Homework #4: Planning optimal flying tours over a terrain [160 points]
Due Date: Tuesday, 9 March 1999 — no late days will be available for this homework

This is the last homework. It consists of one programming problem composed of four parts. This is a much more open-ended assignment, allowing you to combine the modeling and algorithmic tools you have learned about in CS348a in a variety of ways.

Problem 1. [160 points]

Planning flying tours over a terrain

In this problem you will experiment with algorithmic techniques for building triangulated irregular network (TIN) models of geographical terrains, and then planning smooth sight-seeing tours by flying over these terrains.

Building a terrain from elevation data

In many applications in Geographical Information Systems (GIS) dealing with surface terrains, the raw data available is a *height field*. Such a field consists of a set of surface elevation data at regularly or irregularly distributed points in the xy-plane: if n such points are specified, then we are given the triplets (x_i, y_i, h_i), $1 \leq i \leq n$, where h_i denotes the terrain elevation over the point (x_i, y_i). From these elevation data we desire to recover a surface model of what the terrain is like. A useful and commonly used model of the surface is that of a triangulated irregular network — which is just a triangular mesh whose vertices are the given points in space (x_i, y_i, h_i) such that no two triangles of the mesh have overlapping interiors when projected onto the xy-plane.

In general there will be many possible triangular meshes which fit the given data. We will provide you with a software package from the University of Illinois National Center for Supercomputing Applications which is useful for building and visualizing such TIN models of terrains. This package is called `MinMaxer`, because all the TIN generation algorithms it uses are based on the *edge-insertion* paradigm and the goal of minimizing the maximum cost associated with an edge of the TIN. Such algorithms start from an arbitrary triangulation of the corresponding planar point set $(x_i, y_i, 0)$, $1 \leq i \leq n$, and repeatedly 'improve' the TIN (inherited by the 3-D data points from this planar triangulation) by edge-insertions, until no further improvement is possible. In a single iteration, the triangulation is improved by adding a new 'good' edge (selected according to one of various criteria), deleting all edges which intersect the new edge, and retriangulating the resulting regions. More details are provided in the `MinMaxer` manual, which is available via the class home-page.

You will be asked to construct a TIN terrain model for one or more test sets of elevation data, and then use that model in the subsequent parts of the problem. Although you will not have the actual test data to be used for grading until Monday morning, 9 March 1998, other comparable data sets will be made available.

(a) (25 points)

Get `MinMaxer` to work on the test data we will provide. Among all the types of TINs that `MinMaxer` provides, select one to use for the rest of the assignment. Your selection criteria should be the appropriateness of the method for terrain modeling, and especially for the tour construction problem, as discussed below. Explain and justify your choice.

Planning a smooth sightseeing tour over a TIN surface

Suppose we have already settled on a particular TIN $\mathcal{T}$ to represent the given elevation data. A relatively small number m of the given points $(x_{i_j}, y_{i_j}, h_{i_j})$, $1 \leq j \leq m$ will be marked as being *sights of interest*, or sights for short. With each such point S_j we are given an additional elevation v_{i_j}, $v_{i_j} \geq h_{i_j}$, which denotes the optimum viewing altitude for that sight (for example, this could be related to the size of the sight). The goal in this part of the assignment is to design a smooth path for flying over the terrain $\mathcal{T}$ and viewing all of the sights S_j.

Such a tour τ is *valid* if

- τ does not intersect the terrain,
- τ is a quadratic spline (consisting of parabolic arcs) with C^1-continuity at all junctions, and
- τ crosses a cube aligned with the x, y and z axes of size $2d$, for a given parameter d, around each of the m optimal viewing positions $(x_{i_j}, y_{i_j}, v_{i_j})$, $1 \leq j \leq m$.

The sights are to be visited in the order $S_1, S_2, \ldots, S_m$. Your goal is to implement an algorithm for computing good sight-seeing tours over $\mathcal{T}$. The quality of a given valid tour τ will be judged according to several different measures, as follows:

- The tour should be *short*; you need to compute and report the euclidean length of your tour τ, to within an accuracy of 1%.
- The tour should be *smooth*: you need to compute and report the number of distinct parabolic arcs your tour uses; you also need to calculate and report the maximum curvature of your tour τ, to within an accuracy of 1%.
- The tour should be *safe:* you need to compute and report the closest vertical distance between τ and $\mathcal{T}$, to within an accuracy of 1%.

You will also be judged by how long it takes you to compute a valid tour which optimizes the above three measures. Please realize that it is impossible to optimize all these measures simultaneously. What we expect is that you will write a clear program that attempts to optimize some of the measures given above, preferably in the order specified below, and does a reasonable job on the rest.

(b) (60 points)

Write a program to compute valid sight-seeing tour which does the best you can in optimizing the above criteria. We will weigh safety the most, then length, and then smoothness.

Viewing your tour over the terrain

You need to provide a visualization of the terrain you constructed and an animation of the sight-seeing tour you computed over it. You must provide at least a wire-frame view of your terrain; a false color image is also highly recommended (for example, you can choose to color areas of high elevation white, lower elevations green, then yellow, and so on, to simulate the way actual terrains look). Mark the sights of interest on the terrain in a distinctive way. Then, for the tour animation please, provide two views (figure 1):

an observer's view: this is the view from a distant stationary observer over the terrain, watching you execute the tour, and

a pilot's view: this is the view looking straight down within a certain angular frustum, as you execute the tour.

(c) (30 points)

Provide an animation of your tour, as outlined above.

Tours for unordered sights

Suppose now that the sights S_j can be visited in any order (but each must be visited exactly once). Reimplement the part of your algorithm which finds the tour τ to take advantage of this new freedom.

(d) (45 points)

Show your new tour and outline the advantages of the method you chose. When the optimum sequence of sight visits is obvious, do you follow that sequence?

There are many possible approaches to this problem, as it is clearly related to the famous *traveling salesman problem* (TSP). For a possible approach based on *guillotine rectangular subdivisions* (a kind of BSP tree), see [Mit96, MM95].

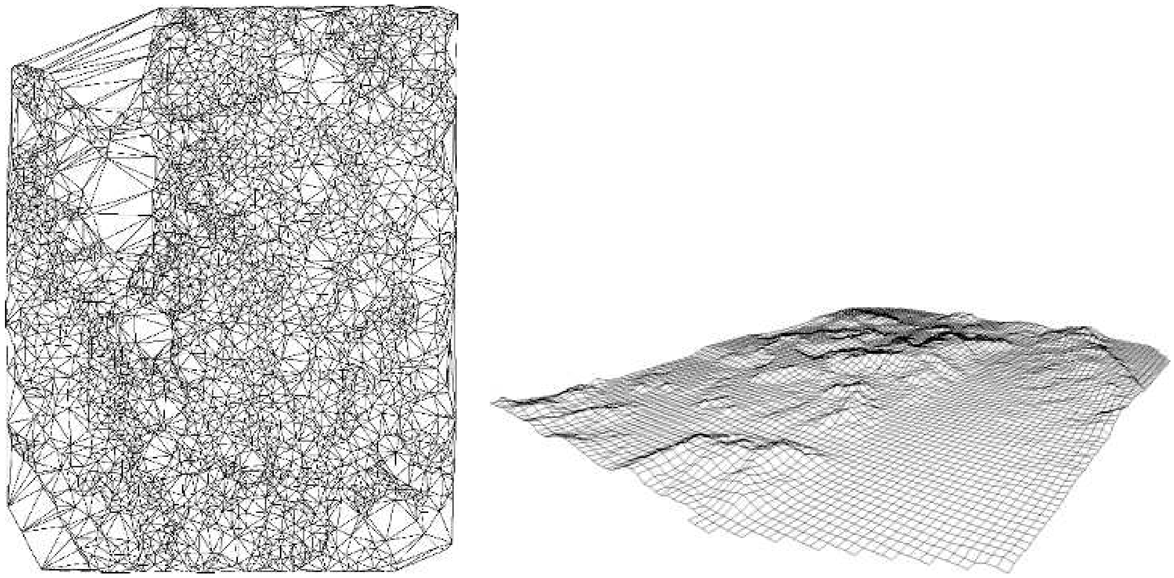

Figure 1: Pilot's and Observer's view.

What you will be given

Height-field data

The height-field data will be given to you as a file of points (one per line) in the format:

`site` x-coord y-coord z-coord

This is the format accepted by `MinMaxer`. Note: `MinMaxer` also requires that all coordinates be integers.

Tour data

Tour data will be given as a file of points (one per line) in the format:

x-coord y-coord z-coord

The x- and y-coordinates will be the same as certain points in the height-field, but the z-coordinate will be an arbitrary floating point number.

MinMaxer package

The `MinMaxer` package is installed in the directory:

```
/usr/class/cs348a/pp/minmaxer
```

The program `minmaxer`, responsible for generating triangulations of data sets, is located in the subdirectory `source.minmaxer`. Try running `minmaxer` with some sample data files, and test different triangulation algorithms. A pre-compiled binary version of `minmaxer` (placed at directory `minmaxer.bin`) contains a graphical interface (SGI only), which allows the visualization of the triangulation algorithms in progress. Documentation of the package is located in the subdirectory `doc`, containing descriptions on how to use the package, and formats for input and output files.

The triangulation functions of `MinMaxer` may also be incorporated into an application program. Internally, triangulations are stored in a quad-edge data structure, and operators that access and modify the data structure are provided in the package.

In order to implement the algorithm that computes the tour, you may find it necessary to traverse the quad-edge representation of the triangulation. In other words, you will need to write an application program that access `MinMaxer` functions, and learn how quad-edge traversal operators work. In the directory `source.triangulation` the file `triangulation.c` has an instructive procedure that outputs all triangles defined by a given quad-edge. Look at the procedure `copyGraphToListOfTriangles()`, which illustrates how the quad-edge operators are used to visit the complete triangulation.

Computing spline properties

In the directory:

```
/usr/class/cs348a/pp/farin
```

you will find the source code that comes with one of the recommended books for the class [Far93]. Some useful code is implemented there, like the *de Casteljau algorithm*, or code to compute the curvature at a given point of the curve.

What to hand in

Remember that in programming assignments you are allowed to work in teams of up to three students, and that each team need hand in only a single write up. On Monday morning, 9 March 1998, you will receive the actual test data that will be used for grading. On or before the due date, you must submit the following information:

- Final version of your source code.
- Tour generated by your program:
  - Control Polygon: coordinates of the quadratic spline control polygon.
  - Evaluation measures computed (length, curvature, safety)
- Paper write-up: Give an overview of your program, explain how your algorithm(s) work, and inform which triangulation algorithm was used by `MinMaxer` to build your TIN .

We will set up a time in the graphics lab during which you will have to demo your program to the instructor and the TA.

References

[Far93] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, 1993.

[Mit96] Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric k-MST problem. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, pages 402–408, 1996.

[MM95] C. Mata and J. S. Mitchell. Approximation algorithms for geometric tour and network design problems. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 360–369, 1995.