Subtiere (thinning) si scheletizare(skeletonization)

Subtierea

Este o operatie prin care sunt indepartati pixeli selectati din regiunile unei imagini binare.

Operatia de subtiere foloseste un sablon binar numit *element de structurare*. Acesta este definit printr-o matrice de 3x3 elemente avand valorile 1 si 0. Originea sa este in centrul sablonului.

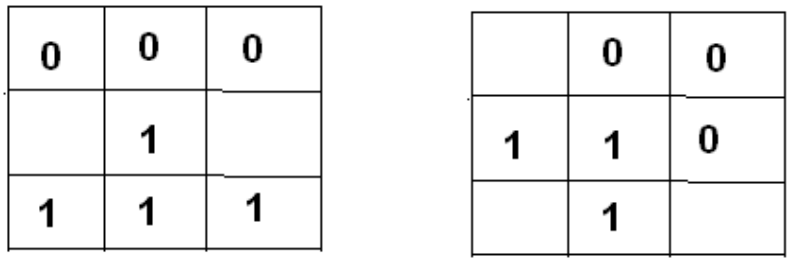
- Procesul de subtiere este iterativ.
- O iteratie consta in translatia originii sablonului peste toti pixelii imaginii si compararea elementelor sablonului cu pixelii acoperiti de el, astfel: daca toate valorile 1 ale sablonului acopera biti egali cu 1 in imagine si daca toate valorile 0 ale sablonului acopera biti egali cu 0 in imagine, atunci pixelul imagine acoperit de originea sablonului este setat la zero, adica devine pixel de fond. Altfel, nu este modificat.
- Procesul se incheie atunci cand intr-o iteratie nu se produce nici o modificare a imaginii (pana la convergenta).
- Elementul de structurare este ales in functie de scopul operatiei de subtiere, deoarece el determina situatiile in care un pixel al unei forme (=1) este transformat in pixel de fond (=0).

Subtierea se utilizeaza pentru scheletizarea regiunilor rezultate din segmentarea imaginilor dar si pentru imbunatatirea rezultatului detectoarelor de frontiere, prin reducerea latimii liniilor la un pixel, fara modificarea lungimii lor. In cel de-al 2-lea caz, subtierea se aplica imaginii binare care se obtine prin aplicarea unui prag asupra matricei marimii fronturilor.

Un algoritm simplu de subtiere a contururilor:

```
repeta
        gata =1;
        pentru fiecare pixel al imaginii
                daca este pixel de contur al unei regiuni (are cel putin un vecin-d in afara
                                                                regiunii)
                atunci
                        daca are mai mult de un vecin in regiune si prin eliminarea sa nu se
                                sparge regiunea
                        atunci
                                Elimina pixelul (se setezaza ca pixel de fond)
                                gata =0;
cat timp (!gata);
```

Algoritmul poate fi implementat cu ajutorul urmatoarelor doua sabloane si a celor obtinute prin rotatia lor cu 90 de grade (in total, 4x2 = 8 sabloane):



In fiecare iteratie se efectueaza operatia de subtiere mai intai cu sablonul din stanga, apoi cu cel din dreapta si cu celelalte 6 rezultate din rotatia lor cu 90 de grade.

Scheletizarea

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 a formelor.

Desenele compuse din linii sau caractere de text trebuie sa fie digitizate la o rezolutie suficient de mare pentru ca liniile sa nu fie intrerupte sau terminatiile lor sa se piarda. La o astfel de rezolutie este posibil ca pe unele portiuni liniile sa aiba latimea mai mare ca 2 pixeli.

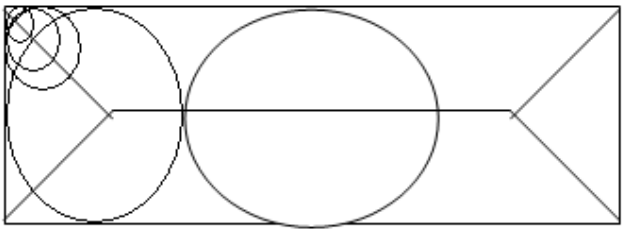
- Scheletizarea permite refacerea structurii liniare a figurii digitizate, fara distrugerea conectivitatii.

Axa mediana sau *axa schelet* a unei regiuni este definita in planul continuu (analogic) astfel:

Fie *R o regiune in planul continuu*, *C conturul regiunii* si *P un punct din R*.

Cel mai apropiat vecin al lui P pe C este un punct M cu proprietatea ca nu exista alt punct in C a carui distanta fata de P sa fie mai mica decat PM. Daca P are mai multi vecini in C cu aceasta proprietate, atunci P este un *punct de schelet al lui R*.

Din definitie rezulta ca punctele schelet sunt centre ale unor cercuri continute in intregime in R, cu proprietatea ca nu exista alte cercuri cu aceleasi centre, de raza mai mare, continute in R.



Reuniunea punctelor de schelet formeaza axa schelet sau axa mediana a lui R.
Scheletul unei regiuni este util deoarece este o reprezentare simpla si compacta a unei forme, care conserva multe dintre caracteristicile topologice si de dimensiune ale formei. Se utilizeaza pentru recunoasterea formelor(de ex. a caracterelor de text, a scrisului manual) estimand caracteristici ale formelor cum ar fi: lungimea si latimea formei, numarul de puncte de jonctiune (puncte in care se intalnesc cel putin 3 linii) si altele.

Transpunerea conceptului de axa mediana in planul discret este dificila deoarece drumul dintre 2 pixeli nu este unic si nu poate fi aplicata definitia distantei euclidiene dintre 2 pixeli.

- Scheletizarea este procesul de determinare a axei schelet a unei forme (regiuni).

- Intrarea acestui proces este o imagine binara in care pixelii de fond au valoarea zero iar pixelii regiunilor au valoarea 1.
- Iesirea procesului de scheletizare este o imagine binara in care numai pixelii axelor schelet au valoarea 1.

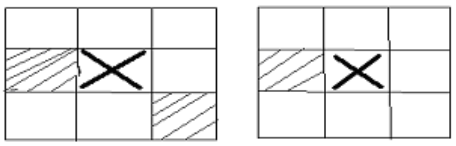
Nu exista o definitie matematica a procesului de scheletizare dar exista numerosi algoritmi. Majoritatea utilizeaza operatori de subtiere prin care se erodeaza treptat marginile regiunilor.

Algoritmi de scheletizare prin subtiere

Scheletizarea prin subtiere poate fi definita euristic ca *o succesiune de erodari ale marginilor unei forme pana cand se obtine scheletul formei***.** Algoritmii de subtiere sunt algoritmi iterativi care indeparteaza pixelii de contur, adica pixelii de tranzitie 0 $\rightarrow$ 1 intr-o imagine binara. Totodata, pixelii sunt indepartati astfel incat conectivitatea formei sa fie conservata iar terminatiile formei sa nu fie scurtate.

Algoritmii de subtiere trebuie sa satisfaca urmatoarele **constrangeri**:

1. *Sa conserve conectivitatea formei***;** pentru aceasta, nu sunt eliminati pixelii de contur care ar putea cauza discontinuitati, de exemplu, pixelul marcat in figura (a).



a b

2. *Sa nu scurteze terminatiile formelor alungite***;** nu trebuie eliminati pixeli ca cel din figura (b)

In fiecare iteratie se viziteaza o singura data fiecare pixel al imaginii, verificandu-se daca poate fi indepartat, cu satisfacerea celor 2 constrangeri.

Pentru aceasta se cerceteaza vecinatatea de 8 pixeli a fiecarui pixel interior regiunii (P=1). Fie urmatoarea notatie:

P8	P1	P2
P7	P	P3
P6	P5	P4

- *se numara pixelii interiori din vecinatatea lui P (inreg);*
- *daca inreg ≤ 2, P nu poate fi eliminat din regiune caci el apartine unei terminatii sau conecteaza 2 parti ale unei forme;*
- *daca inreg = 8, P nu poate fi indepartat, deoarece aceasta ar conduce la erodarea formei;*
- *daca 2< inreg<8:*
  - Se numara tranzitiile 0 → 1 din vecinatatea lui P: P8, P1, P2, P3, P4, P5, P6, P7,P8
  - Daca nrtranz = 1 inseamna ca in fereastra de 3x3 pixeli exista o singura componenta conectata. In acest caz se poate indeparta pixelul din centru caci indepartarea sa nu afecteaza conectivitatea locala a celorlalti pixeli din fereastra.

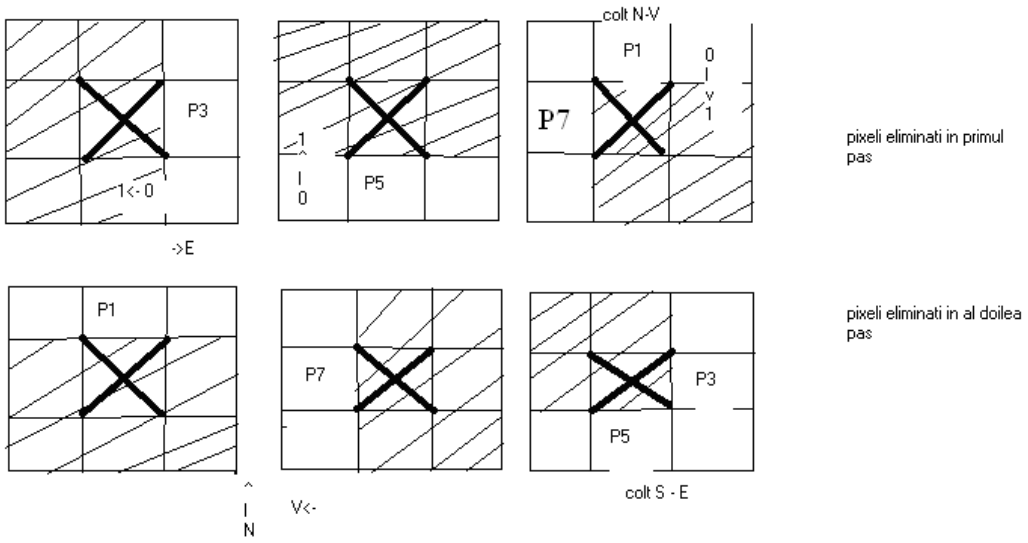
Algoritmul se termina atunci cand in iteratia curenta nu s-a mai indepartat nici un pixel.

```
void subtiere1 ( imagine x,  int N1, int M1, int N2, int M2 )
{
  // In fiecare iteratie imaginea este parcursa o singura data;
  // algoritmul de subtiere intr-un singur pas
  int I, j, k, l, inreg, tranz, gata, m, y [9];

do
```

```
        {
        gata = 1;
        for ( k = N1 + 1; k<N2 – 1; k ++ )
                for ( l = M1 + 1; l < M2 – 1; l ++ )
                        if ( x[k] [l] = = 1 )// pixel in regiune
                          {
                            // numara pixelii = 1 din vecinatatea de 3 x 3 (pixeli interiori)
                            inreg = 0;
                            for ( i = -1; i <=1; i++ )
                                 for ( j = -1; j <=1; j++ )
                                    if ( x[k+i] [l+j] = = 1) inreg++;
                                 if(inreg>2 && inreg<8)
                                          { // calculeaza numarul de tranzitii;
                                            m = 0;
                                            for (i = -1; i <=1; i++ )
                                                    y[m++] = x[k-1][l+i];
                                            y[3] = x[k][l+1];
                                            y[4] = x[k+1][l+1];
                                            y[5] = x[k+1][l];
                                            for ( i = 1, m = 6; i >= -1; i-- )
                                                    y[m++] = x[k+i][l-1];
                                            tranz = 0;
                                            for ( m=0; m<=7; m++ )
                                                    if ( y[m] = = 0 && y[m+1] ==1 ) tranz++;
                                            //daca numarul de tranzatii este 1, se elimina
                                             //pixelul curent;
                                            if ( tranz = =1 ) { x[k][l] = 0; gata = 0; }
                                          }
                                 }

  } while (!gata);
}
```

Dezavantajul algoritmului este ca *nu subtiaza regiunile simetric*. Daca imaginea este parcursa pe randuri si de la stanga la dreapta liniile regiunii subtiate sunt localizate in partea de sud -– est a marginii obiectului, deoarece pixelii de frontiera din partile de nord – vest sunt eliminati primii. De aceea, *rezultatul produs de algoritm nu este satisfacator atunci cand regiunile din imagini sunt relativ mari si convexe.*

Se poate obtine o subtiere simetrica aplicand o varianta a algoritmului care opereaza in doi pasi, alternativi.



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

Notam cu :

- N(P0) numarul de pixeli interiori din vecinatatea de 3 x 3 a pixelului curent, P0
- T (P0) numarul de tranzitii 0 $\rightarrow$ 1 in secventa de pixeli care formeaza periferia ferestrei: p1, p2, p3, …, p8,p1

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

Pas 1 :

(2 < N (P0) < 8) && T (P0) == 1 && (P3 == 0 || P5 == 0 || (P1 == 0 && P7 == 0))

Pas2 :

(2 < N (P0) < 8) && T (P0) == 1 && (P1 == 0 || P7 == 0 || (P3 == 0 && P5 == 0))

In fiecare pas, pixelii care indeplinesc conditia de a fi eliminati sunt marcati pentru eliminare si numai dupa parcurgerea intregii imagini sunt eliminati.

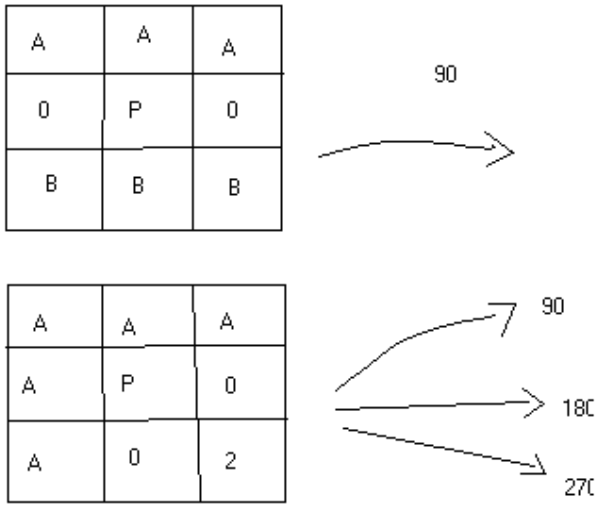
```
void subtiere2 (imagine x, int N1, int M1, int N2, int M2)
{ int I, j, k, l, inreg, tranz, gata, y [9], pas = 0;
  imagine z; // buffer de marcare pixeli de eliminat
 * alocare memorie pentru z;
do
        { gata = 1;
          pas = (pas + 1)%2;
          // sterge buffer-ul de marcare
          for ( k = 0; k < N2 – N1 - 1; k ++ )
                for ( l = 0; l < M2 – M1 - 1; l ++ )
                        z[k][l] = 0;
          for ( k = N1 + 1; k<N2 – 1; k ++ )
             for ( l = M1 + 1; l < M2 – 1; l ++ )
                if ( x[k] [l] = = 1 )
                  {
                    inreg = 0;
                    for (I = -1; I<=1; I++ )
                         for (j = -1; j<=1; j++ )
                           if ( x[k+i][l+j] == 1 ) inreg++;
                    if(inreg>2 && inreg<8)
                    {   * formeaza vectorul y (cu pixelii perimetrului ferestrei);
                       * determina numarul de tranzatii  0 → 1
                       if ( tranz = =1 )
                            if (pas == 0 )
                                if( y[3] == 0 || y[5] == 0 || (y[1] == 0 &&   y[7] == 0))
                                        { z[k-N1-1] [l-M1-1] = 1; gata = 0;}
```

```
                        else // pas =1
                            if ( y[1] == 0 || y[7] == 0 || (y[3] == 0 &&   y[5] == 0))
                                { z[k-N1-1] [l-M1-1] = 1; gata = 0;}
                    }    //if inreg
                }
        //se elimina din regiune pixelii marcati
        for ( k = N1 + 1; k< N2 – 1; k++ )
            for ( l = M1 + 1; k< M2 – 1; l++ )
                if ( z[k-N1-1] [l –M1-1] == 1 ) x[k][l] = 0;
} while (!gata);
*dealocare matrice z
}
```

Algoritm de scheletizare bazat pe notiunea de pixel multiplu

Scheletul unei regiuni este o regiune liniara deci o regiune alcatuita numai din pixeli multipli.

Algoritmii clasici de scheletizare considera drept pixeli schelet pixelii multiplii care satisfac conditia (1) din definitia pixelului multiplu (vezi “Regiuni liniare in spatiul discret”), adica satisfac unul dintre cele 6 sabloane.



Intr-un grup de pixeli marcati cu aceeasi litera cel putin unul are valoarea diferita de zero.

Imaginea de intrare este binara. Se considera ca pixeli de contur pixelii care au un vecin-d egal cu zero. Numai pixelii de contur pot fi pixeli de schelet.

In fiecare iteratie se parcurge imaginea marcand pixelii de contur si pixelii de schelet, pe toate cele 4 laturi ale fiecarei regiuni existente in imagine. Dupa ce toata imaginea a fost traversata, pixelii marcati ca fiind de contur sunt eliminati (li se da valoarea zero).

In descrierea algoritmului se apeleaza functia ***sablon*** care primeste coordonatele unui pixel si indicele unuia dintre cele 6 sabloane. Functia intoarce 1 daca vecinatatea pixelului satisface sablonul.

Exemplu:

0	2	0		A	A	A
0	P	0	satisface sablonul	0	P	0
1	0	0		B	B	B

```
void subtiere3 (imagine x, int N!, int M1, int N2, int M2)
{ int k, l, gata, D;
  do
    { gata = 1;
      for ( D = 0; D<7; D+=2 ) // D = 0, 2, 4, 6
      {
        for ( k = N1 + 1; k< N2 – 1; k++ )
          for ( l = M1 + 1; k< M2 – 1; l++ )
             if (x[k][l] == 1 &&  vecin (x, k, l, D) == 0)  // pixel de contur
                 { for ( i = 0, j = 0; i < 6 && !j; i++ )
                          j= sablon(x, k, l, i);
                   if ( j) // pixel schelet
                      x[k][l] = 3; // se marcheaza ca pixel schelet
                   else { x[k][l] = 2;     // pixel contur
                           gata = 0; // mai sunt pixeli de contur ne-eliminati
                        }
                 }
       }// for D

     //se elimina pixelii de contur
     if ( !gata )
         for ( k = N1 + 1; k<N2 – 1; k ++ )
             for ( l = M1 + 1; l < M2 – 1; l ++ )
                 if ( x[k] [l] = = 2 )
                          x[k][l] = 0;
} while (!gata)

}
```