Project: "Recognition of digital circle"

Introduction

The objective of this project is to implement a convex programming based algorithm for the recognition of digital circles.

We expect from you:

- A short report with answers to the "formal" questions and a description of your implementation choices and results.
- A C++ project (`CMakeLists.txt` plus several **commented** cpp program files).

1 Digital disk

Let $I \subset \mathbb{Z}^2$ be a rectangular digital image. Let $Z \subset I$ be a digital set and $\bar{Z} = I \setminus Z$ be its complement set.

A digital set Z is a *digital disk* if and only if there exists a circle of center $\omega \in \mathbb{R}^2$ and of radius $\rho \in \mathbb{R}$ such that:

$$\begin{cases} \forall z \in Z, \ \|z - \omega\|^2 \leq \rho^2 \\ \forall z \in \bar{Z}, \ \|z - \omega\|^2 \geq \rho^2 \end{cases} \tag{1}$$

Question 1 *Let us consider the Gauss digitization of a convex shape $X \subset \mathbb{R}^2$ at gridstep h:*

$$Dig(X, h) = X \cap (h \cdot \mathbb{Z}^2).$$

Show that if X is an Euclidean disk, then $Dig(X, h)$ is a digital disk. Show however that the converse is not true.

Question 2 *There exists a unique circle passing by three digital points in general position. Show that we can test whether another digital point lies INSIDE, ON or OUTSIDE such a circle with integer-only computations and without explicitly computing its center and radius. You may have a look at "in-circle test", broadly used in computational geometry, e.g. to compute the Delaunay triangulation of a point set.*

Question 3 *Implement a function that checks whether two given digital sets are separated by a given circle passing by three digital points or not.*

2 Convex programming

Let us now consider algorithm 1. It is a randomized and recursive algorithm that checks whether two point sets are separable by a circle in expected linear-time (see [Wel91] for instance).

Instead of having two point sets, we consider below only one set of points, denoted by S. Points of S are either marked as $-$ (inner points) or marked as $+$ (outer points).

The idea of algorithm 1 consists in removing a random point p from S and recursively compute a separating circle of the remaining points.

- if p is an inner points (resp. outer point) and it is located INSIDE (resp. OUTSIDE) or ON the returned separating circle C, ie. C is separating for $S \cup \{p\}$, there is nothing else to do.

- otherwise,

    1. either the two input sets are not circularly separable at all: C is undefined.
    2. or there exists a separating circle passing by p, which is recursively computed.

Algorithm 1: separatingCircle(S, B)

Input: S, the set of inner and outer points,
B, the set of points lying on the boundary of the separating circle
Result: a separating circle C (which may be undefined if the two sets are not circularly separable)

if *S is empty or B contains three points* **then**
    $C \leftarrow$ smallest circle passing by the points of B ;
else
    choose random $p \in S$;
    $C \leftarrow$ separatingCircle($S - \{p\}$, B) ;
    if *C is defined and is not separating for S* **then**
        $C \leftarrow$ separatingCircle($S - \{p\}$, $B \cup \{p\}$) ;
return C ;

Question 4 *Is there always a unique separating circle for any S ? Discuss possible cases, especially degenerate ones.*

Question 5 *Implements a class that represents a smallest circle (possibly undefined) passing by the points of a given set.*

Question 6 *Implement algorithm 1. Provide test files.*

3 Experiments

Question 7 *In order to check whether two connected digital sets Z and $\bar{Z}$ are circularly separable, it is enough to consider only the digital boundaries of Z and $\bar{Z}$. Implement a function that takes as input the common contour of Z and $\bar{Z}$ and that checks whether Z and $\bar{Z}$ are circularly separable or not. You may use* DGtal *and more precisely the class* `GridCurve` *that can return the set of boundary digital points (see e.g.* `IncidentPointsRange`*).*

Question 8 *Perform a running time analysis of your recognition function.*

- *Implement a function that constructs the contour of a disk of a given radius.*
- *Output the running time of your recognition function for disks of increasing radius.*
- *Plot the graph of the running times against the size of the contour.*
- *Do you observe the expected linear-time complexity ?*

4 Extra works

Question 9 *Modify your recognition procedure in order to have an on-line algorithm, which takes input points two by two (one belonging to the boundary of Z and one belonging to the boundary of $\bar{Z}$) and updates the current separating circle on the fly. What is the time complexity of your procedure ?*

Question 10 *Use your on-line procedure to partition a contour into digital circular arcs and to compute the whole set of maximal digital circular arcs.*

References

[Wel91] Emo Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science*, volume 555 of *Lecture Notes in Computer Science*, pages 359–370. Springer, 1991.