Visual Surface Interpolation: A Comparison of Two Methods.

by

Terrance E. Boult[†]

CUCS-189-85

Columbia University Computer Science Department
NYC, NY 10027. tboult@cs.columbia.edu

§0 Abstract

We critically compare 2 different methods for visual surface interpolation. One method uses the reproducing kernels of Hilbert spaces to construct a spline interpolating the data, such that this spline is of minimal norm. The other method, presented in Grimson (1981), recovers the surface of minimal norm by direct minimization of the norm with a gradient projection algorithm. We present the problem that each algorithm is attempting to solve, then briefly introduce both methods. The main contribution is an analysis of each algorithm in terms of the worst case running time (serial processor), space complexity, and rough estimates of the running time and space costs for massively parallel implementations. We then conclude with a discussion of the differences in the internal representation of the surface in both algorithms.

[†]This work supported in part by DARPA grant N00039-84-C-0165 and in part by NSF grant MCS-782-3676

§1 Introduction.

It has been shown that when presented with sparse depth data (say from random dot stereograms) the human visual system infers a smooth surface passing through these data points. The problem of computer visual surface interpolation is to take a sparse set of depth values and calculate the surface passing through these points that seems to model the surface that humans infer from those same data points. Grimson (1981) presented a computational model of this process in the human visual system, and suggested an algorithm that may be used to recover the perceived surface from the depth data.

Although it may be fruitful from a psychological point of view, to develop algorithms that may be biologically realizable, this restriction may increase the computational cost of the algorithms. Therefore we compare, without regard to biological feasibility, two methods for the solution of this visual surface interpolation problem with the intent to determine which is a more efficient algorithm for use in computer vision.

The first of these methods is that presented in Grimson (1981). Grimson's approach was to represent the surface by a grid of depth values, and to use nonlinear programing techniques and directly minimize the "quadratic variation" or bending energy of the surface. Because the problem was to interpolate the given data, Grimson employed a constrained optimization algorithm called the gradient projection algorithm. Because of this we shall refer to Grimson's approach as the *gradient projection based algorithm.*

The second method we shall examine is the method of reproducing kernels. This method uses the reproducing kernels of Hilbert or semi-Hilbert spaces to calculate splines of minimal norm. The use of surfaces of minimal norm as the visual surface interpolating the depth data is done in spirit of the minimization approach used in Grimson (1981). The use of reproducing kernels to recover splines of minimal norm is not a new idea. It has been studied by Duchon (1976a, 1976b), Meinguet (1979a, 1979b) and more recently by Franke (1982, 1983), Franke and Nielson (1980) (all but Meinguet called them thin plate splines). However, the method has not previously been given serious consideration for visual surface interpolation – probably because it seems unlikely that the human visual system uses such an approach.

In section 2 we derive an precise formulation of the visual surface interpolation problem. Section 3 presents details of both of the above algorithms. In section 4 we present and compare algorithmic properties (time, space and parallel time complexity, optimality and accuracy of the solution) of both methods. Section 5 is a discussion of the representational advantages of splines in functional form over simply having a grid of depth values. In section 6 we discuss the extensibility of both methods to other spaces of functions, and other norms. Section 7 presents our conclusions.

§2 The Problem.

A naïve formulation of the visual surface interpolation problem might be :

> to find "the best approximation" to a surface using only the knowledge of a number of given points thereon, where we require the surface to be interpolatory, i.e. to pass through all the given data.

A major difficulty with this formulation is that it is not well posed, inasmuch as the information does not uniquely determine a solution. In fact, given any set (of zero measure) of points on a surface there are infinitely many surfaces interpolating those points. To alleviate this problem, we must somehow restrict the class of allowed surfaces and/or give some method of ranking the "plausibility" of a surface.

One of the classical ways of insuring that a problem has a unique solution (applied to visual surface interpolation in Grimson (1981) and Kender, Lee, and Boult (1985)) is to use a functional on the surface as a measure of the "unreasonableness" of the surface, and to restrict the allowed class of surfaces to make it a Hilbert or semi-Hilbert space and make the functional a norm or semi-norm on this space. This formulation insures that there exists a unique solution to the problem of finding a surface from the allowed class which minimizes the functional (and hence is the most reasonable). Throughout this paper we shall *assume* that this type of formulation is appropriate for the problem of visual surface interpolation. We shall not investigate which classes of surfaces are most appropriate, nor which functionals may be good measures of the unreasonableness of a surface. Readers in this aspect of the problem may consult Boult (1986).

In what follows we choose to define "best approximation" in terms of minimal error. We *assume* that error can be measured by a norm with respect to the given class of functions. The norm might be the **sup** norm (i.e. the maximal difference between the actual surface and the approximation), or the L^2 norm (integral of the square of the difference at each point). The error may be measured in either a relative (e.g. error of 5%) or an absolute sense (e.g. the surfaces never differ by more than .1 mm) depending on the goals of the user. Finally there error may be measured in the worst case, or on the average (with respect to some measure).

Combining these assumptions a precise formulation of the problem of visual surface interpolation from sparse depth data becomes :

Let F_1, the space of allowed surfaces, be a Hilbert or semi-Hilbert space. Let F_2 be the elements of F_1 restricted to a finite domain D (since we are only interested in recovering a finite portion of a possibly infinite surface). Let $\Theta(f)$: $F_1 \rightarrow \Re$, be a functional measuring the "unreasonableness" of a surface (i.e. the more reasonable a surface f, the smaller $\Theta(f)$), where Θ is a norm on F_1 (a semi-norm if F_1 is semi-Hilbert). Let $N(f) \equiv \{z_1,\ldots,z_k\} \equiv \{ f(x_1,y_1), \ldots, f(x_k,y_k) \}$ be the allowed information (i.e. the allowed input to solve the problem is k depth values.) Then the *visual surface interpolation problem* is to find (using only N(f)) $f^* \in F_1$, such that $\Theta(f^*) = \min_{g \in F_1} \Theta(g)$.

Kender, Lee and Boult (1985) show (as a special case of work on information based complexity see Traub and Woz'niakowski (1980), or Traub, Wasilkowski and Woz'niakowski (1983)) that given the above formulation the surface minimizing the functional $\Theta(f)$ will also be the minimal error surface with respect to the class F_1 for almost *any error* norm.

One functional to measure unreasonableness that is used by both Grimson (1981) (who called it quadratic variation) and Kender, Lee and Boult (1985) is given by :

$$\Theta(f) \equiv \left[\iint_{\Re^2} \left(\frac{\partial^2 f}{\partial x^2}\right)^2 + 2 \cdot \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 + \left(\frac{\partial^2 f}{\partial y^2}\right)^2 \right]$$

We note that this is just one particular choice for the functional and that this functional is the norm or semi-norm for a number of different classes, see Boult (1985a). It is known that different functionals (and the associated classes for which they are norms) give rise to different interpolation problems, and hence to different interpolating surfaces. The reader interested in other norms and their associated classes should consult Grimson (1981), Boult (1985a) or Boult (1986).

§2.1 Allowed Information.

We now consider the allowed form of the information $N(f) \equiv \{z_1,\ldots, z_k\} \equiv \{f(x_1,y_1), \ldots, f(x_k,y_k)\}$. Each piece of information consists of 2 components, a function value and the location of that evaluation. Throughout this paper we shall assume we are given the value (height above or depth below a reference plane) of the surface at known points in x–y space. Note that this pre - cludes the use of surface gradients, normals, curvature, etc.. This pure depth data might be the result of a stereo based process, a rangefinder or be synthetically generated. We consider two separate ways of determining the other component, the locations for the information. The first method is to obtain the information from triangulation between matched points in the zero crossings of the Laplacian of the Gaussian of a stereo pair of intensity images (here after $\nabla^2 G$ zero crossing information). This type of information was proposed by Marr and Poggio (1979) as that available in the human visual system, and was used by Grimson in the development of his computational study of surface interpolation in the human visual system. Another method of choosing the location of the information is to use some fixed and regular pattern, e.g. a regular square grid of r points per side, each point separated by a distance h, (thus the number of depth samples is $k = r^2$). This regular grid information would be very difficult, if not totally impossible, to obtain in a passive stereo system but is easily obtained from active ranging systems. The major difference then between the two types of information is the location of the information samples; which may be effected by the availability of an active ranging system (an option not open to the human visual system).

We note that information derived from the zero crossings of $\nabla^2 G$, yields locations (both the number of and position of) that depend in a very nonlinear way on the surface viewed. This extra information, (i.e. the knowledge that information is evaluated at the location of the zero crossings of the intensity image) is *not* used by any algorithm known to this author. After a casual reading, it might seem that Grimson's algorithm should take advantage of this extra information, inasmuch as the surface consistency constraint Grimson (1981, p130), shows a relationship between the location of the zero crossings and the variation of a surface. However, Grimson's algorithm is based on the choice a functional (quadratic variation) that does not truly embody the surface consistency constraint, because it minimizes the total variation of the surface and not the variation between zero crossings. Note that it is not necessarily true that the interpolating surface with minimal total variation also has minimal variation between each set of zero crossings. To see this consider a interpolating surface that has almost zero variation between all but one pair of zero crossings (and hence generally satisfies the surface consistency constraint), but whose variation between that pair is arbitrary large (maybe the surface is not even continuous at one of the zero crossings). Such a surface may have arbitrarily large total surface variation but may have minimal variation between zero crossings (except the one pair).

Since neither the reproducing kernel algorithm nor the gradient projection based algorithm make special use of $\nabla^2 G$ type information, we shall freely compare them with respect to both $\nabla^2 G$ zero crossings and regular grid information. Here after, we shall let the number of information points (regardless of its origin) be denoted by k, and the set of information by $N(f) \equiv \{z_1, \ldots z_k\} \equiv \{f(x_1, y_1), \ldots, f(x_k, y_k)\}$.

§2.2 The Desired Output.

The final component in the formalization of the problem is to specify what it means to find an approximation. i.e. we must consider the representation of the desired solution. Though there are many representations we might chose, we shall examine only those two used by the methods under consideration.

The first and simplest representation of the surface is as function values at some predefined points (e.g. on a 2-d mesh). This is the representation used by the gradient projection based algorithm. In this algorithm, the grid is a uniform 2d mesh, large enough to include all information points. We shall let the total number of points in this grid be n.

The other representation of the solution surface we shall consider is as a function of x and y, which can be evaluated at any point. Obviously given this representation the first representation can be recovered but not visa versa.

Note that the user may be interested in recovering the interpolated surface at fewer than the n points used in the first representation. Hereafter let p be the number points at which the interpolatory surface is to recovered. We need not require that the p solution points contain or be contained in the k information points.

Finally we note that it would be improper to compare two different methods if they were calculating the surface in different representations. Therefore, throughout sections 3-6 we shall assume the reproducing kernel method is used first to calculate its spline representation then the spline is evaluated at the p points the solution is desired at, which is a subset of the n points used in the gradient projection based algorithm.

§3 Description of the Two Methods of Solution.

In this section we briefly describe the two methods of solution to the surface interpolation problem, which we shall be comparing in this paper. We shall refer to the two methods as the *gradient projection based algorithm* and *reproducing kernel algorithm*. We start by discussing the theoretical basis that they have in common.

Neither method actually requires that $\Theta(f)$ be quadratic variation as in (2.1), only that it be a norm or semi-norm on the space F_1. Both methods rely on the theorem from functional analysis that states if $\Theta(f)$ is a norm over the F_1 and F_1 is a Hilbert space then there exists a *unique* function from F_1 minimizing $\Theta(f)$. (If $\Theta(f)$ is a semi-norm and F_1 is only a semi-Hilbert space then the solution exist and is unique up to a member of the null space of $\Theta(f)$.)

The two methods differ in how they minimize the functional $\Theta(f)$ (with respect to the class of functions F_1) and in their representation of the solution.

§3.1 The Gradient Projection Based Algorithm.

We now examine the gradient projection based algorithm as discussed in Grimson (1981). Inherent in the development of this algorithm is the representation as "explicit depth values at all locations within a Cartesian grid of uniform spacing" Grimson (1981, p180). It is also assumed that the information is given at points within this grid, and for simplicity that the the grid is square with size m x m (where m = $\sqrt{n}$). In the following discussion each grid point is represented by its coordinate location (i,j), $1 \leq i,j \leq m$, and the solution surface is represented as its value at each grid point, i.e. $s_{i,j}$. Grimson begins by deriving a discrete analogue of the functional $\Theta(f)$, and then solves the discrete minimization problem given by:

$$\text{minimize} \quad \sum_{i=1}^{m-2} \sum_{j=0}^{m-1} (s_{i-1,j} - 2s_{i,j} + s_{i+1,j})^2$$

$$+ \sum_{i=0}^{m-1} \sum_{j=1}^{m-2} (s_{i,j-1} - 2s_{i,j} + s_{i,j+1})^2 \qquad (3.1)$$

$$+ \sum_{i=0}^{m-2} \sum_{j=0}^{m-2} (s_{i,j} - s_{i+1,j} - s_{i,j+1} + s_{i+1,j+1})^2$$

subject to $s_{i,j} = f(i,j) \ \forall \ f(i,j) \in N(f)$.

To solve this problem he uses a nonlinear programing algorithm called the gradient projection algorithm (actually he seems to use a modified version of this algorithm usually known as Goldfarb's algorithm see Avriel (1976). To implement this algorithm he develops stencils (see Grimson (1981, p183-184)) to allow the calculation of the gradient of the objective function. To determine the amount to move in this direction, one must calculate the minimum of the objective function in that direction. To do this Grimson calculates the value α that minimized the expression:

$$\sum_{i=1}^{m-2} \sum_{j=0}^{m-1} (s_{i-1,j} - 2s_{i,j} + s_{i+1,j} + \alpha d_{i-1,j} - 2\alpha d_{i,j} + \alpha d_{i+1,j})^2$$

$$+ \sum_{i=0}^{m-1} \sum_{j=1}^{m-2} (s_{i,j-1} - 2s_{i,j} + s_{i,j+1} + \alpha d_{i,j-1} - 2\alpha d_{i,j} + \alpha d_{i,j+1})^2 \qquad (3.2)$$

$$+ \sum_{i=0}^{m-2} \sum_{j=0}^{m-2} (s_{i,j} - s_{i+1,j} - s_{i,j+1} + s_{i+1,j+1} + \alpha\, d_{i,j} - \alpha\, d_{i+1,j} - \alpha\, d_{i,j+1} + \alpha\, d_{i+1,j+1})^2$$

where $d_{i,j}$ is the negative of the value obtained from the convolution of the appropriate stencil (see Grimson (1981, p183-184)) with $s_{i,j}$ (i.e. the negative gradient direction or direction of steepest decent of minimizing the surface variation). He the concludes that $\alpha = \alpha_1 / \alpha_2$ where

$$\alpha_1 = \sum_{i=1}^{m-2} \sum_{j=0}^{m-1} (s_{i-1,j} - 2s_{i,j} + s_{i+1,j})^2 (d_{i-1,j} - 2d_{i,j} + d_{i+1,j})^2$$

$$+ \sum_{i=0}^{m-1} \sum_{j=1}^{m-2} (s_{i,j-1} - 2s_{i,j} + s_{i,j+1})^2 (d_{i,j-1} - 2d_{i,j} + d_{i,j+1})^2 \qquad (3.3)$$

$$+ \sum_{i=0}^{m-2} \sum_{j=0}^{m-2} ((s_{i,j} - s_{i+1,j} - s_{i,j+1} + s_{i+1,j+1})^2 (d_{i,j} - d_{i+1,j} - d_{i,j+1} + d_{i+1,j+1})^2)$$

and

$$\alpha_2 = \sum_{i=1}^{m-2} \sum_{j=0}^{m-1} (d_{i-1,j} - 2 \cdot d_{i,j} + d_{i+1,j})^2$$

$$+ \sum_{i=0}^{m-1} \sum_{j=1}^{m-2} (d_{i,j-1} - 2 \cdot d_{i,j} + d_{i,j+1})^2 \qquad (3.4)$$

$$+ \sum_{i=0}^{m-2} \sum_{j=0}^{m-2} (d_{i,j} - d_{i+1,j} - d_{i,j+1} + d_{i+1,j+1})^2$$

Thus the complete gradient projection based algorithm employed by Grimson consists of the following 5 steps:

Step 1: Determine a feasible initial surface (any surface interpolating the information will do).

Step 2: Compute the negative of the gradient direction (the $d_{i,j}$ above) by taking the the convolution of the current approximation (the $s_{i,j}$'s) with the stencils (setting the $d_{i,j} = 0$ if i,j is an information point).

Step 3: Compute α_1 and α_1 (from formulas (3.3) and (3.4) above) and then set $\alpha = \alpha_1 / \alpha_2$.

Step 4: Refine surface approximation (i.e. for each i,j set $s_{i,j} := s_{i,j} + \alpha \cdot d_{i,j}$).

Step 5: If $d_{i,j} \leq \varepsilon \ \forall \ i,j \leq m$ then approximation is complete
Else goto Step 2;

§3.2 The Method of Reproducing Kernels.

The method of reproducing kernels calculates a spline function that *exactly* solves the continuous problem of finding the function from F_1 minimizing $\Theta(f)$. There are at least two differ - ent algorithms based on the use of reproducing kernels, we shall present only one. The interested reader may consult Boult (1985b) or Boult (1986) for a more detailed discussion of both algorithms based on reproducing kernels. The following discussion of reproducing kernels for interpolation is based on the theoretical work of Meinguet (1979a, 1979b).

For this method to be appropriate it is sufficient to have F_1 be a semi-Hilbert space and $\Theta(f)$ the associated semi-norm with null space Π_1. (Throughout this paper Π_1 is the space spanned by $\{1,x,y\}$). To insure uniqueness of the solution we must assume that the information $N_k(f)$ contains a $\Pi_1$unisolvent subset, i.e. there exists a set J of indices (a subset of the index set I = 1... k) and associated information points x_j,y_j with information values z_j such that for each element of J (there are 3 in the present case) there exists a unique $p_j(x,y) \in \Pi_1$ such that for all j, j' from J, $p_j(x_j,y_j) = 1$ and $p_j(x_{j'},y_{j'}) = 0$ if $j \neq j'$. Note that if N_k contains evaluations 3 or more non-colinear points, then $N_k(f)$ will contain a $\Pi_1$unisolvent subset. (This restriction on the information having at least 3 non-colinear points also applies to the gradient projection based algorithm.)

In the development of this method we use that fact that we can separate the space F_1 into $X_0 \oplus \Pi_1$. $X_0 \equiv \{g \in F_1\colon g(x_j,y_j) = 0, \ \forall \ j \in J\}$ where $\oplus$ is a (topological) direct sum. With this decomposition X_0 is a Hilbert space with $\Theta(\cdot)$ as a norm (not a semi-norm). Given the repro - ducing kernel $K_0((s,t);(x,y))$ of X_0 (which can be expressed in terms of the reproducing kernel of F_1 and the functions $p_j(x,y)$, see Boult (1985b)) the spline surface, of minimal $\Theta(f)$ norm, which interpolates the information N(f) is given by :

$$\sigma_M(x,y) = \sum_{i \in I-J} \gamma_i \cdot K_0((x_i, y_i); (x, y)) + \sum_{j \in J} z_j \cdot p_j(x_j,y_j) \qquad (3.5)$$

where the coefficients γ_i can be calculated from the (k-3) by (k-3) dense linear system :

$$\sum_{i\in A-J} K_0(x_k,y_k;\, x_i,y_i)\cdot\gamma_i \;=\; z_i - \sum_{j\in J} z_j\cdot p_j(x_j,y_j) \quad \forall\, k\in I. \qquad (3.6)$$

The reproducing kernel method then consists of the three following steps:

Step 1: Calculate the matrix of coefficients for the left hand side of equation (3.6).

Step 2: Compute γ_i, i =1..k-3, the solution to equation (3.6).

Step 3: Compute the value of interpolating surface at all solution points using equation (3.5).

Note that Step 1 and 2 are necessary parts of the algorithm, whereas step 3 is simply to allow comparison of this method with the gradient projection based algorithm. Also note that for fixed regular data it is possible to precompute the Cholesky decomposition of the coefficient matrix (which is determined entirely by the location of the information), and then step 2 is simply the calculation of the γ_i's using back substitution.

A proof that the above spline is of minimal norm can be found in Meinguet (1979a, 1979b).

§4 Comparison of Computational Issues of The Two Methods.

In this section we provide an analysis and comparison the two visual surface interpolation methods on a number of computational issues. These issues and the subsection in which they are treated are:

§4.1 Time complexity,
§4.2 Space complexity,
§4.3 Inherent Parallelism and Parallel Time Complexity,
§4.4 Optimality and Accuracy of Solution.

For all of the comparisons we shall assume that k, p, n are defined as in sections 2 and 3. When we refer to the steps of the gradient projection based algorithm and the reproducing kernel method, we are referring to the steps as defined in sections 3.1 and 3.2 respectively. A synopsis of the results can be found in Table 4.1.

§4.1 Time Complexity.

First lets us estimate an upper bound on the worst case running time (assuming each arithmetic operation costs unity) of the reproducing kernel method when the information is from $\nabla^2 G$ zero crossings. Obviously step 1 costs $O(k^2)$, and step 3 costs $O(kp)$. For step 2, using Cholesky decomposition (the matrix is positive definite), the cost will be $\frac{1}{6}k^3$. Therefore the worst case cost is $\frac{1}{6}k^3 + O(k^2 + kp)$. (A careful analysis of the current implementation results in a cost of $\cong \frac{1}{6}k^3 + 70kp + O(k^2)$, but this depends on the choice of the space F_1, norm $\Theta(f)$ and the associated reproducing kernel.)

Now we note that if the information is gathered on a regular grid, then we can do the Cholesky decomposition once (a precomputation), and store the results. Given this decomposition we can reduce the cost of step 2 to $O(k^2)$, and the overall cost to $O(k^2 + kp)$. (In fact, given the decomposition for a grid of size $r \times r$ we also have the decomposition for all smaller grids.)

Now let us estimate an upper bound on the worst case running time of the gradient projection based algorithm. Step 1 of that algorithm obviously cost $O(n)$. Examination of the stencils given in Grimson (1981, p183) yields a cost per iteration for step 2 of approximately 26n. For step 3, equations (3.3) and (3.4) yield a per iteration cost of approximately 24n. Finally for step 4 costs 2n per iteration. Thus the total per iteration cost of the algorithm is $\cong 50n$. We take the number of iterations to be n, which is the upper bound on the number iteration of the Goldfarb's algorithm (the nonlinear programing algorithm on which the method is based) for problems of this type (see Avriel (1976, p436)). (Note that this is far better than the $O(n^2)$ iterations which Terzopoulos (1984, p103) suggests the gradient projection based algorithm takes.) Combining the above we arrive at a total estimated cost for the gradient projection based algorithm of approximately $50n^2$. (We note that the number of iterations will actual depend on the number of, value of, and location of the information. Thus one may be able to get a better estimate for fixed regular data. However one can easily show that no placement of data can result in less than $\sqrt{(n/k)}$ iterations and worst case placement of fixed data can easily be shown to result in at least $2\sqrt{n}$ iterations. Both of these bounds are trivial, and the actual lower bound is probably $O(n)$.)

Thus for $\nabla^2 G$ zero crossing information the reproducing kernel method is faster whenever $\frac{1}{6}k^3 + 70kp < 50n^2$. And for grid data, the reproducing method is faster when $O(k^2) + 70kp < 50n^2$.

§4.2 Space Complexity.

The space required for the reproducing kernel method is $.5k^2 + O(k)$ for steps 1 and 2 (independent of the type of information), and $k+p + O(1)$ for step 3. (This assumes that the user needs all p values at the same time. If the user can use the points sequentially, then the space for step 3 is simply $k + O(1)$.)

The space complexity of the gradient projection based algorithm can be calculated by examining equation (3.2). Although algorithms trying to obtain minimal time complexity may use more space, each iteration of the algorithm can be programed using only $2n + O(\sqrt{n})$ space. Note that no savings in space is obtained if the user only requires the solution points one at a time.

Therefore the reproducing kernel method will use less space whenever $\min(k^2,k+p) < 2n$, (i.e. whenever $k < \sqrt{(2n)}$ because $p \leq n$).

§4.3 Inherent Parallelism and Parallel Time Complexity.

In this subsection we examine the sources of parallelism in both of the methods, and estimate their parallel time complexity. True values may vary depending on the instruction set of the parallel machine being used, its topology, its memory limitations, number of processors and its mode of operation (SIMD or MIMD).

There are four different sources of parallelism in the reproducing kernel method. The first is the evaluation of the spline function at one point, which involves the evaluation and summation a weighted kernel function at each of the k information points. This can be parallelized in a straight forward SIMD fashion to run in time $O(\log k)$.

The second source of parallelism in the reproducing kernel method is the evaluation of the p surface solution points. These points can easily be evaluated simultaneously, again in a SIMD fashion, resulting in a factor of p speedup. Combining the first two parallelizations we could speed up the surface reconstruction (given the coefficients of the spline) from 70kp to $O(\log k)$.

The third form of parallelism come from the calculation of the coefficients of the spline. Given the decomposition of the coefficient matrix, we can compute the coefficients in parallel in time k.

The final type of parallelism is that inherent in the solution of a kxk linear system. This has been studied elsewhere and in general one can gain a speed up factor of k.

Thus our estimate of the parallel running time for the reproducing kernel algorithm is $O(k^2)$ if we must decompose the matrix coefficients (as we must for $\nabla^2 G$ information), and $O(k)$ if the decomposition is precomputed (as it is may be for regular gird data).

The parallelisms inherent in the gradient projection based algorithm include the calculation of the gradient direction, local calculation of each of the terms needed for the calculation of the parameter α, and updating the surface. These will reduce the number of operations per iteration by an estimated amount of 26, 20, 2 respectively. Furthermore, given the local terms for the calculation of the parameter α, we can speed up that calculation by a factor of $(\log n) / n$ by using log reduction for the summation. Note that the number of iterations cannot be reduced by parallel implementation. The total estimated time complexity of the parallel gradient projection based is $O(n \cdot \log n)$.

Based on these estimates, a parallel implementation of the reproducing kernel method would be faster than a parallel implementation of the gradient projection based algorithm when $O(k^2) < O(n \cdot \log n)$ and the location of the data is allowed to vary, or when the data is fixed and $O(k) < O(n \cdot \log n)$.

§4.4 Optimality and Accuracy of Solution.

Both methods under consideration started off with the idea of finding the surface of minimal norm ($\Theta(f)$) over F_1, a Hilbert space (or semi-Hilbert space). This had the advantage of resulting in a unique specification of the surface to recover. As mentioned before it is known that such a surface is also a minimal error solution among all surfaces in the class F_1 that interpolate the data, and that this minimal error property holds for almost any reasonable definition of error, see Kender, Lee, and Boult (1985). Thus theoretically both methods are attempting to find an optimal error interpolant from F_1.

The reproducing kernel method theoretically does calculate this optimal error surface. The errors in the coefficients of the spline surface, introduced by the approximate solution of the linear system, however result in the algorithm reconstructing a different surface. The magnitude of the error in these coefficients depend on the condition number of the linear system, which in turn depends on the placement of the information points. Initial experiments suggest that for a regular grid of information the condition number is approximately $19.5\ k^2$. Because Cholesky decompo - sition and back substitution are numerically stable (given proper implementation), we know the resulting coefficients differ from the true spline coefficients by at most $e' \equiv c' \cdot 2^{-t} \cdot 19.5\ k^2$, where t is the number of bits in the mantissa of the floating point representation on the machine and c' is a fixed constant depending of the floating point implementation. Then the maximum error of any surface reconstruction (from the optimal surface, not from the surface generating the information) is $< k \cdot e' \max(K_0)$. Note that this is totally independent of the number of reconstruction points, but depends on the distance of the reconstructed points from the information points.

The gradient projection based algorithm however leaves its theory behind. The first step in the method is the discretizaton of the functional to minimize. This discretizaton is well studied in mathematical physics, and the error introduced by it is $O(h^2)$ where h is the distance between grid points. The method then attempts to minimize this discrete functional without regard for the space F_1, thus its solution may not even be a "feasible solution". (Note that this is not as simple a problem to over come as it might seem because the discretized version of $\Theta(f)$ is no longer a norm or semi-norm on F_1 so there is not even an assurance of a surface minimizing the discretized version of $\Theta(f)$ existing, let alone being reachable by a sequence of surfaces from F_1.) Finally there is the error introduced by the gradient projection portion of the algorithm, and by terminating the algorithm before it has computed the exact solution (to the perturbed problem.) Currently we do not have estimates on the error of the algorithm, but the work of Terzopoulos (1984) suggests that the error does goes to zero, albeit very slowly, as both the number of points and number of iterations grows.

	Reproducing(Fixed)	Reproducing(Varying)	Gradient Projection
Time Complexity	$O(k^2 + kp)$	$({}^1/_6\, k^3+O(k^2+ kp)$	$50\, n^2$
Space Complexity	$(.5k^2+O(kp))$	$(.5k^2+O(kp))$	$(2n+O(\sqrt{n}))$
Parallel Time Complexity	$O(k)$	$O(k^2)$	$O(n \log n)$

Table 4.1: Comparison of essential properties of 2 algorithms. Here k is the number of information (depth) samples; p is the number of points in the desired solution; n is the number of points in the grid used by the gradient projection based algorithm. The fixed and varying in the titles refer to situtations where the location of the data is fixed and varying respectively. This affects the performance of the reproducing kernel algorithm.

§5 Advantages of Spline Representation.

In this section we discuss the advantages of the spline representations over the mesh / grid representation used by the gradient projection based algorithm.

The first advantage of the functional spline representation (as a weighted sum of kernel functions) is that we can easily compute a estimate of any functional on the "true surface" (e.g. an integral or a derivative of the surface that generated the information points) by applying said functional to the spline as a function. In fact, provided that the functional is linear and that the space F_1 is sufficiently smooth, the estimation so obtained is an optimal error estimate (see Traub and Woz'niakowski (1980)). Using this fact, we can easily compute the orientation of the surface at any point, and even estimate the bending energy (however since this is not a linear functional, i it not necessarily an optimal estimate). These values might be used to segment the image, or locat surface discontinuties.

A second advantage of the spline representation is that it can easily be used in a system that a focus of attention. It can easily generate depth values at any points, and if the system dec (after looking at some initial depth values) that it would like to look at a portion of the surfa more detail, there is no need to recalculate the spline, simply evaluate the spline function at th desired points. Along these lines, the system can also update its idea of a surface, by adding information point, calculating the updated spline coefficients (this costs only k^2) and updating the surface points.

A third advantage of this representation is that it is less orientation dependent than a grid of values. If the visual system were to rotate or translate (as long as it does not change the relative order and spacing of the information points) then the coefficients of the spline are the same, and the spline is simply rotated or translated as well. This property does not hold for a grid of data.

A final advantage is that this representation is generally more compact (in space terms) than the grid representation. The spline is defined by 3k values, these are the k coefficients and the location of the k information points. Given these values, one can reconstruct the spline or compare this spline with another spline. This could then be used as a means of saving the reconstructed surface. Also given advantage three above, the spline representation might be used in a surface recognition algorithm.

§6 Extensibility of the Interpolation Methods.

In Grimson (1981), Grimson argued, rather convincingly, that the correct "unreasonableness" functional was quadratic variation. He however considered only a particular form of functional, and there may be more appropriate functionals which are not of the form he considered. Also, the space in which we attempt to minimize the functional has some effect on the interpolating surface, and we should consider other possible spaces for reconstruction, even for the quadratic variation functional.

Inasmuch as the reproducing kernel method finds the surface of minimal norm from a Hilbert space (or semi-norm and semi-Hilbert), in theory it can be applied to any "unreasonableness" functional which is the norm (semi-norm) of such a space. However, to do this we must be able to construct the reproducing kernel of said space, and this can be technically *very* difficult. Thus we can easily apply it only in those situations when the reproducing kernel is already known. Fortunately there are a number of such spaces some of which may be appropriate for visual surface reconstruction, see Boult (1985a). In fact, a number of these kernels exist for higher dimensions allowing the algorithm to be extended into arbitrary dimensions.

Given the different reproducing kernel, the only change to the algorithm is to replace all evaluations of the old kernels with the appropriate evaluations of the new kernel. If the null space of this new space is different from that of the space F_1, then we must also change the functions p(x,y) used by the algorithm.

To extend the gradient projection based algorithm to another functional, one would first have to develop the new discretized version of the functional. Given this discretization, one would then derive how to compute the negative of the gradient function, and the parameter α. Note that if the functional is not quadratic, these last two modification may be very difficult. Modifying the algorithm to compute the minimization with respect to another class of functions is not even possible since it approximates the surface without regard to a space of functions. It would also be very difficult if not impossible to add this feature into the algorithm, since it would involve verifying that the surface produced by each iteration was a member of the given class of functions.

§7 Conclusions.

In this paper we have compared two different algorithms for visual surface interpolation. And with the possible exception of biological feasibility, we found that if the information was sparse, the reproducing kernel algorithm surpassed the gradient projection based algorithm in almost every important algorithmic aspect. If, however, the number of information points was comparable to the number of points at which we are to recover the interpolation surface, then the gradient projection based algorithm may be superior.

Given that the problem of visual interpolation is generally posed as one with sparse information, we believe that the reproducing kernel method will be a superior algorithm for computer vision uses.

§8 Acknowledgements.

I would like to thank David Lee and John Kender, both of whom played major roles in the current investigaton, and into the application of the reproducing kernel method to visual surface interpolation.

§9 References.

Boult, Terrance, (1985a): Smoothness Assumptions in Human and Machine Vision: Their Implications for Optimal Surface Interpolation, Columbia University, Computer Science De - partment Technical Report.

Boult, Terrance, (1985b): Reproducing Kernels for Visual Surface Interpolation, Columbia University Computer Science Department Technical Report.

Boult, Terrance, (1986): Information Based Complexity: Applications in Nonlinear equations and Computer Vision, Doctoral dissertation, in preparation.

Franke, R., and Nielson, G. (1980): Smooth Interpolation of Large Sets of Scattered Data, *International Journal for Numerical Methods in Engineering*, Vol **15**, 1691-1704.

Franke, R. (1982): Scarttered Interpolation: Tests of Some Methods, *Mathematics of Computation*, **38** #157, 181-200.

Franke, R. (1984): Thin Plate Splines with Tension, to appear *CAGD*.

Grimson, W.E.L., (1981): *From Images to Surfaces: A Computational Study of the Human Early Visual System*, MIT Press, Cambridge, MA.

Kender, John, David Lee and Terrance Boult, (1985): Information Based Complexity Applied to the 2 1/2 D Sketch', Columbia University Computer Science Department Technical Report.

Marr, David and Thomas Poggio, (1979): A computational Theory of Human Stereo Vision, *Proc. R. Soc. Lond.* **B 204**, 301-328.

Meinguet, Jean, (1979a): Multivariate Interpolation at Arbitrary Points Made Simple, *Journal of Applied Mathematics and Physics 30*, 292-304.

Meinguet, Jean, (1979b): Basic Mathematical Aspects of Surface Spline Interpolation, *ISNM 45: Numerische Integration*, 211-220, G. Hämmerilin ed., Basel: Birkhäuser Verlag.

Terzopoulos, Demetri, (1983): Multi-level Computational Processes for Visual Surface Reconstruction, *Computer Vision, Graphics, and Image Processing* 24, 52-96.

Traub, J.T. and H. Woz'niakowski, (1980): *A General Theory of Optimal Algorithms*, Academic Press NY.

Traub, J.T., G. Wasilkowski, and H. Woz'niakowski, (1983): *Information, Uncertainty and Complexity*, Addison Wesley, MA.