Segmentation and Image Analysis - Practical work 5

The aim of this work is to analyze the geometry of a digital contour : normal estimation, curvature estimation, extraction of salient points.

Images and source code (2D case)

— 2D test images : `disk.pgm`, `blob.pgm`, `star1.pgm`, `star2.pgm`
— Java classes for analyzing a 2D digital contour : `FreemanCode.java`, `Path.java` (to be filled), `DigitalLineSegment.java` and `DigitalTangent.java`
— An `ImageJ` plugin for visualizing a 2D digital contour : `Contour_Analysis.java`.

FreemanCode

This class corresponds to the Freeman code of a 4-connected path. It contains the first point of the path and the sequence of elementary directions defining it. The available methods are :

1. The constructor `FreemanCode(ImageProcessor ip, int gl)` processes a gray level image and builds the Freeman code of the interpixel boundary of the first 8-connected component of gray level `gl` reached by scanning the image from left to right and from top to bottom.
2. `int getLength()` : length of the path (number of elementary directions of the code).
3. `Point getP0()` : first point of the path.
4. `int getDeltaX (int ind),int getDeltaX (int ind)` : variations of x and y for a step of the path. The index `ind` must verify $0 \leq$ `ind` $<$ `length`.
5. `int getCode (int ind, int shift)` : elementary direction on a closed path.

Path

This class contains the sequence of integer points of a 4-connected path (typically the interpixel boundary of an 8-connected region in the 2D grid). It contains the following internal class :

```
// in the case of an interpixel boundary (_x,_y) stands for the
half-integer point (_x+1/2, _y+1/2)
// The x-axis is oriented to the right, the y-axis is oriented upwards
private class ContourPoint{
  int _x;
  int _y;

// tangent vector
  int _tx;
  int _ty;
}
```

The class `ContourPoint` will be enriched with an additional local information : the curvature estimated at this point.

The class `Path` contains the following methods :

1. The constructor `Path(FreemanCode)` builds the sequence of points of a path given its Freeman code and computes the tangent vector at each point.
2. `int getLength()` : length of the path (number of points).
3. `int getContourX(int ind), int getContourY(int ind)` return the abscissa and ordinate of the point of index `ind` of the contour.

4. `int getContourTX(int ind), int getContourTY(int ind)` return the two components of the tangent vector computed at the point of index `ind` of the contour.
5. `int getPMin(), int getPMax()` return the two opposite corners of the contour bounding box.

Classes for the tangent computation

The class `DigitalLineSegment` represents a segment of a 4-connected digital line segment in the first quadrant (arithmetic definition).

The class `DigitalTangent` corresponds to a linear part of a digital contour, centered on a contour point.

Curvature computation

Fill in the class `Path.java` by adding the curvature computation at each contour point. The curvature will be computed by convolving the tangent orientation with the derivative of a Gaussian function.

$$\kappa(i) = \widehat{\theta}(i) * G'_\sigma \qquad G'_\sigma(x) = \left(\frac{-x}{\sigma^3\sqrt{2\Pi}} e^{\frac{-x^2}{2\sigma^2}}\right)$$

The size of the computation window is $m = 3\sigma$ with $\sigma = 3$ for example.

Reminder : convolution $f(i) * g(i) = \sum_{k=i-m}^{k=i+m} f(k)g(i-k)$

Use `gnuplot` to visualize the result. If the file `data` contains the curvature values of a contour in the following way :

```
0 0.209415
1 0.210887
2 0.197836
3 0.174762
4 0.147319
...
```

the curvature profile can be displayed with :

```
echo "plot 'data' with lines" | gnuplot -persist
```

Salient points

Use the curvature profile to detect the salient points of a 2D contour : they correspond to local extrema of the curvature profile. An option is to select the local extrema among the points with a curvature greater than a predefined threshold (absolute value).

Add the visualization of the salient points in the plugin `Contour_Analysis.java`.

Curvature computation on the surface of a 3D digital object

Compute the curvature at each boundary point of a 3D sphere with the "integral invariants" method. Evaluate the precision of the results (take into account the radius of the sphere and the radius of integration).