The derivation of two-dimensional surface shape from shadows

MICHAEL HATZITHEODOROU

Department of Computer Science
Columbia University
New York, NY 10027

CUCS - 349 - 88

May 1988

Abstract. We study theoretical and implementation issues that arise when solving the shape from shadows problem. In this problem, the shadows created by a light falling on a surface are used to recover the surface itself. The problem is formulated and solved in a Hilbert space setting. We construct the spline algorithm that interpolates the data and show that it is the best possible approximation to the original function. The optimal error algorithm is implemented and a series of tests is shown. We additionally show that the problem can be decomposed into subproblems and each one can be solved independently from the others. This decomposition is suited to parallel computation and can result in considerable reductions in the cost of the solution.

Typeset by $\mathcal{AMS}$-TEX

1. Introduction.

Research in the various *Shape from X* areas has produced a series of methods that recover a surface from different types of information available about it. The information that is most widely used in these Shape from X methods includes depth values, shading, texture, etc.

We propose another approach for surface reconstruction. This is the *Shape from Shadows* problem. Assume that we have a surface that is lighted by a light source. The light source will cast shadows on this surface (see fig. 1.) Then the light will move to a new position where it will cast new shadows. We collect the different images of the shadowed surfaces, at these various times. From those we obtain the location of the start and the end of the shadow, plus some additional information about the surface function. Given any series of images containing shadows, we want to construct an algorithm that produces an approximation to the surface with the smallest possible error.

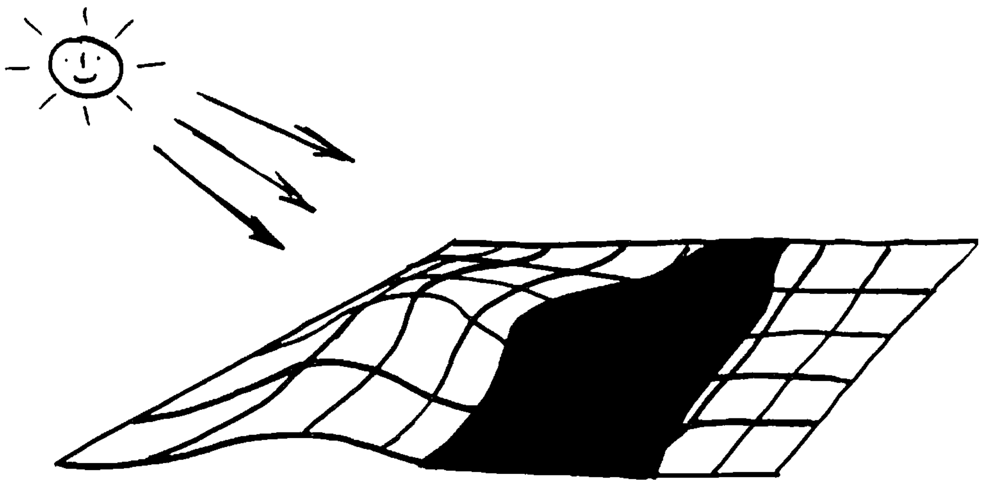

Figure 1.

Very little work has been done on the use of shadows for the reconstruction of the surface shape. In this paper we will extend the work presented in [5] where the solution to the one-dimensional version of the problem is obtained. In this one-dimensional version *surface slices*, intersections of the surface with planes perpendicular to the $x - y$ plane, were recovered. We generalize here the approach taken in [5] and present a method that will approximate the entire surface function, instead of a finite number of functions of one variable. The only other work we are aware of, is the one proposed in [6] where the problem is solved using a relaxation method.

Shadows are a very strong piece of information. The process that uses shadows is not affected by texture or by surface reflectance. Furthermore, our imaging system does not need a grey scale or color capabilities; it is sufficient for it to be able to distinguish between black and white. Also, noise in the form of bright spots inside a dark area and vice-versa can be filtered out easily. From the above, it is evident that shadows yield a powerful tool to be used in the reconstruction process.

We propose an algorithm that recovers the surface and minimizes the worst possible error within a factor of 2. The algorithm that minimizes the error is the spline algorithm.[1] We choose to construct the spline algorithm in steps using a process that converges to the optimal error algorithm. The justification behind this stepwise construction is that, in general, the optimal error algorithm might need many iterations to construct, but in most practical cases the initial version of the algorithm, or the one resulting from a few iterations, already obtains the optimal error. We therefore construct a process that has low cost, while achieving the smallest possible error.

We implement the optimal error algorithm and present a series of numerical runs so that we can see the performance of the algorithm in practice. The resulting approximations were very close to the function from which we obtained the data, and that can be immediately seen from the pairs of original and reconstructed surface that we provide. We also propose a parallel implementation of the algorithm that will considerably improve the running time.

The organization of the rest of the paper will be the following : In section 2, we will formulate the problem; we will define a function space in which our surface must belong. We will also define more precisely the information that can be extracted from the shadows.

In section 3, we will define the optimal error algorithm. We show that this is the spline algorithm which always exists and is unique. This will guarantee that the shape from shadows problem under this formulation is well-posed. The definition of a well-posed problem can be found in [3, 11]. In contrast to many other *shape from X* algorithms, our formulation does not require any regularization (see [9, 10] for a review of vision problems requiring regularization.) Understandably, if we suspect that the data are noisy, we might want to use a smoothing algorithm. In this case, the formulation that is produced by our approach is similar to the one that could be obtained from the application of regularization theory.

Section 4 deals with the implementation of the optimal algorithm. Its performance is analyzed in terms of the error it creates in recovering a surface. We will show sample runs that achieve a good approximation with a small number of data.

In section 5 we discuss the cost of the proposed algorithm. We show how to take advantage of the structure of the data and modify the algorithm, in order to obtain significant cost improvements. Furthermore, the algorithm can now be implemented in parallel, resulting in an even further reduction in the running time.

2. Formulation of the problem.

In the introduction of the paper we said that our aim is to recover a surface. Formally, a surface can be seen as a real function of two variables $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$ belonging in the space of functions F_0. Our aim is to obtain an approximation $x \in F_0$ to our function $f \in F_0$ using the data that we can derive from the shadows. We want the approximation x to be as close to f as possible.

[1] We will define the *worst case-error*, the *spline algorithm*, and all the other needed concepts later.

2.1 Function Space.

Let,

$$F_0 = \{f \mid f : [0,1]^2 \longrightarrow \mathbb{R},\ D^{1,1}f \text{ absolutely cont.}, \|D^{2,2}f\|_{L_2} \leq 1\}, \tag{2-1}$$

be the space that contains the functions f that we want to approximate.[2] [3] The norm $\|\cdot\|_{L_2}$ is defined as $\|f\|_{L_2} = \sqrt{\int_0^1 \int_0^1 |f(x,y)|^2 dxdy}$, and $D^{i,j} = \frac{\partial^{i+j}}{\partial x^i \partial y^j}$.

Also, define the bilinear form $\langle\cdot,\cdot\rangle$ to be such that,

$$\langle f, g\rangle = \int_0^1 \int_0^1 D^{2,2}f(x,y)\, D^{2,2}g(x,y)\, dxdy, \tag{2-2}$$

and the norm $\|\cdot\|$ to be such that,

$$\|f\| = \langle f, f\rangle^{1/2}. \tag{2-3}$$

Clearly $\langle\cdot,\cdot\rangle$ defined above is a semi-inner product and $\|\cdot\|$ is a semi-norm. If we pose the additional requirements $f(0,y) = 0,\ f(x,0) = 0$ and $D^{1,0}f(0,y) = 0,\ D^{0,1}f(x,0) = 0$ on our function, then the bilinear form $\langle\cdot,\cdot\rangle$ is an inner product and $\|\cdot\|$ a norm. Consequently, F_0 equipped with $\langle\cdot,\cdot\rangle$ is a semi-Hilbert space or a Hilbert space respectively.

2.2 Information.

In the next step we will extract from the image(s) the information, that is contained in the shadows, and which will be denoted by $N(f)$.[4] Assume that the light falls on the surface along the x-axis. Clearly, from the position of the light source, we can immediately obtain the partial derivative of the function f, with respect to x, at the point (x_i, y_i), (x_i, y_i) being the beginning of the shadow (see fig. 2). We can also obtain the difference between the two function values $f(x_i, y_i) - f(\bar{x}_i, y_i)$, at the beginning and at the end of the shadow respectively, given by $\frac{\partial f}{\partial x}(x_i, y_i)(x_i - \bar{x}_i)$.

For a light falling on the surface along the y-axis we can obtain similar information. In particular, for a given shadowed area starting at (x_i, y_i), and ending at $(x_i, \bar{y}_i)$, we can obtain the partial derivative of f with respect to y and the difference $f(x_i, y_i) - f(x_i, \bar{y}_i)$ which is given by $\frac{\partial f}{\partial y}(x_i, y_i)(y_i - \bar{y}_i)$.

Assuming that $l_i(x,y)$ is the straight line segment passing through the points (x_i, y_i), $(\bar{x}_i, y_i)$, an additional piece of information can be obtained. It holds (see fig. 2) that,

$$f(x, y_i) < l_i(x, y_i), \quad \forall x \in [x_i, \bar{x}_i]. \tag{2-4}$$

[2]The bound of 1 in $\|D^{2,2}f\|_{L_2}$ is assumed without loss of generality. However, as we already mentioned, any fixed bound is equally good.

[3]The use of the interval $[0,1]^2$ is not restrictive either. Any interval $[a,b] \times [c,d]$ for some a, b, c, and d is equally good.

[4]The concept of information is considerably different from the concept of data. The data vector is a vector of fixed values, while information is an operator. We will use the term somewhat imprecisely. The user is referred to [12, 13, 14, 15] for a more detailed discussion on the concept of information operators.

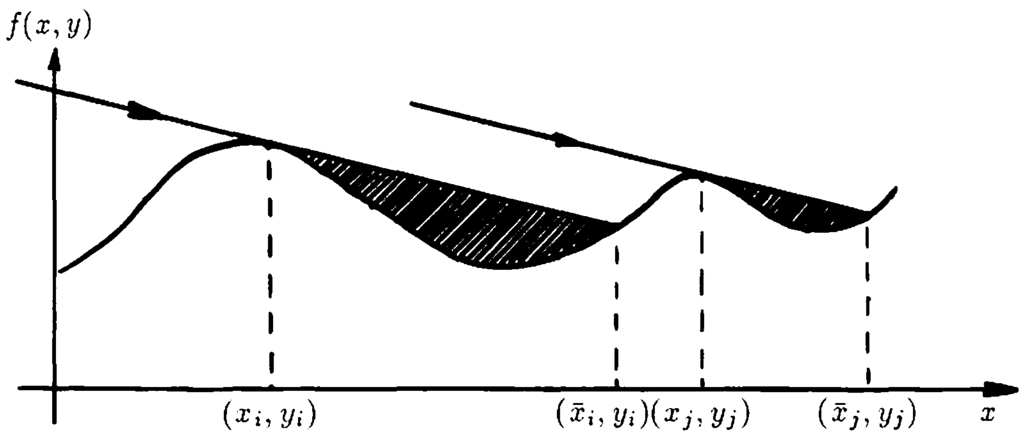

Figure 2.

If the light falls in the direction along the y-axis, then the obtained inequality is

$$f(x_i, y) \ < \ l_i(x_i, y), \quad \forall y \in [y_i, \bar{y}_i]. \tag{2-5}$$

So, formally, the information $N(f)$ contains triplets of the form,

$$\langle \frac{\partial f}{\partial x}(x_i, y_i), f(x_i, y_i) - f(\bar{x}_i, \bar{y}_i), f(x, y_i) < l_i(x, y_i) \rangle, \quad \bar{y}_i = y_i \tag{2-6}$$

or of the form,

$$\langle \frac{\partial f}{\partial y}(x_i, y_i), f(x_i, y_i) - f(\bar{x}_i, \bar{y}_i), f(x_i, y) < l_i(x_i, y) \rangle, \quad \bar{x}_i = x_i. \tag{2-7}$$

Note that the third item in the above triplets is a consistency condition.

In each one of the images in our sample there are 0, 1 or more shadowed areas. From each one of those shadowed areas we can obtain a triplet of the form (2-6) or (2-7). If we group all the data resulting from this sampling we obtain the vector,

$$\begin{aligned} N(f) = [&\frac{\partial f}{\partial x}(x_1, y_1), \ldots, \frac{\partial f}{\partial x}(x_k, y_k), \frac{\partial f}{\partial y}(x_{k+1}, y_{k+1}), \ldots, \frac{\partial f}{\partial y}(x_n, y_n), \\ &f(x_1, y_1) - f(\bar{x}_1, \bar{y}_1), \ldots, f(x_n, y_n) - f(\bar{x}_n, \bar{y}_n), \\ &\quad f(x_1, y_1) - f(t_1, s_1), \ldots, f(x_1, y_1) - f(t_{m_1}, s_{m_1}), \ldots, \\ &\qquad f(x_2, y_2) - f(t_{m_2}, s_{m_2}), \ldots, f(x_n, y_n) - f(t_{m_n}, s_{m_n})]^{\top}, \end{aligned} \tag{2-8}$$

where $m_1, \ldots, m_n$ are the number of points (t_i, y) in every interval $[x_i, \bar{x}_i] \times y$ for which (2-4) holds, or points (x, s_i) in $x \times [y_i, \bar{y}_i]$ for which (2-5) holds, and $m_1 + \cdots + m_n = m$. Clearly, while we know that there are exactly $2n$ pieces of information in the first part of

$N(f)$, we cannot bound the cardinality of the last part because it can be the case that $m \longrightarrow +\infty$.

3. Solution of the problem - The optimal algorithm.

We now proceed to the solution of the problem. We want, given information $N(f)$, to obtain an algorithm that will provide an approximation to our function f. An *algorithm* φ is defined as any mapping from the space of all permissible data vectors to the space F_0.

3.1 Algorithm error.

The error of an algorithm for any fixed function f is given by $\|f - \varphi(N(f))\|$. We would like to know what is the largest possible error that can be made by the algorithm, i.e. we want the error of the algorithm for the worst possible function f.

DEFINITION 3.1. *The worst case error of an algorithm φ is,*

$$e(\varphi, N(f)) = \sup_{\bar{f} \in F_0} \left\{ \|\bar{f} - \varphi(N(\bar{f}))\|, \; N(\bar{f}) = N(f) \right\}. \qquad (3\text{-}1)$$

Intuitively, the error of an algorithm is obtained by an adversary type of argument where the adversary chooses the function f so that the distance between f and φ is maximized.

Clearly, when solving any problem we would like to have an algorithm that will minimize $e(\varphi, N(f))$.

DEFINITION 3.2. *An algorithm φ^* that has the property,*

$$e(\varphi^*, N(f)) = \inf_{\varphi} \left\{ e(\varphi, N(f)) \right\}, \qquad \forall f \in F_0 \qquad (3\text{-}2)$$

is called a strongly optimal error algorithm.

The quantity at the right side of (3-2), i.e. the infimum of the error of all algorithms solving the problem given information $N(f)$, is a property of the problem itself, and does not depend on the particular algorithm used at any moment. This quantity, gives the inherent uncertainty of the problem for given information, and is called the ***radius of information***. Clearly, the error of the strongly optimal algorithm equals the radius of information.[5]

3.2 The spline algorithm.

We propose the spline algorithm φ^s for the solution of our problem. Splines have been known to give the optimal solution to many interesting problems [1, 2, 7, 8, 12, 13].

[5]One point that must be mentioned here is that the radius of information describes, as we said before, the ***inherent uncertainty of the problem*** and has a specific value, say R. However small or large R may be, there is no procedure that will guarantee error less than that.

DEFINITION 3.3. *A spline σ is an element in the space of functions F_0 such that,*

(1) $N(\sigma) = \vec{y}$.
(2) $\|\sigma\| = \min_{f \in F_0}\{\|f\|, \quad N(f) = \vec{y}\}$.

The meaning of (1) is that the spline must interpolate the data, and (2) says that the spline is the function that minimizes $\|\cdot\|$. The spline algorithm is the process that constructs the spline.

In a Hilbert space setting one can obtain a closed form for the spline algorithm. In our particular case, the spline algorithm is given by,

$$\varphi^s(x, y) = \sum_{i=1}^{2n} a_i g_i(x, y) + \sum_{j=1}^{m} c_j h_j(x, y), \tag{3-3}$$

where $\{g_i\}_{i=1,\ldots,2n}$ and $\{h_j\}_{j=1,\ldots,m}$ are such that,

$$D^{2,2} g_i(x, y) = \frac{(x_i - x)_+^0 (y_i - y)_+ - (x_{i-1} - x)_+^0 (y_{i-1} - y)_+}{\sqrt{x_i - x_{i-1}}}, \qquad i = 1, \ldots, k \tag{3-4}$$

when the light falls along the x-axis, and

$$D^{2,2} g_i(x, y) = \frac{(y_i - y)_+^0 (x_i - x)_+ - (y_{i-1} - y)_+^0 (x_{i-1} - x)_+}{\sqrt{y_i - y_{i-1}}}, \qquad i = k+1, \ldots, n \tag{3-5}$$

for light along the y-axis, where $(a - b)_+^0 = 1$ for $a = b$ and 0 otherwise,

$$D^{2,2} g_{n+i}(x, y) = (\bar{x}_i - x)_+ (\bar{y}_i - y)_+ - (x_i - x)_+ (y_i - y)_+ - (x_i - x)_+^0 (\bar{x}_i - x_i)(y_i - y)_+, \qquad i = 1, \ldots, k \tag{3-6}$$

$$D^{2,2} g_{n+i}(x, y) = (\bar{y}_i - y)_+ (\bar{x}_i - x)_+ - (y_i - y)_+ (x_i - x)_+ - (y_i - y)_+^0 (\bar{y}_i - y_i)(x_i - x)_+, \qquad i = k+1, \ldots, n \tag{3-7}$$

where $(a - b)_+ = a - b$ for $a > b$ and 0 otherwise, and

$$D^{2,2} h_j(x, y) = (t_j - x)_+ (s_j - y)_+ - (x_i - x)_+ (y_i - y)_+ - (x_i - x)_+^0 (t_j - x_i)(y_i - y)_+, \qquad \text{some } i, \quad j = 1, \ldots, m, \tag{3-8}$$

$$D^{2,2} h_j(x, y) = (s_j - y)_+ (t_j - x)_+ - (y_i - y)_+ (x_i - x)_+ - (y_i - y)_+^0 (s_j - y_i)(x_i - x)_+, \qquad \text{some } i, \quad j = 1, \ldots, m. \tag{3-9}$$

The functions $\{g_i\}_{i=1,\ldots,2n}$ and $\{h_j\}_{j=1,\ldots,m}$ are the *representers* of the functionals that construct the information $N(f)$, properly modified to have a small area of support.

The coefficients a_i and c_j are chosen so that the definition of the spline is satisfied, that is, φ^s interpolates the data, and also minimizes the norm $\|\cdot\|$. If the area of support of every g_i is disjoint from the area of support of every other and furthermore $\langle g_i, g_j \rangle = \delta_{ij}$, the Krönecker delta, then the coefficients a_i are given directly by the theory. This is not the case in this setting where the coefficients a_i are obtained by solving a system of linear equations and the coefficients c_j are obtained by directly minimizing $\|\cdot\|$. We will describe the implementation of the algorithm in section 4. The derivation of the minimization problem is somewhat tedius and can be found in the appendix.

For the spline algorithm the following very strong theorem holds [8, 13].

THEOREM 3.1. *Let F_0 be a Hilbert space, $f \in F_0$ and information $\vec{y} = N(f)$. Then, the spline algorithm interpolating the data $\vec{y}$ exists, is unique, and achieves error at most twice the radius of information.*

From Theorem 3.1 we can obtain two very important results. First, our problem under the proposed formulation is well-posed. This property [3, 11] is always desirable when solving a problem. Computer vision problems tend to be ill-posed and considerable effort has been spent by the vision community towards the correct formulation that will yield well-posedness (See [4, 9, 10] for a survey.)

Second, the spline algorithm has a worst case error that is within a factor of 2 from the radius of information. The algorithm that achieves that, is called *almost strongly optimal* [12]. If the problem is *linear*[6] then the spline algorithm φ^s has a worst case error equal to the radius of information and is, therefore, the strongly optimal algorithm.

We can choose to ignore the existence of the inequalities (2-4) and (2-5) and not include them in the information $N(f)$. If we choose to do so, we essentially assume that $m = 0$ and the shape from shadows problem is a linear problem. Then, the spline algorithm becomes

$$\varphi^s(x, y) = \sum_{i=1}^{2n} a_i g_i(x, y). \tag{3-10}$$

As expected, the part $\sum_{j=1}^{m} c_j h_j(x, y)$ can now be omitted.

If on the other hand, $m > 0$ then, the problem is non-linear, and φ^s is an almost strongly optimal algorithm.[7]

We saw before that the construction of the algorithm has to be done in steps. First, we obtain the coefficients a_i and subsequently the coefficients c_j. We also mentioned that the cardinality of the non-linear part of the information m is not known a-priori. We

[6] For an exact definition of a linear problem see [8, 13, 14, 15].

[7] Up to this moment there does not exist a general theory that will help construct strongly optimal algorithms for non-linear problems. These algorithms are in general difficult to construct and are derived on a per problem basis.

would like to keep its value low to reduce the costs involved in solving the minimization problem. To obtain this we propose to break down the implementation of the algorithm in additional steps.

4. Application of the algorithm - Numerical runs.

The spline algorithm of section 3 has been applied and its performance has been tested in practice.

4.1 Algorithm implementation.

From our early experience we have concluded that in many cases, the non-linear part of the information is not needed.

Stage 1:

Therefore, we begin the implementation of the spline algorithm by assuming that $m = 0$. We first construct the values of the coefficients a_i. This is done by solving the system of equations,

$$\mathbf{G}\,\vec{a} = \vec{y}, \tag{4-1}$$

where $\mathbf{G} = \{(g_i, g_j)\}_{i,j=1}^{2n}$ and $\{g_i\}_{i=1,\dots,2n}$ are given by (3-4), (3-5), (3-6) and (3-7). The system is solved by a direct method without the need for pivoting since it is symmetric, positive definite and has a nice structure that reduces the number of calculations.

As a next step, we use the computed values of the a_i' s to construct the spline algorithm.

Third, we check to see whether the non-linear constraints are violated.

If the non-linear constraints (2-4) and (2-5) are not violated, which is usually the case, we do not need to do anything else. We have already obtained the approximation $\varphi^s(x, y)$ to the function f.

Since we have not used the non-linear part of the information, the problem is linear hence the spline algorithm achieves the radius of information.

Stage 2:

If the constraints (2-4) or (2-5) are violated, then we do not have a sufficiently good approximation, which means that we must obtain the coefficients c_j of (3-3). To do so, we have to solve the minimization problem derived in section 3.3.

We will consequently proceed as follows. We will take a few points (t_i, s_i) in the shadowed intervals where the constraints are violated. For these points we solve the minimization problem. Then we check again for violations of the non-linear constraints. If there are violations we repeat Stage 2. We select a few more points from the new interval(s) in which (2-4) or (2-5) are violated, and we add them to the sample. The minimization is repeated for the new set of points and the new coefficients are derived. At the same time, the a_i' s and the old c_j' s are modified.

We perform the minimization for a few points at a time for various reasons.

(1) Theoretically, the cardinality of the non-linear information m can be arbitrarily large. In practice though, we rarely need to use more that one or two points per

shadowed area. Taking a few points at a time we can minimize the cost of the algorithm.

(2) At each new iteration we do not need to undo our previous work, but we simply modify the existing coefficients while deriving the new ones.

4.2 Test runs.

We have constructed a broad series of functions, and have run the algorithm using these as test surfaces.

From early test runs, we have observed that smooth functions can be approximated easily with almost non-observable error, using a small number of sample points.

The functions that are the most difficult to approximate, are the ones that have as few derivatives as possible. In the class F_0 these functions are piecewise quadratic polynomials which are constructed as the product of one-dimensional quadratic polynomials. We will show the performance of the algorithm φ^s on these functions.

We start the series of test runs with a function consisting of 100 polynomial pieces. We use two different light angles from each direction, two along the x-axis and two along the y-axis. The function and the reconstruction can be seen in figure 3.

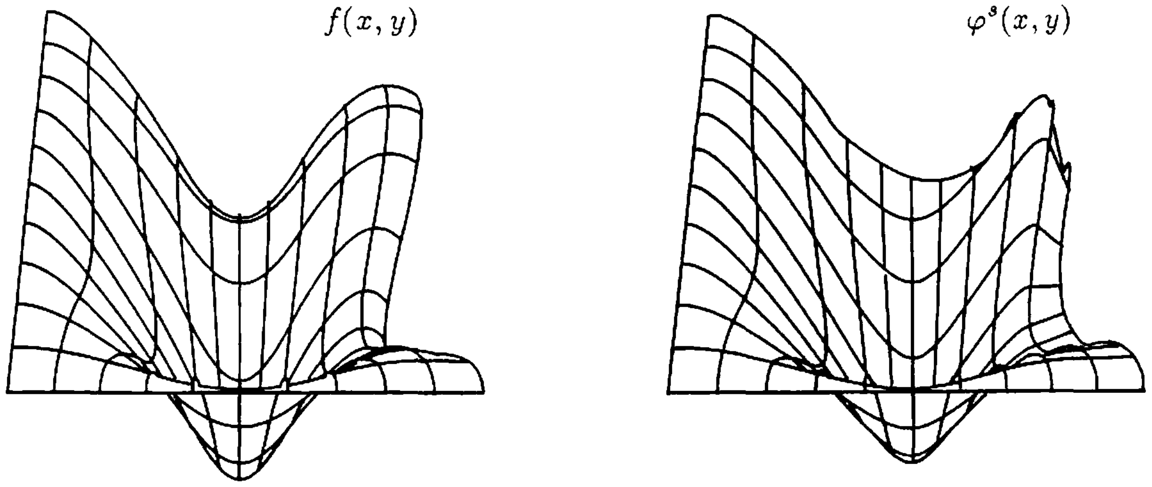

Figure 3.

In figure 4 we show a function consisting of 200 polynomial pieces. We again draw the reconstruction together with the function for comparison purposes. The information was obtained by using 4 different lighting angles in each direction.

Finally, in figure 5 we show one of the most difficult functions that we have constructed. It consists of 400 polynomial pieces with large jumps in the second derivatives.[8] The

[8]Which means that $||D^{2,2}f|| \leq 1$ does not hold. As we mentioned earlier, this does not change the mathematical formulation. Nevertheless, the visual effect it creates in the quality of the reconstruction is significant.

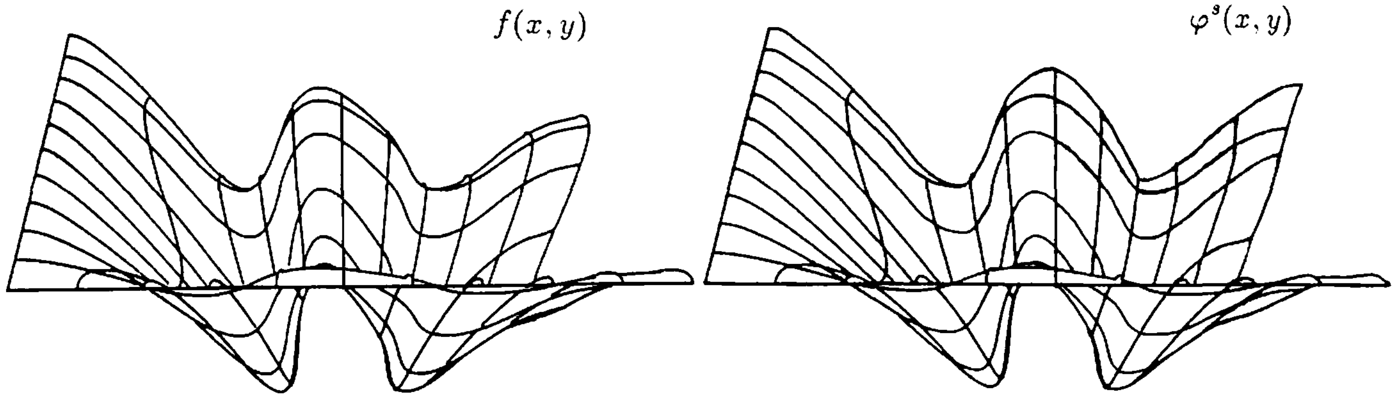

Figure 4.

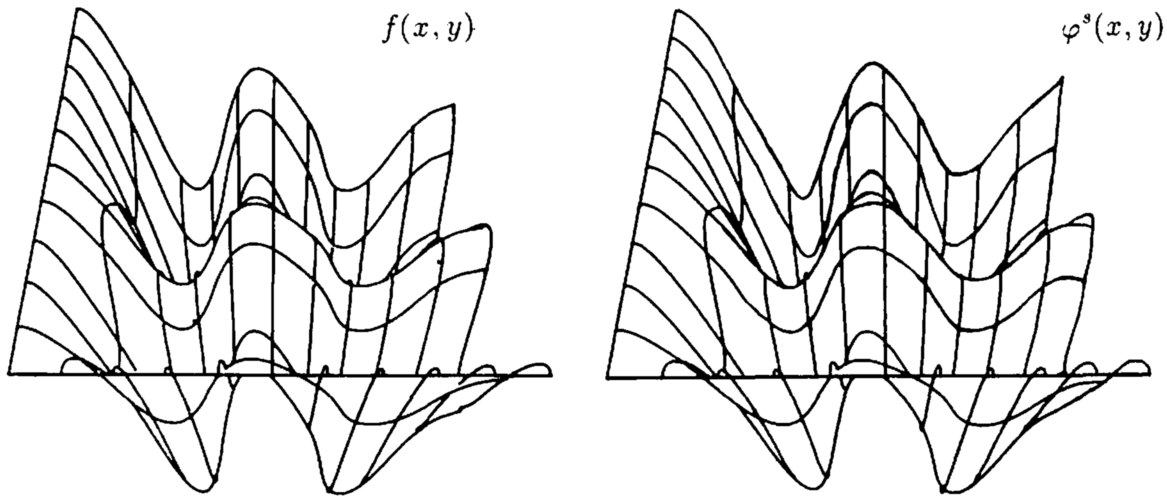

Figure 5.

function has been approximated using a sample created from lights falling from six different light angles in each direction.[9]

5. Cost of the algorithm - Speed improvements.

5.1 Algorithm cost.

Let us now discuss the speed performance of our algorithm. The spline algorithm, as defined in section 3, is linear in terms of its input. Thus, if we knew the coefficients a_i and c_j then, $cost(\varphi^s)$ would be $O(n)$.

In our case, the coefficients of the spline algorithm are not known, and must be constructed. To achieve this we must solve a system of linear equations, and sometimes, a minimization problem. These costs dominate the cost of the algorithm.

In particular the solution of the system (4-1) has a cost $O(n^3)$. The cost of the non-

[9]Theoretically, light falling from one direction only could be used, but in this case the problem is identical to the one-dimensional one [5].

linear minimization is considerably higher in the general case. Due to the special structure of the problem, whose derivation we show in the appendix, we can use a variation of the feasible directions approach. The proposed method has a cost of O(n). We are in the process of investigating the properties of this method.

5.2 Speed improvements.

In section 5.1 we have discussed the cost of a very straightforward implementation of the spline algorithm described in section 4.1. We now show that a slight improvement in the implementation of the algorithm can yield a significant speedup. This speedup can be achieved only if the function we want to recover can be split in distinct sections that we will from now on call *valleys*. A valley is defined by two local maxima of the function, but also depends on the specific sampling. For example, the function of figure 4 has 2 valleys. In particular, we say that the function f, under some fixed sampling, has k valleys if we can define k partitions $\Pi_1, \Pi_2, \ldots, \Pi_k$ of the functions $\{g_i\}_{i=1,\ldots,2n}$, given by (3-4) and (3-5), such that the union of the areas of support of all the functions in each partition is disjoint from the union of the areas of support of the functions in every other partition.

We can detect the existence of any number of valleys in time O(n) and subsequently, we can solve k problems of sizes $n_1, n_2, \ldots, n_k$ respectively, instead of solving one problem of size n, where $n = n_1 + n_2 + \cdots + n_k$.

To connect the pieces resulting from each of the k problems we need constant time per problem, hence combining can be done in time O(k).

Therefore, the total cost of this algorithm, which we will denote φ_i^s, will be O($k\nu^3$), where $\nu = \max\{n_1, \ldots, n_k\}$.

5.3 Parallel implementation.

Since splitting the problem into individual subproblems and combining the resulting surfaces is straightforward and cheap to implement, one immediate extension to the above set-up of the problem is to assign one individual subproblem to a different processor and solve the initial problem in a parallel or in a distributed environment.

Again, splitting into k subproblems requires time O(n) and combining the individual solutions into one requires time of O(k). Then every processor will require time O(n_i^3), $i = 1, \ldots, k$ resulting in a total cost for the parallel version φ_p^s of our algorithm of O(ν^3), where $\nu = \max\{n_1, \ldots, n_k\}$.

6. Conclusion - Future work.

There are certain issues that we would like to investigate in the future. We would like to study ***optimal*** and ***adaptive information***. The study of optimal information will determine the light placement that will minimize the error of the algorithm for a fixed number of samples. Having adaptive information will permit, given a number of samples, to choose where to place a new light so as to reduce the error as much as possible.

Although the work presented in this paper is not directly related to signal processing, the construction of a complete system will also have to address many issues that arise when the information is obtained from the shadows.

We solved the problem of recovering a surface from the shadows it casts on itself when lighted by a light source positioned at various locations.

We proposed a formulation that results in a well-posed problem and we have consequently proceeded into solving it. We proposed an optimal error algorithm which additionally achieves a low time cost, especially if a clever but simple breakdown of the problem is used.

The work described in this paper, extends the results on the reconstruction of one-dimensional surface slices. This new approach can be very desirable, especially if our purpose is to recover the entire shape of the surface.

The one-dimensional model can still be used if our aim is to recover the function value at just one point, in which case we only have to use the slice passing through this point. Also, if we only have lights along one axis then, the one- and two-dimensional models become equivalent.

7. Acknowledgements.

I would like to thank Prof. John Kender for many valuable suggestions and discussions.

I would also like to thank Prof. Greg Wasilkowski, currently with the University of Kentucky at Lexington, whose guidance made this work possible.

I. Appendix - The minimization problem.

We want to minimize $\|\sigma\|^2$ where σ is given by (3-3). We can write,

$$\begin{aligned}\|\sigma\|^2 &= \langle\sigma,\sigma\rangle \\ &= \sum_{i_1=1}^{n}\sum_{i_2=1}^{n} a_{i_1}a_{i_2}\langle g_{i_1}, g_{i_2}\rangle + 2\sum_{i=1}^{n}\sum_{j=1}^{k} a_i c_j\langle g_i, h_j\rangle + \sum_{j_1=1}^{k}\sum_{j_2=1}^{k} c_{j_1}c_{j_2}\langle h_{j_1}, h_{j_2}\rangle \\ &= \vec{a}^{\mathsf{T}}\mathbf{G}\,\vec{a} + 2\,\vec{a}^{\mathsf{T}}\mathbf{P}^{\mathsf{T}}\vec{c} + \vec{c}^{\mathsf{T}}\mathbf{H}\,\vec{c}, \qquad \text{(I-1)}\end{aligned}$$

where $\mathbf{G} = \{\langle g_i, g_j\rangle\}_{i,j=1,\dots,2n}$, $\mathbf{P} = \{\langle h_j, g_i\rangle\}_{\substack{j=1,\dots,m\\ i=1,\dots,2n}}$, and $\mathbf{H} = \{\langle h_i, h_j\rangle\}_{i,j=1,\dots,m}$.

Also,

$$\begin{aligned}\langle\sigma, g_s\rangle = \sum_{i=1}^{n} a_i\langle g_i, g_s\rangle + \sum_{j=1}^{k} c_j\langle h_j, g_s\rangle &= y_s \\ \Longrightarrow \qquad \mathbf{G}\,\vec{a} + \mathbf{P}\,\vec{c} &= \vec{y} \\ \Longrightarrow \qquad \vec{a} &= \mathbf{G}^{-1}(\vec{y} - \mathbf{P}\,\vec{c}), \qquad \text{(I-2)}\end{aligned}$$

and,

$$\begin{aligned}&\langle\sigma, h_s\rangle = \sum_{i=1}^{n} a_i\langle g_i, h_s\rangle + \sum_{j=1}^{k} c_j\langle h_j, h_s\rangle \le A_s \\ \Longrightarrow \quad &\mathbf{P}^{\mathsf{T}}\vec{a} + \mathbf{H}\,\vec{c} \le \vec{A} \\ \overset{\text{(I-2)}}{\Longrightarrow} \quad &\mathbf{P}^{\mathsf{T}}\mathbf{G}^{-1}\vec{y} - \mathbf{P}^{\mathsf{T}}\mathbf{G}^{-1}\mathbf{P}\,\vec{c} + \mathbf{H}\,\vec{c} \le \vec{A} \\ \Longrightarrow \quad &(\mathbf{H} - \mathbf{P}^{\mathsf{T}}\mathbf{G}^{-1}\mathbf{P})\,\vec{c} \le \vec{A} - \mathbf{P}^{\mathsf{T}}\mathbf{G}^{-1}\vec{y}. \qquad \text{(I-3)}\end{aligned}$$

Now, if we substitute (I-2) for $\vec{a}$ in (I-1) we obtain,

$$\begin{aligned}\|\sigma\|^2 &= \vec{c}^{\mathsf{T}}\mathbf{H}\,\vec{c} + 2\left(\mathbf{G}^{-1}(\vec{y} - \mathbf{P}\vec{c})\right)^{\mathsf{T}}\mathbf{P}\,\vec{c} + \left(\mathbf{G}^{-1}(\vec{y} - \mathbf{P}\vec{c})\right)^{\mathsf{T}}\mathbf{G}\,\left(\mathbf{G}^{-1}(\vec{y} - \mathbf{P}\vec{c})\right) \\ &= \qquad \dots \\ &= \vec{c}^{\mathsf{T}}\mathbf{H}\,\vec{c} - \vec{c}^{\mathsf{T}}\mathbf{P}^{\mathsf{T}}\mathbf{G}^{-1}\mathbf{P}\,\vec{c} + \vec{y}^{\mathsf{T}}\mathbf{G}^{-1}\vec{y} \\ &= \vec{c}^{\mathsf{T}}(\mathbf{H} - \mathbf{P}^{\mathsf{T}}\mathbf{G}^{-1}\mathbf{P})\,\vec{c} + \vec{y}^{\mathsf{T}}\mathbf{G}^{-1}\vec{y}. \qquad \text{(I-4)}\end{aligned}$$

Since $\vec{y}^{\mathsf{T}}\mathbf{G}^{-1}\vec{y}$ has a known fixed value for any given problem, it remains to minimize $\vec{c}^{\mathsf{T}}\left(\mathbf{H} - \mathbf{P}^{\mathsf{T}}\mathbf{G}^{-1}\mathbf{P}\right)\vec{c}$ given the conditions in (I-3). This is a non-linear minimization problem. We can solve this problem using a feasible directions method. The matrix $\mathbf{Q} = \left(\mathbf{H} - \mathbf{P}^{\mathsf{T}}\mathbf{G}^{-1}\mathbf{P}\right)$ is positive definite, hence the problem is convex. Therefore, we are guaranteed to find the global minimum. At the same time, it can be noticed, that the constraints that we have are linear[10] which permits us to speed up the minimization method considerably. We are currently working on an even faster method for the solution of this problem.

[10] Even more, they are simple constant bounds on the variables.

References

[1] Anselone, P. M., and Laurent, P. J., *A general method for the construction of interpolating or smoothing spline functions*, Nummer. Math. 12 (1968).

[2] Atteia, M., *Fonctions spline généralisées*, C.R. Acad. Sci. Paris 261 (1965).

[3] Hadamard, J., *Sur les problèmes aux derivées partielles et leur signification physique*, Princeton Univ. Bulletin 13 (1902).

[4] Hatzitheodorou, M. G., *The Application of Approximation theory methods to the solution of computer vision problems*, Columbia University, Comp. Science dep. (1988).

[5] Hatzitheodorou M. G., and Kender, J. R., *An optimal algorithm for the derivation of shape from shadows*, Proceedings IEEE CVPR (1988).

[6] Kender, J. R., and Smith, E. M., *Shape from darkness. Deriving surface information from dynamic shadows*, Proceedings AAAI (1986).

[7] Holmes, R., *R-splines in Banach spaces : I. Interpolation of linear manifolds*, J. Math. Anal. Appl. 40 (1972).

[8] Michelli, C. A., and Rivlin, T. J, *A Survey of optimal recovery*, in "Optimal estimation in Approximation theory," Plenum Press, 1977.

[9] Poggio, T., *Computer vision*, MIT Artificial Inteligence Lab (1986).

[10] Poggio, T., and Torre, V., *Ill-posed problems and regularization analysis in early vision*, MIT Artificial Inteligence Lab (1984).

[11] Tikhonov, A. N., and Arsenin, V. Y., "Solutions of ill-posed problems," V.H.Winston and Sons, 1977.

[12] Traub, J. F., Wasilkowski, G., and Woźniakowski. H., "Information, Uncertainty, Complexity," Addison-Wesley, 1983.

[13] Traub, J. F., and Woźniakowski, H., "A general theory of optimal algorithms," Academic Press, 1981.

[14] Woźniakowski, H., *A survey of information-based complexity*, Journal of Complexity 1 (1985).

[15] Woźniakowski, H., *Information-based complexity*, Annual Review of Computer Science 1 (1986).