

TU Eindhoven

Faculteit Wiskunde en Informatica

Project description OGO 2.1 (2I025) — najaar 2009

## **Curve reconstruction**

Project coordinators: M. de Berg and K. Buchin

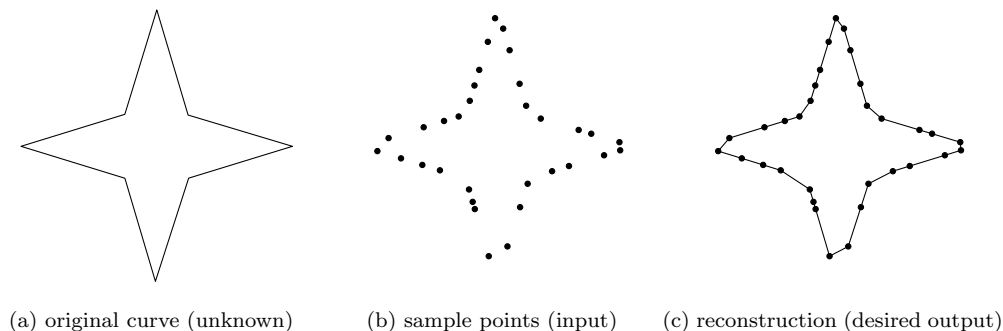


Figure 1: The curve reconstruction problem.

## 1 Introduction

In the process of creating digital models from real-world objects, an important task is to reconstruct shapes from point samples. In this OGO project we will be considering one such shape reconstruction task, namely curve reconstruction. The task is illustrated in Figure 1: The original curve (a) is not given, instead the input consists of a set of sample points from the curve (b). The goal is to determine the order in which the points lie on the curve. Connecting the points in this order yields a polygonal reconstruction of the curve (c).

Different versions of the curve reconstruction problem exist, see Figure 2. For instance, we can consider open curves (Figure 2(a)). In contrast, the curve in Figure 1 is a closed curve. An algorithm that works well on closed curves might fail on open curves. Also, instead of one original curve there might be several (Figure 2(b)), in which case there is the additional problem of identifying which points come from which curves. Furthermore, self-intersections of the curve (Figure 2(c)) make the curve reconstruction problem more difficult.

## 2 The Task

### 2.1 Problem Statement

Your task is to design and implement algorithms for reconstruction a curve or curves in the plane, that is, algorithms for finding the order, in which a given set of sample points lie on the curve(s). More specifically, you should have one algorithm for each of the following sub-tasks:

**closed:** one closed curve (non-self-intersecting)

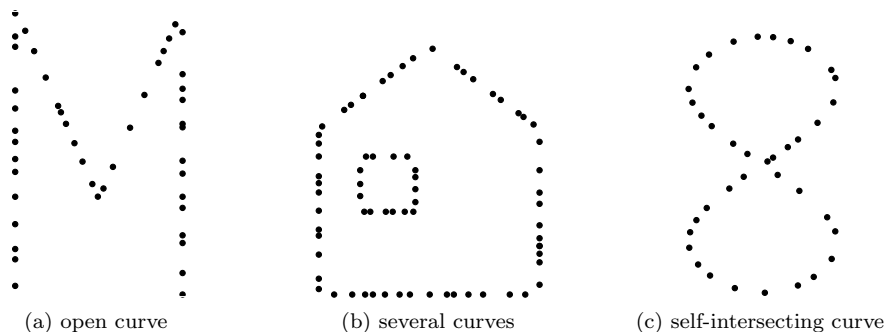


Figure 2: Variants of the curve reconstruction problem.

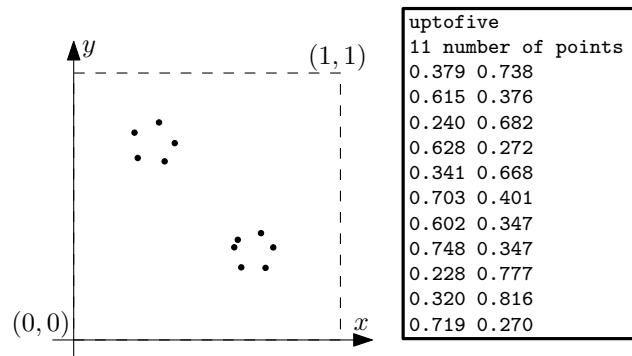


Figure 3: Input format.

**open:** one open curve (non-self-intersecting)

**selfintersecting:** one closed, self-intersecting curve

**uptofive:** up to five closed curves (non intersecting, non-self-intersecting)

An *open* curve has a different start and end point, while a *closed* curve does not have a start or end point. A curve is *self-intersecting* if it visits a location more than once.

The algorithms will be evaluated by running them on a set of test cases. The algorithms will be compared by the number of test cases that they reconstruct correctly. For each of the sub-tasks there will be the same number of test cases, and for each test case we give you the information to which sub-task it corresponds to. Early on in the project, we will provide you with a similar set of test cases, for which we will also provide you with the correct output.

Your algorithms should be efficient. The algorithms should terminate in a reasonable amount of time. For a test case with 10.000 sample points it should terminate in 5 minutes (on a computer with at least a 3GHz Intel Pentium 4 processor and 1 GB RAM).

The program to solve the tasks should be implemented in Free Pascal. It should read an input file containing one task, and write the ordered points to an output file. For testing it is important that it is called in the following way:

```
CurveRecon.exe < [input filename] > [output filename]
```

## 2.2 Input Format

The input file for a test case has the following structure. The first line contains one of the keywords specifying the sub-task, i.e., **closed**, **open**, **selfintersecting**, or **uptofive**. The next line contains a positive integer corresponding to the number of sample points, followed by the text **number of points**. Each of the following lines contains two floating-point numbers (separated by a whitespace), which correspond to the coordinates of the points. The numbers use a point as decimal separator. Figure 3 shows an example.

## 2.3 Output Format

The output file for a test case should have the following structure. The first line contains a positive integer corresponding to the number of curves followed by the text **number of curves**. The next line contains a positive integer corresponding to the number of points of the first curve, followed by the text **number of points**. Each of the following lines contains two floating-point numbers separated by a whitespace, which correspond to the coordinates of the points. The numbers use a point as decimal separator. If there are several reconstructed curves, this is followed by the same data for each of the remaining curves.

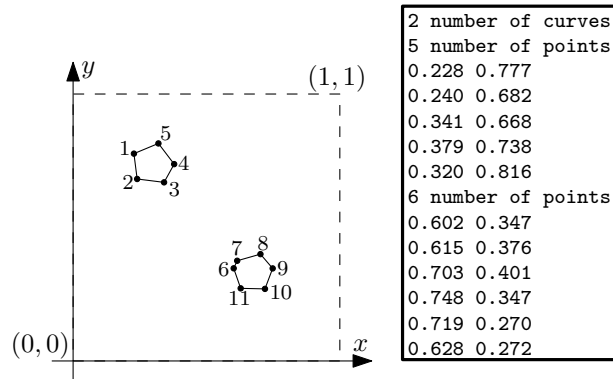


Figure 4: Output format.

The points of a curve are sorted in the order in which they lie on the curve. For an open curve the first point is the *lexicographically smaller* end point. That is, it is the end point with the smaller  $x$ -coordinate, and if both end points have the same  $x$ -coordinate, then it is the point with the smaller  $y$ -coordinate. For a closed curve the first point is the lexicographically smallest sample point of the curve. The first point has two neighbors, and the second point is the lexicographically smaller one of these neighbors. Then the remaining points are given in the corresponding order. The curves are sorted by the lexicographical order of their first points, where the curve with the lexicographically smallest first point is given first. Figure 4 shows a reconstruction of the input of Figure 3 together with the corresponding output file.