Solving the shape-from-shading problem on the CM-5*

Michael J. Brooks Wojciech Chojnacki Anton van den Hengel
{mjb,wojtek,hengel}@cs.adelaide.edu.au
Department of Computer Science
University of Adelaide
Adelaide, SA 5005, Australia
Centre for Sensor Signal and Information Processing
Signal Processing Research Institute
Technology Park, The Levels, SA 5095, Australia

Abstract

We consider the problem of recovering surface shape from image shading for the situation in which a distant overhead "sun" illuminates a Lambertian surface. An iterative scheme is presented which requires no prerequisite shape information. This scheme forms the basis for a parallel algorithm implemented on a CM-5. *Performance of the* CM-5 *implementation is compared with that of a sequential implementation running on a Sun Sparc 2. Also considered are the complexity and scalability of the parallel algorithm as a function of image size and number of processors, respectively.*

1 Introduction

A monochrome photograph of a smooth object will typically exhibit brightness variation, or *shading*. Of interest to researchers in computer vision has been the problem of how object shape may be extracted from image shading. This *shape-from-shading problem* has been shown to correspond to solving a first-order partial differential equation [6].

Nearly all shape-from-shading methods need to be given substantial prior knowledge of surface shape—the very information being sought [7]. In contrast with this, fundamental results on uniqueness reveal that under certain conditions shape may be determined from shading without the prior knowledge of some height values or surface normals (see [4], [11], [3]). Relying upon this observation, we present in this paper a massively parallel shape-from-shading algorithm, operating in a restricted environment, that requires no explicit initial shape information. (See also [5], [12] and [10] for related shape-from-shading work, without consideration of parallelism.)

The shape-from-shading problem may be expressed as the need to find a function $u(x,y)$, representing surface depth in the direction of the z-axis, that satisfies the *image irradiance equation* $R(u_x,u_y) = E(x,y)$ over Ω. Here E is an image formed by (orthographic) projection of light onto a plane parallel to the xy-plane, situated above the surface, R is the reflectance map [8] relating image brightness to surface orientation, and Ω is the image domain. In this formulation, it is implicitly assumed that light sources are infinitely far away, and that internal surface reflections are disallowed.

Assume that a distant overhead point-source illuminates a Lambertian surface of unit albedo. Then the corresponding image irradiance equation is

$$(u_x^2 + u_y^2 + 1)^{-1/2} = E(x,y). \tag{1}$$

Noting that $0 < E(x,y) \leq 1$, we may safely let

$$\mathcal{E}(x,y) = E^{-2}(x,y) - 1 \tag{2}$$

and express (1) as the eikonal equation

$$u_x^2 + u_y^2 = \mathcal{E}(x,y). \tag{3}$$

Any C^2 function satisfying (3) over Ω will be referred to as a *solution*. With each solution to (3) there are associated *base characteristic curves* defined as projections onto the image domain of the solution surface's lines of steepest ascent with respect to the illuminant direction (this direction, in our setting, coincides with that of the z-axis). A point in an image at which E attains 1 (that is, at which image brightness is maximal) is called *singular*. It is readily seen from equations (2) and (3) that if $X \in \Omega$ is singular, then $u_x(X) = u_y(X) = 0$ for every solution u;

*CM-5 is a trademark of *Thinking Machines Corporated.*

0-8186-7134-3/95 $04.00 © 1995 IEEE

in particular, if $u_{xx}(X)u_{yy}(X) > u_{xy}^2(X)$, then u has a local maximum or local minimum at X.

We may now state the result upon which our approach is based:

Theorem 1 *If a C^1 function u satisfies (3) in a domain Ω, and X and Y are points in Ω that can be joined by a base characteristic curve wholly contained in Ω, then the relative depth $|u(X) - u(Y)|$ between points on the graph of u, having X and Y as images, is given by*

$$|u(X) - u(Y)| = \min_{\gamma} \int_{\gamma} \sqrt{\mathcal{E}}\, dl, \tag{4}$$

where the minimum is taken over all piecewise smooth curves γ in Ω joining X and Y, and the integration is meant with respect to the standard line measure.

A proof may be found in [1] (cf. also [2]).

The significance of this theorem is as follows. Suppose that two points in the image are linked via a base characteristic curve that is wholly contained in the image. An expression is derived whose integral along the base characteristic curve connecting these points equals the absolute value of the difference of the surface heights at the points in question. Moreover, the expression is such that its integral along any other curve in the domain connecting the two points is not smaller than this value. Thus the relative height of the two points can be determined by finding the smallest of those integrals taken over all paths in the domain that connect the two points.

We now specify two classes of surfaces whose shape recovery will be based on Theorem 1. One of these will comprise *convex* surfaces and the other will comprise *concave* surfaces. By definition, a surface is convex (concave) if it has an isolated maximum (minimum) and is such that every point in the surface's image can be joined by a base characteristic curve with the singular point at which surface height is maximal (minimal). Note that the above definition is not intrinsically geometric as it depends on the direction from which the surface is illuminated.

For every pair of points X, Y in the image domain Ω, put

$$d(X, Y) = \min_{\gamma} \int_{\gamma} \sqrt{\mathcal{E}}\, dl, \tag{5}$$

where the minimum is taken over all piecewise smooth curves γ in Ω joining X and Y. It is readily inferred from Theorem 1 that if $\mathcal{E}$ has a solitary singular point S, then any convex solution of (3) is given by

$$u(X) = u(S) - d(X, S) \qquad (X \in \Omega) \tag{6}$$

and any concave solution u is given by

$$u(X) = u(S) + d(X, S) \qquad (X \in \Omega).$$

Since any concave (convex) solution u can be represented as $u = -v + u(S) + v(S)$, where v is a convex (concave) solution to (3), we shall hereafter confine ourselves to considering convex solutions.

2 Iterative Scheme

We now develop a numerical method for obtaining shape from shading using the above results.

Let $\mathcal{E}$ correspond to a shading pattern in a discrete rectangular domain Ω, having a solitary singular point S. We aim to determine the height map of a convex shape having a maximum at S. For simplicity, we assume that the surface's height at S is zero. Let X be an arbitrary point in Ω. Denote by $\mathcal{N}(X)$ the neighbourhood of X comprising the nearest neighbours of X in a rectangular grid and X itself (typically, $\mathcal{N}(X)$ will comprise nine points). For $X, Y \in \Omega$, let $d_{X,Y}$ be a numerical approximation to the integral of $\sqrt{\mathcal{E}}$ over the interval joining X with Y. Note that d depends only on image values, and may therefore be pre-computed and stored. Given a non-negative integer n and $X \in \Omega$, let $U_X^{(n)}$ denote the n-th estimate of height at X of the sought-after shape. Implementing (6), we propose the following iterative scheme for computation of successive height values:

$$U_X^{(n+1)} = \max_{Y \in \mathcal{N}(X)} \{U_Y^{(n)} - d_{X,Y}\} \qquad (n = 0, 1, \ldots)$$

with the initial values being prescribed as

$$U_X^{(0)} = \begin{cases} 0 & \text{if } X = S, \\ -\infty & \text{if } X \in \Omega \setminus \{S\}. \end{cases}$$

Here, the scheme terminates when, for some n, $U_X^{(n)} = U_X^{(n+1)}$ for all $X \in \Omega$; that is, when no changes are made to the height estimates in computing the next iterate.

The above computation proceeds as follows. Consider a point $X \in \Omega$. For each of X's (typically eight) nearest neighbours, a new candidate is obtained for the height at X by adding the neighbour's height to the integral of $\sqrt{\mathcal{E}}$ over the interval joining the neighbour and X. A new height value for X is then obtained by choosing the maximum of the set comprising the estimates obtained from the neighbours, along with X's previous height value. At the edges of the image, less than eight neighbours may be involved in the calculation.

Initially all points in the height map are assigned a large negative height value. The singular point is assigned a height of zero. As the algorithm proceeds new height estimates are propagated out from the singular point. Heights

across the image are recomputed as many times as is necessary to reach a steady state. In this way, complex and twisting paths are eventually explored if the base characteristics exhibit such a form.

3 Parallel Implementation

We now describe a parallel implementation of the iterative scheme discussed above.

In this implementation, eight auxiliary arrays are employed, each having the same dimensions as the image array. The auxiliary arrays store new estimates of height values for the depicted surface corresponding to calculation in the direction of each of the eight neighbouring points. For example, the first array computes a height estimate at any given point by extrapolating inward from its Northern (N) neighbour. The second array is similar, except that the North-Western (NW) neighbour is used. Likewise, the remaining arrays store data associated with neighbours in compass directions W, SW, S, SE, E, and NE, respectively. Calculating the auxiliary arrays involves adding the current estimates of surface height (shifted in the appropriate direction) to the stored integrals.

At each iteration a new height estimate is calculated as the element-wise maximum of the eight auxiliary arrays and the current estimate. The values of the auxiliary arrays are then re-calculated on the basis of this new estimate. The array operations described are a good match to the data parallel programming style supported by the CM-5.

We now express the algorithm more formally. For each non-negative n and compass direction $\vec{l}$, denote by ${}^{\vec{l}}U^{(n)}$ the n-th estimate of the aforementioned eight arrays. The above iterative scheme may now be rewritten in the form

$$U_X^{(n+1)} = \max_{Y \in \mathcal{N}(X)} {}^{\overrightarrow{YX}}U_X^{(n)},$$

where ${}^{\overrightarrow{YX}}U_X^{(n)}$ denotes the height estimate at X obtained via integration from Y to X, and is given by

$${}^{\overrightarrow{YX}}U_X^{(n)} = U_Y^{(n)} - d_{X,Y};$$

we adopt here the convention that ${}^{\overrightarrow{XX}}U_X^{(n)} = U_X^{(n)}$. The iterative scheme may be terminated when

$$\sum_{X \in \Omega} U_X^{(n+1)} = \sum_{X \in \Omega} U_X^{(n)}$$

rather than when the respective elements of the two arrays are all equal. This is possible because the values held at each position in the array cannot decrease in height over successive iterations. When the sum fails to increase, the process is therefore complete. The advantage of this approach is that summation of an array's values is a primitive and parallel operation of $O(\log(N^2))$ on the CM-5.

Minimisation across nine arrays necessitates no communication, only computation local to each of the CM-5's processors, thus requiring little processor time. In contrast, re-building the eight arrays requires communication of data between processors. Given the amount of data needed to compute a height estimate, this communication accounts for a significant proportion of the processor time consumed by the method. The fact that this operation must take place eight times for each iteration, and that it must take place selectively to allow for edge conditions, adds further the expensiveness of this computation.

4 Performance Analysis

We now consider the time complexity of the algorithm in relation to an image size of $N \times N$ pixels. This is given by the product of the time taken to complete a single iteration and the number of iterations required for the algorithm to converge. We consider each factor in turn.

In carrying out a single iteration, it is necessary to perform a relatively simple local operation (involving nearest-neighbour values) at each of the N^2 pixels (we shall not concern ourselves with the simpler but idiosyncratic processing needed at the boundary). A sequential machine would thus require a processing time of $O(N^2)$ per iteration. Were we fortunate enough to have an array processor machine with sufficiently many processors to enable all N^2 local operations to be executed in parallel, the time complexity would be reduced to $O(1)$ per iteration, providing that the time taken to carry out a local operation is independent of image size. It will prove convenient to term such an array processor an ideal-parallel machine.

The time complexity of a single iteration is more difficult to determine when the algorithm is to run on a real parallel machine with a limited number of processors. Let the number of processors in some parallel machine be λ. In the event that $N^2 \leq \lambda$, we retain ideal-parallel complexity. However, as N^2 increases beyond λ, the time complexity becomes difficult to assess, until, as $N^2 \to \infty$, time complexity tends to that for the sequential case. We shall not attempt here to derive a theoretical expression for time complexity when N^2 is greater, but not vastly greater, than λ.

In the event that M iterations are required for the algorithm to terminate, then it clearly has sequential time complexity of $O(MN^2)$, and ideal-parallel time complexity of $O(M)$. The actual number of iterations required to process an image is dependent upon the nature of the image itself, and corresponds to the length in pixels of

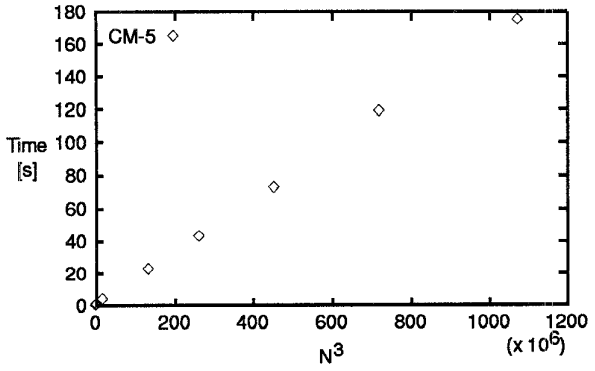

Figure 1. Time vs. cube of image size (CM-5)

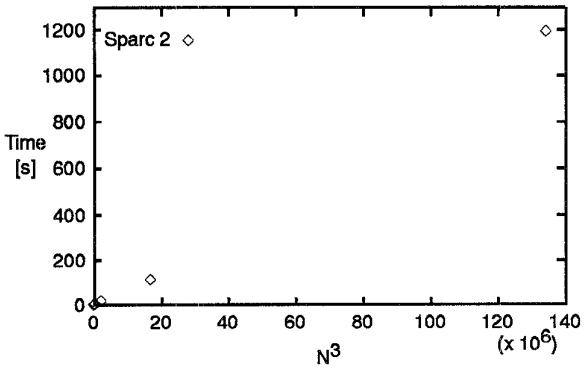

Figure 2. Time vs. cube of image size (Sparc 2)

the longest base characteristic contained within the image. In the case of a Lambertian sphere illuminated and viewed from overhead, all base characteristics are straight lines. An image of all or part of such a sphere therefore requires no more than N iterations. Although there could exist pathological images where the number of iterations necessary to determine the corresponding height map approaches N^2, for a large class of images the number of iterations required is $O(N)$. For this class of images the algorithm therefore has sequential time complexity of $O(N^3)$, and ideal-parallel time complexity of $O(N)$.

On our particular CM-5, computation is distributed across 128 vector-unit processors. These processors are in groups of four, each group being associated with one of 32 nodes. Since all images to be processed contain greater than 128 pixels, a high pixel-to-processor ratio is inevitable and communication costs become significant (largely because communication between processors of different groups is expensive). Determining the time complexity of the algorithm running on such a machine is fraught with difficulty. For this reason we now turn to a comparative performance analysis of both sequential and parallel implementations.

The sequential version of the scheme was run on a Sparc Station 2. Timing data were obtained using standard C facilities. Recall that the parallel version of the method was run on a CM-5 with 32 nodes and 128 vector units. Here, timing data were obtained using standard C* facilities. When dealing with various $N \times N$ images, our implementation always terminated well within $2N$ iterations.

Figures 1 and 2 depict the execution times, for various image sizes, of the parallel version on the CM-5 and the sequential version running on a Sparc 2, respectively. The axes of both graphs represent the time taken to process an image versus the cube of image width. Over the modest number of trials conducted the data points for the respective machines appear almost linear, suggesting a time complexity of $O(N^3)$ for both machines. While it may seem odd that the algorithms running on sequential and parallel machines exhibit similar orders of complexity, this is to be expected given that in all tests $\lambda \ll N^2$. Nevertheless, there is a significant speedup for large image sizes associated with the modest parallelism.

Figure 3 shows execution time against cube of image size for both CM-5 and Sparc 2. Here it may be inferred that the implementations have similar complexity $O(N^3)$. Table 1 explicitly lists the data displayed in Figure 3, and reveals speedup factors ranging between 3.5 and 55 for execution on the CM-5 over the Sparc 2. These factors could be expected to increase further with increased image size; however, additional comparison was not practicable given that the Sparc 2 was unable to process the 1024×1024 element array due to lack of physical memory.

The variation in speedup factors is due largely to the nature of the architecture of the CM-5. Each vector processing unit consists of a five-stage pipeline. Efficient use of each unit requires that the pipeline be kept full. This in turn requires the processing of a large number of pixels at each node. A high pixel-to-processor ratio clearly ensues when images increase sufficiently in size.

The communication model employed within the CM-5 also serves to increase the speedup factor as image size increases (assuming a fixed number of processors). The latency of communication between vector units belonging to different groups or nodes is considerably greater than the latency of communication between units from the same group. A higher pixel-to-processor ratio will therefore tend to increase the proportion of communication which is local to a given group.

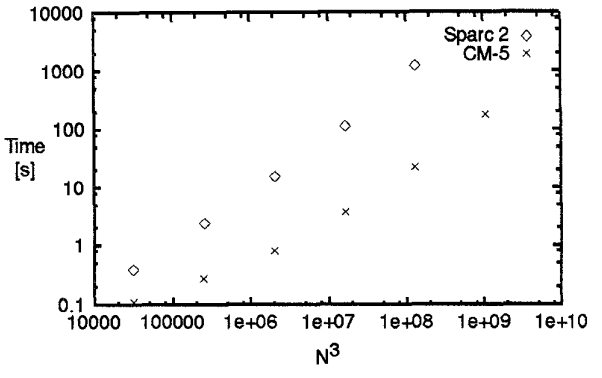

Figure 3. Time vs. cube of image size (CM-5 and Sparc 2)

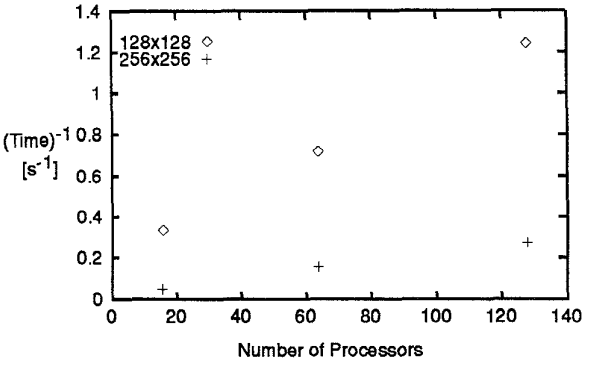

Figure 4. Speed with increasing number of processors

N	Execution Time [s]		Ratio of Exec. Times
	Sparc 2	CM-5	
32	0.383	0.11	3.5
64	2.333	0.27	8.6
128	15.24	0.81	18.9
256	109.79	3.71	29.6
512	1191.1	22.51	52.9
1024	—	175.14	—

Table 1.

Finally, Figure 4 presents data on the scalability of the parallel implementation with respect to number of processors. The number of data points here is limited by the ability to reconfigure the CM-5 (since only 16, 64 or 128 processors are possible), but the graph suggests that, over the range of trials, execution time is approximately inversely proportional to the number of processors. Note that were the number of processors to approach the number of pixels, this relationship could no longer be expected to hold for an architecture such as that of the CM-5.

5 Acknowledgements

This research was partially supported by the Australian Research Council. The authors are grateful to the South Australian Centre for Parallel Computing for permitting access to the CM-5, and to Wayne Agutter for his advice.

References

[1] M. J. Brooks and W. Chojnacki, *Direct computation of shape from shading*, Tech. Report 2176, Institut National de Recherche en Informatique et en Automatique, Sophia Antipolis, France, January 1994.

[2] ———, *Direct computation of shape from shading*, in ICPR '94 [9], pp. 114–119.

[3] M. J. Brooks, W. Chojnacki, and R. Kozera, *Impossible and ambiguous shading patterns*, International Journal of Computer Vision 7 (1992), no. 2, 119–126.

[4] A. R. Bruss, *The eikonal equation: some results applicable to computer vision*, Journal of Mathematical Physics 23 (1982), no. 5, 890–896.

[5] P. Dupuis and J. Oliensis, *Direct method for reconstructing shape from shading*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (Champaign, IL, June 15–18, 1992), IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 453–458.

[6] B. K. P. Horn, *Obtaining shape from shading information*, The Psychology of Computer Vision (P. H. Winston, ed.), McGraw-Hill, New York, 1975, pp. 115–155. (Also appears in [7].)

[7] B. K. P. Horn and M. J. Brooks (eds.), *Shape from shading*, MIT Press, Cambridge, MA, 1989.

[8] B. K. P. Horn and R. W Sjoberg, *Calculating the reflectance map*, Applied Optics 18 (1979), no. 11, 1770–1779.

[9] *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, Jerusalem, Israel, October 9–13, 1994, IEEE Computer Society Press, Los Alamitos, CA, 1994.

[10] R. Kimmel and A. M. Bruckstein, *Global shape from shading*, in ICPR '94 [9], pp. 120–125.

[11] J. Oliensis, *Uniqueness in shape from shading*, International Journal of Computer Vision 6 (1991), no. 2, 75–104.

[12] J. Oliensis and P. Dupuis, *A global algorithm for shape from shading*, Proceedings of the Fourth International Conference on Computer Vision (Berlin, Germany, May 11–14, 1993), IEEE Computer Society Press, Los Alamitos, CA, 1993, pp. 692–701.