TD7: DSS Covering and Normal Vector Estimation

In this TP, the idea is to implement a multigrid convergent normal vector estimator. We assume that you have:

- DGtal and DGtalSkel tools updated and running (TP5)
- A function that constructs the Gauss digitalization of an Euclidean disc (TP5)
- A function that tracks the border of a DigitalSet and construct a `std::vector<Point>` of border points (TP6)

Exercise 1 Digital Tangent

By using the example file `DSSExample.cpp` and `http://libdgtal.org/doc/stable/classDGtal_1_1ArithmeticalDSS.html`, we first focus on the decomposition of a contour (`std::vector<Point> sequence`) into maximal DSS. The functions are the following:

- Init a DSS at the "begin()" sequence iterator
- While the `DSS.extendForward()` returns true, we increment an iterator copy "iter" of the `sequence.begin()`
- When the DSS recognition fails, we init a new DSS at "iter" and repeat the above process until "iter" reaches the end of the sequence (`sequence.end()`)

Question: Implement such DSS decomposition and visualize each DSS segments as shown in `DSSExample.cpp`

We want to implement the digital tangent estimator, which works in the following way:

- Given position `iter` on the contour, we init a DSS segment at this point and extend the DSS at both sides (`extendForward()` and `extendBackward()`)
- Returns the DSS slope (`getA()` and `getB()`).

Remarks: The DSS is defined by $\mu \leq ax - by < \mu + max(|a|, |b|)$. At this point, even if the sequence encodes a closed contour, we suppose that there nothing "backward" the `sequence.begin()` and nothing after the `sequence.end()`.

Questions:

- Implement this estimator.
- If you repeat such process at each position on the sequence, what would be the computational cost to estimate all digital tangents?

Using the above algorithm, the estimations near the sequence extremities are biased and do not take into account the fact the contour is closed (i.e. tangent near the `sequence.begin()` could contain points near the `sequence.end()`. To handle extremities, we can use the concept of `Circulator` instead of `Iterator`.

Questions:

- Have a look to the `DSSCirculatorExample.cpp` in the `DGtalSkel` folder where we give you an example of Circulator construction in DGtal and its use to define DSS on circular contours.
- Update your code to be valid on closed contours.

Remarks:

- To display an arrow $(p, p + dp)$ using Board2D, you can use the following `draw` function:

```
Board2D board;
RealPoint p(0.0,0.4), dp(0.33,0.66);
Display2DFactory::draw( board , dp , p);
```

- Be careful `RealPoint` (with "double" coordinates) differs from `Point` (Integer coordinates)
- Dereferencing an iterator ("*iter") of std::vector returns a Point.

Exercise 2 Maximal DSS Covering (from TP6)

In this exercise, we implement the maximal DSS covering of a contour.

The algorithm is the following:

- We init a DSS (starting from `sequence.begin()`) and extend it in the "forward" direction. When the DSS recongnition stops, we have out first maximal DSS.
- To get the other ones, remove one point at the left of the DSS (`dss.retractBackward()`) and extend this new DSS forward.
- We repeat this process until reaching the end of the sequence.

Questions:

- If both `extend` and `retract` are in $O(1)$, what is the complexity of the maximal covering algorithm?
- Implement this algorithm and display all the maximal DSS (cf `DSSExample.cpp`).
- When computing the maximal DSS, compute the max, min and mean lenght (using the l_∞ distance, between `getBackPoint()` and `getFrontPoint()` DSS points).
- Create a small executable parametrized by a grid step h which digitizes a disk at step h and returns the max/min/mean values. Then, using a small script and gnuplot, display the behavior of this quantities when $h \to 0$. Using logscale mode in gnuplot[1], can you experimentally observed the theoretical bounds on maximal DSS we discussed during the lecture ?

[1] `set logscale x` and `set logscale y`