

Extragerea contururilor regiunilor

Prof. univ. dr. ing. Florica Moldoveanu

Definitii(1)

- Doi pixeli sunt **vecini direcți (vecini-d)** dacă celulele lor au o latură comună.
- Doi pixeli sunt **vecini indirecti (vecini-i)** dacă celulele lor se ating numai într-un colț.
- Termenul **vecin-N**, $0 \leq N \leq 7$, desemnează un pixel aflat în poziția N față de un pixel P:

3	2	1
4	P	0
5	6	7

vecini-d sunt vecini-N, cu N=par

vecini-i sunt vecini-N, cu N=impar

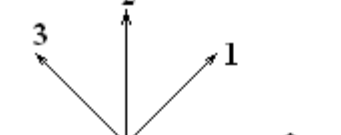
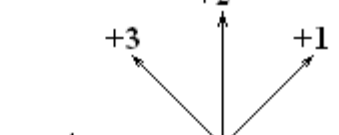
- O **cale** este o secvență de pixeli vecini A_{k-1}, A_k, A_{k+1} , astfel încât:
 pentru $k > 1$, A_{k-1} este un vecin al lui A_k
 pentru $k < n$ (numarul de pixeli din cale), A_{k+1} este un vecin al lui A_k
- O **cale-d** este o cale în care toți pixelii sunt vecini-d.
- O **cale simplă** este o cale în care toți pixelii sunt distincti și nici un pixel nu are mai mult de 2 vecini-d în cale.
- O **cale închisă** este o cale în care primul pixel coincide cu ultimul.

Definitii(2)

- Un **set de pixeli**, S , este **conectat** (sau **conectat-i**), dacă pentru fiecare pereche de pixeli A și B din S , există o cale în care A și B sunt primul respectiv ultimul element, iar toți ceilalți pixeli ai căii aparțin lui S .
- Un **set de pixeli este conectat-d**, dacă fiecare pereche de pixeli din set poate fi conectată printr-o cale-d inclusă în set.
- O **regiune** este un set de pixeli conectat în care toți pixelii au aceleași caracteristici: intensitate/culoare/adâncime.
- Un **pixel de contur** al unei regiuni este un pixel care are cel puțin un vecin-d în afara regiunii.

Reprezentarea contururilor

- Prin lista adreselor absolute ale pixelilor care-l compun.
- Adresa absoluta a pixelui de start urmata de deplasamente relative de la un pixel la urmatorul.
 - Deplasamentele relative pot fi reprezentate prin:
 - Coduri de înlănțuire absolute (cod Freeman)
 - Coduri de înlănțuire diferențiale

<p>Cod de înălțuire (Freeman)</p>  <p>Valorile codului Freeman sunt valori absolute.</p>	<p>Cod de înălțuire diferențial</p>  <p>Valorile codului diferențial sunt valori relative. Ele exprima schimbarea de direcție in pasul curent.</p>
--	--

Exemplu:

```

graph LR
    1((1)) --> 2((2))
    1((1)) --> 3((3))
    2((2)) --> 4((4))
    3((3)) --> 4((4))
    4((4)) --> 5((5))
  
```

Codul Freeman

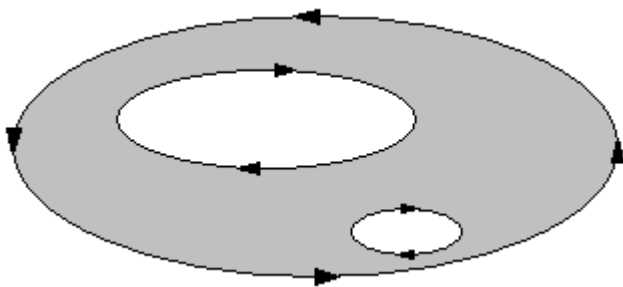
0, 1, 0, 6, 0, 5

Codul de înlănțuire diferențial

0, +1, -1, -2, +2, -3

Extragerea conturului unei regiuni(1)

- Conturul unei regiuni este traversat pe o cale închisă, fiind posibil întotdeauna să se aleagă o astfel de cale.
- **Pixelul de start** se poate alege scanand imaginea de sus in jos și de la stanga la dreapta:
este un pixel care *aparține regiunii considerate* (are intensitatea/eticheta pixelilor regiunii) și
este *pixel de contur* (are cel puțin un vecin-d in afara regiunii).
- Traversarea constă într-o deplasare pixel cu pixel, în fiecare pas fiind ales ca pixel de contur următor pixelul din regiune care este cel mai in dreapta față de pixelul curent.
 - In acest fel, interiorul regiunii se afla intotdeauna in stanga pixelului selectat (conturului).



Cu aceasta regula de traversare, contururile exterioare sunt parcurse in sens trigonometric, iar cele interioare in sensul acelor de ceas.

Extragerea conturului unei regiuni (2)

- Punctul de start se alege astfel încât vecinul său 4 să nu aparțină regiunii.

➤ Punctul de start ales poate fi un pixel izolat!

3	2	1
4	S	0
5	6	7

Notății:

S - pixelul de start

C - pixelul curent

D - direcția de căutare a următorului pixel de contur, unde $0 \leq D \leq 7$:

urm – adresa pixelului din vecinatatea celui curent, care ar putea fi următorul pixel de contur

contur - vector în care se memorează direcția de deplasare în fiecare pas; în final se va obține reprezentarea prin cod de înlănțuire (Freeman) a conturului traversat.

urm in R = true, daca pixelul urm are valoarea pixelilor regiunii (apartine regiunii).

Operatiile \oplus și \ominus sunt operatii modulo 8. Astfel:

$$0 \ominus 1 = 7$$

$$6 \oplus 2 = 0$$

Extragerea conturului unei regiuni (3)

```
unsigned char * contur;
```

```
int n=0, D; // D: directia de cautare curenta – in prima iteratie D=6
```

```
// In fiecare iterație se caută următorul punct de contur
```

```
// printre vecinii  $D \ominus 1$ ,  $D$ ,  $D \oplus 1$  ai pixelului curent.
```

```
// Functia PunctContur memoreaza in vectorul contur
```

```
// directia de deplasare în urmatorul punct de contur si intoarce:
```

```
// - fie urmatorul punct de contur
```

```
// - fie punctul de contur curent, C, daca nu s-a gasit un punct in regiune pe directiile  $D \ominus 1$ ,  $D$ ,  $D \oplus 1$ 
```

```
Punct PunctContur (Punct C) // Punct este o adresa de pixel
```

```
{ Punct urm;
```

```
urm = vecin(C,  $D \ominus 1$ ); // adresa vecinului lui C de pe directia  $D \ominus 1$ 
```

```
if (urm in R) // pixelul urm are valoarea pixelilor regiunii
```

```
{ contur[n++] = (unsigned char) ( $D \ominus 1$ );  $D = D \ominus 2$ ; return urm; }
```

```
urm = vecin(C, D);
```

```
if (urm in R)
```

```
{ contur[n++] = (unsigned char)D; return urm; }
```

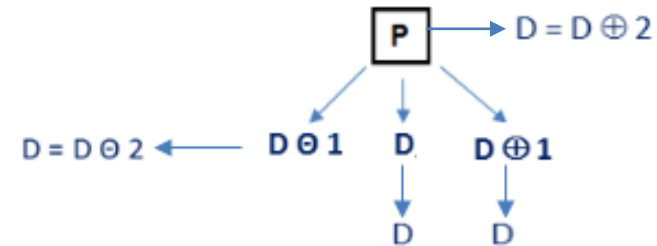
```
urm = vecin(C,  $D \oplus 1$ );
```

```
if (urm in R)
```

```
{ contur[n++] = (unsigned char)( $D \oplus 1$ ); return urm;}
```

```
 $D = D \oplus 2$ ; return C; // nu s-a gasit un pixel in regiune pe directiile  $D \ominus 1$ ,  $D$ ,  $D \oplus 1$ 
```

```
}
```



Extragerea conturului unei regiuni (4)

- Inițial $D=6$; vecinul 4 al pixelului de start nu apartine regiunii
- Daca dupa primele 3 iteratii nu s-a gasit un alt punct de contur, inseamna ca punctul de start este punct izolat.
- Algoritmul se termina atunci cand pixelul curent este cel de start.

```
int ExtrageContur (Punct S)
```

```
{ Punct C = S; // punctul de start contur
```

```
  D = 6; // directia de cautare initiala
```

```
  // cauta al doilea punct de contur
```

```
  for (int k = 0; k<3; k++)
```

```
  { C = PunctContur(C); //intoarce urmatorul punct de contur sau punctul curent
```

```
    if (C !=S) break; }
```

```
  if( k==3) return 0; // S este punct izolat
```

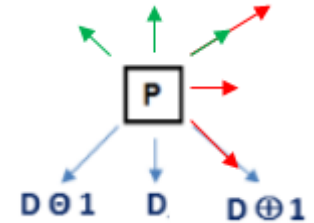
```
  // traverseaza conturul
```

```
  while( C != S)
```

```
    C = PunctContur(C);
```

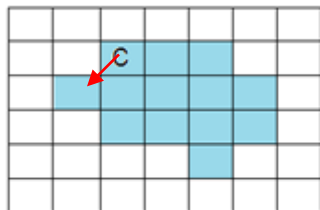
```
  return 1;
```

```
}
```



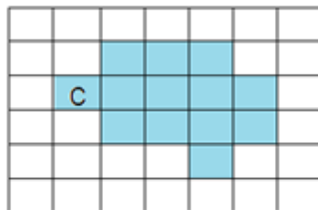
Iteratia:

1



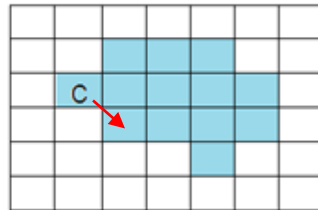
$D=6$; $D \leftarrow D-2$;
 $\text{contur}[n++] \leftarrow 5$

2



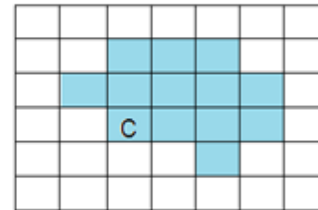
$D=4$; $D \leftarrow D+2$

3



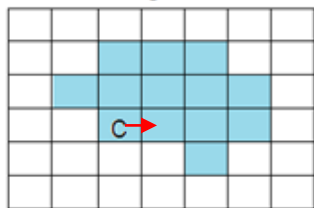
$D=6$; $\text{contur}[n++] \leftarrow 7$

4



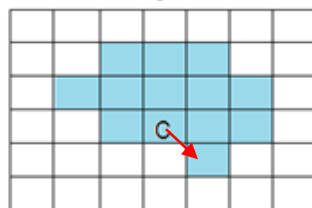
$D=6$; $D \leftarrow D+2$

5



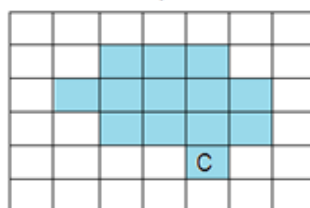
$D=0$; $\text{contur}[n++] \leftarrow 0$

6



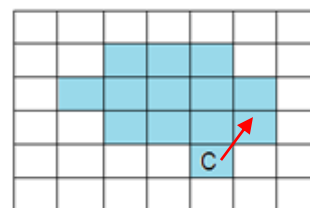
$D=0$; $D \leftarrow D-2$
 $\text{contur}[n++] \leftarrow 7$

7



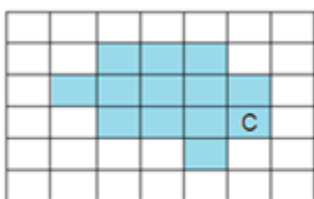
$D=6$; $D \leftarrow D+2$

8



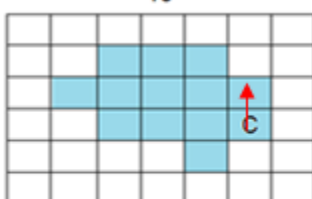
$D=0$; $\text{contur}[n++] \leftarrow 1$

9



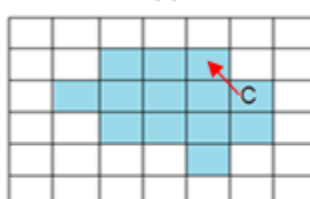
$D=0$; $D \leftarrow D+2$

10



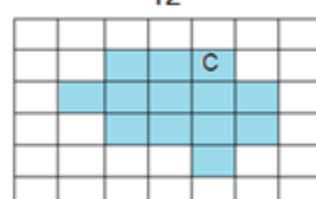
$D=2$;
 $\text{contur}[n++] \leftarrow 2$

11



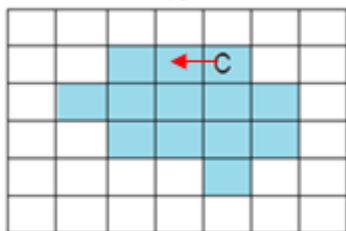
$D=2$;
 $\text{contur}[n++] \leftarrow 3$

12



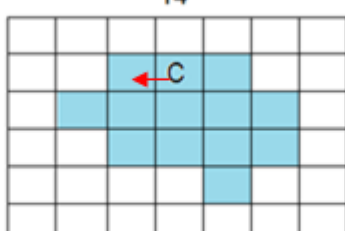
$D=2$; $D \leftarrow D+2$

13



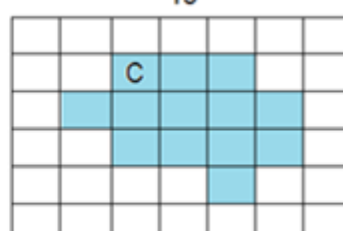
$D=4$; $\text{contur}[n++] \leftarrow 4$

14



$D=4$; $\text{contur}[n++] \leftarrow 4$

15

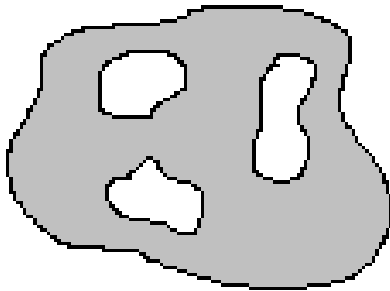


$C=S$

Vectorul contur:
5, 7, 0, 7, 1, 2, 3, 4, 4

Extragerea tuturor contururilor unei regiuni(1)

- Pentru simplificare, consideram ca imaginea segmentata este binara:
 - pixelii regiunilor au valoarea =1;
 - pixelii din afara regiunilor au valoarea =0
- Aceasta nu este o restrictie, algoritmul putand fi generalizat.



O regiune poate avea mai multe contururi interioare care marginesc „gauri” ale regiunii.

Se foloseste functia **ExtrageContur**, modificata astfel incat:

- punctul de start al fiecarui contur să fie memorat într-un vector, **PStart**
- codurile de înlănțuire ale contururilor sa fie delimitate in vectorul **contur** prin valoarea 8
- la traversarea unui contur, valoarea fiecarui pixel de contur se incrementeaza, devenind 2 sau > 2 (in cazul in care pixelul apartine mai multor contururi traversate). In acest fel se pot recunoaste conturile deja traversate.

Extragerea tuturor contururilor unei regiuni(2)

```
Punct * PStart; //vectorul punctelor de start
int ncont=0; // numarul curent de contururi
unsigned char ** Img; // imaginea segmentata
// Functia ExtrageContur modificata pentru extragerea tuturor contururilor unei regiuni
int ExtrageContur (Punct S)
{ Punct C = S;
  D = 6; // directia de cautare initiala
  // cauta al doilea punct de contur
  .....
  if( k==3) return 0; // S este punct izolat
  PStart[ncont++] = S; Img[S.y][S.x]++;
  // traverseaza conturul
  while( C != S)
  { Img[C.y][C.x]++ ;
    C = PunctContur(C);
  }
  contur[n++] = 8;
  return 1;
}
```

Extragerea tuturor contururilor unei regiuni(3)

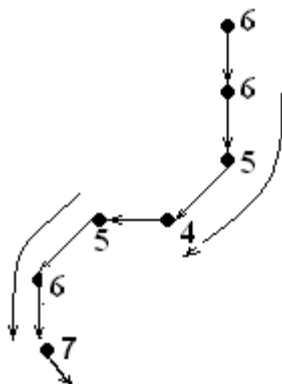
- Se extrage mai intai conturul exterior.
- Contururile interioare se gasesc **parcurgand regiunea de la stanga spre dreapta**, plecand din pixelii conturului exterior sau ai unui contur interior.

Pasii algoritmului:

1. Se extrage conturul exterior în vectorul **contur** si se incrementeaza pixelii săi
1. Se parcurge vectorul **contur**, cautând un punct din care va începe parcurgerea regiunii pentru găsirea contururilor interioare.

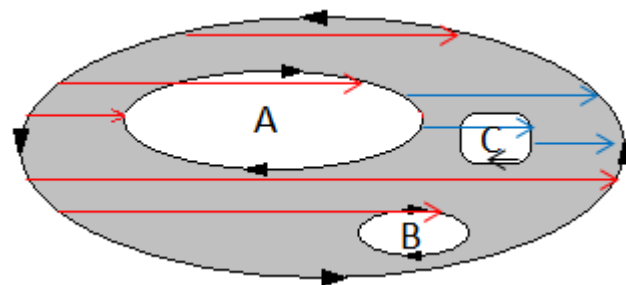
→ *Parcurgerea regiunii are loc întotdeauna spre dreapta, de aceea punctul de start pentru parcurgere se alege de pe un **arc coborâtor**.*

Arc coborâtor pe
conturul exterior



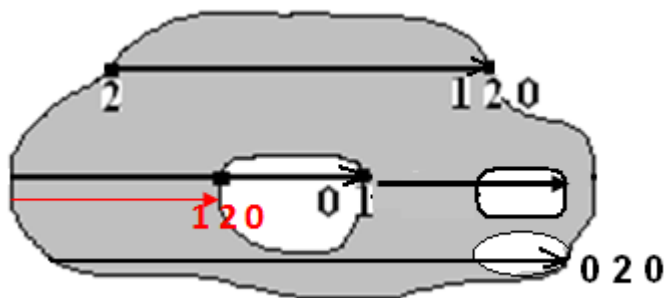
→ *Condiția ca un punct sa fie punct de start parcurgere regiune este:*

- codul punctului să fie cuprins între 5-7 si
- codul punctului *anterior* să fie cuprins între 4-7



Extragerea tuturor contururilor unei regiuni(4)

3. Se parcurge regiunea din punctul de start determinat, spre dreapta, cautand fie un punct de start pentru un contur interior, fie marginea regiunii (ieșirea din regiune).



Parcurgerea regiunii pe linia curenta se termina la intalnirea secvenței:

1,2,0 - contur deja traversat

0,1 sau 0,2,0 (contur interior netraversat)

4. Daca s-a intalnit secventa 01 sau 020, se apeleaza functia **ExtrageContur**, care adaugă conturul in vectorul **contur** si incrementeaza valoarea pixelilor conturului:

➤ pe liniile imagine urmatoare el va fi identificat ca un contur deja traversat prin intalnirea secventei 1,2,0.

5. Dupa parcurgerea conturului exterior, se continua cu parcurgerea contururilor interioare memorate in vectorul **contur**, cautand puncte de start pentru parcurgerea regiunii.

➤ Contururile interioare netraversate la parcurgerea conturului exterior vor fi traversate la parcurgerea contururilor interioare memorate in vectorul **contur**.

Extragerea tuturor contururilor unei regiuni(5)

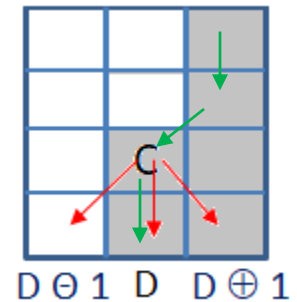
Traversarea contururilor interioare

Regula de traversare: in fiecare pas este ales ca pixel urmator pixelul din regiune care este cel mai in dreapta față de pixelul curent.

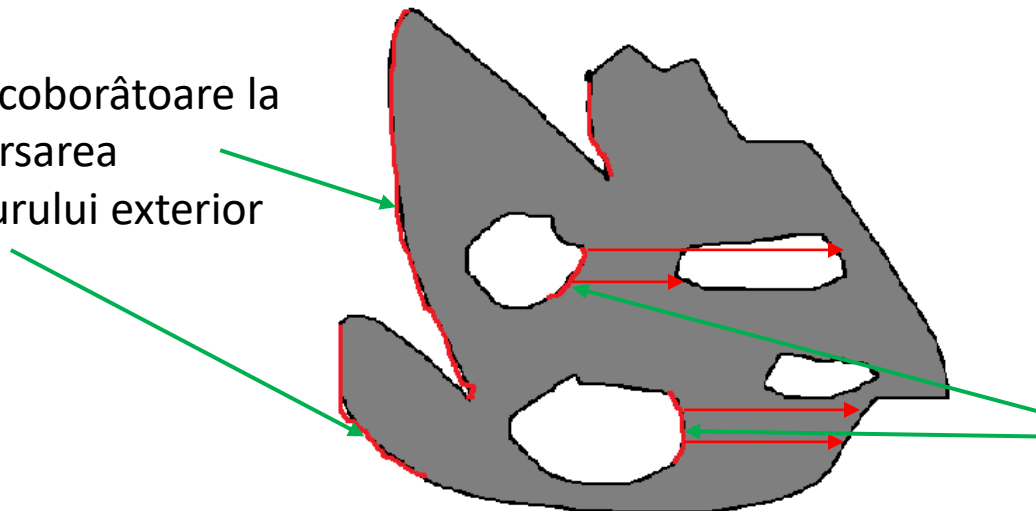
➤ Interiorul regiunii se afla intotdeauna in stanga pixelului selectat (conturului).

➤ Daca C este pixelul curent, directiile in care se cauta urmatorul

pixel de contur sunt:



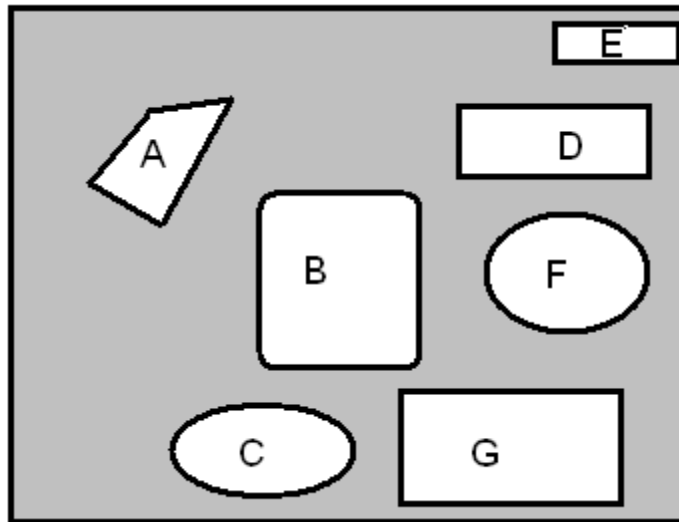
Arce coborâtoare la traversarea conturului exterior



Arce coborâtoare la traversarea contururilor interioare

Extragerea tuturor contururilor unei regiuni(5)

Exemplu:



Continutul vectorului contur dupa parcurgerea tuturor contururilor regiunii din figura.

contur exterior, 8, conturul E, 8, conturul A, 8, conturul B, 8,

conturul G, 8, conturul C, 8 , conturul D, 8, conturul F, 8

Extragerea tuturor contururilor unei regiuni(6)

```
Punct * PStart; int ncont=0;
unsigned char ** Img;
unsigned char * contur; int D; int n=0; // numarul de elemente din vectorul contur
int ExtrageToateContururile(Punct S)
{ // S: punctul de start pentru parcurgerea conturului exterior
  int A, B, C, I, c, c0, int ks =0; Punct P, PS;
  if( ! ExtrageContur(S)) return 0; // punctul de start este un punct izolat
  I=0; // index in vectorul contur
  c0 = 6; // directia punctului de contur anterior (initial D = 6)
  P = S; // punctul de contur curent
  while(I < n-1) // ! sfarsit lista coduri de contururi
  { c = contur[I++]; //codul de contur curent
    if( c==8) // start contur nou
    { c0 = 6; P = PStart[ks++]; c = contur[I++];}
    else
      P = PunctUrm(P, c); // calculeaza adresa urmatorului punct de contur
    if( 4 <= c0 <= 7 && 5 <= c <= 7) // P este punct de start parcurgere spre dreapta
```


Extragerea tuturor contururilor unei regiuni(7)

//parcurge regiunea spre dreapta

```
{ x = P.x; y = P.y; gata = 0;
  while( ! gata)
  { A = (int)Img[y][x]; B = (int)Img[y][x+1]; C = (int)Img[y][x+2];
    if( A==0 && B==1 || A==0 && B==2 && C==0) // B este punct de start contur interior
      { PS.y = y; PS.x = x+1;
        ExtrageContur(PS);
        gata =1; //sfarsit traversare regiune pe linia curenta
      }
    else
      if(A==1 && B==2 && C==0) // contur deja traversat
        gata = 1; //sfarsit traversare regiune pe linia curenta
      x++;
  } // while (! gata)
}

c0 = c; // se avanseaza in vectorul contur; c devine codul punctului anterior
} //while (l<n-1) – sfarsit lista contururi
```

Segmentarea imaginilor prin clusterizare

Prof. univ. dr. ing. Florica Moldoveanu

Segmentarea prin algoritmul K-Means (1)

Obiectivul algoritmului de clusterizare K-Means este de a minimiza suma distanțelor dintre fiecare element dintr-un set de date asignat unui cluster și centrul clusterului.

Element din clusterul j

Centrul clusterului j

$$J = \sum_{j=1}^k \sum_{i=1}^{n_k} \underbrace{\|x_i^{(j)} - c_j\|}_{\text{Distanța}}^2$$

K – numărul de cluster

n_k – numărul de elemente din clusterul k

- **Pixelii cu caracteristici similare dintr-o imagine pot fi grupați în “cluster”.**
- Setul de date clasificat este setul de intensități/culori din imagine.
- Centrele de cluster – culori reprezentative pentru culorile din imagine.
- Fiecare pixel din imagine este asignat centrului de cluster care este cel mai apropiat de culoarea sa.

Segmentarea prin algoritmul K-Means (2)

Pașii algoritmului K-Means pentru segmentarea unei imagini:

1. Se alege numărul de clustere, K.
2. Se aleg centrele celor K clustere. Centrele sunt culori/nivele de intensitate.
3. **repetă**
 - 3.1. Se asignează fiecare pixel centrului de cluster care este cel mai apropiat de culoarea sa.
 - 3.2. Se actualizează centrul fiecărui cluster calculând media culorilor pixelilor din cluster.**până când**
 - nici un pixel nu-si schimbă centrul de cluster **sau**
 - s-a executat numărul maxim de iteratii
4. Se atribuie tuturor pixelilor dintr-un cluster intensitatea/culoarea centrului clusterului.

- Regiuni cu aceeași culoare rezultate prin segmentarea k-means pot să nu fie adiacente spațial.
- Rezultatul segmentării este influențat de K și alegerea centrelor de cluster inițiale.
- Este consumator de timp → rar folosit în segmentarea imaginilor.

Complexitatea: $O(K*N*I)$ – N nr de pixeli din imagine, I – nr de iteratii în pasul 3

Reprezentarea vizuala a clusterizarii K-Means

Centrele initiale: cele 4 puncte verzi din stanga.



Sursa: <http://shabal.in/visuals/kmeans/5.html>

Exemple

Imaginea originala



<https://www.kdnuggets.com/2019/08/introduction-image-segmentation-k-means-clustering.html>



Imaginea segmentata
cu $K = 3$



Imaginea segmentata
cu $K = 5$



Imaginea segmentata
cu $K = 7$

K-means pentru superpixeli (1)

- ❖ Un “superpixel” este un grup compact de pixeli cu caracteristici similare.
- ❖ Divizarea unei imagini în superpixeli de o anumită dimensiune poate fi folosită ca o **etapă de preprocesare** în algoritmi de segmentare.

SLIC (**Simple Linear Iterative Clustering**) este un algoritm de generare superpixeli care folosește o adaptare a algoritmului k-means pentru segmentarea imaginilor.

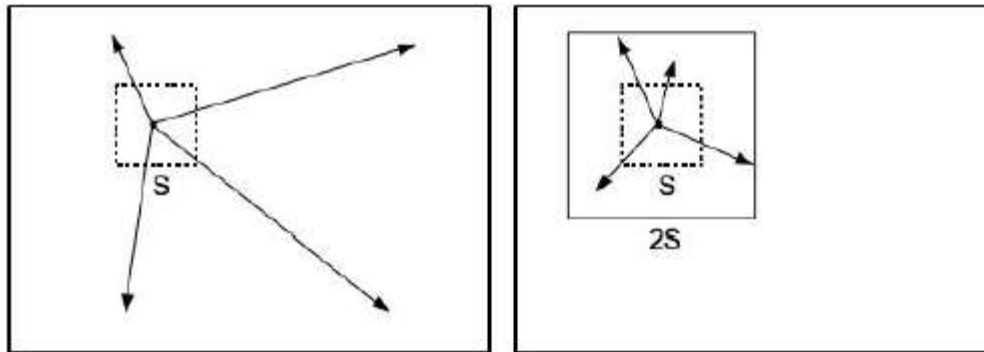


Imagini segmentate în superpixeli de dimensiuni 8x8, 16x16, 32x32 pixeli aproximativ.

<https://core.ac.uk/download/pdf/147983593.pdf>

K-means pentru superpixeli (2)

- Pixelii sunt grupati in clustere pe baza a doua criterii de similaritate:
 - Intensitate sau culoare
 - Poziție
- Centrele clusterelor sunt alese pe o grilă regulată cu latura $S = \sqrt{N/K}$, unde N este numarul de pixeli ai imaginii iar K numarul de superpixeli produsi de algoritm.
- Pixelii asignați unui cluster sunt căutați într-o regiune limitată, de $2*S$ in jurul fiecarui centru de cluster.



K-means standard:

Intr-o iteratie, pentru fiecare centru de cluster se cauta in intreaga imagine

Complexitatea $O(K*N)$

SLIC: Intr-o iteratie, pentru fiecare centru de cluster se cauta intr-o zona limitata

- Complexitatea algoritmului k-means pt gruparea pixelilor in superpixeli in SLIC :

Aria de cautare pt un cluster (superpixel) = $2S*2S = 4*N/K$ ($S = \sqrt{N/K}$)

Complexitatea intr-o iteratie = $K * (4*N/K) = 4N \Rightarrow O(N)$

In SLIC numarul de iteratii este limitat (empiric) la 10.

Aproximarea poligonală a frontierelor

Prof. univ. dr. ing. Florica Moldoveanu

Curs Sisteme de Prelucrare Grafică – UPB, Automatică și Calculatoare
2021-2022

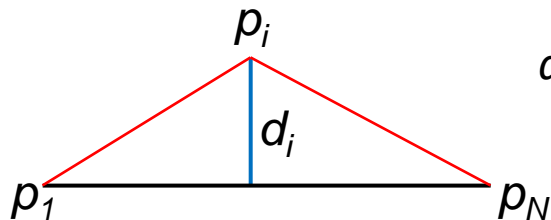
Aproximarea poligonală a unei frontiere (1)

- ❖ Algoritmii de extragere a frontierelor/contururilor regiunilor produc frontiere reprezentate printr-o secvență de adrese de pixeli (absolute, relative).
 - Reprezentarea este afectată de zgomot.
 - Poate conține caracteristici nerelevante pentru forma frontierei.

- ❖ Aproximarea poligonală:
 - Simplifică forma frontierei, fără efecte de netezire
 - Conduce la eliminarea zgomotului
 - Elimină caracteristicile nerelevante, păstrându-le pe cele relevante pentru forma frontierei
 - Conține pixeli ai frontierei – nu introduce pixeli noi
 - Poate fi folosită ca metodă de preprocesare înainte de extragerea caracteristicilor unei forme

Aproximarea poligonală a unei frontiere (2)

Fie p_1, p_2, \dots, p_N pixelii unei frontiere, care poate fi o curbă deschisă oarecare sau o parte dintr-o curbă închisă (conturul unei regiuni).



d_i – distanța de la pixelul p_i la segmentul de dreaptă (p_1-p_N)

Definiție: d_i este eroarea de aproximare a frontierei prin segmentul (p_1-p_N) în pixelul p_i

Definiție: Eroarea maximă de aproximare a frontierei prin segmentul (p_1-p_N) este

$$E_{\max} = \max_{2 \leq i \leq N-1} (d_i)$$

Metode de poligonalizare:

1. Prin **divizarea recursivă a frontierei** până la nivel de segmente de frontiera care se pot aproxima prin segmente de dreaptă; pixelii în care se efectuează divizarea devin vârfuri ale liniei poligonale.
2. Prin parcurgerea frontierei și extragerea pixelilor care devin vârfuri ale liniei poligonale.

Poligonalizarea prin divizarea recursiva a frontierei(1)

- Algoritmul se aplica unei frontiere deschise.
- Dacă frontiera este un contur închis, punctele p_1 și p_N se aleg astfel încât să fie situate la distanța maximă pe contur ($p_1 - p_N$ reprezintă axa majora a formei marginite de contur).



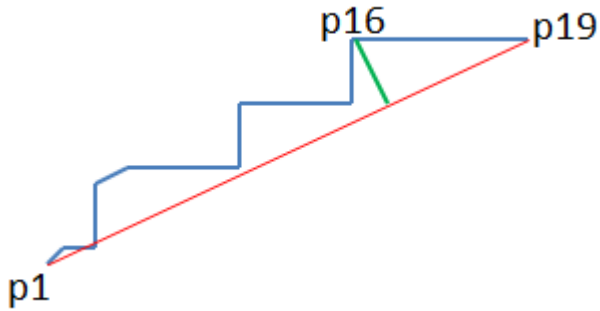
- Se efectuează poligonalizarea separat, pentru fiecare parte a conturului.

Algoritmul este recursiv

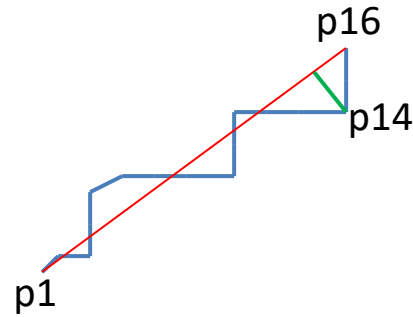
Fie p_1 și p_N punctele extreme ale frontierei la un apel al funcției de poligonalizare.

- Dacă frontiera poate fi aproximată prin segmentul $p_1 - p_N$ ($E_{max} < prag$) atunci se iese din recursivitate
- altfel
 - se determină pixelul $p(x,y)$, aflat la distanță maximă de segmentul ($p_1 - p_N$)
 - se apelează recursiv funcția de poligonalizare pentru segmentul de frontieră ($p_1 - p$)
 - se adaugă p la lista de varfuri a liniei poligonale
 - se apelează recursiv funcția de poligonalizare pentru segmentul de frontieră ($p - p_N$)

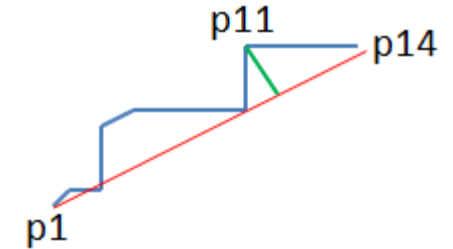
Poligonalizarea prin divizarea recursiva a frontierei(2)



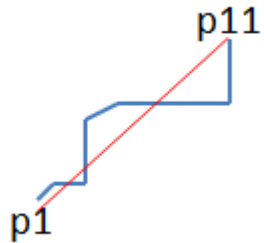
$$E_{\max} = D(p16) > \text{prag}$$



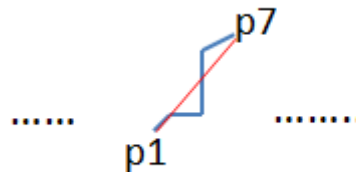
$$E_{\max} = D(p14) > \text{prag}$$



$$E_{\max} = D(p11) > \text{prag}$$



$$E_{\max} > \text{prag}$$



$$E_{\max} \leq \text{prag}$$

iesire din recursivitate

Varfurile liniei poligonale: p1, p7,...,p11,p14,p16,p19

Poligonalizarea prin divizarea recursiva a frontierei(3)

Intrarea algoritmului: lista adreselor pixelilor frontierei

Ieșirea: vârfurile liniei poligonale care aproximează frontiera, memorate într-o lista înlantuita

O implementare in C:

```
typedef struct PIXEL {int, x,y;} PIXEL;  
typedef struct CEL { PIXEL p; struct CEL * urm} CEL; // o celula a listei inlantuite  
typedef struct LISTV { CEL * start; CEL * end;} LISTV; // pointeri catre prima si ultima celula ale listei
```

```
void Poligonalizare1(PIXEL * pixeli, LISTV varfuri, int i1, int i2, double prag)
```

```
{ // i1 si i2 sunt indicii care delimiteaza frontiera in vectorul "pixeli" la apelul curent
```

```
    int i, j, imax; double d = 0, dmax; boolean gata = true;
```

```
    for (i=i1+1; i<=i2-1 ; i++)
```

```
    { // calculeaza distanta de la pixelul i la segmentul determinat de pixeli[i1] si pixeli[i2]
```

```
        d = Distanta(pixeli, i1, i2, i);
```

```
        if (d > prag)
```

```
            { gata = false; break;}
```

```
    }
```

Determinarea varfurilor prin divizarea recursiva a frontierei(4)

```
if (gata)
    return; // toti pixelii frontierei sunt la o distanta mai mica decat pragul
// Determina pixelul aflat la distanta maxima de segmentul pixeli[i1] - pixeli[i2]

dmax = d; imax = i;

for (j=i+1; j<=i2-1; j++) // se continua parcurgerea frontierei
{
    d = Distanta(pixeli, i1, i2, j);

    if (d > dmax) { dmax = d; imax = j; }
}

Poligonalizare1(pixeli, varfuri, i1, imax, prag);

AdaugaVarf (varfuri, pixeli[imax]);

Poligonalizare1(pixeli, varfuri, imax, i2, prag);

}
```

Poligonalizarea prin divizarea recursiva a frontierei(5)

Apel:

LISTV Polig;

PIXEL *Frontiera;

int nfront; // **numarul pixelilor de frontiera**

double Prag;

Polig.start = NULL; Polig.end = NULL;

AdaugaVarf (Polig, Frontiera[0]);

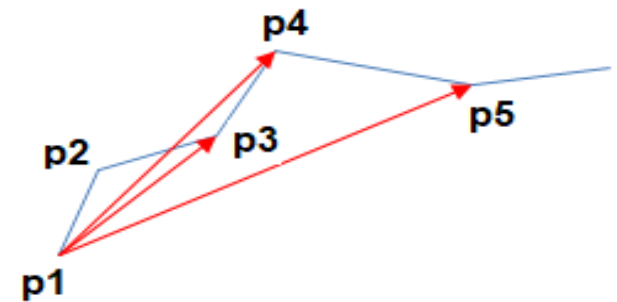
Poligonalizare1(Frontiera, Polig, 0, nfront-1, Prag);

AdaugaVarf (Polig, Frontiera[nfront-1]);

- Dacă pragul este mic, poligonul va avea un număr mare de varfuri.
- Alegerea pragului depinde de scopul poligonalizarii.
- **Principalul avantaj** al metodei: detectează punctele de inflexiune ale frontierei, care devin vârfuli ale liniei poligonale.

Poligonalizare prin parcurgerea frontierei (1)

- Se pleacă din p_1 și se avansează în lista de pixeli până într-un punct p_i , în care eroarea de aproximare a frontierei p_1-p_i printr-un segment de dreaptă depășește pragul.
- Se memorează p_{i-1} în lista vârfurilor liniei poligonale.
- Se repetă procesul plecând din p_{i-1} .



$E_{\max} = D(p_4) > \text{prag}$

Lista de varfuri: p_1, p_4, \dots

- Dacă frontiera este închisă, se alege p_1 ca punctul de inflexiune cel mai proeminent.

Algoritm:

Intrare: lista pixelilor de frontieră

Ieșire: lista vârfurilor liniei poligonale care aproximează frontiera

Poligonalizarea prin parcurgerea frontierei (2)

```
void Poligonalizare2(PIXEL * pixeli, LISTV varfuri, int start, int n, double prag)
{ // n este indicele maxim in vectorul "pixeli";
  // start este indicele primului pixel al frontierei curente
  double d, dmax, int l, end=start+1;
  do{
    end++; dmax =0; //dmax: eroarea maxima de aprox. a frontierei prin segmentul start-end
    for (i=start+1; i<end; i++)
      { // calculeaza distanta de la pixelul i la segmentul determinat de pixelii (start, end)
        d = Distanta(pixeli, start, end, i);
        if (d > dmax) dmax = d;
      }
  } while (dmax <= prag && end != n);
  if (end < n)
    { AdaugaVarf (varfuri, pixeli[end-1]); Poligonalizare2(pixeli, varfuri, end-1, n, prag); }
  else
    AdaugaVarf (varfuri, pixeli[n]); // adauga ultimul punct de frontiera in lista de varfuri
}
```

Poligonalizarea prin parcurgerea frontierei (3)

LISTV Polig;

PIXEL *Frontiera;

int nfront; // numarul pixelilor de frontiera

double Prag;

Polig.start = NULL; Polig.end = NULL;

Apel:

AdaugaVarf (Polig, Frontiera[0]);

Poligonalizare2(Frontiera, Polig, 0, nfront-1, Prag);

➤ **Principalul dezavantaj** al metodei:

- vârfurile poligonului nu coincid cu punctele de inflexiune ale frontierei.

Operații morfologice

Prof. univ. dr. ing. Florica Moldoveanu

Curs Sisteme de Prelucrare Grafică – UPB, Automatică și Calculatoare
2021-2022

Morfologie

❖ Morfologia: studiul formelor

- Imaginile binare contin numeroase defecte, mai ales cele obtinute prin aplicarea unui prag.
- In prelucrarea imaginilor binare:

operatiile morfologice se folosesc pentru eliminarea defectelor si simplificarea imaginilor in vederea recunoasterii formelor:

- Operatiile morfologice pot fi extinse pentru imagini in mai multe nivele de gri.
- Principalele operatii morfologice:
 - Erodarea
 - Dilatarea

Operatii morfologice

Operatie morfologica:

- **Intrările :**

- **$I(x,y)$ - o imagine binara:** 0 – pixeli de fond; 1 – pixeli din regiuni
- **Un element structural** (șablon) - contine, in general, trei tipuri de valori: 0, 1, neutra; valorile neutre nu au efect la utilizarea șablonului
- Exemplu de element structural:

2	1	2	← valoare neutră
0	1	1	← originea șablonului
0	0	2	

- **Iesirea, $O(x,y)$: o imagine binara**

- **Executia:**

- Se translateaza șablonul cu originea sa peste fiecare pixel $I(x,y)$ al imaginii de intrare, comparandu-se pixelii acoperiti de șablon cu valorile șablonului egale cu 0 sau 1.
- In functie de rezultatul comparatiei si scopul operatiei:

$$O(x,y) \leftarrow I(x,y) \text{ sau } O(x,y) \leftarrow 0 \text{ sau } O(x,y) \leftarrow 1.$$

Erodarea (Erosion):

➤ Erodarea frontierelor regiunilor – de exemplu, pentru extragerea scheletului unei regiuni

❖ Elementul structural contine numai valori 1 si neutrale.

- Pentru fiecare pixel $I(x,y)$ al unei regiuni din imaginea de intrare ($I(x,y) = 1$)
 - Se suprapune șablonul cu originea sa peste pixelul $I(x,y)$
 - Daca fiecare element al șablonului =1 coincide cu pixelul imaginii de intrare acoperit de el, atunci $O(x,y) \leftarrow I(x,y)$, altfel $O(x,y) \leftarrow 0$ (devine pixel de fond)
 - Fie urmatoarele elemente structurale:

1	1	1
1	1	1
1	1	1

Pentru regiuni cu conectivitate de ordin 8

	1	
1	1	1
	1	

Pentru regiuni cu conectivitate de ordin 4

- Efectul operatiei de erodare cu aceste sabloane: **orice pixel care nu este inconjurat de pixeli ai unei regiuni este eliminat (devine pixel de fond);** astfel, **pixelii aflatii la frontierele regiunilor sunt eliminati.**

Dilatarea (Dilation)

- Este operatia duală erodarii :
 - Erodarea regiunilor - echivalenta cu dilatarea zonelor cu pixeli de fond.
 - Dilatarea regiunilor - echivalenta cu erodarea zonelor cu pixeli de fond.
- **Pentru fiecare pixel de fond al imaginii de intrare ($I(x,y)=0$)**
 - Se suprapune șablonul cu originea sa peste pixelul $I(x,y)$
 - **Daca cel puțin un element al șablonului coincide cu pixelul imaginii de intrare acoperit de el, atunci $O(x,y) \leftarrow 1$ (devine pixel de regiune)**
- Efectul operatiei de dilatare folosind elementul structural:

1	1	1
1	1	1
1	1	1

sau

	1	
1	1	1
	1	

devine pixel de regiune orice pixel de fond care are in vecinatatea sa un pixel de regiune.

Astfel, regiunile cresc iar gaurile din interiorul regiunilor se micsoreaza.

Morfologia matematica

- In morfologia matematica (Mathematical morphology) operatiile morfologice sunt definite folosind teoria multimilor, considerand imaginea de intrare (A) si elementul structural (B) ca multimi.
- **Operatiile morfologice de baza**, erodarea și dilatarea sunt definite in morfologia matematica astfel:

Erodarea multimii A cu multimea B: $A \ominus B = \bigcap_{b \in B} A_{-b}$.

Dilatarea multimii A cu multimea B : $A \oplus B = \bigcup_{b \in B} A_b$.

- Folosind operatorii de erodare si dilatare sunt definiti alti operatori morfologici:

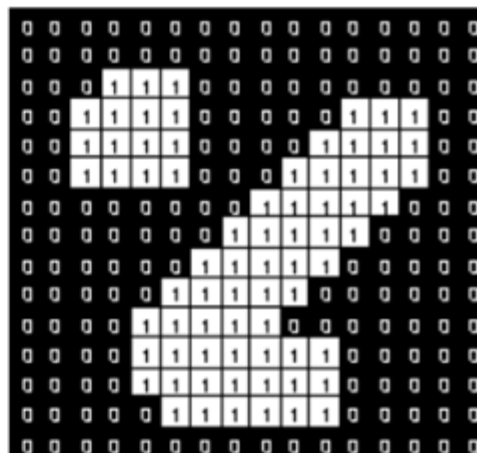
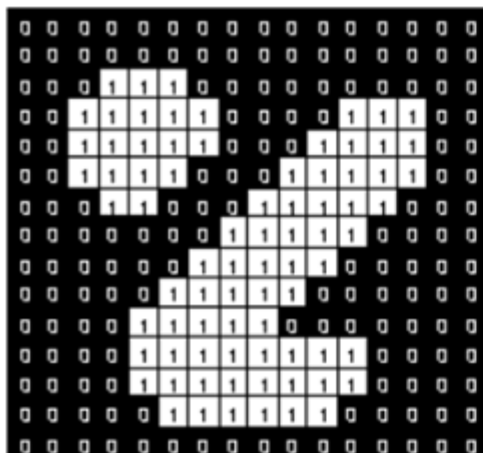
Deschiderea (Opening) - erodarea multimii A cu multimea B, urmata de dilatarea rezultatului folosind B: $A \circ B = (A \ominus B) \oplus B$.

Inchiderea (Closing) - dilatarea multimii A folosind B, urmata de erodarea rezultatului folosind B: $A \bullet B = (A \oplus B) \ominus B$.

Deschiderea

Deschiderea: $A \circ B = (A \ominus B) \oplus B$.

- Prin erodare: pixelii din regiuni complet acoperiti de elementul structural sunt conservati; cei de la frontiere sunt eliminati.
 - Dupa dilatarea rezultatului cu acelasi element structural, rezulta regiuni care contin complet elementul structural
 - Efectul depinde de elementul structural.
 - Este eficienta in eliminarea zgomotului de tip “sare”
- ❖ **Produce regiuni care contin complet elementul de structurare, eliminand regiunile mici (care nu contin elementul de structurare).**



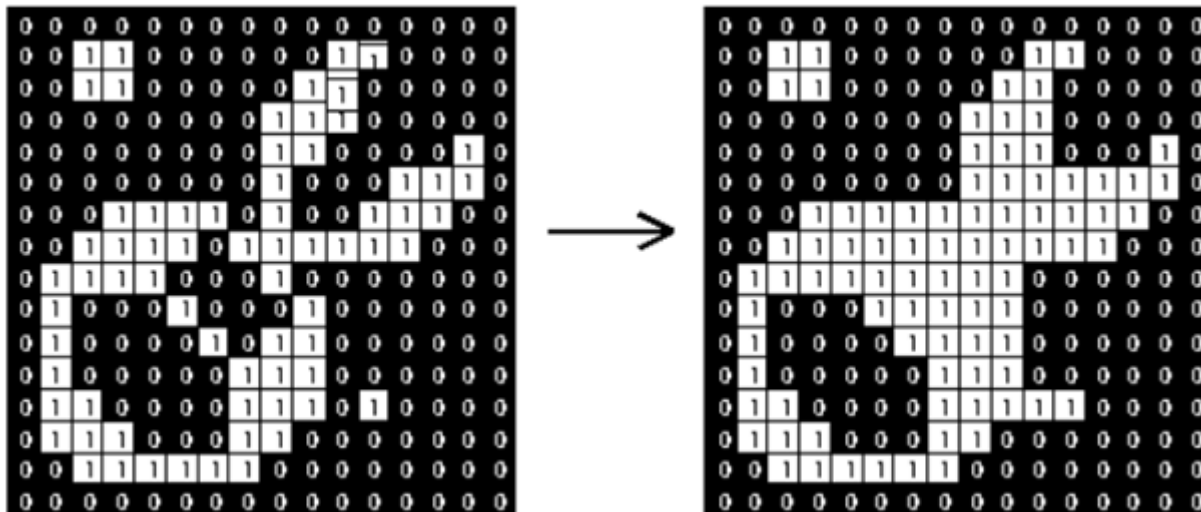
Elementul structural

1	1	1
1	1	1
1	1	1

Inchiderea

Inchiderea: $A \bullet B = (A \oplus B) \ominus B$.

- Ca si dilatarea, tinde sa largeasca marginile regiunilor si sa micsoreze gaurile din regiuni.
- Este eficienta in eliminarea zgomutului de tip “piper”.
- Efectul depinde de elementul de structurare:
- ❖ **Pastreaza zonele de fond care au forma similara cu cea a elementului de structurare sau care contin complet elementul de structurare eliminand toate celelalte zone de pixeli de fond.**



Transformarea “Hit-and-miss”

- Este o operatie morfologica binara generala, care poate fi folosita pentru identificarea unor șabloane de pixeli de regiune sau pixeli de fond dintr-o imagine.
- Elementul structural poate contine: 0, 1 (pixeli de fond si pixeli de regiune), valori neutrale.
- Pentru fiecare pixel $I(x,y)$ al imaginii de intrare
 - Se suprapune șablonul cu originea sa peste pixelul $I(x,y)$
 - **Daca fiecare element al șablonului (0 sau 1) este identic cu pixelul peste care este suprapus, atunci $O(x,y) \leftarrow 1$, altfel $O(x,y) \leftarrow 0$.**

- Exemplu: șablonul

	1	
0	1	1
0	0	

 poate fi folosit pentru identificarea pixelilor de colț ai regiunilor, mai exact a colțurilor convexe stanga –jos

Pentru identificarea tuturor colțurilor convexe, se proceseaza imaginea de intrare de 4 ori, folosind de fiecare data șablonul specific:

Rezulta 4 imagini de iesire care pot fi combinate prin operatia OR.

	1	
0	1	1
0	0	

0	0	
0	1	1
	1	

	1	
1	1	0
	0	0

	0	0
1	1	0
	1	

Subțierea (*Thinning*)

- Operație de erodare iterativă, prin care sunt îndepărtati succesiv pixeli de frontieră ai regiunilor unei imagini binare.
- Este utilă pentru:
 - scheletizarea regiunilor rezultate din segmentarea imaginilor, în scopul recunoașterii formelor
 - subțierea frontierelor produse de detectoarele de frontiere (ex. detectorul Sobel), prin reducerea lățimii liniilor la un pixel, fără modificarea lungimii lor

repetă

gata = 1

pentru fiecare pixel al imaginii de intrare $I(x,y)$:

dacă $I(x,y)$ este pixel de frontieră (are cel puțin un vecin pixel de fond)

și are mai mult de un vecin în regiune (pixel = 1)

și prin eliminarea sa nu se distruge conectivitatea formei

atunci $I(x,y) \leftarrow 0$ (devine pixel de fond)

gata = 0

cat timp (! gata)

Subțierea ca operație morfologică (1)

Condiția de eliminare a unui pixel de regiune este implementată folosind elemente de structurare cu valori 0 și 1.

Pot fi folosite următoarele două șabloane și cele obținute prin rotația lor cu 90, 180 și 270 de grade (în total, $4 \times 2 = 8$ șabloane):

0	0	0
	1	
1	1	1

	0	0
1	1	0
	1	

repetă

gata = 1

***initializare matrice de marcăre, M, cu valoarea 0;**

pentru fiecare șablon de subțiere, S_i , $i = 1, 8$

pentru fiecare pixel $I(x,y)=1$ al imaginii de intrare

- se aplică șablonul S_i cu originea suprapusă peste pixelul $I(x,y)$

- **dacă** toate elementele 0 și 1 ale șablonului sunt egale cu valorile pixelilor acoperiți de șablon
atunci

$M(x,y) \leftarrow 1$ (se marchează pentru eliminare pixelul acoperit de originea șablonului)
gata = 0

pentru fiecare pixel $M(x,y)=1$ al imaginii de marcăre

$I(x,y) \leftarrow 0$ (se elimină pixelul de regiune)

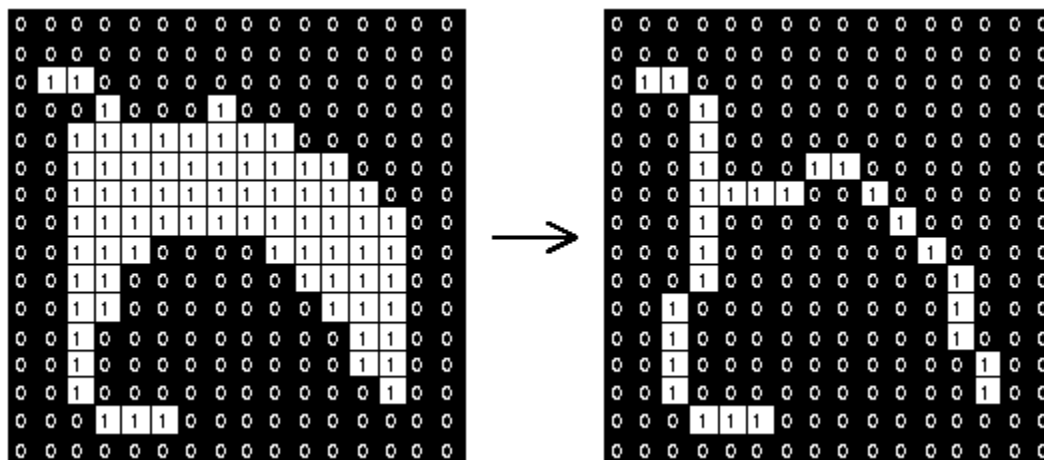
cat timp (! gata)

Subțierea ca operatie morfologica (2)

- Pentru ca subțierea sa fie simetrica:
 - In fiecare iteratie se aplica mai intai sablonul din stanga, apoi cel din dreapta, apoi celelalte 6 rezultate din rotatia celor 2 cu 90, 180 si 270 de grade.
 - Intr-o iteratie, pixelii imaginii sunt vizitati de 8 ori!
- Rezultatul subțierii unei regiuni prin aplicarea celor 8 șabloane este un “schelet” conectat. Exemplu:

0	0	0
	1	
1	1	1

	0	0
1	1	0
	1	



<http://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm>

Subțierea ca operație morfologică (3)

- Scheletul produs prin cele 8 sabloane poate contine “pinteni scurți”, produși de iregularitățile frontierei.
- Aceștia pot fi îndepărtați printr-o operație numită “pruning” (tăierea crengilor din partea inferioară a tulpinii arborilor), care este de fapt un alt tip de subțiere. Operația poate fi implementată folosind următoarele 2 sabloane și cele obținute prin rotația lor cu 90 grade:

0	0	0
0	1	0
0		

0	0	0
0	1	0
		0

- Aceste sabloane trebuie aplicate numai în câteva iterații ale procesului de subțiere
- Aplicarea lor până la convergență poate conduce la eliminarea tuturor pixelilor, cu excepția acelor care formează cicluri închise.

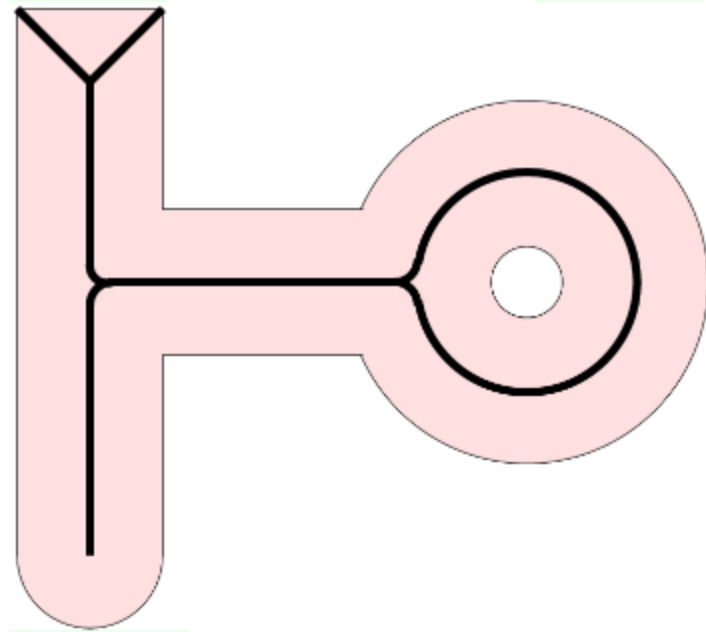
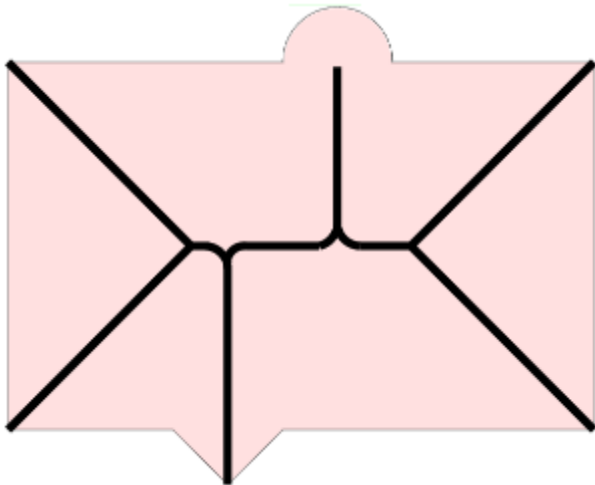
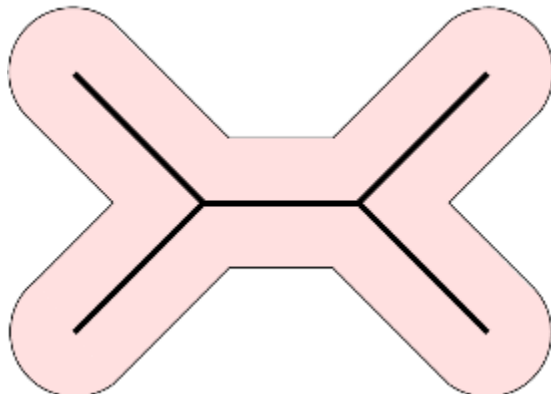
Scheletizarea (skeletonization)

Prof. univ. dr. ing. Florica Moldoveanu

*Curs Sisteme de Prelucrare Grafică – UPB, Automatică și Calculatoare
2021-2022*

Scheletizarea(1)

❖ Multe forme, mai ales cele subtiri, pot fi descrise prin versiunile lor subtiate, alcatuite din linii conectate, aflate, in mod ideal, de-a lungul **axei mediane**. Exemple:



<http://www.inf.u-szeged.hu/~palagyi/skel/skel.html>

Scheletizarea(2)

- ❖ **Scopul scheletizarii:** extragerea axei schelet (axa mediana) a unei regiuni, in scopul recunoasterii forme regiunii.
- ❖ **Axa schelet a unei regiuni este o aproximare in spatiul discret a scheletului unei forme.**

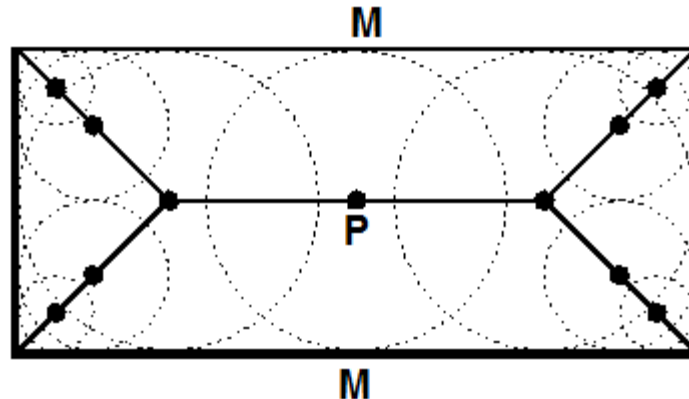
Trebuie sa satisfaca 2 cerinte:

- **Topologica:** sa conserve caracteristicile topologice ale forme: lungimea si latimea forme, nr. de puncte de jonctiune si alte caracteristici
- **Geometrica:** sa se situeze cat mai aproape de axa mediana a forme originale
- Se utilizeaza pentru:
 - recunoasterea caracterelor de text din documente scanate (unde, la o rezolutie mare, caracterele apar cu o grosime de mai multi pixeli)
 - recunoasterea scrisului manual si a semnaturilor
 - recunoasterea de forme diverse.

Scheletizarea (3)

❖ Axa schelet (axa mediana) a unei regiuni, R

- In planul continuu, axa schelet este alcatuita din puncte P care sunt centre ale unor cercuri continute în întregime în R , cu proprietatea ca nu exista alte cercuri cu aceleasi centre, de raza mai mare, continute în R



- In planul discret, axa schelet se obtine prin 2 tipuri de metode:
1. Calculand distantele aproximative ale pixelilor regiunii fata de pixelii de frontiera – pixelii de schelet sunt cei aflati la distanta maxima locala față de cei mai apropiati pixeli de frontiera
 2. Printr-un proces iterativ de erodare a marginilor formei (subtiere)

Metrice de distanta in planul discret

- Folosite pentru aproximarea distantei dintre doi pixeli.

Distanta Euclidiană

- Daca adresele celor doi pixeli sunt (x_1, y_1) si (x_2, y_2) atunci distanta Euclidiană este:

$$D_{\text{Euclid}} = ((x_2 - x_1)^2 + (y_2 - y_1)^2)^{0.5}$$

Distanta “City Block”

- Cunoscuta si ca “distanta Manhattan”.
- Deplasarea dintre doi pixeli se poate face mergand numai pe liniile grilei de pixeli.

Deplasarile diagonale nu sunt permise.

$$D_{\text{City}} = |x_2 - x_1| + |y_2 - y_1|$$

Distanta “Tabla de sah”(Chessboard Distance)

- Este inspirata din jocul de sah (miscarea regelui): o deplasare diagonala conteaza la fel ca o deplasare orizontala sau verticala.

Transformarea distanță (Distance transform)

Chessboard Distance – cont

$$D_{\text{Chess}} = \max(|x_2 - x_1|, |y_2 - y_1|)$$

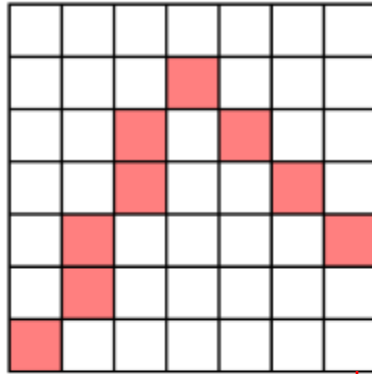
- Distanțele “City block” și “Chessboard” se calculează mult mai rapid decât distanța Euclidiană.
- Sunt utilizate atunci când viteza de calcul este mai importantă decât acuratețea calculelor.

Transformarea distanță

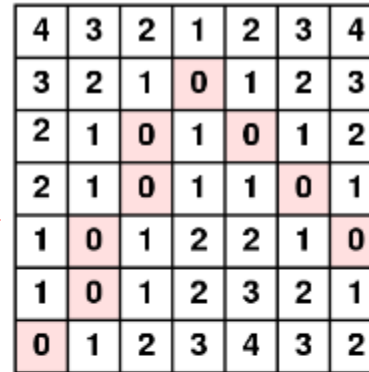
- Se marchează în imaginea de intrare, I , pixelii de interes (pot fi pixelii conturului unei regiuni).
- Fiecare pixel (x,y) din imaginea transformată (imaginea de ieșire) are ca valoare distanța pixelului față de cel mai apropiat pixel de interes.
- Implementarea iterativă a transformării constă în utilizarea unor măști cu ajutorul cărora se pot calcula iterativ distanțele de la pixelii imaginii la pixelii de interes.

Transformarea distanță (2)

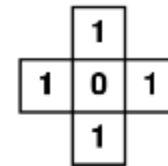
Rezultatele transformării cu diferite masti



Pixelii de interes marcati



$$D_{\text{City}} = |x_2 - x_1| + |y_2 - y_1|$$

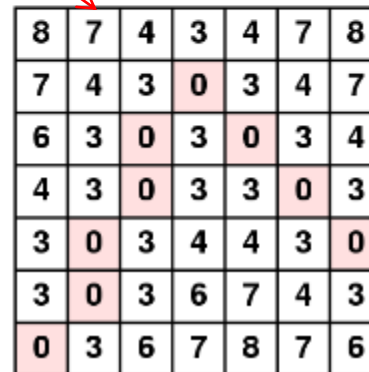


masca "city block"

$$D_{\text{Chess}} = \max(|x_2 - x_1|, |y_2 - y_1|)$$



masca
"chess-board"



masca "3-4"

Transformarea distanță (3)

Implementarea iterativa a transformarii distanta

repetă

gata = 1

pentru fiecare pixel $I(x,y)$ al imaginii

- plaseaza masca centrata pe pixelul (x,y)
- pentru fiecare element $m[k][l]$ al mastii care acopera vecinatatea pixelului (x,y)
 - calculeaza $sum[k][l] = \text{suma dintre } m[k][l] \text{ si valoarea pixelului acoperit}$
- calculeaza $summin = \min(sum[k][l])$
- daca $I(x,y) \neq summin$ atunci
 - gata = 0;
 - $I(x,y) \leftarrow summin$

cat timp(! gata) // s-a modificat imaginea

Transformarea distanță (4)

1	1	1	1	1	1	1
1	1	1		1	1	1
1	1		1		1	1
1	1		1	1		1
1		1	1	1	1	
1		1	1	1	1	1
	1	1	1	1	1	1

Imaginea initiala
(pixelii de interes
au valoarea 0)

	1	
1	0	1
	1	

masca "city block"

2	2	2	1	2	2	2
2	2	1	0	1	2	2
2	1	0	1	0	1	2
2	1	0	1	1	0	1
1	0	1	2	2	1	0
1	0	1	2	2	2	1
0	1	2	2	2	2	2

Dupa prima iteratie

3	3	2	1	2	3	3
3	2	1	0	1	2	3
2	1	0	1	0	1	2
2	1	0	1	1	0	1
1	0	1	2	2	1	0
1	0	1	2	3	2	1
0	1	2	3	3	3	2

Dupa a 2-a iteratie

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

Dupa a 3-a iteratie

Scheletizarea folosind transformarea distanta

- În imaginea segmentată binară pixelii regiunilor au valoarea 1 iar cei de fundal valoarea 0.
 - Imaginea este transformată în mai multe iteratii, executand algoritmul anterior, folosind o masca de aproximare a distantei.
 - **Pixelii din imaginea finala care au valoarea locala maxima sunt considerati pixeli de schelet.**
- Numarul de iteratii este proportional cu cea mai mare distanta de la un pixel de regiune la cel mai apropiat pixel de frontiera.

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

Imaginea segmentata
Pixelii de interes au
valoarea 0



0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	2	2	2	2	1	0
0	1	2	2	2	2	1	0
0	1	2	2	2	2	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

Dupa prima iteratie

masca
'chess-board'

1	1	1
1	0	1
1	1	1



0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	2	2	2	2	1	0
0	1	2	3	3	2	1	0
0	1	2	2	2	2	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

Dupa a 2-a iteratie

Scheletizarea prin subtiere (1)

❖ Subtierea este un proces iterativ de erodare a marginilor unei regiuni.

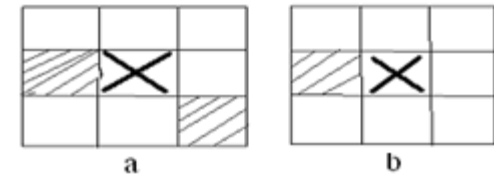
❖ Pixelii de contur într-o imagine binară: pixeli de tranziție $0 \rightarrow 1$

❖ Erodarea = îndepărtarea pixelilor de contur astfel încât:

1. să fie conservată topologia formei

(a) - să nu fie distrusă conectivitatea formei

(b) - să nu fie scurtate terminatiile formei



2. să fie conservată geometria formei: axa schelet să fie cât mai apropiată de axa mediană a formei

Scheletizarea prin subtiere (2)

- Intrarea algoritmului: imagine binara, in care pixelii regiunilor au valoarea 1.
- Iesirea algoritmului: imagine binara in care numai pixelii axelor schelet au valoarea 1.
- In fiecare iteratie se viziteaza o singura data fiecare pixel al imaginii, verificandu-se daca poate fi indepartat, cu satisfacerea celor 2 constrangeri de conservare a topologiei(a si b):
- Pentru aceasta se cerceteaza vecinatatea de 8 pixeli a fiecarui pixel de regiune (=1).

P8	P1	P2
P7	P	P3
P6	P5	P4

Imaginea este parcursă de la stanga spre dreapta, de sus in jos, eliminand pixelii de contur ai regiunilor cu satisfacerea celor 2 constrangeri de topologie.

Dezavantajul algoritmului: nu subtiaza imaginea simetric

- Axa schelet este localizata in partea de sud - est a regiunii, deoarece pixelii de frontiera din partea de nord – vest sunt eliminati primii.

→ rezultat nesatisfacator atunci cand obiectele din imagine sunt relativ mari si convexe.

Scheletizarea prin subtiere (3)

repetă

gata = 1

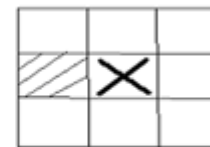
pentru fiecare pixel P (x,y) al imaginii: $0 \leq x \leq x_{\max}-1$, $0 \leq y \leq y_{\max}-1$

- dacă $P(x,y) = 1$ (pixel al unei regiuni)
 - se numără pixelii din regiune din vecinătatea lui P: $N(P)$
 - dacă $(N(P) \leq 2)$, P nu poate fi eliminat din regiune căci el aparține unei terminatii sau conectează 2 parti ale unei forme;

P8	P1	P2
P7	P	P3
P6	P5	P4



a



b



- dacă $N(P) = 8$, P nu poate fi îndepărtat, deoarece aceasta ar conduce la distrugerea formei – P nu este pixel de contur;
- dacă $2 < N(P) < 8$:

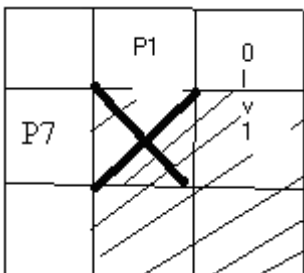
- se numără tranzițiile $0 \rightarrow 1$ din secvența de pixeli: P8, P1, P2, P3, P4, P5, P6, P7, P8

- dacă $NrTranz = 1$ (în fereastra de 3x3 pixeli există o singură componentă conectată) atunci

$P(x,y) \leftarrow 0$; (se elimină pixelul din centru căci îndepărtarea sa nu afectează conectivitatea locală a celorlalți pixeli din fereastră)

gata = 0

cat timp(! gata) // au mai fost îndepărtati pixeli de contur

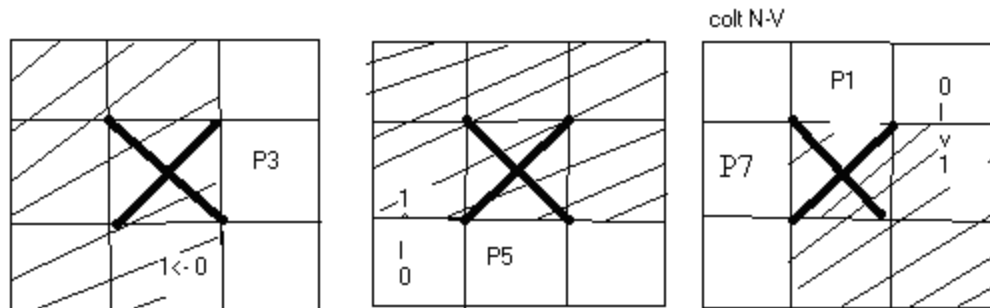


Scheletizarea prin subtiere (4)

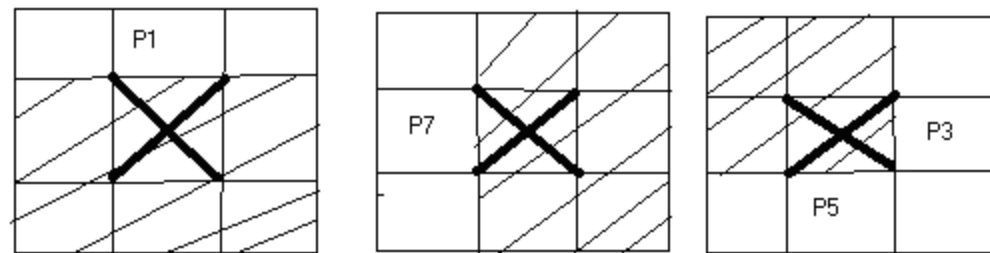
Subtierea in doi pasi (subtiere simetrică – axa schelet in apropierea axei mediane a formei):

❖ **Intr-o iteratie imaginea este vizitata de 2 ori (in 2 pasi):**

- In primul pas sunt eliminati pixelii care apartin unei frontiere de est, de sud sau unui colt de nord - vest.
- In pasul al doilea sunt eliminati cei care apartin unei frontiere de nord, de vest sau unui colt de sud - est.



pixeli eliminati in primul pas



pixeli eliminati in al 2-lea pas

V<-

colt S - E

Scheletizarea prin subtiere (5)

Notam cu :

$N(P)$ numarul de pixeli interiori (=1) din vecinatatea de 3×3 a pixelului curent, P

$T(P)$ numarul de tranzitii $0 \rightarrow 1$ in secventa de pixeli care formeaza periferia ferestrei:

$P1, P2, P3, \dots, P8, P1$

P8	P1	P2
P7	P	P3
P6	P5	P4

Atunci cele 2 conditii de eliminare a pixelilor in cei doi pasi sunt exprimate prin predicatele:

Pas 1 :

$(2 < N(P) < 8) \ \&\& \ T(P) == 1 \ \&\& \ (P3 == 0 \ || \ P5 == 0 \ || \ (P1 == 0 \ \&\& \ P7 == 0))$

Pas2 :

$(2 < N(P) < 8) \ \&\& \ T(P) == 1 \ \&\& \ (P1 == 0 \ || \ P7 == 0 \ || \ (P3 == 0 \ \&\& \ P5 == 0))$

In fiecare iteratie, pixelii care indeplinesc conditia de a fi eliminati sunt marcati pentru eliminare intr-o alta matrice imagine si numai la sfarsitul iteratiei sunt eliminati (setati la zero in matricea imaginii subtiate). Matricea de marcare este initializata la inceputul fiecarei iteratii.

Scheletizarea prin subtiere (6)

repetă

gata = 1

*initializare matrice de marcare, M, cu valoarea 0;

pentru (pas =1; pas<=2; pas++)

pentru fiecare pixel P (x,y) al imaginii: $0 \leq x \leq x_{\max}-1$, $0 \leq y \leq y_{\max}-1$

dacă $P(x,y) = 1$ (pixel de regiune)

dacă $2 < N(P) < 8$ && $T(P) == 1$

dacă $pas == 1$ && $(P3 == 0 \mid \mid P5 == 0 \mid \mid (P1 == 0 \ \&\& \ P7 == 0))$

$M(x,y) \leftarrow 1$; //marcheaza pixelul (x,y) pentru eliminare

gata = 0;

else //pas ==2

dacă $(P1 == 0 \mid \mid P7 == 0 \mid \mid (P3 == 0 \ \&\& \ P5 == 0))$

$M(x,y) \leftarrow 1$; gata =0;

//pentru fiecare pixel

// pentru pas=1,2

pentru fiecare pixel M (x,y) al imaginii de marcare

dacă $M(x,y) = 1$

$P(x,y) \leftarrow 0$

cat timp (! gata) //nu au mai fost indepartati pixeli de contur

P8	P1	P2
P7	P	P3
P6	P5	P4

Analiza formelor din imagini

Prof. univ. dr. ing. Florica Moldoveanu

Curs Sisteme de Prelucrare Grafică – UPB, Automatică și Calculatoare
2021-2022

Analiza formelor din imagini

Aplicatii:

- Analiza imaginilor medicale, aplicatii industriale, arheologie, recunoasterea fețelor, animatie, etc
- Regasirea formelor (**shape retrieval**): gasirea unor forme (dintr-o baza de date) similare cu o forma data
- Recunoasterea si clasificarea formelor (**pattern recognition and classification**): se determina daca o forma data satisface un model sau in care clasa de forme poate fi clasificata. Exemple: recunoasterea scrisului de mână, clasificarea amprentelor digitale, a celulelor biologice, etc.

Metode de analiza:

- ❖ Bazate pe caracteristici ale conturului
- ❖ Bazate pe caracteristici de suprafață (regiune)
- ❖ Bazate pe descriptori de caracteristici locale
- ❖ Bazate pe cunoasterea structurii de ansamblu a formei (șabloane de biti sau vectori, blocuri constructive); exemplu: recunoasterea caracterelor de text.

Descriptori de forme(1)

Descriptori de forma: caracteristici ale formelor ce pot fi folosite în analiza, recunoasterea si clasificarea formelor.

- **Bazati pe contur:**
 - Structurali:
 - Reprezentarea prin coduri de inlantuire
 - Reprezentarea poligonala (varfurile amplasate in apropierea maximelor de curbura)
 - Reprezentarea prin curbe B-spline (permite cautarea de caracteristici ce se extind pe zone mari ale conturului – mai putin sensibila la zgomot), s.a tipuri de curbe
 - Globali
 - Perimetrul, curbura locala, energia curburii
 - Semnaturi (reprezentarea conturului printr-o functie de o variabila)
 - Descriptori Fourier
- **Bazati pe suprafata (regiune):** aria, compactitatea, închiderea convexa, dreptunghiul încadrator, dreptunghiul încadrator minimal, excentricitatea, elongatia, descriptori topologici, s.a.

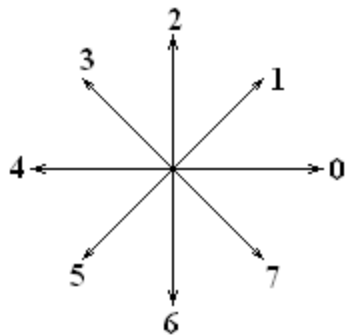
Descriptori de forme (2)

Proprietati dorite ale descriptorilor de forme, pentru recunoasterea formelor din imagini

- Unicitatea: fiecare forma sa aiba o reprezentare unica pentru a fi recunoscuta
- Invarianta la transformari geometrice afine: translatie, rotatie, scalare, oglindire
- Senzitivitatea: să poata fi deduse usor diferentele dintre forme similare
- Abstractizarea detaliilor (robustetea la zgomot): să descrie caracteristicile de baza
- Invarianta la ocluziune: atunci cand anumite parti ale unei forme sunt acoperite de alte obiecte, caracteristicile vizibile ale formei sa permita recunoasterea formei originale.

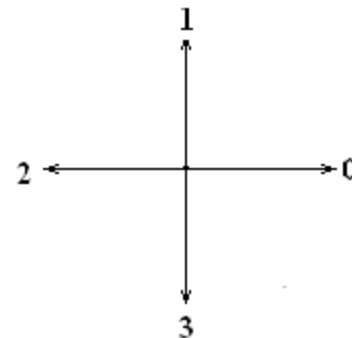
Reprezentarea prin coduri de înlantuire(1)

Reprezentare de nivel coborat, **utila în analiza caracteristicilor locale**, în imagini cu nivel de zgomot redus

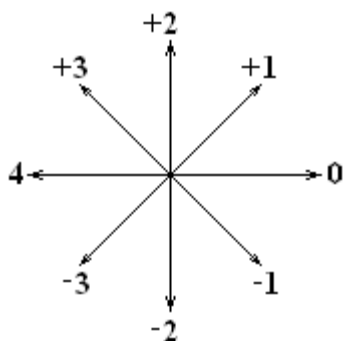


Codul de înlantuire absolut:
schimbarea absolută a direcției
de deplasare de la un punct al
conturului la următorul

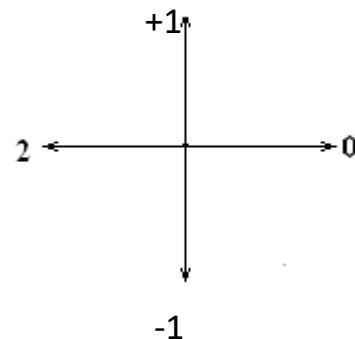
Pentru un contur conectat-8 (8-connected)



Pentru un contur conectat-4



Codul de înlantuire diferential:
schimbarea relativă de direcție
de la un punct al conturului la
următorul.



În cazul parcurgerii conturului în sens trigonometric.

Reprezentarea prin coduri de înlantuire(2)

➤ **Codul de înlantuire diferential se poate obtine din codul de înlantuire absolut scazand fiecare cod din precedentul sau.**

Exemplu:

Codul absolut al unui contur conectat-8: 5,4,5,4,4,5,4,6,7,6,7,...2,2,2

Codul diferential, pentru parcurgerea in sens trigonometric: -1, 1,-1, 0, 1, -1, 2, 1, -1, 1,, 0, 0, 3

➤ ultimul cod diferential = primul cod absolut – ultimul cod absolut

➤ Daca operatia de scadere este modulo 8 (pentru un contur conectat-8), atunci elementele codului diferential sunt numere pozitive cuprinse intre 0 si 7 → pot fi reprezentate pe 3 biti ca si elementele codului absolut.

Reprezentarea prin coduri de înlantuire(3)

Fie c_0, c_1, \dots, c_{M-1} codul de înlantuire absolut

si $c'_0, c'_1, \dots, c'_{M-1}$ codul de înlantuire diferential echivalent, pentru parcurgerea
in sens trigonometric

$c'_i = c_{i+1} \ominus c_i$ unde \ominus este operatia de scadere modulo 8 sau modulo 4 $0 \leq i \leq M-1$

$c'_{M-1} = c_0 \ominus c_{M-1}$

Exemplu:

- Codul absolut: 5,4,5,4,4,5,4,6,7,6,7,...2,2,2
- Codul diferential obtinut prin operatii de scadere modulo 8:

7 (4-5+8), 1, 7, 0, 1, 7, 2, 1, 7, 1, ..., 0, 0, 3

Reprezentarea prin coduri de înlantuire(4)

Proprietati ale reprezentarii prin coduri de înlantuire:

- Conturul reprezentat prin cod de înlantuire poate fi reconstruit dându-se:
 - punctul de start, pentru codul absolut
 - punctul de start și direcția absolută inițială, pentru codul diferențial
- La compararea a 2 forme trebuie să se înceapă din același punct
 - Mutarea punctului de start în codul de înlantuire are ca efect rotația codului
- Este invariantă la translație
- Nu este invariantă la scalare - este dependentă de rezoluția imaginii din care a fost extrasă (grila de eșantionare).
- Reprezentarea prin cod de înlantuire absolut nu este invariantă la rotație
- Reprezentarea prin cod de înlantuire diferențial este invariantă la rotații multiplu de 90 grade.
- Este sensibilă la zgomet – zgometul poate produce reprezentări diferite ale aceleiași forme

Reprezentarea prin coduri de înlantuire(5)

Reprezentarea prin “*Numar de forma (Shape number)*”

- Se considera elementele codului de inlantuire diferential ca cifre ale unui numar in baza 8, respectiv 4
- Se rotesc elementele codului (care formeaza o secventa circulara) astfel incat numarul rezultat sa aiba valoarea maxima (pentru a se evita calculul valorii numarului echivalent codului, se compara codurile pe baza ordonarii lexicografice)
- Codul de inlantuire obtinut este normalizat in raport cu punctul de start:
se numeste ***numar de forma***
 - **Compararea a 2 contururi reprezentate prin numar de forma nu depinde de punctul de start**

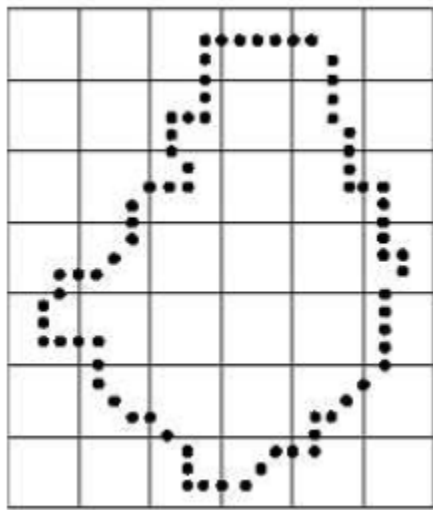
Cresterea robustetei la zgomot: netezirea codului de inlantuire

- Se efectueaza o netezire a formei, urmata de reprezentarea sa prin cod de inlantuire.
- Pentru netezire se poate aplica un filtru de netezire pe imaginea formei
- Se pierde din informatie, dar ajuta in eliminarea zgomotului si la compararea a 2 forme.

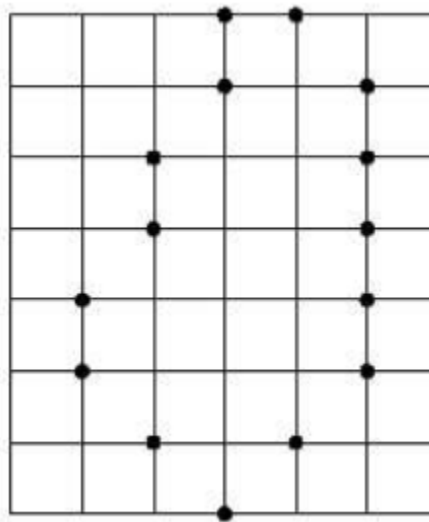
Reprezentarea prin coduri de înlantuire(6)

Dependenta codului de rezolutia grilei de esantionare: re-eșantionarea codului

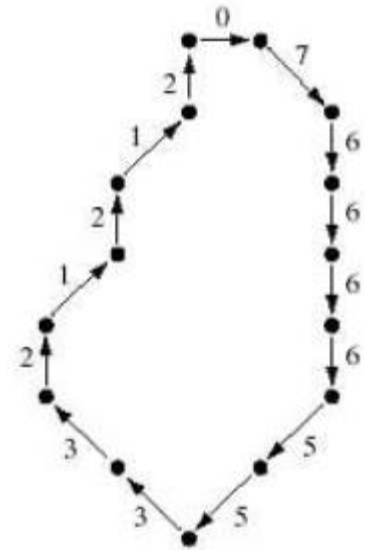
- ❖ 2 forme reprezentate prin cod de inlantuire pot fi comparate daca s-au obtinut pe aceeasi grila de esantionare: codurile pot fi re-esantionate la aceeasi rezolutie



Conturul reconstruit pe baza codului initial



Conturul esantionat pe grila mai rara



Codul re-esantionat pe grila mai rara

- ❖ Prin re-esantionare:
 - Se netezesc variatiile mici
 - Lungimea codului de inlantuire este mai mica
 - Se pot pierde structuri semnificative pentru obiectele mici.

Detectia caracteristicilor locale (1)

folosind codul de înlantuire

- Caracteristica locala: caracteristica reprezentata printr-o secventa relativ mica de coduri de înlantuire într-un contur.
- Pot fi recunoscute diferite tipuri de arce. Utila in: recunoasterea defectelor pe placile de circuite imprimate, analiza contururilor in imaginile medicale, ş.a.
- Tipuri de arce in codul de inlantuire: linii drepte, arce circulare, colţuri, creste
- Detectia se bazeaza pe utilizarea de expresii regulate

Linii drepte

- Pe una dintre directiile codului de înlantuire differential: $\rightarrow\rightarrow\rightarrow\rightarrow$

se reprezinta printr-o secventa de zerouri: 0, 0, 0, .., 0 sau 0^n

- Cu directii oarecare:

$(0^m, +1, -1)^k$ sau $(0^m, -1, +1)^k$, m poate sa varieze cu 1 de la segment la segment

- din cauza zgomotului, codul poate sa nu fie atat de regulat, dar va fi caracterizat de prezenta perechilor (+1, -1) sau (-1, +1) intr-o secventa de zero.

Detectia caracteristicilor locale (2)

Arce circulare:

$(0^m, +1)^k$ sau $(0^m, -1)^k$



Colțuri:

0, -3, 0.. sau 0, +3, 0..

+1, +1, +1, +1,.. sau -1, -1, -1 -1,....

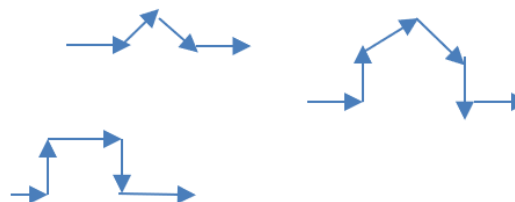


Creste: protuberante sau adancituri in contur

de interes in aplicatiile in care contururile trebuie sa fie netede – crestele indica defecte

- **Exemple de protuberante:**

- 0, +1, -2, +1, 0
- 0, +2, -1, -2, -1, +2, 0
- 0, +2, -2, -2, +2, 0



Expresia regulata prin care pot fi recunoscute crestele anterioare:

$0, (+1 \text{ sau } +2), (\text{șir de cel mult 3 valori negative}), (+1 \text{ sau } +2), 0$

- **Exemple de intrânduri:**

- 0, -1, +2, -1, 0
- 0, -2, +1, +2, +1, 0

Descriptori de forme globale bazati pe contur (1)

Se presupune reprezentarea conturului prin cod de inlantuire absolut:

x_1, x_2, \dots, x_n unde $0 \leq x_i \leq 7$

❖ Perimetrul

$$P = \sum n_i \quad \begin{array}{ll} n_i = 1, \text{ daca } x_i \bmod 2 = 0 & (0, 2, 4, 6) \\ = \sqrt{2} \text{ daca } x_i \bmod 2 = 1 & (\text{deplasari diagonale}) \end{array}$$

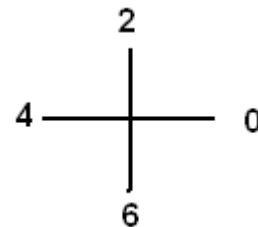
❖ Latimea si inaltimea formei

(1) Pentru o regiune conectata – d (toti pixelii regiunii sunt vecini directi):

- codurile 0 si 4 contribuie la latime
- codurile 2 si 6 contribuie la inaltime

$$\text{Latime} = \sum w_i$$

$$\text{Inaltime} = \sum h_i$$



- pentru o parcurgere in sensul acelor de ceas, codurile 0/2 contribuie la latime/inaltime:

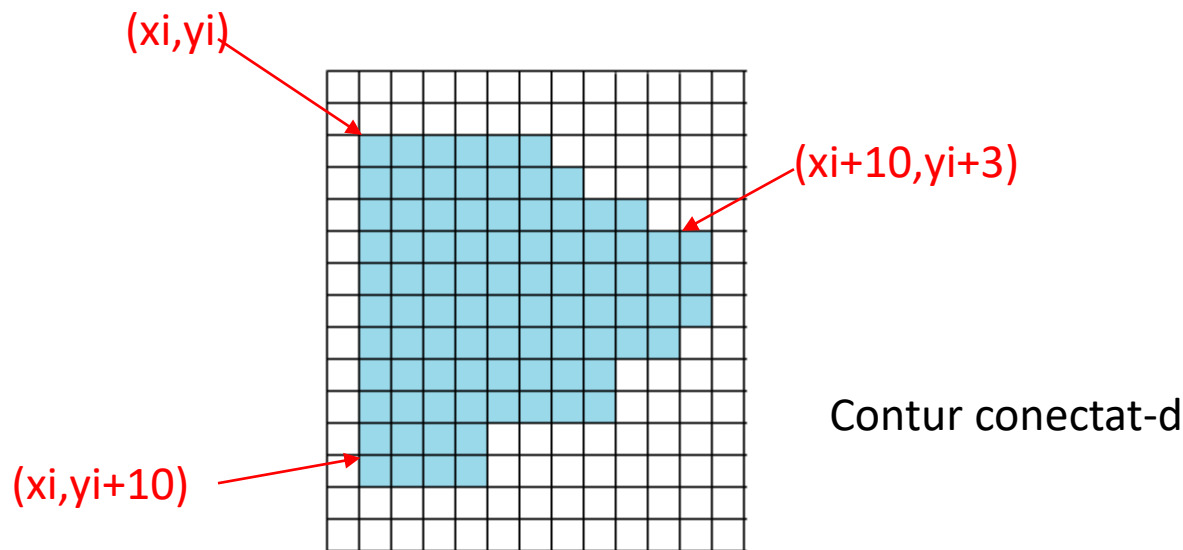
$$w_i = 0 \text{ daca } x_i = 2, 4, 6$$

$$= 1 \text{ daca } x_i = 0$$

$$h_i = 0 \text{ daca } x_i = 0, 4, 6$$

$$= 1 \text{ daca } x_i = 2$$

Descriptori de forme globale bazati pe contur (2)



0,0,0,0,0,6,0,6,0,0,6,0,0,6,6,4,6,4,4,6,6,4,4,4,4,6,6,4,4,4,2,2,2,2,2,2,2,2,2,2,2

Latime = 10 (numarul de cifre 0)

Inaltime = 10 (numarul de cifre 2)

Obs: se considera adresa coltului stanga-sus al fiecarui pixel

Descriptori de forme globale bazati pe contur(3)

Latimea si inaltimea formei (cont)

(2) pentru o regiune conectata –i:

- codurile 0, 1, 7 sau 3, 4, 5 contribuie la latime
- codurile 1,2,3 sau 5,6,7 contribuie la inaltime

$$\text{Latime} = \sum w_i$$

$$\text{Inaltime} = \sum h_i$$

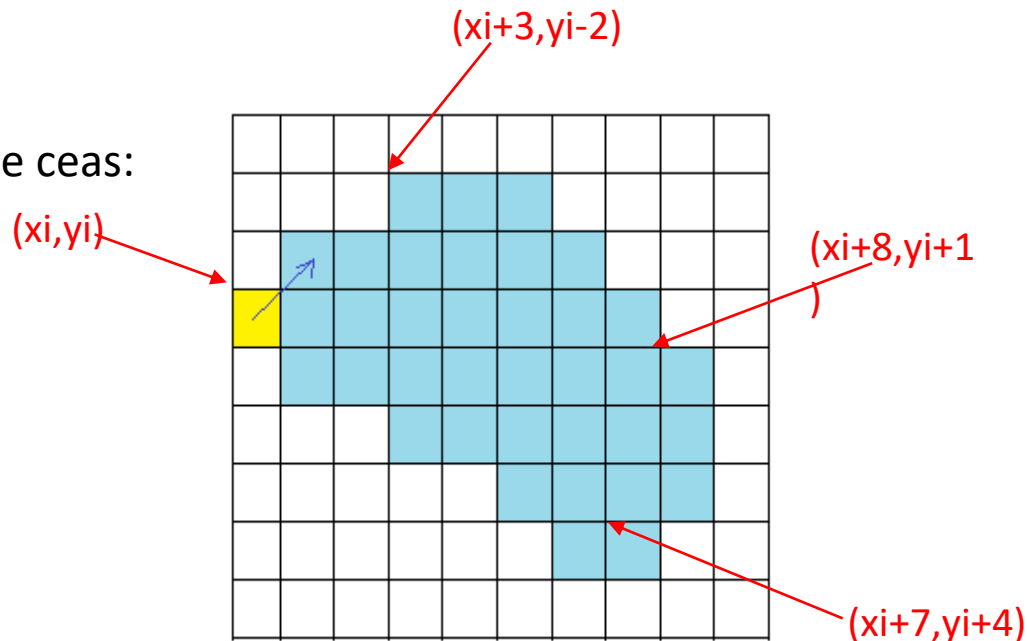
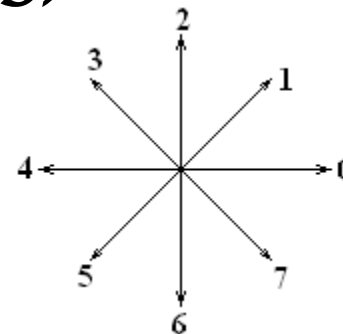
- pentru o parcurgere in sensul acelor de ceas:

$$w_i = 1 \text{ daca } x_i = 0, 1, 7$$

$$= 0, \text{ altfel}$$

$$h_i = 1 \text{ daca } x_i = 1, 2, 3$$

$$= 0, \text{ altfel}$$



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

Latime = 8

Inaltime = 6

Descriptori de forme globale bazati pe contur (4)

❑ **Descriptori utili in aplicatii in care contururile au forme aproximativ circulare** (ex.aplicatii biomedicale, unde se analizeaza contururile celulelor)

❖ **Curbura locala in punctul i al conturului (o aproximare)**

$$K(i) = (x_i - x_{i-1}) / (L(x_i) + L(x_{i-1}))$$

- numaratorul: schimbarea locala a directiei tangentei

- numitorul: aproximeaza lungimea locala a curbei, in jurul punctului i al conturului

$L(x_i) = \frac{1}{2}$ pentru x_i par; $= \sqrt{2}/2$ pentru x_i impar

❖ **Energia curburii (bending energy) intr-un punct n:**

$$E(n) = (1/P) (\sum_{i=0, n-1} |K(i)|^2) \quad 1 \leq n \leq N$$

P este perimetrul formei

$K(i)$ este curbura locala in punctul i al conturului

Cercul are energia curburii minima in orice punct, dintre toate formele cu acelasi perimetru:

$$E_{cerc} = (2\pi/P)^2$$

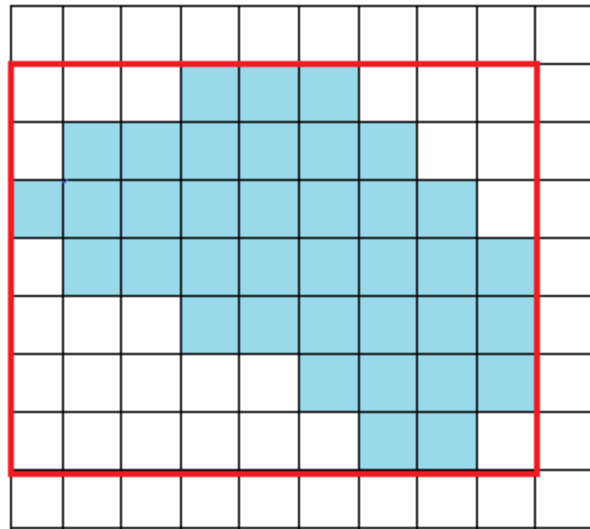
❖ **Energia curburii normalizata, intr-un punct n:**

$$E(n) = 1 - E_{cerc}/E_{obiect}(n) = 1 - 4\pi^2 / (P * (\sum_{i=0, n-1} |K(i)|^2))$$

Descriptori de forme globali bazati pe suprafata (1)

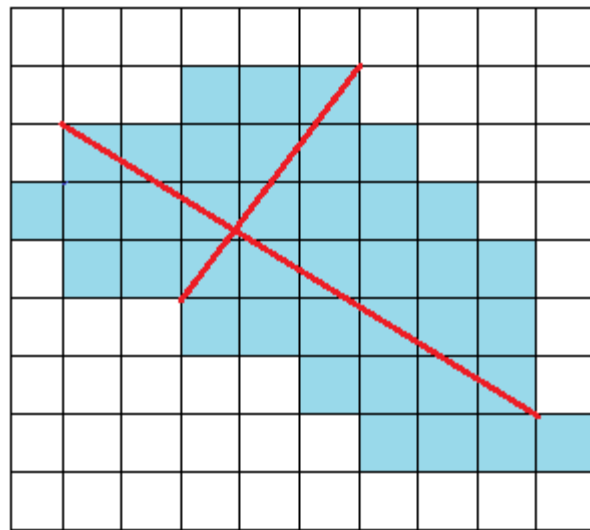
❖ Dreptunghiul încadrator (bounding box):

- încadrează forma și are laturile paralele cu axele OX, OY



❖ Axele majora și minora ale formei:

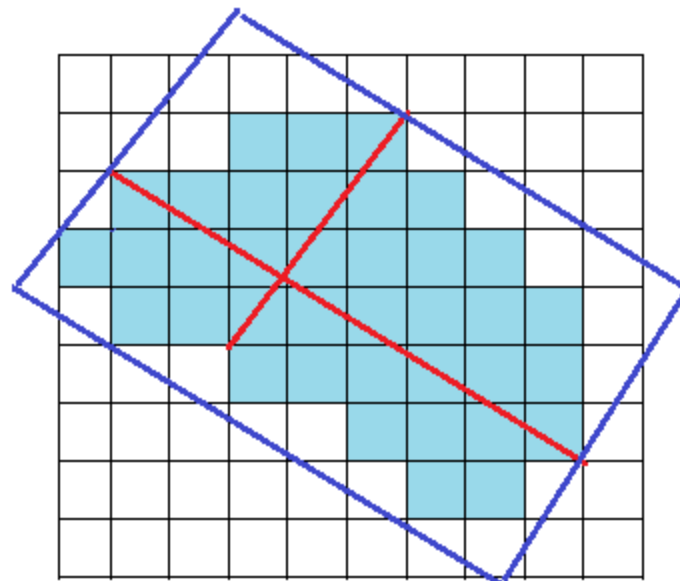
- axa majora:
linia dreaptă care unește punctele de pe contur aflate la distanța maximă
- axa minora:
linia dreaptă, perpendiculară pe axa majora, care unește puncte aflate pe contur la distanța maximă



Descriptori de forme globali bazati pe suprafata (2)

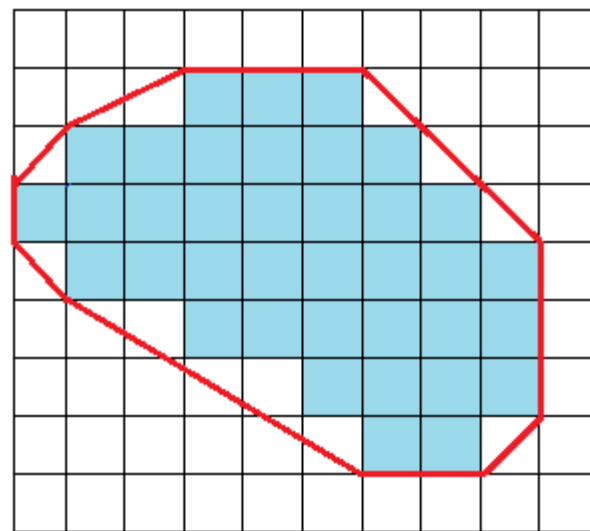
❖ Dreptunghiul încadrator minimal:

dreptunghiul încadrator minimal al formei,
cu laturile paralele și egale cu axele
majora/minora.



❖ Inchiderea convexa (convex hull):

poligonul convex minimal care încadrează forma.



Descriptori de forme globale bazati pe suprafata (3)

- ❖ **Diferenta convexa**: diferenta dintre suprafata inchiderii convexe si suprafata formei
 - Ex: permite sa se faca diferenta intre forma literei O si a literei C



- ❖ **Aria (suprafata formei)**: numarul de pixeli ai formei (se poate determina in algoritmi de segmentare)

- ❖ **Compactitatea (sau circularitatea)** - permite compararea unei regiuni cu o suprafata circulara:

$$= (\text{perimetrul conturului})^2 / (4\pi * \text{aria})$$

➤ pentru o forma circulara, compactitatea =1: $(2 * \pi * r)^2 / (4\pi(\pi * r^2))$

Descriptori de forme globale bazati pe suprafata (4)

❖ **Descriptori topologici**

- dau o informatie globala utila asupra unei forme
- pot fi furnizati de algoritmi de extragere contururi
- Exemplu de descriptor topologic: **numarul Euler**

$E = C - H$, unde C este numarul de componente conectate (regiuni) iar H numarul de gauri (contururi interioare)

- exemple: pentru litera A: $C=1, H=1 \rightarrow E=0$, pentru litera B: $C=1, H=2 \rightarrow E=-1$,
pentru litera C: $C=1, H=0 \rightarrow E=1$

A

B

C

Descriptori de forme globale

Centroidul unei forme (centrul de gravitație)

Centroidul unei regiuni :

$$gx = \frac{1}{N} \sum_1^N xi$$

$$gy = \frac{1}{N} \sum_1^N yi$$

(xi, yi) sunt pixelii regiunii ($f(xi, yi) = 1$)

Centroidul unui contur:

$$gx = \frac{1}{6A} \sum_0^{N-1} (xi + xi+1)(xi * yi+1 - xi+1 * yi)$$

$$gy = \frac{1}{6A} \sum_0^{N-1} (yi + yi+1)(xi * yi+1 - xi+1 * yi)$$

(xi, yi) sunt punctele conturului

A este aria conturului: $A = \frac{1}{2} |\sum_0^{N-1} (xi * yi+1 - xi+1 * yi)|$

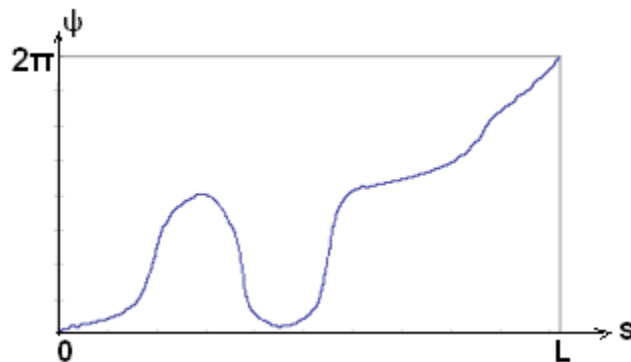
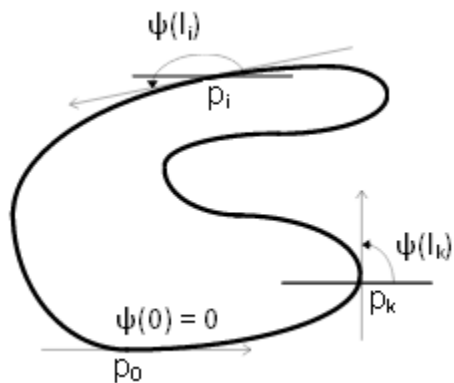
Obs: se considera triunghiurile in care se descompune forma pe baza adreselor punctelor conturului.

Descriptori de forme globale de tip semnatura (1)

- Descriptorii de tip semnatura: descriptori de contur
- **O semnatura este o functie 1D discreta care reprezinta un contur.**
- Motivatie: **descriptorii semnatura sunt invariati la scalare, rotatie, translatie**

❖ **Curba ψ -s – reprezentarea tangentiala**

- Pentru fiecare punct al conturului, se memoreaza directia vectorului tangent la contur in acel punct.
- Unghiul tangentei, ψ , intr-un punct al conturului este functie de lungimea de arc, s , de-a lungul conturului, pana in acel punct.



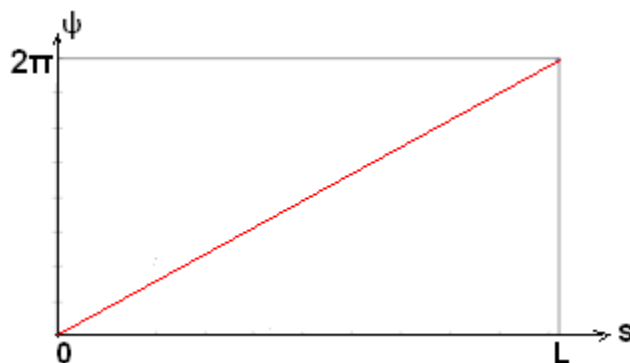
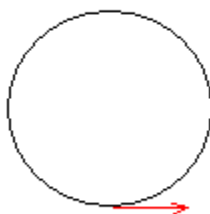
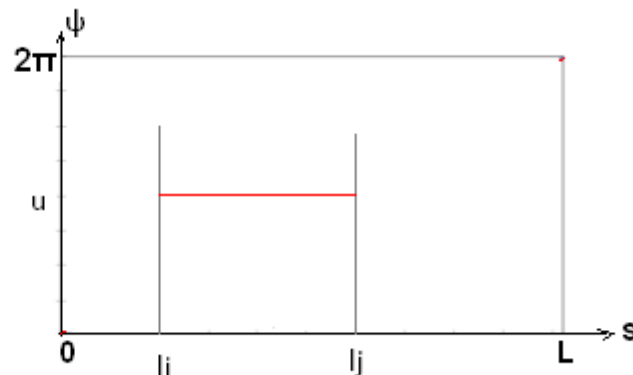
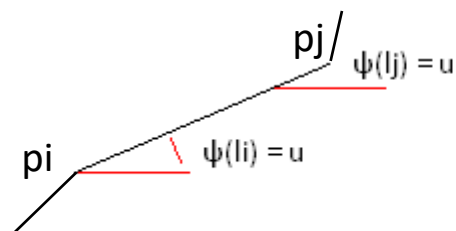
$\Psi(s)$ = unghiul tangentei in punctul conturului in care lungimea de arc este s

L : lungimea conturului
 p_0 : punctul de start la traversarea conturului

Descriptori de forme globali de tip semnatura (2)

- Unghiul tangentei in fiecare punct al conturului este relativ la directia tangentei in punctul de start (care poate fi ales in mod arbitrar).
- Orientarea tangentei in fiecare punct este data de sensul deplasarii pe contur (in figura anterioara, trigonometric).

$(l_j - l_i) =$
lungimea
segmentului
de dreapta



Descriptori de forme globale de tip semnatura (3)

Avantajele reprezentarii prin curbe ψ -s fata de reprezentarea prin coduri de inlantuire

- Ca si codul de inlantuire este **invarianta la translatie**.
- Spre deosebire de codul de inlantuire care este limitat la directii de 45 si 90 de grade, vectorii tangenti pot fi calculati cu precizia dorita.
- Este **invarianta la rotatie** (nu depinde de orientarea formei) – unghiul tangentei intr-un punct este relativ la unghiul tangentei in punctul de start.
- Daca normalizam reprezentarea la lungimea conturului (s variaza de la 0 la 1, indiferent de lungimea L), reprezentarea devine **invarianta la scalare**.
- Permite analiza de forme complexe, alcatuite din linii si diferite tipuri de arce.

Descriptori de forme globale de tip semnatura (4)

Pentru compararea a 2 curbe ψ -s trebuie sa se inceapa din acelasi punct de start.

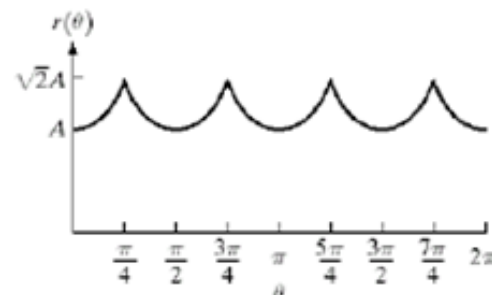
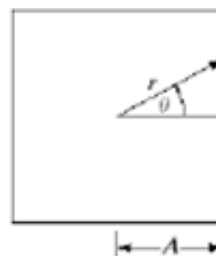
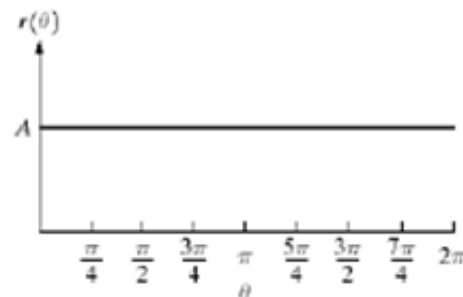
Un algoritm pentru compararea a 2 contururi reprezentate prin curbe ψ -s [1]:

1. Se foloseste reprezentarea ψ -s a fiecarui contur normalizata cu lungimea.
3. Pentru fiecare punct de start posibil al celei de a 2-a curbe (fiecare punct al celei de a 2-a curbe), se compara prima curba cu a 2-a curba incepand din acel punct de start.
4. Se selecteaza cea mai buna dintre potriviri: aceasta da punctul de start pentru a 2-a curba.
5. Se compara cele 2 curbe pornind din punctul de start determinat pentru a 2-a curba.

Descriptori de forme globali de tip semnatura (5)

❖ Curba r - s – reprezentarea radiala

- este similara curbei ψ - s .
- r este distanta de la un punct al conturului, P, la centroidul figurii.
- s este lungimea arcului de curba din punctul de start pana in punctul P, reprezentata prin unghiul arcului de curba parcurs din punctul de start pana in punctul P.
- In punctul de start $\theta=0$



- Invarianța la rotație se poate obtine printr-o translație, astfel incat valoarea maxima a functiei sa fie în $\theta=0$, dacă exista o astfel de valoare unica.
- Invarianța la scalare: se mapeaza toate valorile functiei in intervalul $[0,1]$

Recunoasterea formelor folosind sabloane(1)

- **Șablon:** o structura de imagine cunoscuta, de dimensiuni relativ mici (ex. 64x64 pixeli)
- Recunoasterea consta in **calculul potrivirii** (matching) dintre șablon si imaginea peste care este suprapus.

- **Măsura de similaritate** cea mai cunoscuta: **cross-correlation**

Pentru un șablon cu lățimea $2w+1$ si înălțimea $2h+1$:

$$C(x,y) = \sum \sum I(x+i,y+j) * S(i, j) , -w \leq i \leq w, -h \leq j \leq h$$

- I este imaginea iar S este șablonul cu centrul suprapus peste pixelul (x,y)
- Suma este calculata pe zona de imagine acoperita de șablon
- $C(x,y)$ este o măsura a similarității dintre imaginea acoperita de șablon si forma descrisa de șablon, poziționat in (x,y)
- $C(x,y)$ are valoarea maxima acolo unde imaginea si șablonul se potrivesc cel mai bine

Recunoasterea formelor folosind sabloane(2)

Cross corelatia normalizata:

Cross-corelatia este sensibila la variatii ale iluminarii in $I(x,y)$ → se normalizeaza

$$C(x,y) = \frac{(\sum \sum (I(x+i,y+j) - IM(x,y)) * (S(i,j) - SM))}{(\sum \sum (I(x+i,y+j) - IM(x,y))^2 * (S(i,j) - SM))^2)^{0.5}}$$

- $IM(x,y)$ este media intensitatilor pixelilor din zona imagine peste care este suprapus sablonul
- SM este media intensitatilor pixelilor din sablon
- Metoda se utilizeaza in contexte in care sablonul se regaseste in imagine exact sau destul de apropiat; ex: recunoasterea caracterelor de text cu un singur font.
- Dezavantaj: senzitivitatea la scalare si rotatie.
- Cu o proiectare atenta a sablonului, metoda poate da rezultate bune si in aplicatii mai complexe, cum ar fi recunoasterea caracteristicilor fețelor umane (ochii, gura, nasul)

Referinte si lecturi suplimentare

1. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/boundary-rep-desc.pdf
2. https://www.math.uci.edu/icamp/summer/research_12/classification/shape_descriptions_survey_part3.pdf
3. <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.5279&rep=rep1&type=pdf>
4. <https://www.ece.uvic.ca/~aalbu/computer%20vision%202009/Lecture%2022.%20Shape%20description-contours.pdf>
5. <https://etd.lib.metu.edu.tr/upload/3/1068873/index.pdf>