TD9: Homotopic Skeletons

In this TP, the idea is to implement various thinning algorithms preserving the homotopy of the input set. First, go to the `DGtalSkel` git folder and open the `PGMObject.cpp`. This file illustrates the definition of `Object` in DGtal: such class is a wrapper around a `DGtalSet` adding topological information (Jordan pair). Hence, `Object` instances have topological features such as simple point computation. In this file, we also illustrate the PGM loader in DGtal (already described in the DM).

Exercise 1 Ultimate Homotopic Thinning

Question Implement the ultimate homotopic thinning algorithm as described in the lecture (Slide 58[1]). You would have to consider the following data structures: `std::queue<Point>` or `std::set<Point>`. Accessing and iterating on such `std::` containers can be done as follows

```
std::queue<Point> myQueue;
...
myQueue.push( Point(2,3) );  //insert a point
Point p = myQueue.front();   //get the point at "front"
myQueue.pop();               //remove the "front" point
for(std::queue<Point>::const_iterator it = myQueue.begin(), itend =
    myQueue.end(); it != itend; ++it)
   // (*it) is a point of the container
```

Question Test the thinning algorithm for the (4,8)- and (8,4)-adjacencies on the given image examples

Exercise 2 Anchor Points, Branches and Skeleton pruning

Question Implement the following Anchor point definition: p is an anchor point if it has only one component in X. Update the previous thinning algorithm to take into account anchor points (cf Slide 59).

Question Once the thinning algorithm stops, we would like to classify the topology of remaining point. Generate a board a output of the skeleton with different colors in order to distinguish extremities, junctions and other points (definitions of such points are quite straightforward).

Question Similarly, we can define a branch as the set of pixels between an extremity and a junction. Parts of the skeleton between two junctions are thus not considered in this definition. Propose an algorithm to extract all branches of a skeleton (and display each branch with different color for instance).

Question A first approach to simplify the skeleton is to perform small branch removal. Implement the following pruning algorithm: for each branch, we count its number of pixels and we remove the branch if the number is less than a given threshold. Experiment this skeleton pruning on the given image examples. Are these skeletons more stable within image classes ?

[1] `http://liris.cnrs.fr/david.coeurjolly/cours/ENS2012/html/slides/c-gd-volumique.html#58`