2D Local Sample-Based Interpolation as a Tool for Approximation of 3D Point Sets*

Vladimir SAVCHENKO**

We describe a novel algorithm for local approximation of scattered surface points, implementing a 2D finite element interpolation algorithm in combination with approximation of coordinates using quadrics for conversion of noisy data to sufficiently smooth data sets. The applied methods and time performance of the developed algorithm are discussed. Experimental results are included to demonstrate the functionality of our approximation technique.

Key Words: Finite Element Interpolation, Quadric, Scattered Data

1. Introduction

Many recent studies in computer graphics (CG) and computer-aided geometric design (CAGD) have focused on exploring surface reconstruction from point sets obtained from computer vision techniques. Nevertheless, it is still one of the challenging problems of CG and CAGD to obtain sufficiently smooth surface data which can be used for the visualization of various data. Image generators traditionally use polygons as database primitives of 3D visual environments, but many polygonal meshes exhibit effects of noise. Approaches using methods of interpolation of scattered data based on minimum-energy properties have recently received increased attention for geometrical design problems, computer animation, and medical applications. While providing good restoration qualities, such methods sometimes produce excessive smoothing. Uniting this concept with simultaneous smoothing has become an important problem in CG and CAGD because 3D range data often suffers from noise (See recent work of Carr et al.\(^{(1)}\)). We do not address here the problem of global reconstruction; we consider the question of local reconstruction of scattered 3D data with simultaneous denoising.

This research proposes an approach based on 2D finite element interpolation for conversion of noisy data to sufficiently smooth data sets with simultaneous resampling of the data in the neighborhood of processed points as can be needed in CAGD applications — for instance, to provide constructive solid geometry operations on denser sets of points. The objective of our work is to investigate possibilities of smooth interpolation of sets of scattered data by a finite element method (FEM), which can be considered a blurring filter in analogy to image processing techniques. This work continues an approach discussed in Ref. (2) which uses patterns for smooth global reconstruction of elevation data given in an image raster. Nevertheless, there are significant distinctions from the previous research. The developed algorithm is applied to scattered data without any supposition about topological connectivity of points. The technique, in fact, is based on local fairing 3D data; that is, an algorithm for local approximation of scattered surface points is used. However, for visualization of points (at this stage of our project) we assume that topological connectivity is provided. This is not a serious limitation, since a fast and memory efficient algorithm\(^{(3)}\) can be used for surface reconstruction (triangulation).

Our method is purely point-based, and the processing pipeline is based on the following assumptions:

- Input 3D data, representing a set of points with arbitrary coordinates \((x, y, z)\) from the surface of a geometric object, can be used for extrapolation and interpolation.
- Surface approximation on a set of square patches requires only \((x, y, z)\) positions of processed points.

The technique presented here uses the FEM as a numerical approach to processing of scattered data. FEMs are widely used in various engineering applications. The reason for the selection of the FEM approximation of scattered data is that this method is very well formalized and consists of several clearly defined steps\(^{(4)}\). An attractive feature of FEMs is also that there is a very rich set of basis functions developed for various applications to attain desired accuracy of approximation. Another attractive feature of FEMs is that manual intervention by an end user is
possible. New simulation levels can be achieved by combining the proposed technique with the use of available reference points and break lines to approximate a surface, as discussed by Ebner and Reiss(5). A possible modification is the adding, for instance, of lines along which the interpolation function is allowed to be discontinuous, as discussed in Ref. (6); slope and curvature information can be incorporated. In spite of the simplicity of their implementation, FEMs require complicated and time-consuming operations, including mesh generation, construction of a system of linear algebraic equations (SLAE), and its solution. Thus, for processing a huge amount of 3D data we need to apply a general acceleration scheme to scattered 3D points. Moreover, it is known that results of FEM interpolation depend on the mesh quality. Instead of using Delaunay triangulations (in which mesh large and thin triangles can appear) and solution of a global SLAE, we use a so-called pattern-dependent scheme and precalculated decomposed matrix, which serves as a smoothing filter for the data processing. Unfortunately, it is not straightforward to apply FEM for extrapolation purposes, so for some “exceptional” points we approximate coordinates using quadrics.

Our contribution in this paper is a realization of a total framework for approximation of 3D points based on the use of a finite element algorithm.

The next section gives an overview of the related works. Applied methods are described in section 3. Section 4 presents application examples. Section 5 contains some conclusions.

2. Related Works

To show that the problem of surface reconstruction and approximation is far from solved we provide here a full survey of related research.

The benefits of using radial-basis functions (RBFs) and compactly supported RBFs have been recognized in many studies (See, for instance, literature devoted to surface reconstruction(7)-(12)). Such methods based on global reconstruction produce sufficiently good approximation of a surface, but they suffer from three drawbacks. They take a great amount of time. Artifacts or “ghost” objects can appear as a result of extracting a geometric shape from an implicitly defined function(13). Fine textures of surfaces may be smoothed out by resampling that depends on application of a polygonization method for implicitly defined geometry objects.

Questions of surface reconstruction are related to the problem of surface smoothing. Recently, in CG and CAGD communities, attention has been paid to mesh smoothing. In Ref. (14) a signal processing approach to mesh smoothing was proposed. Laplacian smoothing is considered in Ref. (15) as time integration of the heat equation on an irregular mesh and a curvature flow approach is used to remove undesirable noise. The algorithm described there is very fast, but over-smoothing can be observed in Fig. 5 of that reference. Subdivision schemes(16) can be thought of as an alternative approach to the problem. Methods exploiting anisotropic diffusion techniques became rather popular recently (See, for instance, Ref. (17) and references therein). The technique of Tasdizen et al. is based on filtering normals of the surface, rather than processing the positions of points on a mesh. The method is applied for smoothing complex, noisy surfaces, while preserving (and enhancing) sharp, geometric features, and applies to any shape that can be modeled as an isosurface. A fast mesh-smoothing algorithm based on multi-resolution techniques, in combination with constrained minimization of the discrete energy functional, was proposed in Ref. (18). Using mesh hierarchies, where components of the geometric shape at each level of detail are characterized by a fairing, solves the problem. It allows interactive response times for moderately complex models, up to about 5 K triangles. Nevertheless, off-line preprocessing (an incremental mesh decimation algorithm) is applied. Many authors have considered very carefully questions of geometric distortion during smoothing and problems of precise shaping. See, for instance, the recent papers of Zhang and Fiume(19) and Yagou et al.(20), where algorithms to preserve the sharp features of a mesh were presented, and also the article by Belyaev et al.(21) and references therein. These techniques provide excellent feature-preserving behavior, but a number of issues regarding their application remain open, and require more thorough consideration. For general example, techniques which strongly preserve features can also produce additional artifacts.

A shortcoming of the methods discussed above is that they do not provide resampling of data sets. Nevertheless, the motivation of this work is also the following. Because of their low computational cost, Laplacian and smart Laplacian smoothing are the most commonly used techniques to simultaneously improve quality of meshes and smooth 3D data. But both methods may produce badly shaped or even inverted elements. In Ref. (22) an effective method, called an angle-based approach, with low computational cost was proposed. The quality of a mesh optimized with this method is much better than after Laplacian and smart Laplacian smoothing, and the likelihood of obtaining inverted elements (with negative area) is reduced. But in Ref. (23) it was shown that this is true mostly for meshes with regular connectivity. For instance, these methods fail to smooth such a mesh as is shown in Fig. 8(b), producing inverted triangles. When the mesh contains strongly distorted elements, an angle-based approach also may fail. Thus, we seek a method that does not exploit repositioning of (x,y) local coordinates but guarantees a valid mesh after processing.
In Ref. (1), scattered range data are smoothed by fitting a RBF to the data and convolving with a smoothing kernel (low-pass filtering). The proposed method exhibits very good smoothing features. However, the technique is computationally very expensive. In spite of that, the algorithm allows for producing data resampling.

An approach based on approximation of local “elevation data” by compactly supported RBFs was used in Ref. (24). The method is based on space mapping techniques so, while allowing obtaining rather good smoothing results for noisy surfaces, the approach can not be applied to approximating an arbitrary given point. In fact, RBFs can be used for data approximation in 2D space. Nevertheless, \( N^2 \) and \( N^3 \) operations, where \( N \) is the number of local points, are needed for matrix construction and solution of a system of algebraic equations, respectively.

Recently, sampled point clouds have received much attention in the CG community for visualization purposes (see Refs. (25) and (26) and references therein), where a moving least-square method (MLS)(27) is used. Approximation of a single point is based on using a radial weight. We use piecewise linear approximation over the selected parameter, and the computational cost of the MLS can be high.

The partition-of-unity method (PUM) for the construction of local interpolations was pioneered by Shepard(28) and later extended by Franke and Nielson(29). In recent years, it has received much attention due to the works of Melenk and Babuska(30) and Krysl et al.(31) Application of PUM for goals of geometry modeling was considered in a recent article by Ohtake et al.(32) In fact, PUM does not provide overall smoothness. According to our experiments(33) with PUM, reconstructed surfaces often exhibit clearly visible “bumps,” so a surface generated by PUM should be refined by applying smoothing algorithms.

The problem of global surface reconstruction by FEM has been considered in many articles, especially for elevation data reconstruction (See, for instance, Ref. (2) and references therein). Nevertheless, we have been unable to find any application of FEMs to local approximation of scattered data.

### 3. Interpolation Algorithm

The problem of constructing interpolating functions for scattered data is well known. The book by Vasilenko(34) presents a good theoretical and practical introduction to this challenge. We do not consider here this problem in detail, but only sketch the basic idea of the interpolation schema and describe the software algorithm.

#### 3.1 Interpolation schema

The problem of constructing interpolating functions by FEM for 2D space \( \Omega \) is stated as follows: We are given data points \( P_i(x,y), i = 1,2,\ldots,N \), are scattered, in the sense that there are no assumptions about the disposition of the independent data. Data set \( r \) is associated with the points. We must construct a smooth function \( \sigma_r(x,y) \) which takes on the value \( r_j \) at points \( P_j(x,y) \in \Omega \), if possible, or satisfies the condition:

\[
||A\sigma_r - r||^2 = \text{min},
\]

where \( A \) is some linearly bounded operator. Along with this condition, the function \( \sigma_r(x,y) \) has minimum energy of all functions that interpolate values \( r_j \). This conforms to the following minimum condition, which defines operator \( T \):

\[
\int_\Omega [\sigma_{xx}^2 + \sigma_{yy}^2]d\Omega = \text{min}.
\]

In our case, we solve the general problem of smooth approximation of scattered data. For the solution of this problem, according to Ref. (34), two matrices \( T \) and \( A \) must be constructed. After that, a nonsingular linear system of algebraic equations \((\alpha T + A)\sigma = f \) can be solved with the application of many iterative or direct methods; \( \alpha \) is a smoothness weight. We use piecewise linear approximation over the rectangular mesh. The bilinear basis function \( \omega_{ij}(x,y) = \omega_i(x) \omega_j(y) \) corresponds to each node of the rectangular mesh. Unless we adjust the notation to make vector \( \sigma = [\sigma_{ij}] \) one-dimensional, matrices \( T \) and \( A \) will be quadruply indexed:

\[
t_{ijkl} = (T_{ij},T_{kl}), \quad a_{ijkl} = (A_{ij},A_{kl}),
\]

or, more explicitly,

\[
t_{ijkl} = \int_\Omega \left[ \frac{\partial \omega_i(x)}{\partial x} \frac{\partial \omega_j(y)}{\partial y} + \frac{\partial \omega_i(x)}{\partial y} \frac{\partial \omega_j(y)}{\partial x} \right] dx dy,
\]

\[
a_{ijkl} = \sum_{m=1}^{N} \omega_i(x_m) \omega_j(y_m) \omega_k(x_m) \omega_l(x_m),
\]

where \( P_m \) has coordinates \((x_m,y_m), m = 1,2,\ldots,N \). Taking into account that

\[
t_{ijkl} = d_{ik} u_{jl} + u_{kl} d_{ij},
\]

where \( u_{ps} = \int \omega_p(x) \omega_s(x) dx, d_{ps} = \int \omega_p(x) \omega_s(x) dx, \) and indexes \( p,s \in \{i,j,k,l\} \), elements of the matrix \( T \) can be calculated.

Taking into account explicit form of \( \omega_i(x) \) (see Fig. 1) integrating gives us:

\[
u_{11} = u_{mm} = h/3, \quad d_{11} = d_{nn} = 1/h,
\]
\[
\begin{align*}
u_{ii} &= 2h/3, \quad i \neq 1, \quad i \neq n, \quad d_{11} = 2/h, \quad i \neq 1, \quad i \neq n, \\
u_{ij} &= h/6, \quad d_{ij} = -1/h.
\end{align*}
\]

Figure 2 depicts three possible situations to calculate the coefficients \( t_{ijkl} \):
- (a) corner point;
- (b) boundary but not corner point;
- (c) interior point.

All remaining coefficients for \(|i-k| > 1\) or \(|j-l| > 1\) are equal zero. Coefficients \( a_{ijkl} \) are equal zero for \(|i-k| > 1\) or \(|j-l| > 1\) because of \( \omega_i(x) \omega_k(x) \equiv 0 \) or \( \omega_j(x) \omega_l(x) \equiv 0 \).

The solution of the data approximation can be found in the form:
\[
\sigma_{h}(x,y) = \sum_{i,j} \sigma_{ij} \omega_{ij}(x,y).
\]

3.2 Programming

The software algorithm consists of the following steps:
1. Delaunay triangulation. At this stage, for the current processed point \((x,y,z)\),
   - select \( N \) nearest neighborhood points from \( M \) points of the data set;
   - define the nearest plane for neighborhood points from the octal tree for the point \((x,y,z)\), rotate these points according to the estimated tangent plane (see Ref. (35)) so that the nearest plane is perpendicular to the \( z \)-axis, and project the original neighborhood points onto the \( OXY \) plane;
   - produce 2D local Delaunay triangulation, as shown in Fig. 3.
2. Calculating elevation data. At this stage, initially the closest point belonging to the border of a triangulated area (in fact, we calculate a convex hull) is determined. Coordinates of the point define a size of a pattern area (white square in Fig. 3).

The center of the pattern area is located at the processed point. Further, the pattern area is scaled to unit size, and all neighborhood coordinates are also scaled in conformity to the chosen scaling factor. If the pattern area is outside the triangulated area (see Fig. 4) we skip to step 7. According to our experiments, 2–5% of model points are “outside” points. Otherwise we put calculated data on the grid of size selected according to the chosen network pattern (see Fig. 3) and compute local \( z \)-coordinates as distances between intersection points of linear components perpendicular to the local tangent plane and triangles. We call this approach a pattern-dependent scheme. In our application, 16-point network pattern \((x, y)\) coordinates) are used to fill the area, as shown in Fig. 3, and to construct the logical structure of an FEM matrix. Actually it is possible to use other network patterns, such as a random sequence, Sobol’s quasi-random LP-\( \tau \) sequence(36), or patterns for halftone approximation(37). We store such patterns in a look-up table as pre-processed data and use them for constructing the logical structure of the FEM matrix. During the summation for calculating the matrix, all points \((x_m, y_m)\) which do not belong to the intersection of carrier functions \( \omega_i(x) \omega_j(y) \) and \( \omega_k(x) \omega_l(y) \) should be discarded. To avoid multiple verification of point \( P_m \) membership in this collection, it is reasonable to presort the initial data set, placing points according to the cells’ numbering. Such sorting also allows us to effectively calculate the components of the right part of the system of linear equations:
Results of the data sorting are placed in a two-dimensional array of pointers in which pointers to the first point \((x,y)\) for \(i\)-th cell and the number of points belonging to the cell are stored. An additional array contains pointers to the sorted data array of points.

3. **Numerical assembly.** This step calculates the values of all elements of the FEM matrix according to the logical structure of the matrix.

4. **Cholesky factorization.** The problem of approximating the data reduces to the solution of a system of linear algebraic equations: \(Ax = b\), where \(A\) is a band matrix of coefficients, \(x\) is a vector of unknown node values, and \(b\) is a vector of right parts.

For decomposition of the matrix \(A\) into a product of the lower and the upper triangular matrices, \(A = LL^t\), Cholesky decomposition is used\(^{(37)}\). Steps 3 and 4 are performed only once, at the preprocessing stage, after which the triangular matrix is used as the look-up table. Therefore, for every processed point we scale the pattern area to unit size.

5. **Calculating the right part for the resulting linear algebraic system, solving lower and upper triangular matrices.** Calculating the right part and solving triangular matrices is performed for every point for the approximation of \(z\) components.

6. **Interpolation over the local mesh.** The \(z\)-value for the computed point is calculated. Skip to step 8.

7. **Quadric fitting.** The solution to calculate the \(z\) values for the point shown in Fig. 4 is based on local approximation of the point coordinates by quadratic polynomials. For construction of the quadrics for all neighboring points we use the least-squares method, which detailed description may be found in Ref. \(^{(39)}\).

8. **Produce the inverse rotation of the transformed point and add the result to the initial \(z\)-coordinate.**

In practice, the technique requires that near a processed point there exists a neighborhood that is homeomorphic to a disc.

All described parts of our software tool have been implemented in C++ with floating point precision. The measured processing time is about 0.001 second per processed point on a 500 MHz PIII. The computational complexity of the algorithm, determined by Delaunay triangulations and forward/back steps, is about \(O(MN^2)\). Our experiments show that it is reasonable to use 7–11 neighboring points and \(\alpha = 1\) to obtain satisfactory visual appearance.

### 4. Examples

#### 4.1 Smoothing examples

Figure 5 shows an example of processing real-world data. Our algorithm, based on local data approximation, attains overall smoothness and provides sufficiently good feature-preserving behavior. Figure 5 (b) shows that almost all features of the object were preserved without excessive smoothing or underlining.

Figure 6 also shows that the algorithm attains overall smoothness and preserves most fine features of the object. Volume characteristics are preserved without special treatment. For instance, the difference between the volume of the original “Monk” model and its approximation is 0.1%.

Real-world data often exhibit shot noise. Figure 7 shows that high frequencies (see enlarged squares from frontal area of the model before and after processing) with low amplitude are globally attenuated by the FEM filter.

For comparison, Fig. 7 (c) shows the surface smoothing results using a very effective least-square error method pro-
Fig. 7 (a) The “Stoned” model, whose surface was constructed from range data (courtesy of R. Scopigno and M. Callieri of the Institute CNUCE). Model size: 88,478 points. (b) Approximation of “Stoned” model, \( N = 10 \). (c) Result of data processing based on smoothed surface normals.

Fig. 8 (a) Contour map of Mount Bandai, Japan. Number of initial points: 21,930 (438 \( \times \) 421 raster size). (b) Fragment (from the top right) of the mesh of Mt. Bandai. Time of Delaunay triangulation of contour maps (number of triangles after Delaunay triangulation: 43,663) and discovery of intersection points: 29.4 sec. (c) Ray tracing of elevation data. Number of quadrilateral polygons: 172,540. (d) Result of ray tracing after data smoothing. Time of interpolation: 9.6 sec. (e) Polygonal visualization of resampled and randomly selected data (number of triangles — 73,961).

4.2 Resampling

Data can be obtained or derived from different sources of information, including existing contour maps. They are still a rich source of such data; an example is shown in Fig. 8(a). Despite a flurry of activity in the generation and visualization of terrain data from scattered data points, this problem remains difficult an computationally expensive, far from complete solution, because desirable characteristics of an interpolation or approximation algorithm are not obvious, as different attributes conflict with each other. Clearly visible artifacts affect almost all recent terrain models. Undesirable synthetic effects, such as terracing, appear during interpolating contour lines with the application of PDE methods and thin plate interpolation while minimizing the energy of bending; over-smoothing can be observed when a polynomial or a multi-level B-spline interpolation is applied; and so forth. Traditionally, scattered data visualization of elevation data consists of three major steps: tessellation, modeling, and rendering. The most commonly used tessellation is the Delaunay triangulation. Unfortunately, this method has some serious drawbacks resulting in a confusing image because clearly visible “traces” of triangulation can be observed, as seen in Fig. 8(b). To attenuate this effect, we implement the approach discussed above. We process elevation data using the described algorithm and put calculated data on a grid of the initially specified 438 \( \times \) 421 size. Reduced sets of interpolated elevation data can be used.

5. Conclusion

In this article we propose a novel approach for approximation of irregularly sampled data that combines the local FEM interpolation technique and the least-squares fit of the quadric. Practical applications yield good results regarding volume preservation and visual appearance. Our approach’s computational complexity and required storage are linear in the number of points of the shape and our algorithm exhibits satisfactory time performance. Nevertheless, a shortcoming of its current implementation (based on the VTK(42)) is its computation weight, and we plan an optimization of the developed software. Proper reconstruction of surfaces is possible only if the surfaces are "sufficiently" sampled, as noted by Gopi and Krishnan(3).

Resampling results shown in Fig. 8 demonstrate the applicability of our approach to elevation data, and our immediate goal is to extend the approach for resampling 3D point sets. Piecewise linear approximation achieving only \( C^0 \)-continuity was used. The proposed technique and developed tool can be improved by the use of more complex elements for better accuracy of surface approximation.

References


(2) Savchenko, V. and Sedukhin, S., Pattern Dependent Reconstruction of Raster Digital Elevation Models from Contour Maps, Proc. of VII’01, IASTED Int.


