

Figure 7: PLS's on a 100-point sample of  $F_1$  with  $\sigma = 0$ : (a)-(b) Delaunay PLS ( $l_1$  distance = .019). (c)-(d) LOP PLS ( $l_1$  distance = .017). (e)-(f) Optimal PLS at  $\lambda = .0001$  ( $l_1$  distance = .017).

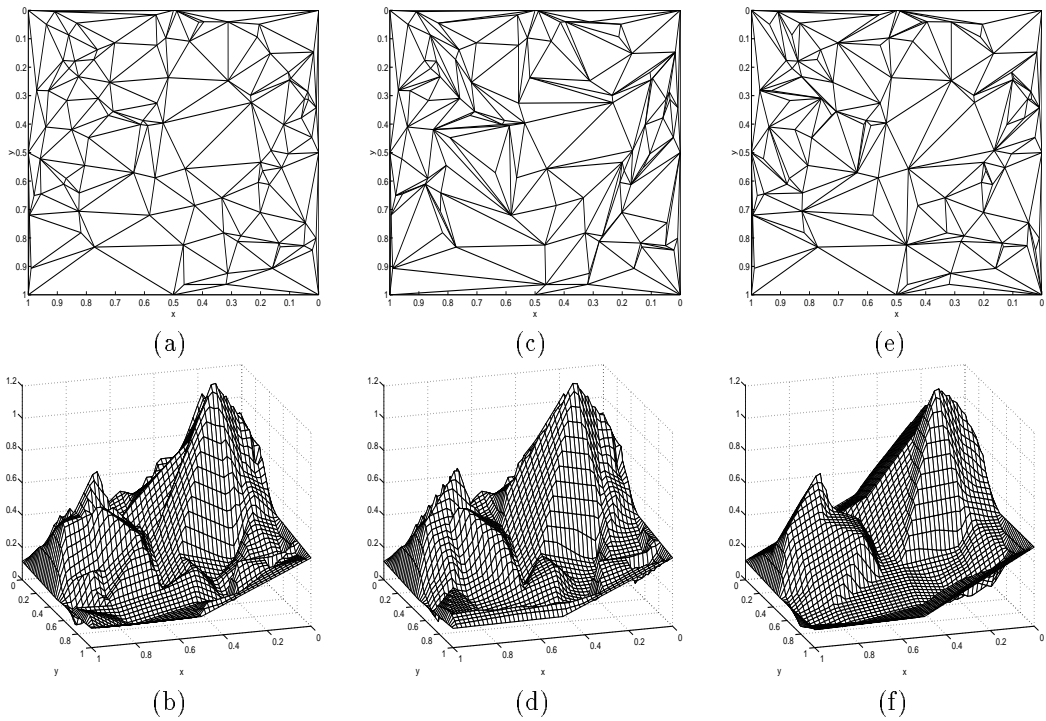


Figure 8: PLS's on a 100-point sample of  $F_1$  with  $\sigma = .1$ : (a)-(b) Delaunay PLS ( $l_1$  distance = .065). (c)-(d) LOP PLS ( $l_1$  distance = .071). (e)-(f) Optimal PLS at  $\lambda = .01$  ( $l_1$  distance = .055).

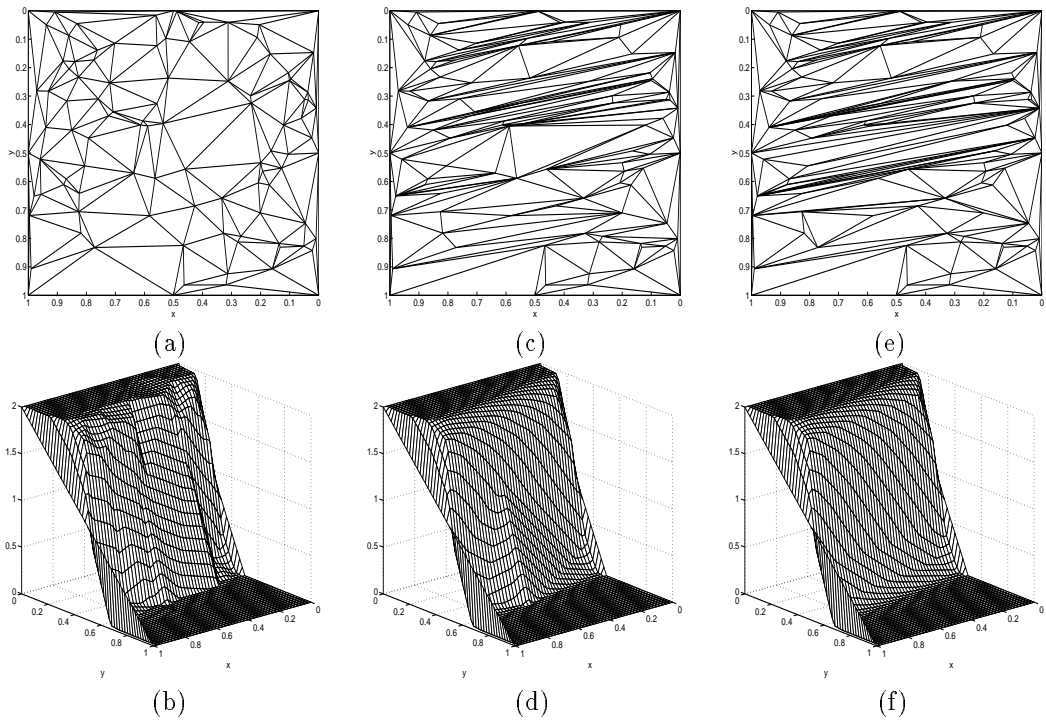


Figure 5: PLS's on a 100-point sample of  $F_8$  with  $\sigma = 0$ : (a)-(b) Delaunay PLS ( $l_1$  distance = .050). (c)-(d) LOP PLS ( $l_1$  distance = .026). (e)-(f) Optimal PLS at  $\lambda = .005$  ( $l_1$  distance = .011).

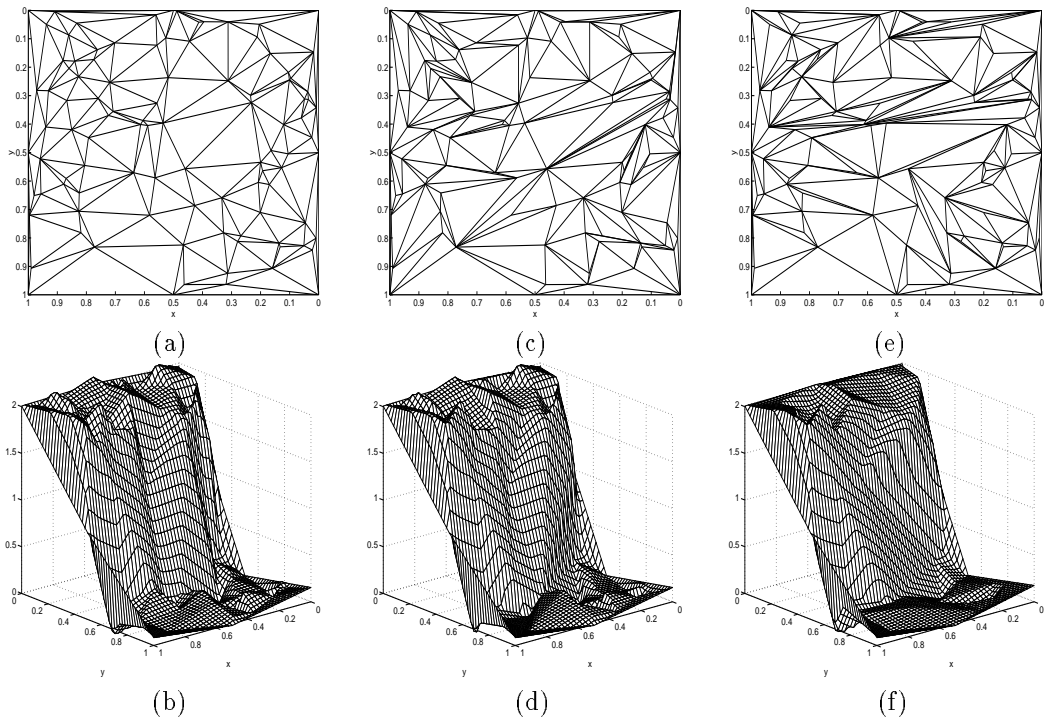


Figure 6: PLS's on a 100-point sample of  $F_8$  with  $\sigma = .1$ : (a)-(b) Delaunay PLS ( $l_1$  distance = .098). (c)-(d) LOP PLS ( $l_1$  distance = .097). (e)-(f) Optimal PLS at  $\lambda = .01$  ( $l_1$  distance = .075).

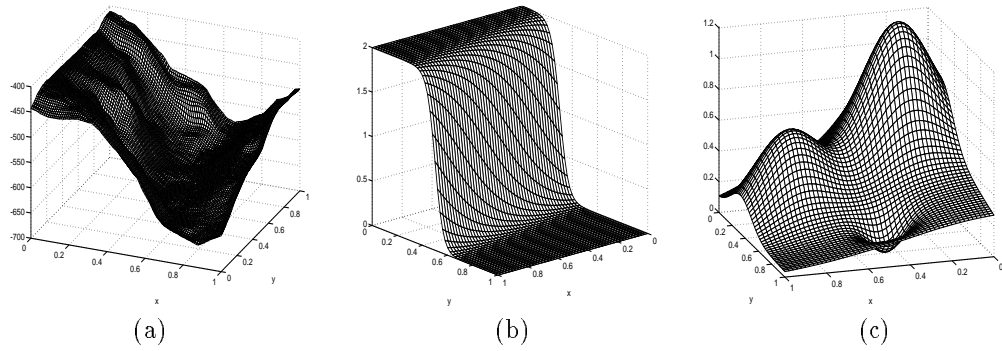


Figure 3: Test cases: (a) DTM. (b)  $F_8$ . (c)  $F_1$ .

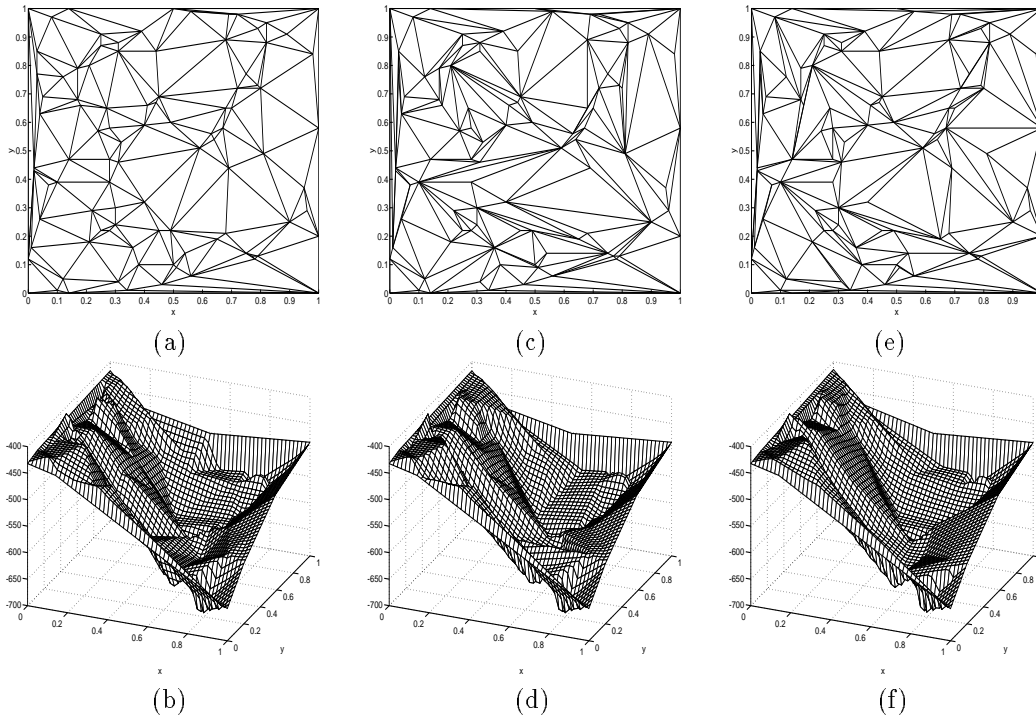


Figure 4: PLS's on a 100-point sample of DTM data (Fig. 3(a)) with  $\sigma = 15m$ : (a)-(b) Delaunay PLS ( $l_1$  distance = 13.3). (c)-(d) LOP PLS ( $l_1$  distance = 14.0). (e)-(f) Optimal PLS at  $\lambda = 150$  ( $l_1$  distance = 12.8).

## 6 Summary and Conclusions

We have presented an algorithm generating a good piecewise-linear approximation of a surface over some triangulation from noisy samples. The algorithm produces the best results in one of the following cases:

1. The sampled function has a clear preferred direction (like  $F_8$ ): It seems that the flexibility in the PLS values at the triangulation vertices enables the LOP to perform more edge swaps than when the heights are constrained to fixed values. These additional swaps improve the PLS. This is true even when there is no noise.
2. The noise is significant: The first LOP damages the PLS quality because this procedure is very sensitive to the sample values, which are very inaccurate. Using our algorithm when first adjusting the PLS heights reduces some of the noise, enabling the LOP to perform better.

There are a few possible variations on our algorithm, including the type of metric used to measure the distance, the traversal order of the PLS vertices during the first step of the algorithm and the traversal order of PLS convex quadrilaterals during the second step of the algorithm. The exact threshold of  $\sigma$  before which the samples are considered relatively accurate, therefore beneficial to reverse the order of the algorithm steps, is not yet clear.

## Acknowledgments

The second author wishes to thank Nira Dyn and David Levin for helpful discussions on the subject of the paper. The DTM data used in our experiments was produced and kindly made available by John Hall of the Israel Geological Survey.

## References

- [1] Special issue on softcopy photogrammetric workstations. *Photogrammetric Engineering and Remote Sensing*, January 1992.
- [2] D. Cohen and C. Gotsman. Photorealistic terrain imaging and flight simulation. *IEEE Computer Graphics and Applications*, 14(2):10–12, 1994.
- [3] P. Craven and G. Wahba. Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross validation. *Numerische Mathematik*, 31:377–403, 1979.
- [4] E.F. D’Azevedo. Optimal triangular mesh generation by coordinate transformation. *SIAM J. Sci. Stat. Comput.*, 12(4):755–786, 1991.
- [5] N. Dyn, D. Levin, and S. Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA Journal of Numerical Analysis*, 10:137–154, 1990.
- [6] R. Franke. Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, 38:181–200, 1982.
- [7] C.L. Lawson. Transforming triangulations. *Discrete Math.*, 3:365–372, 1972.
- [8] T. Lyche and K. Morken. Knot removal for parametric B-spline curves and surfaces. *Computer Aided Geometric Design*, 4:217–230, 1987.
- [9] E. Nadler. Piecewise linear best  $l_2$  approximation on triangulations. In C.K. Chui, L.L. Schumaker, and J.D. Ward, editors, *Approximation Theory V*, pages 499–502. Academic Press, 1986.
- [10] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.
- [11] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C (Second Edition)*. Cambridge University Press, 1992.
- [12] E. Quak and Schumaker L.L. Least squares fitting by linear splines on data dependent triangulations. In P.J. Laurent, A. Le Méhauté, and L.L. Schumaker, editors, *Curves and Surfaces*, pages 387–390. Academic Press, 1991.
- [13] C. Reinsch. Smoothing by spline functions. *Numerische Mathematik*, 10:177–183, 1967.
- [14] S. Rippa. Long and thin triangles can be good for linear interpolation. *SIAM J. Numer. Anal.*, 29(1):257–270, 1992.
- [15] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen. Decimation of triangle meshes. In *Proceedings of SIGGRAPH’92*, pages 65–70. ACM, 1992.
- [16] G. Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.

An approach similar to this was proposed by Reinsch [13] in the context of  $C^2$  smoothing splines. Finding this  $\lambda_{opt}$  requires a search procedure that applies the PLS construction algorithm for different values of  $\lambda$ , calculates  $H(\lambda)$  and re-estimates  $\lambda$  accordingly.

## 5 Experimental Results

We tested our algorithm on one set of real DTM data and samples of two different “synthetic” test functions  $f : [0, 1]^2 \rightarrow \mathbb{R}$ . For the DTM, we started with a set  $\{(x_i, y_j, z_{i,j})\}_{i,j=1}^{100}$  of  $100 \times 100$  data points on a regular grid in the Dead Sea area (Fig. 3(a)). We randomly selected 100 points from this set and added to them Gaussian noise with  $\sigma = 15m$ . These 100 samples, and a chosen  $\lambda$ , served as input to our algorithm, producing a PLS  $f'$  as output. To estimate the PLS *quality* we calculated the  $l_1$  distance  $\|f - f'\|_1$ , as in (2). The two test functions were taken from [5], who adopted them from [6] and [8]:

$$\begin{aligned} F_1(x, y) &= \frac{3}{4} \exp \left[ -\frac{(9x-2)^2 + (9y-2)^2}{4} \right] \\ &+ \frac{3}{4} \exp \left[ -\frac{(9x+1)^2}{49} - \frac{9y+1}{10} \right] \\ &+ \frac{1}{2} \exp \left[ -\frac{(9x-7)^2 + (9y-3)^2}{4} \right] \\ &- \frac{1}{5} \exp [-(9x-4)^2 - (9y-7)^2] \end{aligned}$$

$$F_8(x, y) = \tanh(-3g(x, y)) + 1$$

where:

$$g(x, y) = 0.595576(y + 3.79762)^2 - x - 10$$

The function  $F_1$  is composed of two Gaussian peaks and a sharp Gaussian dip (Fig. 3(c)). The function  $F_8$  simulates a sharp rise, whose contour lines are the parabolas  $g(x, y) = \text{const}$  (Fig. 3(b)). Each of the functions was sampled on 100 random points distributed uniformly in  $[0, 1]^2$  and contaminated with Gaussian noise of variance  $\sigma^2$ . These samples, and a chosen  $\lambda$ , served as input to our algorithm. Again, the performance of the algorithm was measured by the  $l_1$  distance (1). In practice, this integral was computed numerically by Monte-Carlo integration. Note that this can be computed only for synthetically generated samples, such as ours, where  $f$  is available. To evaluate our results, and contrast them with those of [5],

for each input data set we constructed three PLS's. The first obtained using  $z'_i = z_i$  and  $T =$  the Delaunay triangulation (referred to as the *Delaunay PLS*). The second with  $z'_i = z_i$  and  $T =$  the triangulation obtained by applying the LOP procedure on the Delaunay triangulation as suggested in [5] (referred to as the *LOP PLS*). The third is the final PLS obtained using our algorithm (referred to as the *Optimal PLS*).

### 5.1 Accurate Data

Although our algorithm was designed primarily for noisy sample sets, we applied it also on relatively accurate (very small  $\sigma$ ) samples of  $F_1$  and  $F_8$ . With this type of input, the LOP procedure on the Delaunay triangulation seems to always improve the PLS, so it is advantageous to swap the steps of our algorithm, such that the first step improves the triangulation, and the second the PLS values at the triangulation vertices. Fig. 5 demonstrates that when  $\sigma = 0$ , although the LOP PLS reduces the distance by 48% relative to the Delaunay PLS (as has been shown already in [5]), allowing the sample values to move by applying our algorithm with a small value of  $\lambda$ , results in a PLS with distance further improved by another 58% (relative to the LOP PLS). This is especially true for functions with a clear preferred direction, such as  $F_8$ , as the extra freedom allows the triangulation to align itself in this preferred direction. For  $F_1$ , as demonstrated by Fig. 7, there is no significant improvement.

### 5.2 Noisy Data

The main benefit of our algorithm was in the case of relatively noisy data (approximately 10% error in the elevation values). Fig. 4 shows our results on the DTM data, and Figs. 8 and 6 show our results on  $F_1$  and  $F_8$ . These are for the optimal values of  $\lambda$ , found experimentally.

The results of our procedure on the noisy DTM data were not as good as those obtained for the noisy synthetic data. This is probably due to the fact that the original terrain surface is not very smooth, and there are no significant preferred directions.

For  $F_1$  and  $F_8$ , the LOP PLS is not an improvement over the Delaunay PLS (for  $F_1$  it is even worse). In contrast, the optimal PLS produced by our algorithm reduces the distance relative to the Delaunay PLS by 15% and 23% respectively.

In all cases, the optimal value of  $\lambda$  seems to be a little smaller than that predicted by (7), as was also observed by Craven [3] in the case of  $C^2$  smoothing splines.

$t_2 = \{p_2, p_3, p_4\}$  be two triangles of the PLS  $f'$ , with common edge  $\overline{p_2p_3}$  (see Fig. 1). Let  $n_i$  be the vector normal to  $t_i$ ,  $i = 1, 2$ , and  $ABN(e)$  of edge  $e = \overline{p_2p_3}$  be the Angle Between the Normals  $n_1$  and  $n_2$ . Then

$$R(f') = \sum_{e \in \{\text{edges of } f'\}} ABN(e) \quad (4)$$

$ABN(e)$  may be thought of as an estimate of the *discrete* curvature of the surface  $f'$  at that edge. The relative importance of the two components of the cost function is controlled by the positive scalar parameter  $\lambda$ . For  $\lambda = 0$ , the infidelity of the PLS  $f'$  to the sampled data dominates, so any PLS with  $f'(x_i, y_i) = z_i$  minimizes  $C$ . For very large  $\lambda$ , the roughness of the surface dominates, so any constant-valued PLS minimizes  $C$ .

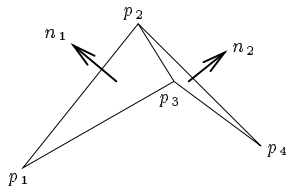


Figure 1: Two triangles of a PLS and their normals.

### 3 The Optimization Algorithm

Our algorithm generates an initial PLS and iteratively improves it. The PLS  $f_2$  is an *improvement* of PLS  $f_1$  if  $C(f_2) < C(f_1)$ . The input to the algorithm is  $\{(x_i, y_i, z_i)_{i=1}^n\}$  – the set of data samples, and  $\lambda$  – the smoothing parameter. The output of the algorithm is  $T$  – a triangulation of  $V = \{(x_i, y_i)_{i=1}^n\}$ , and  $\{(z'_i)_{i=1}^n\}$  – the values of the PLS at the points of  $V$ . An outline of the algorithm is :

- Generate an initial PLS.
- Repeat
  - (1) Improve the PLS values at points of  $V$  (keeping the triangulation fixed).
  - (2) Improve the triangulation of  $V$  (keeping the PLS values at  $V$  fixed).
- Until (there is no change in the PLS)

The initial PLS  $f'$  is  $T =$  the Delaunay triangulation of  $V$ , and  $z'_i = z_i, i = 1, \dots, n$ . To improve the PLS values at the set of vertices  $V$ , we systematically scan the vertices and set  $z'_i$  to be the  $z$  minimizing

$$h(z) = (z_i - z)^2 + \lambda \sum_{e \in E_i} ABN(e) \quad (5)$$

where  $E_i = \{\text{the edges of } f' \text{ incident on } (x_i, y_i)\}$  and  $f'(x_i, y_i) = z_i$ . Note that the second term of  $h(z)$  implicitly depends on  $z$ . The  $z$  minimizing the one-dimensional function  $h(z)$  may be found by standard numerical optimization procedures. We used the simple golden section method ([11], Chap. 10). It is easy to see that assigning this value to  $z'_i$  improves the PLS.

To improve the triangulation of  $V$ , we use the same LOP (Lawson Optimization Procedure) used in [5]: for every edge  $e$  which is a diagonal of a convex quadrilateral of  $T$ , replace  $e$  by the other diagonal of the quadrilateral (replacing the two triangles by two others) if this improves the resulting PLS (see Fig. 2).

It is not obvious that this algorithm converges. However, we have found in all our experiments that this is indeed the case, requiring 6 iterations on the average. We do not know whether the minimum achieved is global.

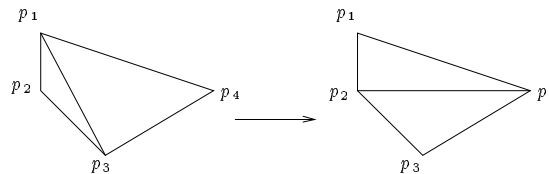


Figure 2: Swapping edges in a convex quadrilateral.

### 4 Determining the “Optimal” $\lambda$

The output of our algorithm obviously depends on the value of  $\lambda$  used. A large  $\lambda$  causes the surface to be smooth, while a small  $\lambda$  forces the  $f'(x_i, y_i)$  to be close to  $z_i$ , strongly constraining the solution. An important practical question is how to determine the optimal value of  $\lambda$  so that the resulting PLS  $f'$  will indeed be a good approximation of the original (unknown) surface  $f(x, y)$ . Define

$$e^2(\lambda) = \frac{1}{n} \sum_{i=1}^n (z_i - z'_i(\lambda))^2, \quad (6)$$

where  $z'_i(\lambda)$  are the values of the PLS  $f'(x_i, y_i)$  produced by our algorithm when applied with parameter  $\lambda$ . If  $z'_i$  were indeed the “true” values  $f(x_i, y_i)$ , then  $e^2(\lambda)$  would be a good estimate of the sample error variance, which we know to be  $\sigma^2$ . The optimal  $\lambda$  should then be

$$\lambda_{opt} = \arg \min H(\lambda), \quad H(\lambda) = |e^2(\lambda) - \sigma^2| \quad (7)$$

is no larger than  $d$ . Nadler [9] studied the connection between the the second derivatives of  $f$  at a point and the optimal shape of a triangle in that vicinity with respect to the  $l_2$  distance. D’Azevedo [4] used coordinate transformations to generate optimal triangulations with respect to the  $l_\infty$  distance.

In a similar vein, given a dense sample of  $f$  at  $m$  locations  $\{(x_i, y_i) : i = 1, \dots, m\}$ , it is sometimes required to dilute this sample to a sparse set  $V$  (sometimes called *decimation*) of  $n$  points, and approximate  $f$  as a PLS  $f'$  on some triangulation of  $V$ . The  $l_p$  distance is now measured relative to the original sample set:

$$\|f - f'\|_p = \left[ \frac{1}{m} \sum_{i=1}^m |f(x_i, y_i) - f'(x_i, y_i)|^p \right]^{1/p} \quad (2)$$

Schroeder et al. [15] present an algorithm for decimating a sample set and triangulating the result, while preserving important geometric features. Quak and Schumaker [12], while not specifying how to decimate the sample set, triangulate the  $n$  points to a PLS  $f'$  achieving a local minimum of the  $l_2$  distance  $\|f - f'\|_2$ .

In both cases mentioned above, the objective is clear: the PLS  $f'$  should approximate the explicitly given  $f$  or its dense sample as closely as possible. Our working point, for which the objective is not as clear, is to triangulate an already sparse sample set of locations  $V$ , without removing or adding points. The problem is ill-posed since, theoretically, we have no reason to prefer one triangulation over another. However, we try to minimize some “smoothness” measure, with the hope that since the “true” (but unknown)  $f$  was probably smooth, a smoother PLS has a better chance of approximating it well. Rippa [14] showed that the standard Delaunay triangulation ([10], Chap. 5) of  $V$ , which has many nice mathematical properties, yields suboptimal results in many cases and that long thin triangles (which the Delaunay triangulation avoids) are sometimes good for linear interpolation, contrary to common belief. Dyn et al. [5] suggested a method for the iterative improvement of an initial triangulation, using an edge-swapping technique due to Lawson [7], minimizing a cost function measuring the “roughness” of the PLS.

All the works mentioned above (except [12]) deal only with the case of accurate (non-noisy) samples. We extend the procedure of Dyn et. al to deal with noisy samples. Additionally, even for the case of accurate samples, we demonstrate that results superior to theirs may be achieved by using our algorithm: not constraining  $f'(x_i, y_i) = f(x_i, y_i)$  enables the algorithm to reach a better local minimum of the cost

function than the one obtained when constraining the PLS vertices. For a PLS  $f'$ , define its “cost”  $C(f')$  as

$$C(f') = I(f') + \lambda R(f') \quad (3)$$

where  $I(f')$  measures the inaccuracy of the fit of  $f'$  to the sampled data,  $R(f')$  measures the “roughness” of  $f'$ , and  $\lambda$  is a weighting factor. The best PLS is the  $f'$  minimizing  $C$ . This approach is standard for smoothing splines [16]. For example, in the one-dimensional case,  $g \in C^2[0, 1]$ ,  $I$  is taken to be

$$I(g) = \frac{1}{n} \sum_{i=1}^n (g(x_i) - y_i)^2$$

and  $R$  is

$$R(g) = \int_0^1 g''(x)^2 dx$$

which is an approximation for the average curvature of the surface. In the case of a PLS, which is two-dimensional and possesses noncontinuous derivatives, an expression similar to  $I$  is still applicable, but a discrete analog to  $R$  must be found. Once the cost function is well defined, an efficient procedure to determine the PLS minimizing it must be described.

The rest of this paper is organized as follows: Section 2 elaborates on the cost function  $C$ , Section 3 describes the optimization procedure minimizing  $C$  and Section 4 deals with determining the scalar parameter  $\lambda$  present in  $C$ . The results of numerical experiments are reported in Section 5. In Section 6 we summarize and conclude.

## 2 The Cost Function

What type of PLS is considered “good” ? In classic approximation theory, a common answer is: a good surface is one that passes close to the sampled data and is smooth (implicitly we assume that the original sampled function was smooth). Towards this end, we define the following “cost”  $C(f')$  of a PLS candidate  $f'$ . Given the sample set  $\{(x_i, y_i, z_i)_{i=1}^n\}$

$$C(f') = I(f') + \lambda R(f')$$

where

$$I(f') = \sum_{i=1}^n (z_i - f'(x_i, y_i))^2$$

measures the *infidelity* of the PLS fit to the sampled data, and  $R(f')$  is a measure of the PLS *roughness*, defined as follows: Let  $t_1 = \{p_1, p_2, p_3\}$  and

# Piecewise-Linear Surface Approximation From Noisy Scattered Samples

Michael Margaliot

Dept. of Electrical Engineering  
Technion - Israel Institute of Technology  
Haifa 32000, Israel

Craig Gotsman

Dept. of Computer Science  
Technion - Israel Institute of Technology  
Haifa 32000, Israel

## Abstract

We consider the problem of approximating a smooth surface  $f(x, y)$ , based on  $n$  scattered samples  $\{(x_i, y_i, z_i)_{i=1}^n\}$  where the sample values  $\{z_i\}$  are contaminated with noise:  $z_i = f(x_i, y_i) + \epsilon_i$ . We present an algorithm that generates a PLS (Piecewise Linear Surface)  $f'$ , defined on a triangulation of the sample locations  $V = \{(x_i, y_i)_{i=1}^n\}$ , approximating  $f$  well. Constructing the PLS involves specifying both the triangulation of  $V$  and the values of  $f'$  at the points of  $V$ . We demonstrate that even when the sampling process is not noisy, a better approximation for  $f$  is obtained using our algorithm, compared to existing methods. This algorithm is useful for DTM (Digital Terrain Map) manipulation by polygon-based graphics engines for visualization applications.

## 1 Introduction

Let  $f(x, y)$  be a smooth surface. Assume we are given  $n$  noisy samples  $\{(x_i, y_i, z_i)_{i=1}^n\}$  of  $f$  at scattered locations  $V = \{(x_i, y_i)_{i=1}^n\}$ , such that  $z_i = f(x_i, y_i) + \epsilon_i$ , where  $\epsilon_i$  are independent identically distributed zero-mean Gaussian random variables with known variance  $\sigma^2$ . We wish to construct a PLS (Piecewise Linear Surface), sometimes called a TIN (Triangulated Irregular Network),  $f'$ , defined on some triangulation of  $V$ . The PLS  $f'$  should approximate the original surface  $f(x, y)$  as closely as possible, i.e. the distance  $\|f - f'\|$  should be minimal, for some norm  $\|\cdot\|$ .

This problem arises in the reconstruction of terrain surfaces from random DTM's (Digital Terrain Models) extracted by automatic methods, such as matching stereo image pairs (see the many articles on this subject in [1]). These modern methods obtain terrain

elevation samples wherever possible, usually at feature points, resulting in a data set consisting of points at essentially random locations in the plane. This contrasts with the traditional manual DTM extraction procedures, where elevation samples are obtained on a regular grid, or at significantly correlated locations in the plane. Both manual and automatic DTM extraction methods are inaccurate, so noise is inevitably introduced into the elevation samples.

We choose to reconstruct the terrain as a PLS, namely, a collection of triangles, as these are standard geometric primitives in modern graphics engine hardware. Terrain visualization with texture-mapped aerial imagery are popular graphics applications in visual simulation environments [2]. Two issues must be addressed:

- The PLS  $f'$  is a collection of triangles in 3D space. The topology of the triangulation is identical to the topology of the specific planar triangulation of  $V$  used. As there are an exponential (in  $n$ ) number of different triangulations of  $V$ , each resulting in a different PLS  $f'$ , it is not obvious which is the best for our solution.
- The values  $z'_i = f'(x_i, y_i)$  at the PLS vertices. Since the data is noisy anyway, they do not necessarily have to coincide with the sampled  $z_i$ .

Many researchers have dealt with variants of the optimal surface triangulation problem in the non-noisy case ( $\sigma = 0$ ). In its most pure form, given an explicit function  $f(x, y)$ , and a tolerance  $d$ , approximate  $f$  by a PLS  $f'$  with a minimal number of triangles, such that the  $l_p$  distance between  $f$  and  $f'$ , defined as

$$\|f - f'\|_p = \left[ \int_0^1 \int_0^1 |f(x, y) - f'(x, y)|^p dx dy \right]^{1/p} \quad (1)$$