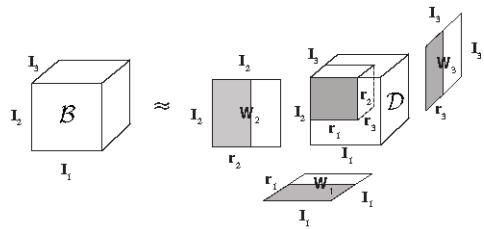
alábbi módon (lásd 3.1 ábra):

$$B = D \times_{n=1}^3 W^{(n)},$$

ahol

$$B \in \mathbb{R}^{I_1 \times I_2 \times I_3}$$

16.1. ábra - A HOSVD illusztrációja három változós esetben. D az ún. magtenzor a mátrixok pedig a dimenziónkénti ortonormált mátrixokat jelölik, melyek oszlopai az approximációt alkalmazott egy változós függvények diszkretizált változatai.



2.1. Felhasznált és javasolt irodalom

[1] Szeidl L., Várlaki P. *HOSVD Based Canonical Form for Polytopic Models of Dynamic Systems* Journal of Advanced Computational Intelligence and Intelligent Informatics, 13:(1) pp. 52-60, 2009.

[2] Rövid, A. Szeidl, L., Várlaki, P. *Polylinear form of functions on HOSVD basis in relation to Fourier image processing* in Proceedings of the 14th WSEAS International Conference on Applied mathematics, ISBN: 978-960-474-138-0, pp. 239-244, 2009.

17. fejezet - Letölthető melléklet

A tananyagot az Óbudai Egyetem Mérnök Informatikus MSc hallgatói egy szemeszteren keresztül hallgatták, aminek eredményeként az ismertetett módszerek egy részét implementálták és a megvalósítás részleteiről "Képfeldolgozási algoritmusok párhuzamosítása" címmel dokumentációt is készítettek.

A fejlesztésben részt vevő hallgatók:

- Kalázdi Bálint
- Könyvesi Gábor
- Kurucz András
- Szabó Balázs
- Szántó Balázs
- Urbán András

Az elkészült anyagok letölthetők a <http://users/nik.uni-obuda.hu/ParhuzamosKepfeldolgozas> címről.