Polyhedral Approximation of Bivariate Functions

Per-Olof Fjällström
Department of Computer and Information Science
Linköping University
S–581 83 Linköping, Sweden

Abstract

Given a set V of n points drawn from a bivariate function and a non-negative scalar d, we want to find a smallest subset, V', of V, such that a polyhedral surface interpolating V', does not deviate more than d from any point in V. A greedy heuristic, requiring $O(n^2 \log n)$ time and $\Theta(n)$ space, is presented.

Key words: Computational geometry, Delaunay triangulation, polyhedral approximation.

1 Introduction

In many applications, e.g computer vision and geographic data processing, we need to approximate functions of two variables, where the function value is initially defined only at irregular locations. A standard method for problems of this kind is to first triangulate the locations. The value of the function at an arbitrary location is then computed by performing an interpolation within the triangle containing the location. It is widely recognized that the so-called Delaunay triangulation is good for these purposes [4,6,5].

In applications with large amounts of data available, it may also be necessary to reduce the initial set of data. Accordingly, several researchers have studied the problem of constructing so-called hierarchical surface models, which provide representations of the data at successively finer level of details. See, for example, De Floriani [1]. The problem studied in this paper is related: how to find an maximum reduction of the given data satisfying an accuracy-based criterion. A more formal description of the problem is given below.

We have a set S of n locations in the x-y plane. To each location $(x_i, y_i) \in S$ there is an associated height, z_i, the value of an underlying primitive function, $z = F(x, y)$. Observe that F is usually known only at locations in S. Let V denote the set of points (x_i, y_i, z_i), $i = 1, \ldots, n$. Let d be a non-negative scalar.

Our general goal is to find a piecewise linear function for which the vertical deviation from the points in V never exceeds d, and the representation requires as little storage space as possible. However, to limit the number of possible solutions, we will study this problem with respect to the following method of polyhedral approximation. Given a set U of points in space, let $z = F_A(U; x, y)$ denote the piecewise linear function obtained as follows:

1. Compute the Delaunay triangulation of R, $DT(R)$, where R is the projection of U onto the x-y plane.
2. Within each triangle of $DT(R)$, F_A is equal to the plane interpolating the heights at the vertices of the triangle.

If we apply this scheme to V, we may get an unnecessarily detailed approximation of F. Instead, we try to find a smallest subset V' of V such that $F_A(V'; x, y)$ never deviates more than allowed from V. In order for $F_A(V'; x, y)$ to have the same domain as $F_A(V; x, y)$, V' must contain all points (x_i, y_i, z_i) such that $(x_i, y_i) \in E(S)$, where $E(S)$ is the set of extreme locations in S. Let us call this set $E(V)$. Our problem can now be formulated as follows:

Given a set V of points drawn from a bivariate function and a non-negative scalar d, find a set $I \subseteq V - E(V)$ of minimum size such that:

$$|F_A(E(V) \cup I; x_i, y_i) - z_i| \leq d, \quad i = 1, \ldots, n \qquad (1)$$

Any set I which satisfies (1) will be called *admissible*, and an optimal set, i.e. an admissible set of minimum size, is denoted I^*. Furthermore, we say that a triangle, $\{(x_i, y_i), (x_j, y_j), (x_k, y_k)\}$, is admissible either if there are no locations in the interior of the triangle or if the heights of the interior locations do not deviate more than d from the plane interpolating the heights at the vertices.

An obvious brute force algorithm for this problem would be to first determine $E(V)$, and then try out all possible subsets of the remaining points in order of increasing size until an admissible set is found. This would require exponential time in the worst case. It is left open here whether a polynomial time algorithm exists or not. However, even if such an algorithm exists, it seems reasonable to assume that it must determine which triangles are admissible, and, for every pair of admissible triangles, if they can coexist in a Delaunay triangulation [1]. Just to represent this information, in e.g. a graph, would require $\Theta(n^6)$ space in the worst case. Since we here assume that n is large, such an algorithm would be prohibitively expensive.

Instead, we will investigate a simple greedy algorithm requiring $O(n^2 \log n)$ and $\Theta(n)$ space. It is based on a very simple idea: we start with a coarse polyhedral approximation, this is iteratively refined until an admissible approximation is obtained. It is possible to construct input for which this algorithm performs poorly. However, experiments indicate that good results can be obtained under conditions likely to occur in many applications.

2 The algorithm

In the following we assume that $E(S)$ is known. Otherwise, the extreme locations of S can easily be computed in $O(n \log n)$ time [7].

Information about the current polyhedral approximation is kept in a simple data structure consisting of a list of records, one record for each triangle. Each record contains a pointer to a list of all locations lying in the interior of the triangle. Also, in each record there is information indicating at which interior location the height deviates most from the current approximation, and how large this deviation is. The algorithm consists of the following steps:

1. Compute $DT(E(S))$. Together with the heights at locations in $E(S)$, this triangulation defines the initial polyhedral approximation.

2. Initialize the data structure for $DT(E(S))$, i.e. for each triangle in $DT(E(S))$ it is determined which locations lie in the interior of the triangle, what the maximum deviation is, and for which location it occurs. If some location is found to lie on the edge between two triangles, it is assigned to one of them.

3. Determine the maximum deviation, *maxdev*, over all triangles in the current triangulation.

4. If *maxdev* is less than or equal to d, the current approximation is admissible and we are done, otherwise refine the current triangulation as follows:

   (a) Let p be a location for which the deviation is equal to *maxdev*. This location is now to be inserted into the current triangulation. First, find all triangles whose circumcircles contain p. Together these triangles form a star-shaped polygon R with p in its kernel.

   (b) Update the data structure, i.e. the triangles found in the previous step are replaced by the triangles formed by inserting edges from p to the boundary vertices of R. Also, for each of the new triangles, we determine the interior locations, the maximum deviation, and where this deviation occurs.

   (c) Go to Step 3.

It is easy to see that most of the computations involved in this algorithm are the same as those performed in the iterative method for computing the Delaunay triangulation [3,8]. In the iterative method, the insertion of the k:th location p may lead to the insertion of $O(k)$ new edges incident to the inserted location. Hence, a total time of $O(n^2)$ is required. In our case, we must also determine which locations are interior to the new triangles. This is easy; sort the new edges angularly around p, then use binary search to determine in which triangle a location lies. Obviously, we only need to consider the locations interior to the replaced triangles. In our method, we then need $O(n \log k)$ time to insert the k:th location and a total of $\Theta(n^2 \log n)$ time in the worst case.

3 Performance analysis

In the following, let $\tilde{I}$ be the set of points inserted by our method until (1) is satisfied.

We will first show that the worst case performance of our method is poor: the ratio $|\tilde{I}|/|I^*|$ is $\Theta(n)$ in the worst case. Consider the case illustrated in Figure 1. Here, $E(S)$ is equal to the vertices of the unit square. These locations are assigned the height 0. All the remaining locations lie along one of the diagonals. One of these locations lie on the center of the square, and is assigned an arbitrary non-zero height, say $5d$. Let us call this point p. The remaining points are alternatingly lying a distance d above or below the line segments connecting p to the extreme points of the diagonal. This construction can be seen more clearly in

[1] Two triangles can coexist in a Delaunay triangulation only if no vertex of one of the triangles lies in the interior of the circumcircle of the other triangle.

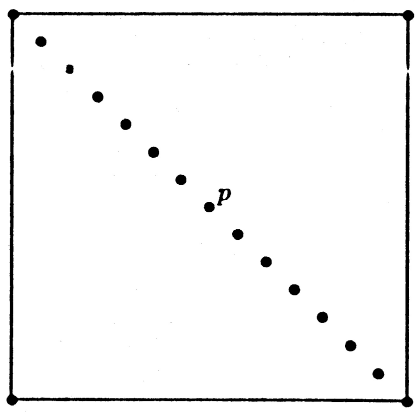

Figure 1: Worst-case input.

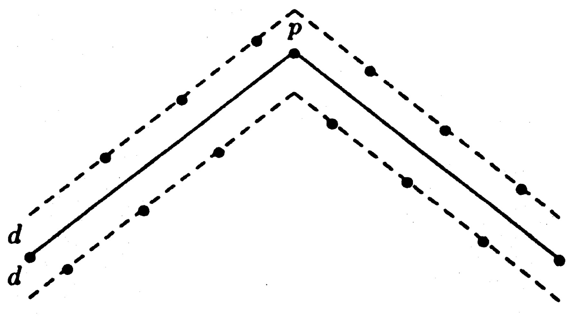

Figure 2: Cross-section along diagonal.

Figure 2, showing a cross-section along the diagonal. Obviously, $I^* = \{p\}$. However, if the height of the location immediately to the left of p is greater than $5d$, then our method will first choose this point. It is now easy to see that our method will be forced to insert every point, except p, until an admissible set is found.

This negative worst-case behaviour does, however, not imply that the method is useless. In the following, we present experimental evidence that the method performs well under conditions often occurring in practice.

To generate test data we have used the function:

$$\begin{aligned} F(x,y) &= 0.75e^{-((9x-2)^2+(9y-2)^2)/4} + \\ &\quad 0.75e^{-((9x+1)^2/49+(9y+1)/10)} + \\ &\quad 0.50e^{-((9x-7)^2+(9y-3)^2)/4} - \\ &\quad 0.20e^{-((9x-4)^2+(9y-7)^2)}. \end{aligned}$$

This function has been used by other researchers to test non-linear interpolants for scattered data interpolation [2,9]. Data is generated in the domain of the unit square, where this function has two maxima, one minimum and a saddle point. In the following a *test case* refers to a set S of n locations in the plane and a set h of heights at these locations. S will always contain the vertices of the unit square, whereas the remaining locations are chosen randomly within the unit square.

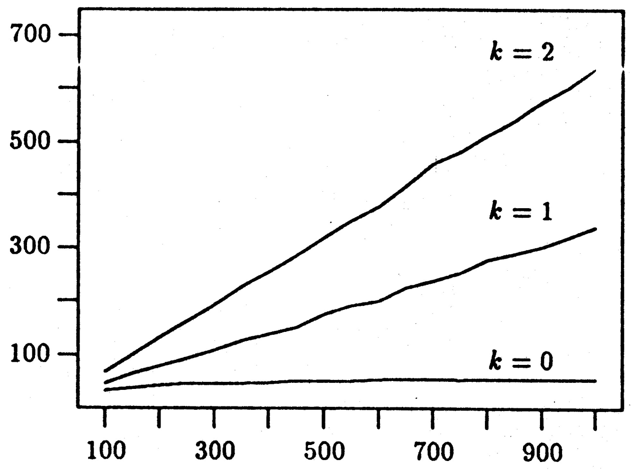

Figure 3: Median size of $\tilde{I}$ for increasing values of n.

Set h will consist of more or less perturbed values of F.

We ran three series of tests, corresponding to an increasing perturbation of function values: for series k, $k = 0,1,2$, the height at a location (x_i, y_i) was chosen randomly in the interval $[F(x_i,y_i) - kd, F(x_i,y_i) + kd]$. The value of d was set to 0.05. In each series, for $n = 100, 150, \ldots, 1000$, we generated 21 test cases for each value of n. In Figure 3, we have, for each series, plotted the median value of $|\tilde{I}|$ for each value of n. We observe, that with no perturbation, i.e. $k = 0$, the median value is essentially constant.

The results presented so far does not tell us very much about how a solution obtained by our method compares with an optimal solution. Since we know of no other way to find an optimal solution than exhaustive search, we have to estimate the size of an optimal solution. This estimate should be as large as possible without ever exceeding the true value.

To determine if there is no admissible triangulation with v vertices, we can proceed as follows. First, we compute $t(v)$, the corresponding number of triangles. It is well-known that $t(v) = 2(v-1) - e$, where e is the size of $E(S)$. A necessary requirement for the existence of an admissible triangulation with v vertices is that the remaining $n - v$ locations can be *assigned* to the $t(v)$ triangles. To explain what we mean by assigning locations to triangles, assume that we know *max*, the maximum number of locations interior to any admissible triangle. We now assign *max* locations to each triangle until either the accumulated number of assigned locations is at least $n - v$ or we run out of triangles. In the latter case, we can immediately conclude that there is no admissible triangulation with v vertices. Hence, a simple method to estimate the size of an optimal solution would be start with $v = e$, and then increment v until $t(v) \cdot max \geq n - v$. Obviously, this estimate can never exceed the true value. It can, however, be much lower since the number of locations which are actually contained in admissable triangles containing max locations can be small. Instead, we will use a more refined method of assigning locations to triangles.

Each location p is generally contained in a number of admissible triangles. Let n_p be the maximum number of locations contained in any admissible triangle containing p. For $i = 0, \ldots, max$, let m_i be the size of $\{p \in S : n_p = i\}$. Observe, that the total number of locations interior to admissible triangles containing at least j, $j > 0$, locations is at most $\sum_{i=j}^{max} m_i$. To decide if no admissible triangulation with v vertices exists, we start by assigning max locations to triangles. However, this time max locations are assigned only to $t_{max} = \lfloor m_{max}/max \rfloor$ triangles since this is the maximum number of triangles which simultaneously can contain max locations. Next, we assign $max - 1$ locations to triangles. This can be done to at most $t_{max-1} = \lfloor (m_{max-1} + m_{max} - t_{max} \cdot max)/(max - 1) \rfloor$ triangles. Continuing like this, that is, assigning j, $j > 0$, locations to $t_j = \lfloor (\sum_{i=j}^{max} m_i - \sum_{i=j+1}^{max} t_i \cdot i)/j \rfloor$ triangles, we finally arrive at a situation when either the total number of locations assigned to triangles is at least $n - v$ or we run out of triangles.

To estimate the size of an optimal solution, we repeat the above process for increasing values of v until we manage to assign at least $n - v$ locations to triangles. The value of v for which this happens is our estimate of the size of an optimal solution. It should be obvious that this value can never exceed the true value. This makes it possible to compute an upper bound on the ratio $|\tilde{I}|/|I^*|$.

The above method for estimating the size of optimal solutions was applied to the earlier described tests. Since the computation of estimates is rather time consuming, it was necessary to decrease both the maximum value of n and the number of generated test cases for each value of n. In Figure 4, the upper bound on $|\tilde{I}|/|I^*|$ is plotted for increasing values of n. These results suggest that our method gives good results, in particular when the amplitude of the local variations in the given data is small compared to d.

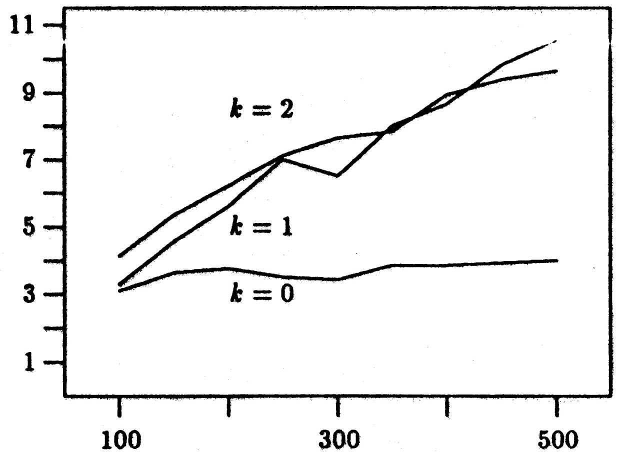

Figure 4: Upper bound on $|\tilde{I}|/|I^*|$ for increasing values of n.

References

[1] L. De Floriani. A pyramidal data structure for triangle-based surface description. *IEEE Computer Graphics and Appl.*, 67–78, March 1989.

[2] R. Franke. Scattered data interpolation: tests of some methods. *Math. Comp.*, 38:181–200, 1982.

[3] P.J. Green and R. Sibson. Computing Dirichlet tesselations in the plane. *The Computer Journal*, 21(2):168–173, 1978.

[4] C.L. Lawson. Software for C^1 surface interpolation. In J.R. Rice, editor, *Mathematical Software III*, pages 161–194, Academic Press, 1977.

[5] A. Lingas. The greedy and Delaunay triangulations are not bad in the average case. *Information Processing Letters*, 22(1):25–31, 1986.

[6] D.H. McLain. Two dimensional interpolation from random data. *The Computer Journal*, 19(2):178–181, 1976.

[7] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, N.Y., 1985.

[8] D.F. Watson. Computing the n-dimensional Delaunay triangulation with applications to Voronoi polytopes. *The Computer Journal*, 24(2):167–171, 1981.

[9] T. Whelan. A representation of a C^2 interpolant over triangles. *Computer Aided Geometric Design*, 3:53–66, 1986.