MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-11278-NTC/364

SKELETONIZATION ALGORITHM IMPLEMENTATION IN C++

Arley Ferreira de Souza

INPE
São José dos Campos
2005

Here, we present the C++ Builder 5 implementation of the expansion by dilation and the skeletonization algorithm suggested in the paper “An Easy Skeletonization Algorithm Implementation”.

The input binary image in both the cases is a bitmap with 8 bits per pixel.

```
void __fastcall Expansion::ExpansionByDilatation8bits(Graphics::TBitmap *BitMap)
{
 unsigned char *LineAux, *Line, *LineAbove, *LineBelow;
 int x, y, i, j;

//Step (1) - it creates an image with domain size (2m + 1) x (2n + 1)
 Graphics::TBitmap *BitMapAux = new Graphics::TBitmap();
 BitMapAux->Assign(BitMap);
 BitMap->Width = 2 * BitMap->Width + 1;
 BitMap->Height = 2 * BitMap->Height + 1;

//Step (2) - it fills out the image with the background color
 for(y = 0; y < BitMap->Height; y++)
 {
  Line = (unsigned char *) BitMap->ScanLine[y];

  for(x = 0; x < BitMap->Width; x++)
    Line[x] = 0x0;
 }

//Step (3) - it puts the objects pixels in the expanded image
 for(y = 0; y < BitMapAux->Height; y++)
 {
  j = 2 * y + 1;

  LineAbove = (unsigned char *) BitMap->ScanLine[j - 1];
  Line = (unsigned char *) BitMap->ScanLine[j];
  LineBelow = (unsigned char *) BitMap->ScanLine[j + 1];

  LineAux = (unsigned char *) BitMapAux->ScanLine[y];

  for(x = 0; x < BitMapAux->Width; x++)
    if(LineAux[x] != 0x0)
    {
     i = 2 * x + 1;

     LineAbove[i - 1] = LineAux[x];
     LineAbove[i] = LineAux[x];
     LineAbove[i + 1] = LineAux[x];
     Line[i - 1] = LineAux[x];
     Line[i] = LineAux[x];
     Line[i + 1] = LineAux[x];
     LineBelow[i - 1] = LineAux[x];
     LineBelow[i] = LineAux[x];
     LineBelow[i + 1] = LineAux[x];
    }
 }

 delete BitMapAux;
}
```

```
void __fastcall Skeletonization::Skeleton8bits(Graphics::TBitmap *BitMap)
{
 try{
    int min, x, y, X = BitMap->Width - 1, Y = BitMap->Height - 1;
    unsigned char *Line;

//it creates the matrix M
    int **M = new int * [BitMap->Height];   //to allocate the lines
    for(int y = 0; y < BitMap->Height; y++)
      M[y] = new int [BitMap->Width]; //to allocate the columns

//it puts the image in a matrix
    for(y = 0; y < BitMap->Height; y++)
    {
     Line = (unsigned char *) BitMap->ScanLine[y];
     for(x = 0; x < BitMap->Width; x++)
       M[y][x] = (Line[x] != 0x0)? 1 : 0;
    }

//Pseudo-code B - it implements the Chessboard distance transformation
    for(y = 1; y < Y; y++) //process in raster mode
      for(x = 1; x < X; x++)
        if(M[y][x] == 1)
        {
         min = M[y-1][x-1];
         if(M[y-1][x] < min)   min = M[y-1][x];
         if(M[y-1][x+1] < min) min = M[y-1][x+1];
         if(M[y][x-1] < min)   min = M[y][x-1];
         M[y][x] = min + 1;
        }

    for(y = Y-1; y > 0; y--) //process in anti-raster mode
      for(x = X-1; x > 0; x--)
        if(M[y][x] > 1)
        {
         min = M[y][x+1];
         if(M[y+1][x-1] < min) min = M[y+1][x-1];
         if(M[y+1][x] < min)   min = M[y+1][x];
         if(M[y+1][x+1] < min) min = M[y+1][x+1];
         if(min+1 < M[y][x])   M[y][x] = min + 1;
        }

//Pseudo-code C - it implements the skeletonization algorithm
    for(y = 1; y < Y; y++) //process in raster mode
      for(x = 1; x < X; x++)
        if(M[y][x] > 0 && abs(M[y-1][x]) <= M[y][x] && abs(M[y][x-1]) <= M[y][x] &&
           (M[y+1][x] <= M[y][x] || M[y-1][x] < 0) && (M[y][x+1] <= M[y][x] || M[y][x-1] < 0))
          M[y][x] *= -1;

    for(y = Y-1; y > 0; y--) //process in anti-raster mode
      for(x = X-1; x > 0; x--)
        if(M[y][x] > 0 && ((M[y+1][x] < 0 && abs(M[y-1][x]) > M[y][x]) ||
           (M[y][x+1] < 0 && abs(M[y][x-1]) > M[y][x])))
          M[y][x] *= -1;

//Pseudo-code D - it implements the skeleton filter
    int transitions, greater;
    for(y = 1; y < Y; y++) //process in raster mode
```

```
      for(x = 1; x < X; x++)
        if(M[y][x] < 0)
        {
         greater = 0;
         transitions = 0;

         if(M[y-1][x-1] < M[y][x]) greater = 1;
         if(M[y-1][x-1] < 0 && M[y-1][x] >= 0) transitions++;
         if(M[y-1][x] < M[y][x])   greater = 1;
         if(M[y-1][x] < 0 && M[y-1][x+1] >= 0) transitions++;
         if(M[y-1][x+1] < M[y][x]) greater = 1;
         if(M[y-1][x+1] < 0 && M[y][x+1] >= 0) transitions++;
         if(M[y][x-1] < M[y][x])   greater = 1;
         if(M[y][x+1] < 0 && M[y+1][x+1] >= 0) transitions++;
         if(M[y][x+1] < M[y][x])   greater = 1;
         if(M[y+1][x+1] < 0 && M[y+1][x] >= 0) transitions++;
         if(M[y+1][x-1] < M[y][x]) greater = 1;
         if(M[y+1][x] < 0 && M[y+1][x-1] >= 0) transitions++;
         if(M[y+1][x] < M[y][x])   greater = 1;
         if(M[y+1][x-1] < 0 && M[y][x-1] >= 0) transitions++;
         if(M[y+1][x+1] < M[y][x]) greater = 1;
         if(M[y][x-1] < 0 && M[y-1][x-1] >= 0) transitions++;

         if(greater == 1 && transitions < 2) M[y][x] *= -1;
        }

      for(y = Y-1; y > 0; y--) //process in anti-raster mode
        for(x = X-1; x > 0; x--)
        if(M[y][x] < 0)
        {
         greater = 0;
         transitions = 0;

         if(M[y-1][x-1] < M[y][x]) greater = 1;
         if(M[y-1][x-1] < 0 && M[y-1][x] >= 0) transitions++;
         if(M[y-1][x] < M[y][x])   greater = 1;
         if(M[y-1][x] < 0 && M[y-1][x+1] >= 0) transitions++;
         if(M[y-1][x+1] < M[y][x]) greater = 1;
         if(M[y-1][x+1] < 0 && M[y][x+1] >= 0) transitions++;
         if(M[y][x-1] < M[y][x])   greater = 1;
         if(M[y][x+1] < 0 && M[y+1][x+1] >= 0) transitions++;
         if(M[y][x+1] < M[y][x])   greater = 1;
         if(M[y+1][x+1] < 0 && M[y+1][x] >= 0) transitions++;
         if(M[y+1][x-1] < M[y][x]) greater = 1;
         if(M[y+1][x] < 0 && M[y+1][x-1] >= 0) transitions++;
         if(M[y+1][x] < M[y][x])   greater = 1;
         if(M[y+1][x-1] < 0 && M[y][x-1] >= 0) transitions++;
         if(M[y+1][x+1] < M[y][x]) greater = 1;
         if(M[y][x-1] < 0 && M[y-1][x-1] >= 0) transitions++;

         if(greater == 1 && transitions < 2) M[y][x] *= -1;
        }

//it puts the skeleton points in the image
      for(y = 1; y < Y; y++)
      {
       Line = (unsigned char *) BitMap->ScanLine[y];
       for(x = 1; x < X; x++)
         if(M[y][x] < 0)
           Line[x] = 0xbb;
```

```
    }

//it deletes the matrix M
    for(int y = 0; y < BitMap->Height; y++)
      delete[] M[y];    //to delete the columns

    delete[] M;         //to delete the lines
   }
 catch(...){}
}
```