A FAST SURFACE INTERPOLATION TECHNIQUE

Technical Note No. 333

August 27, 1984

By: Grahame B. Smith, Senior Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

The research reported herein was supported by the Defense Advanced Research Projects Agency under Contract MDA903-83-C-0027 and by the National Aeronautics and Space Administration under Contract NASA 9-16664. These contracts are monitored by the U.S. Army Engineer Topographic Laboratory and by the Texas A&M Research Foundation for the Lyndon B. Johnson Space Center.

Abstract

A method for interpolating a surface through 3-D data is presented. The method is computationally efficient and general enough to allow the construction of surfaces with either smooth or rough texture.

1. Introduction

In image analysis we are often faced with the fact that the measurements we make in an image only constrain properties of the 3-D world, instead of specifying them. Analysis techniques that recover 3-D shape information from image measurements incorporate very restrictive assumptions about the nature of the world. In our attempts to avoid the need for these restrictions, we have been examining hypothesis-and-test methods. If we assume that we are able to obtain some shape data, from which we can hypothesize an approximate shape model for the world, then we can use this model to predict image features. To proceed from shape data to an approximate shape model we need to "flesh out" the data. In this paper we address the problem of fitting a surface to a set of points whose 3-D locations are known. While our interest centers on fitting a surface to 3-D location data that have been acquired by processing images of that surface, the technique developed has application to a broad class of surface-fitting tasks.

To select a surface-fitting procedure, it is insufficient merely to know the data set and to require that a surface be fitted to the points in that set. We also need to know the desired properties of the surface, the characteristics of the data, and the uses to which the fitted surface will be put. If we are building a surface to allow, say, water runoff estimates to be made, smoothness may be a desired property for that surface. For realistic rendering of a natural surface in computer graphics, however, a fractal surface may be preferable. While the technique we develop can construct either smooth or rough surfaces, our applications generally require the former. Our examples, Figures 3 and 4, show both types.

Besides the desired properties of the fitted surface, the characteristics of the data limit the approach we must adopt to surface construction. In fitting a surface we must balance the influence exerted by the data values themselves, against that exerted by the implicit surface model embedded in any fitting procedure. If our data values are inaccurate and we know the class of surfaces that should fit the data, we can usually let the surface model dominate the construction process. Least-square methods are typical of procedures that prefer a model to data. In general, techniques whose resultant surfaces do not conform exactly to the data are known as approximation methods. Methods that produce surfaces conforming exactly to the data are called interpolation methods.

The selection of an approximation or interpolation method is influenced by properties of the data other than their accuracy. Consider, for example, the terrain data collected by a surveyor. In selecting the places at which to make measurements, he considers the breakpoints of the surface – that is, those places on the surface where the gradient is discontinuous – and his data include measurements at these breakpoints. Surface reconstruction by linear interpolation over triangular surface patches is possible because the surveyor has furnished not only the 3-D data, but also an implicit statement that the surface between his points can be approximated by planar patches. In matching stereo pairs of images, an edge-based matcher provides more than the 3-D data it produces. Like a surveyor's data, it too makes an implicit statement about the continuity of the imaged surfaces. On the other hand, an area-based correlation matcher says less about surface continuity, but has the desirable behavior of providing regularly spaced data.

Such data can usually be processed with considerably less computational effort than data that are irregularly spaced. The volume of data, the regularity of their spacing, the implicit characteristics of their collection procedure, and their accuracy are all essential parameters in selecting a surface-fitting technique. For our applications we choose to investigate interpolation methods. We want methods that will work with irregularly spaced data, but still achieve substantial computational savings if we can use a regular grid of data points. We need to be able to handle thousands of such points. As a rule, we do not want to use implicit properties of the data that stem from their collection procedure.

The uses to which the fitted surface will be put further restricts the set of applicable surface-fitting procedures. If the task at hand is surface area estimation, the accuracy of the surface gradients is not important. Conversely, if we wish to use the fitted surface to generate the latter's image under some known lighting conditions, the surface gradient information then becomes crucial. We can classify the uses of fitted surfaces by the surface derivatives that are needed. An application that does not require surface derivatives to be calculated can usually be satisified by a surface composed of local patches. That is, the surface is fitted locally patch by patch, with each patch determined by a small number of local data points. Such methods have strong surface models and few data are needed to instantiate them. As a consequence, however, the surface derivatives are more a function of the surface model than of the data. The amount of data used to determine the surface patch may be barely sufficient to calculate an average value for the surface derivatives across the whole patch; besides, any variations in derivatives across the patch are caused by the model, not the data. The more data employed, the less is the influence of the surface model on the calculation of surface derivatives. In the extreme case, all the data may be used to determine the surface to be fitted at each locality. Such techniques are called global methods, whereas those that use only local data are local methods. Our applications require that we calculate surface curvature from our fitted surface. The technique we present here is a global method for surface fitting.

In summary, we address the problem of fitting a surface to a large data set composed mostly of regularly spaced data points, but which also includes grid points at which we have no data, and non grid points where data values are known. The data are acquired through a collection process that is assumed to yield accurate values, but for which we choose not to characterize the data further. We require a solution that is smooth and from which we can calculate the first and second surface derivatives. We present details of a global interpolation method that is computationally efficient and appears to applicable to a broad range of tasks. Although the general form of the method applies to non gridded data our computationally efficient algorithm comes from exploiting the regularity of the data points.

We commence by considering the multiquadric method invented by Hardy [1] for modeling natural terrain. In its generalized form, we examine it under the restriction of regularly spaced data points and derive an algorithm to solve for the unknown parameters. We show how to generate the interpolated surface in an efficient manner.

2. Surface Interpolation

2.1 Hyperbolic Multiquadrics

Suppose we have a set of data points, $[(x_i, y_i, z_i)]_{i=0}^{n-1}$ in 3-D space to which we wish to fit a hyperbolic multiquadric surface [1] defined by

$$z(x,y) = \sum_{j=0}^{n-1} c_j [d_j^2(x,y) + h]^{\frac{1}{2}} \quad ,$$

where $d_j^2(x,y) = (x - x_j)^2 + (y - y_j)^2$, h is a user-specified constant, and c_j's are the coefficients that must be determined.

To understand this method, let us suppose that $h = 0$. The data are fitted by placing a cone at each of the n data points so that the cone's axis is aligned with the z axis direction and the cone's apex is in the $z = 0$ plane. That is, the data are fitted with a set of cones, some of which are inverted. The z value of the constructed surface at position (x, y) is calculated by summing the z values contributed by each of the n cones at this (x, y) position.

Each cone has one free parameter, namely, its apex angle; we determine these apex angles by requiring that the constructed surface pass exactly through the data points. In the foregoing expression, the c_j's correspond to the apex angles of the cones. We calculate the c_j's by solving the nxn system of linear equations

$$\sum_{j=0}^{n-1} c_j [d_j^2(x_i, y_i) + h]^{\frac{1}{2}} = z_i \qquad i = 0, ..., n-1 \quad .$$

Note that this fitting technique does not require that the data be regularly spaced; furthermore, when $h \neq 0$, hyperboloids rather than cones are fitted to the data. Cones and hyperboloids are not the only options. Stead [2], for example, has generalized this method, using the form

$$z(x,y) = \sum_{j=0}^{n-1} c_j [d_j^2(x,y) + h]^{\frac{\mu}{2}} \quad .$$

2.2 General Form

We examine surface-fitting techniques that use the general form of the above method, namely,

$$z(x,y) = \sum_{j=0}^{n-1} c_j g(x - x_j, y - y_j) \quad ,$$

where the kernel function g is any function of the parameters $x - x_j, y - y_j$. Clearly, the previously defined functions are particular cases of this form. As before, we determine the c_j's by solving the nxn system of linear equations

$$\sum_{j=0}^{n-1} c_j g(x_i - x_j, y_i - y_j) = z_i \qquad i = 0, ..., n-1 \quad .$$

For large values of n it is not feasible to solve this system of equations. In our applications n may be 10,000. However, for smaller n we have used the above form to "patch" holes in a regular grid of data points. While any kernel function can be employed, we have found it important to match the method used to solve the nxn system of linear equations to the form of the kernel function selected. The numerical difficulties encountered in solving some of the systems of equations produced by a particular kernal can often be averted by exploiting properties of the linear system stemming from the choice of kernel function. For example, if we use the Gaussian function as the kernel, the symmetric positive definite coefficient matrix of the system of linear equations allows solution by the "square-root" method (see, for example [3]), and avoids the numerical problems created by Gaussian elimination. If we impose the restriction that the data points must be gridded, we can find feasible solution techniques even when n is of the order of millions.

2.3 Regular Grid Solution

Consider the problem of fitting the surface

$$z(x,y)=\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}c_{ij}g(x-x_{i,j},y-y_{i,j}) \quad , \tag{1}$$

where $[(x_{i,j},y_{i,j})]_{i=0,j=0}^{n-1,m-1}$ is an nxm regular grid, to the data set $[(x_{i,j},y_{i,j},z_{i,j})]_{i=0,j=0}^{n-1,m-1}$. We can find an expression for calculating the c_{ij}'s in the following manner.

Let $G(u,v)$ denote the discrete Fourier transform (DFT) of $g(x,y)$. Using the shift theorem of DFT theory, we note that the DFT of $g(x-x_{i,j},y-y_{i,j})$ is $G(u,v)e^{-2\pi i(\frac{u}{n}x_{i,j}+\frac{v}{m}y_{i,j})}$. If $Z(u,v)$ denotes the DFT of $z(x,y)$, we can write the DFT of Equation (1), namely,

$$Z(u_{k,l},v_{k,l})=\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}c_{i,j}G(u_{k,l},v_{k,l})e^{-2\pi i(\frac{u_{k,l}x_{i,j}}{n}+\frac{v_{k,l}y_{i,j}}{m})} \qquad \begin{array}{l}k=0,...,n-1\\ l=0,...,m-1.\end{array}$$

Removing $G(u_{k,l},v_{k,l})$ from the summation, we have

$$\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}c_{i,j}e^{-2\pi i(\frac{u_{k,l}x_{i,j}}{n}+\frac{v_{k,l}y_{i,j}}{m})}=\frac{Z(u_{k,l},v_{k,l})}{G(u_{k,l},v_{k,l})} \quad .$$

Taking the inverse DFT of the above expression, we obtain

$$\sum_{k=0}^{n-1}\sum_{l=0}^{m-1}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}c_{i,j}e^{-2\pi i(\frac{u_{k,l}x_{i,j}}{n}+\frac{v_{k,l}y_{i,j}}{m})}e^{2\pi i(\frac{u_{k,l}x_{p,q}}{n}+\frac{v_{k,l}y_{p,q}}{m})}=$$
$$\sum_{k=0}^{n-1}\sum_{l=0}^{m-1}\frac{Z(u_{k,l},v_{k,l})}{G(u_{k,l},v_{k,l})}e^{2\pi i(\frac{u_{k,l}x_{p,q}}{n}+\frac{v_{k,l}y_{p,q}}{m})} \quad .$$

Using $\mathbf{F}^{-1}$ to represent the inverse DFT, $\mathbf{F}^{-1}[\frac{Z(u,v)}{G(u,v)}](x_{p,q},y_{p,q})$ is the inverse DFT of $\frac{Z(u,v)}{G(u,v)}$, calculated at $(x_{p,q},y_{p,q})$. We have

$$\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}c_{i,j}\sum_{k=0}^{n-1}\sum_{l=0}^{m-1}e^{-2\pi i(\frac{u_{k,l}x_{i,j}}{n}+\frac{v_{k,l}y_{i,j}}{m})}e^{2\pi i(\frac{u_{k,l}x_{p,q}}{n}+\frac{v_{k,l}y_{p,q}}{m})}=$$
$$\mathbf{F}^{-1}[\frac{Z(u,v)}{G(u,v)}](x_{p,q},y_{p,q}) \quad .$$

Now

$$\sum_{k=0}^{n-1}\sum_{l=0}^{m-1} e^{-2\pi i(\frac{u_{k,l}x_{i,j}}{n}+\frac{v_{k,l}y_{i,j}}{m})} e^{2\pi i(\frac{u_{k,l}x_{p,q}}{n}+\frac{v_{k,l}y_{p,q}}{m})} \begin{array}{ll} =nm & x_{i,j}=x_{p,q}, y_{i,j}=y_{p,q}, \\ =0 & otherwise. \end{array}$$

Hence

$$c_{i,j} = \frac{1}{nm}\mathbf{F}^{-1}[\frac{Z(u,v)}{G(u,v)}](x_{i,j}, y_{i,j}) \qquad . \tag{2}$$

An alternate way of viewing the above derivation is to note that $g(x_{p,q}-x_{i,j}, y_{p,q}-y_{i,j})$ forms a circulant matrix, and to recall that such matrices are diagonalized by the discrete Fourier transform [4].

2.4 Surface Rendering

Once the $c_{i,j}$'s have been calculated, Equation (1) provides an analytic expression for the constructed surface. We can calculate $z(x,y)$ for any (x,y) position. However, each such calculation involves the sum of nm terms. If it is our intention to use this analytic expression for surface interpolation we may have to calculate this sum a very large number of times. The cost savings gained in computing the $c_{i,j}$ coefficients by means of the DFT (implemented by the fast Fourier transform) will be offset by the cost of these summations. As a rule, if we commence with data on a regular grid we want an interpolated surface on a finer grid. This results in considerable savings which are realized when the DFT is again employed.

Suppose we want to interpolate each grid interval in x and y at an additional number of points so that the final surface is calculated on a rnxsm grid. Consider Equation (1), revised for the new, larger grid:

$$z(x,y) = \sum_{i=0}^{rn-1}\sum_{j=0}^{sm-1} c'_{i,j} g(x-x_{i,j}, y-y_{i,j}) \qquad , \tag{3}$$

where

$$\begin{array}{ll} c'_{i,j} = c_{i\div r, j\div s} & i \,(\mathrm{mod}\ r) = 0, j\,(\mathrm{mod}\ s) = 0, \\ =0 & otherwise. \end{array}$$

That is, we assume the surface is constructed by the placing of objects at each of the new grid points, but zero coefficients are associated with all objects except those placed at the original data points. Now, taking the DFT of Equation (3), we get

$$Z(u_{k,l}, v_{k,l}) = \sum_{i=0}^{rn-1}\sum_{j=0}^{sm-1} c'_{i,j} G'(u_{k,l}, v_{k,l}) e^{-2\pi i(\frac{u_{k,l}x_{i,j}}{rn}+\frac{v_{k,l}y_{i,j}}{sm})} \qquad \begin{array}{l} k = 0, ..., rn-1 \\ l = 0, ..., sm-1, \end{array}$$

i.e.,

$$Z(u_{k,l}, v_{k,l}) = G'(u_{k,l}, v_{k,l}) C'(u_{k,l}, v_{k,l}) \qquad ,$$

where $G'(u,v)$ is the DFT of $g(x,y)$, defined on the finer grid, and $C'(u,v)$ is the DFT of the array $c'_{i,j}$.

The interpolated surface can be formed by taking the inverse DFT of the above expression:

$$z(x_{i,j}, y_{i,j}) = \mathbf{F}^{-1}[G'(u_{k,l}, v_{k,l})C'(u_{k,l}, v_{k,l})] \qquad .$$

Note that, in finding the $c_{i,j}$'s in Equation (2), we took the inverse DFT of $\frac{Z(u,v)}{G(u,v)}$, then stretched these $c_{i,j}$'s by adding zeros at the points corresponding to the new interpolation points, and finally took the DFT of the stretched coefficients to calculate $C'(u,v)$. These steps are in fact unnecessary, for we can calculate the $C'(u,v)$'s directly from the $\frac{Z(u,v)}{G(u,v)}$'s. The similarity therеom of DFT theory [5] is required:

$$C'(u_{k,l}, v_{k,l}) = \frac{1}{rs}\frac{Z(u_{k,l}\,(\mathrm{mod}\ n), v_{k,l}\,(\mathrm{mod}\ m))}{G(u_{k,l}\,(\mathrm{mod}\ n), v_{k,l}\,(\mathrm{mod}\ m))} \qquad \begin{array}{l} k = 0, ..., rn-1 \\ l = 0, ..., sm-1. \end{array} \tag{4}$$

2.5 Algorithm

We can now write down our interpolation algorithm:

1. Given the data array $z(x_{i,j}, y_{i,j})$, find its DFT, $Z(u_{i,j}, v_{i,j})$
2. Find the DFT, $G(u_{i,j}, v_{i,j})$, of the kernel function, $g(x_{i,j}, y_{i,j})$
3. $C(u_{i,j}, v_{i,j}) = \frac{Z(u_{i,j}, v_{i,j})}{G(u_{i,j}, v_{i,j})}$
4. Calculate $C'(u_{i,j}, v_{i,j})$, using Equation (4)
5. Calculate $G'(u_{i,j}, v_{i,j})$ for the larger interpolation grid
6. $Z'(u_{i,j}, v_{i,j}) = C'(u_{i,j}, v_{i,j})G'(u_{i,j}, v_{i,j})$
7. Find the interpolated surface by taking the inverse DFT of $Z'(u_{i,j}, v_{i,j})$.

Note that for selected kernel functions Steps 2 and 5 could be precomputed for standard-size grids. As an alternative, these steps can sometimes be accomplished by analytic means if the analytic form of the kernel function is known.

3. Discussion

For purposes of illustration, let us compare the computational efficiency of this method on a regular grid with the cost of the usual non gridded formulation of the multiquadric. Of course, since the usual formulation deals with irregularly spaced data, we would not expect it to compare favorably with this method; such a comparison nevertheless confirms the advantages of our technique. Consider a square nxn grid of data points on which we want an interpolated surface over a $rnxrn$ grid. The usual multiquadric formulation solves a n^2xn^2 system of linear equation at a cost proportional to n^6, and calculates $rnxrn$ sums of terms at a cost proportional to r^2n^4. If it is assumed that $n > r$, this cost is dominated by the n^6 term.

The algorithm outlined above is dominated by the cost of the DFTs in Steps 5 and 7. We use the fast Fourier transform to implement the DFT. This means that we pad our data with zeros to force the dimension size of the grid to be a power of 2. At worst, our grid is $2rn$x$2rn$. The cost of the DFT is proportional to $4r^2n^2log2rn$. Even if r were as great as n, this cost would be proportional only to $n^4 logn$. From an empirical standpoint, the algorithm outlined is faster for n (and k) of the order of 10.

The outlined algorithm places little limitation on the type of kernel function employed. Not only smooth, but also rough functions may comprise the basic objects from which the surface is built. We have used, inter alia, cones, hyperboloids, and Gaussian-shaped objects, some of which had fractal texture added to them. In Figures 1-4 we show profile plots. Figure 1 shows a real surface, Figure 2 the sampling grid we used to select data points. In Figure 2 the profiles depict what would have been obtained if we had used bilinear interpolation to build the surface. Figure 3 reveals the resultant surface when Gaussian kernel functions were used, while Figure 4 was obtained with a kernel function that had fractal texture added to a Gaussian base. When we compare the fitted surface to ground truth, the average error for the smooth kernel functions used by us, is approximately one percent of the data height range. As with any fitting technique, we cannot construct surface features that are not described by the sampled data.

We indicated that one reason for investigating global surface interpolation techniques was the need to calculate reliable estimates of surface curvature. In our preliminary trials with synthetic surface data the constructed surface appears to allow adequate surface curvature estimation. This will be tested further in future applications.

4. Conclusions

We have presented a method of surface interpolation that is computationally efficient. The reconstructed surface is fitted globally to enable the data rather than an implicit surface model to control the construction process. The method makes it possible to build not only the more customary smooth interpolated surface, but the roughly textured type as well.

Surface reconstruction methods provide a means of using the hypothesis-and-test approach in image analysis. They provide a mechanism for using image information that only constrains rather than specifies 3-D world parameters. The outlined algorithm is a tool for hypothesizing a broad range of surface types.

References

1. Hardy, R.L., Multiquadric Equations of Topography and Other Irregular Surfaces, *J. Geophysical Res.*, Vol. 76, No. 8, 1971, pp. 1905-1915.
2. Stead, S.E., Smooth Multistage Multivariate Approximation, Ph.D. thesis, Division of Applied Mathematics., Brown University, Providence, Rhode Island, 1983.
3. Froberg, C-E., *Introduction to Numerical Analysis*, Addison-Wesley Publishing Co., 1969.
4. Hunt, B.R., The Application of Constrained Least Squares Estimation to Image Restoration by Digital Computer, *IEEE Trans. Computers*, Vol. C-22, No. 9, 1973, pp. 805-812.
5. Bracewell, R.N., *The Fourier Transform and Its Applications*, McGraw-Hill Book Co., 1978.

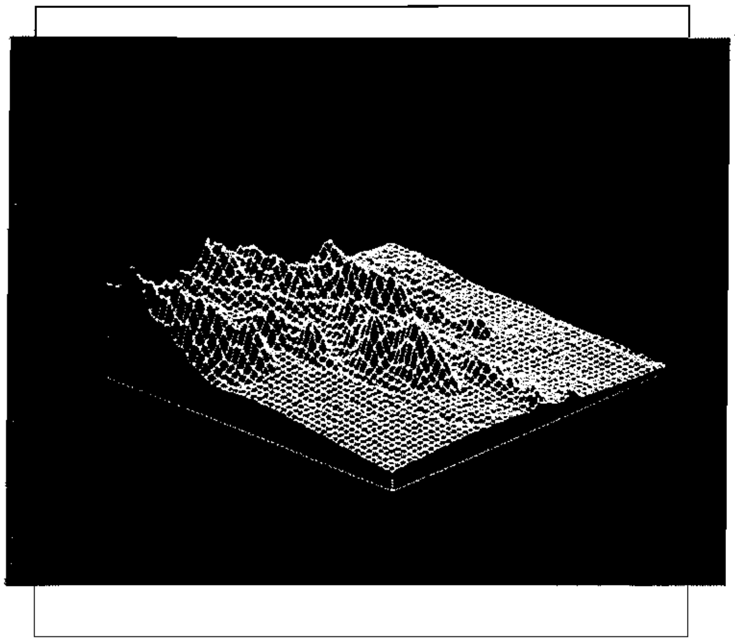

Figure 1. Real Surface Used as Ground Truth

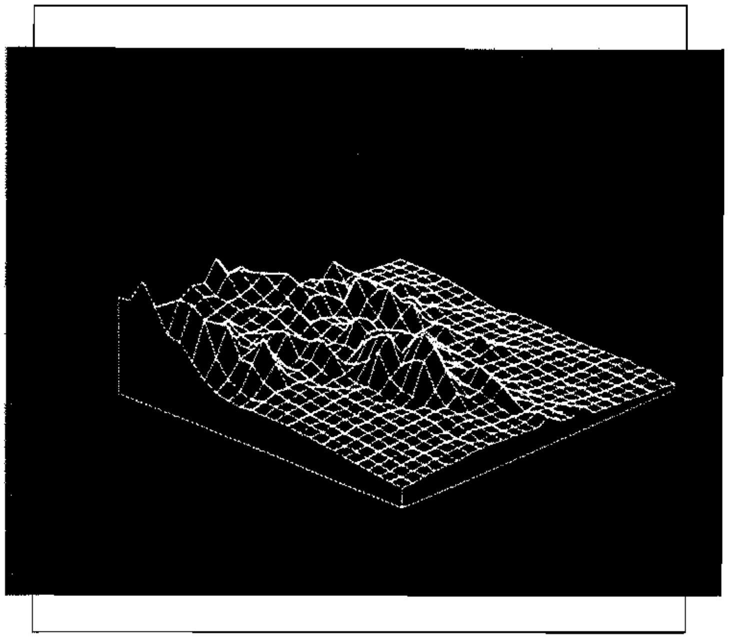

Figure 2. Data Sampled at Grid Intersections

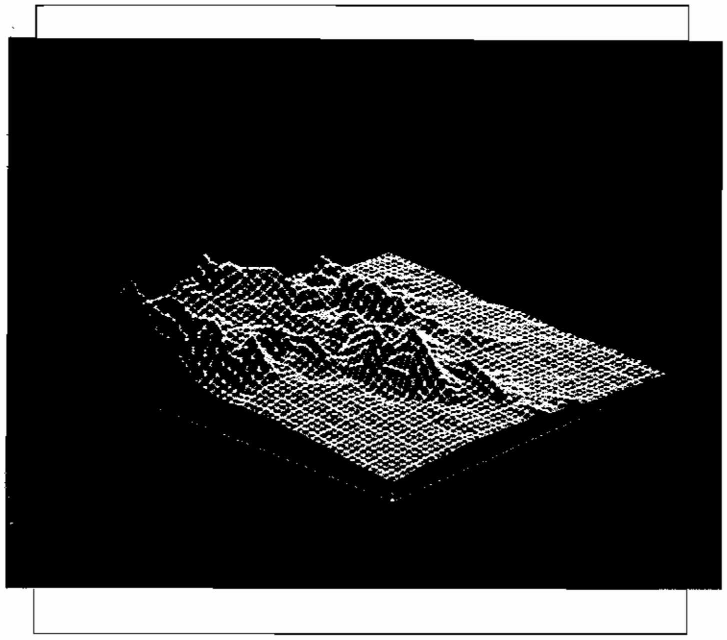

Figure 3. Reconstructed Surface by Means of Gaussian Kernel Functions.

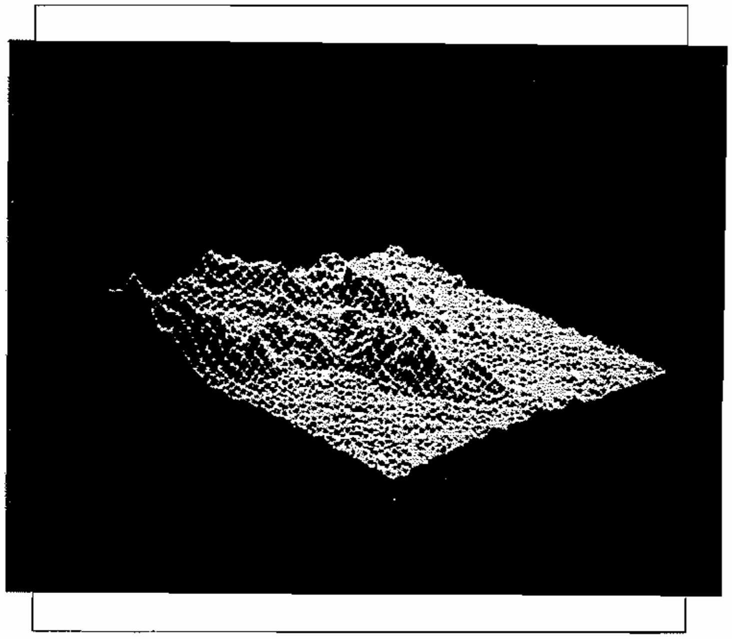

Figure 4. Reconstructed Surface by Means of Fractal Textured Gaussian Kernel Functions.