Computers & Geosciences Vol. 17, No. 7, pp. 985–993, 1991
Printed in Great Britain.

0098-3004/91 $3.00 + 0.00

MEMORY INTENSIVE STATISTICAL ALGORITHMS FOR MULTIBEAM BATHYMETRIC DATA

COLIN WARE,[1] WILLIAM KNIGHT,[1] and DAVID WELLS[2]
[1]Faculty of Computer Science and [2]Department of Surveying Engineering, University of New Brunswick, P.O. Box 4400, Fredericton, New Brunswick, Canada E3B 5A3

(Received 19 February 1991; revised 20 May 1991)

Abstract—A set of algorithms is presented for analyzing and processing the large spatial data sets which are derived from multibeam bathymetry systems. These algorithms are designed to make use of large two-dimensional arrays to enable: (1) the estimation of how many times a particular area has been sampled, (2) the approximation of the bathymetric surface using a weighted running average, (3) the estimation of the data standard deviation continuously over the surface given the assumption of a noisy signal, (4) the breakdown of variance into within-line and between-line variance for situations where multiple ship's survey lines cover the same area and (5) the shoal-biased thinning of data. These algorithms are not restricted to processing bathymetric data, they can be applied to any data which has the property of being (approximately) randomly distributed on a plane with a roughly uniform density.

Key Words: Surface fitting, Ocean mapping.

INTRODUCTION

The advent of multibeam acoustic and optical systems to gather bathymetric information has great potential for enhancing our knowledge of the ocean floor. Applications for these data include: condition surveys for critical navigation channels; the interpretation of the geomorphology of the seabed; dredging surveys to estimate the amount of material which should be removed, and the amount of material which has been removed from a channel; applications in marine biology where biological activity can be correlated to seabed morphology; searching for mines and wrecks; sediment classification, and the preparation of charts for navigation. In order to make high volumes of bathymetric data available to these applications it is necessary to evolve techniques to clean, analyze, and visualize systematically the data.

In the Ocean Mapping Group at the University of New Brunswick we have been addressing the problem of transforming the data from the various bathymetric systems so that it is useful to these applications. One of the problems is the volume of data. Acoustic systems may produce data at the rate of 100,000 depth measurements per hour (de Moustier, 1988), whereas the airborne Lidar systems which use lasers to scan the ocean floor in relatively shallow water may produce 700,000 depth measurements per hour.

As part of this effort we have developed a set of algorithms which make use of a general programming strategy to utilize the relatively large amounts of memory that are available on modern high-performance workstations. By large, we indicate the 8 Mbyte and more which is becoming standard for the lowest cost UNIX workstations. The essence of our programming strategy is to make use of large two-dimensional arrays in the computation of statistical surfaces derived from the bathymetry data, and it is these two-dimensional arrays which make these algorithms memory intensive. However, before describing the algorithms and the statistics they produce it is worth discussing the nature of the data produced by a typical multibeam bathymetric surveying system.

THE NATURE OF THE DATA

The sounding data which we work with can be thought of as having a hierarchical structure. At the top of the hierarchy is the survey (or perhaps the set of data gathered on a single day). A complete survey consists of a set of lines usually laid out in some regular pattern, each of which represents a ship's path over a particular region together with the sounding data gathered along the course of that line. A line consists of a set of transverse profiles, each of which contains a set of n soundings, where n is the number of beams or sample points in the multibeam array of the acquisition system (in the range 12–4000). These n soundings may be arranged in a line orthogonal to the centerline of the ship's line, but they may be offset from this line in a manner which depends on the particular acquisition system. Finally, there is the individual sounding, which for our purposes is the depth computed by the signal processing system. The structure of the data is illustrated in Figure 1. Taken together, the data consist of irregularly spaced soundings with a roughly constant density. This basic structure is common to data from Navitronics, EM100 and Optech SHOALS, Krupp Atlas, Seabeam, and other systems (for a review see de Moustier, 1988).

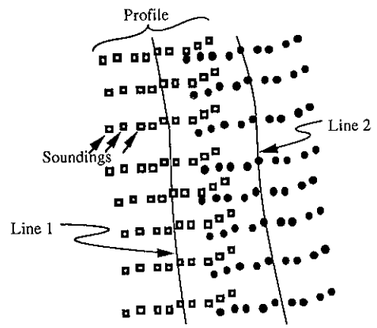


Figure 1. Structure of typical bathymetric data obtained from multibeam systems.

With this dense, high-volume data, the data error may be large relative to the density of the observations. It is usual to see a set of soundings such as those shown in Figure 2 where there is much vertical "noise" which may be attributable to a variety of factors including errors in tide correction, navigation, the algorithms used to detect the bottom, acoustic propagation, and the presence of fish or kelp.

FIVE ALGORITHMS

In the remainder of this paper we present five algorithms for analyzing this type of data and briefly make comparisons with other techniques which are used for similar purposes.

(1) Coverage Estimation

We begin by describing the simplest algorithm. The purpose of this algorithm is to discover how many times the survey vessel has covered a particular area. The output is a regularly gridded digital map of a survey area which indicates how many times each geographical point has been sampled. For the purpose of this algorithm it is necessary to assume that the ship's lines are nonself intersecting.

This algorithm makes use of two large two-dimensional arrays **Buffer_A** and **Buffer_B** which provide a digital approximation to the region of the survey. The size of the arrays determines the resolution of the resulting coverage map. These arrays either can be stored in the frame buffer of a graphics system, or as conventional array declaration.

Major variables

Buffer_A 8 bit unsigned integer (actually 1 bit is all that is needed but 1 bit arrays may not be handled efficiently). This array is used to compute the image of each individual line.
Buffer_B 8 bit unsigned integer. This array is used to compute the coverage plot.

Zero Buffer_B
FOR *each line in survey* **DO**
  BEGIN
  Zero buffer_A
  For *each sounding in line* **DO**
    BEGIN
      Draw a circle filled with the value of 1 around each sounding in **Buffer_A** at the correct location for that sounding.

    END
  Buffer_B←**Buffer_B** + **Buffer_A**
END

Note 1: for this and all subsequent pseudocode all operations on Buffer variables are done element by element thus the line **Buffer_B**←**Buffer_B** + **Buffer_A** indicates that Buffers A and B are added element-by-element and the result is placed in Buffer_B.
Note 2: it is assumed for this and subsequent algorithms that the data positions have been scaled (in latitude or longitude) so that they fit within the array boundaries.
Note 3: the array must be bigger than the bounding box of the soundings by an amount equal to the maximum diameter of the circle.

The essence of this algorithm is to create a coverage plot of each line in the survey by drawing a filled circle around each sounding in the line. The radius of this circle normally should be such that the line occurs as a seamless sheet. This sheet of ones then is added to a second buffer in which the coverage plot is accumulated and the process is repeated for the next line. The result can be color coded as in Figure 3 to show visually how many passes the ship has made over each part of the survey region.

Although this algorithm is simple, it is worth comparing it to the more usual approach to the problem to see why it is efficient. A typical approach to this problem would be to count the soundings within radius r of each grid cell in the coverage buffer. This would be slow because it involves a spatial search (at best its computational complexity is $O(n - \log n)$ to determine the points within radius r of every cell in the buffer. Compare this to the previous algorithm which required only a single pass through the data [this makes it $O(n)$] and where the cost for each sounding is simply that of drawing a circle—which is optimized in hardware for most graphics boards. We have traded large memory requirements for computational speed. This tradeoff is maintained for the other algorithms presented next.

(2) Weighted Moving Average

This algorithm produces a moving weighted average as an estimate of the ocean floor sampled by the soundings. The mean depth μ_p at position p on the

horizontal plane given a set of n soundings is computed using

$$\mu_p = \frac{\sum_{i=1}^{n} x_i w_{ip}}{\sum_{i=1}^{n} w_{ip}} \tag{1}$$

where x_i is the height value for the ith sounding and w_{ip} is the weight value of the ith sounding at position p. One option for the weight function (not the one we select) is $w_{ip} = 1/d_{ip}\alpha$ where d is the horizontal distance from p to the location of sounding and $\alpha = 2$. This function results in interpolation of the data points which may be desirable in some instances but which is inappropriate for noisy data.

The weight function which we select is cone shaped and centered around each sounding

$$w_{ip} = \begin{cases} 1 - d_{ip}/r & \text{for } d_{ip} < r \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where r is the radius of the weight field around each point. This has two advantages. The first advantage is that each point acts only locally; it has no influence outside of radius r. The second advantage is that this method produces a running average which is reasonably smooth (although not differentiable).

The algorithm which we use to compute the moving average, similar to that for the coverage plot, requires that each sounding be processed only once.

Major variables

Buffer_A two-dimensional floating point array: used to compute the numerator of Equation (1).
Buffer_B two-dimensional floating point array: used to compute the denominator of Equation (1).
Weight_field two-dimensional floating point array of size $2r \times 2r$: used to store a digital approximation of the weight field.

Zero Buffer_A
Zero Buffer_B
FOR *each sounding* x_i *in survey* **DO**
  begin
    Add Weight_field *to* **Buffer_A** *centered on sounding position*
    Multiply weight_field *by xi and add to* **Buffer_B** *centered on sounding position*
  END
Buffer_B←Buffer_B / Buffer_A

At this point **Buffer_B** contains a digital approximation to the terrain, although we have considered each data point x_i only once in deriving this surface. For display using conventional pseudocoloring **Buffer_B** should be scaled so that the values lie between 0 and 255.

This method of approximating the seabed provides one free parameter, namely weight field radius r, which can be used as a smoothing factor. For clean data r should be close to the typical intersounding distance. For data which is noisy r should be larger.

Figure 4 shows a shaded perspective view of the surface created using the given algorithm. The fine horizontal ripples are the result of inaccuracies in the data and some of the more unusual features such as the elongated bar in the lower left-hand corner are the result of outliers.

(3) Weighted Standard Deviation Surface

This is an extension to Algorithm (2), to include confidence values for the depth map μ. The weighted vertical standard deviation σ_p for point p on the plane is given by

$$\sigma_p = \sqrt{\frac{\sum_{i=1}^{n} w_{ip}(x_i - \mu_p)^2}{\sum_{i=1}^{n} w_{ip}}} \tag{3}$$

or alternatively

$$\sigma_p = \sqrt{\frac{\sum_{i=1}^{n} w_{ip} x_i^2}{\sum_{i=1}^{n} w_{ip}} - \mu_p^2} \tag{4}$$

where μ_p is the weighted moving average value at point p computed in Algorithm (2).

The given formula provides a method for computing a continuous estimate of the standard deviation of a surface derived from randomly scattered discrete data points. Each sounding has an influence only within a fixed radius r because the same weight filed as in Algorithm (2) is used here, and the influence of the sounding decreases linearly to r.

In the next algorithm we use the form given by Equation (4) for computational efficiency.

Major variables

Buffer_A, **Buffer_B**, **Buffer_C**, **Buffer_D**: two-dimensional floating point arrays used to accumulate sums and sums of squares.
Weight_field a two-dimensional floating point array of size $2r \times 2r$: used to store a digital approximation of the weight field.

Zero Buffer_A
Zero Buffer_B
FOR *each sounding* x_i *in survey* **DO**
  BEGIN
    Add Weight_field *to* **Buffer_A** *centered on sounding position*
    Multiply weight_field *by* x_i *and add to* **Buffer_B** *centered on sounding position*
    Multiply weight_field *by* x_i^2 *and add to* **Buffer_C** *centered on sounding position*
  END
Buffer_C←Buffer_C / Buffer_A
Buffer_B←Buffer_B / Buffer_A

Buffer_D←Buffer_B *Save the running average (depth map) This is the same as the output of Algorithm (2).*
Square (Buffer_B)
Buffer_C←Buffer_C — Buffer_B
SquareRoot (Buffer_C) *This now contains the standard deviation map.*

The results from applying this algorithm are shown in Figure 5 as a pseudocolored sequence where green and orange represent a high standard deviation and gray and blue represent a low standard deviation. It is worth briefly comparing this method with two other methods used to obtain a continuous estimate of the standard deviation of a surface namely binning or gridding (Midthassel, Solvberg, and Pohner, 1988; Herlihy, Matula, and Andreasen, 1988; Varma and others, 1989) and kriging (Olea, 1974). In binning, a grid is placed over the survey region, much as in the present method. However, the grid used is relatively much coarser; typically its size is set so that approximately ten soundings lie within each grid cell. Once the grid is set up, the mean and standard deviation are computed for each grid cell. It is possible to view the given algorithm as a straightforward extension of gridding, one which has the following advantages:

(1) Higher resolution. Conventional gridding treats all the points within a grid cell as if they had the same location, namely the center of the cell. Our weighted averaging technique decreases the influence of each point linearly with distance.

(2) Reduced aliasing artifacts. Consider the situation where the underlying terrain has the form of a step function (e.g. a cliff). If a coarse grid is aligned so that all the points on the top of the cliff lie in one set of grid cells and all the points at the bottom of the cliff lie in another set then the cliff information will have been preserved perfectly. However, this result is dependent critically on both the orientation and the offset of the grid. In the situation of the method described here, the grid cells are smaller and aliasing artifacts of this type will be reduced correspondingly (for our application we believe them to be negligible).

(3) More flexibility. r can be changed easily to match the data characteristics, grid size is less easy to change because it may depend on the size of fixed arrays.

There is an assumption underlying our computation of standard deviation, which is that the true surface is smooth and continuous and has relatively low-frequency variation (in a spatial sense), whereas the noise consists of high-frequency variation. Clearly, a point which is anomalous with respect to it's neighbors (deeper or shallower) will contribute greatly to the local variance. A rough terrain will be indistinguishable from a noisy signal. Ultimately, this is a sampling problem and these two instances can be resolved by increasing the sampling density and reducing the diameter of the weight field. One result is clear: the algorithm will produce a low standard deviation only in regions where both a smooth measured surface and low noise exist, hence it identifies areas which need not be investigated in more detail, either to evaluate topographic features, or to identify noisy data points.

The continuous estimate of the standard deviation has been determined useful in a number of ways. Typically regions with a high standard deviation occur because of the presence of fish or kelp in the survey area, or because the data from a particular ship's line are inconsistent with the data from other lines covering the same region. We use a standard deviation map as part of our data cleaning system to show the hydrographer where regions of high variability exist. We also use the standard deviation information in algorithms which remove outliers (blunders) from the data automatically.

(4) Weighted Variance Surfaces

In a multibeam bathymetry the seabed may be sampled more than once. The area covered by one line overlaps the area covered by other lines. It is useful to know if depth variance estimates are the result of anomalous soundings within a line as opposed to differences between lines. Differences between some entire line and others in the vicinity could be the result of calibration problems relating to positioning, incorrect calibration for tide, incorrect navigation for that line, some temporary factor such as a scattering layer, etc. Inconsistency within a line could result from some transient factor such as fish, stormy weather, moving kelp, air bubbles, temporary malfunctioning of apparatus, etc.

Each actual sounding measurement can be decomposed into three parts:

sounding = overall_weighted_average
+ (line average
− overall_weighted_average)
+ (sounding − line_average)

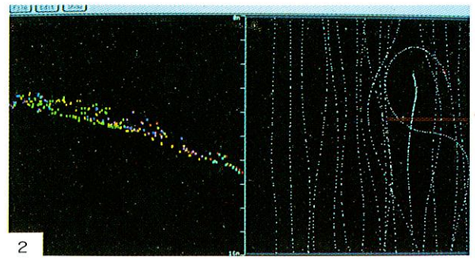
and these parts can be renamed,

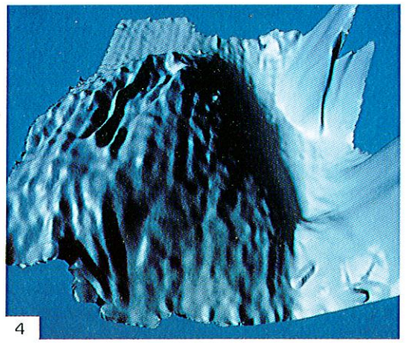
sounding = overall_weighted_average
+ line displacement from overall
+ individual displacement from line.

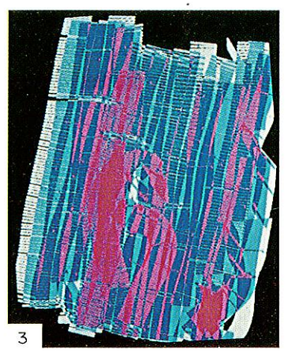
Associated with every point in an approximating array there is a mean square value for each part. These are weighted mean square values, near soundings having more influence than far. The sums of squares add, as do the components. Thus

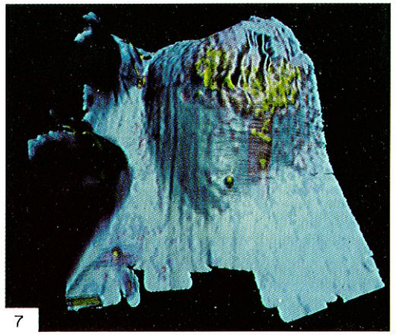
mean square of soundings =
square of overall_weighted average
+ sum of squares of line displacements B^2
+ sum of squares of individual displacements, E^2.

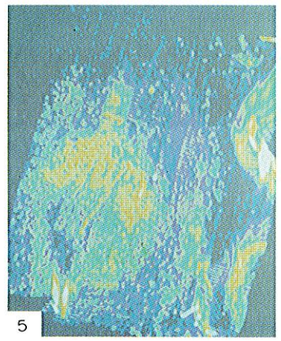
With all weights equal, the given is the standard sum of squares relation of the one way analysis of variance (divided throughout by the sum of the weights), see for example Scheffé (1959).

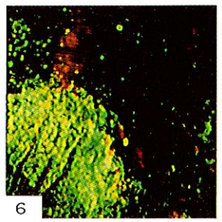

2


4


3


7


5


6

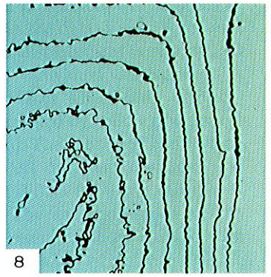

8

Figure 2. Vertical slice through set of soundings obtained using Navitronics Seadig 201.

Figure 3. Color-coded raster plot showing coverage of section of survey. Coverage was between one and six times.

Figure 4. Shaded perspective view of estimated bathymetric surface. Surface was obtained by using running weighted average.

Figure 5. Color-coded raster plot showing depth standard deviation over survey region.

Figure 6. Color-coded bivariate map showing within line variation as red and between line variation as green.

Figure 7. Bathymetric and error information are combined by representing bathymetry as height of surface and depth standard deviation as color of surface.

Figure 8. Contour plot obtained by setting every nth value in color look up table to black and rest to white.

Because

σ^2 = mean square of soundings − square of overall average

as described in Equation (4)

$$\sigma^2 = B^2 + E^2. \tag{5}$$

We determine it useful to compute the root mean square values σ, B, and E which we interpret as follows:

σ^2 = overall standard deviation. The standard deviation quantity discussed under Algorithm (3).

B = root mean square line displacement. Inconsistency amongst different lines. If this is large, suspect that one or more lines are biased up, down, or laterally.

E = root mean square individual displacement. Inconsistency within a single line. If this is large suspect individual outliers, or one or more error prone lines with inconsistent soundings.

Calculations are better described for the mean squares, taking their square roots after the fact. For line and individual displacement these are weighted variances.

The mean square line displacement B_p^2 at position p is calculated using

$$B_p^2 = \frac{\sum_{j=1}^{\text{lines}} w_{jp}(t_{jp} - \mu_p)^2}{\sum_{j=1}^{\text{lines}} w_{jp}} = \frac{\sum_{j=1}^{\text{lines}} w_{jp} t_{jp}^2}{\sum_{i=1}^{\text{lines}} w_{jp}} - \mu_p^2 \tag{6}$$

where t_{jp} represents the weighted running average of the jth at point p, and the w_{jp} is the sum of the weights for the soundings at point p in the jth line.

The mean square individual displacement E_p^2 is given by

$$E_p^2 = \sigma_p^2 - B_p^2 \tag{7}$$

where σ_p^2 is obtained using the same method as that used in Algorithm (3). The resulting algorithm is as follows:

Major variables

Buffer_A, Buffer_B, Buffer_C, Buffer_D, Buffer_E, Buffer_AA, Buffer_BB: two-dimensional floating point arrays used to accumulate sums and sums of squares.
Weight_field: a two-dimensional floating point array of size $2r \times 2r$: used to store a digital approximation of the weight field.

```
Zero Buffer_A, Buffer_B, Buffer_C, Buffer_E
FOR each line t_i in survey DO
  BEGIN
    Zero Buffer_AA
    Zero Buffer_BB
    For each sounding x_i in line DO
      BEGIN
        Add Weight_field centered on sounding position to
          Buffer_AA
        Multiply weight_field by x_i and add to
Buffer_BB centered on sounding position
        Multiply weight_field by x_i^2 and add to
Buffer_C centered on sounding position
      END
    Buffer_A←Buffer_A + Buffer_AA
    Buffer_B←Buffer_B + Buffer_BB
    Buffer_BB←Buffer_BB / Buffer_AA   running average on line t_j
    Square (Buffer_BB)
    Buffer_BB←Buffer_BB * Buffer_AA
    Buffer_E←Buffer_E + Buffer_BB
  END
Compute the weighted total variance in Buffer_C
Buffer_C←Buffer_C / Buffer_A
Buffer_B←Buffer_B / Buffer_A
Buffer_D←Buffer_B save the running average (depth map).
Square (Buffer_B)
Buffer_C←Buffer_C − Buffer_B.
Compute the weighted betweed lines variance in Buffer_E
Buffer_E←Buffer_E / Buffer_A
Buffer_E←Buffer_E − Buffer_B
at this point Buffer_B contains the between lines variation and Buffer_E contains the within lines variation
SquareRoot (Buffer_B) the root weighted mean square deviation map for between lines variation
SquareRoot (Buffer_E) the root weighted mean square deviation map for within lines variation.
```

At present we use **B** and **E** as descriptive statistics to show us various effects in the data. Figure 6 shows a plot in which r.m.s. line displacement is represented by red, whereas r.m.s. individual displacement is represented by green. Places where both displacements are high are represented by yellow (the optical mixture of red and green). The broad red band which is visible in the upper-left quadrant represents the entire ship's line which differs from the other lines covering that region. The mass of green in the lower-left quadrant represents a shoal covered with fish or kelp.

(5) Data Thinning

Athough the previous three algorithms have become progressively more complex this algorithm is perhaps the simplest of all. It implements a shoal biased data-thinning method (Orass, 1978) in a way that is both fast and which provides visual feedback.

Some applications require that the density of soundings coming from the acquisition system be reduced. For example, if a hydrographer wishes to see data from a particular region plotted as spot soundings, the density must be reduced so that the plotted numbers do not overlap. It is a widespread practice to "shoal bias" such a sample, which is to select the shallowest soundings in a particular region sub-

ject to the constraint of interpoint spacing. This biasing is appropriate if the reduced data are intended ultimately to be used in constructing nautical charts, where a conservative depiction of the bottom is appropriate. It is not appropriate where a realistic depiction is important, such as for dredging surveys and geomorphological studies (the running weighted average of the cleaned data may be more appropriate here). The shoal biasing algorithm presented here ensures that all local minimum depths (shoals) will be preserved, unless there is a shallower point within radius r. A deep-biased alternative is obtained simply by reversing the order of the sort.

Sort all soundings x_i by depth shallowest to deepest
Zero Buffer_A
FOR *each sounding* x_i *in survey* **DO**
  BEGIN
    if *(value in* **Buffer_A** *at the location of sounding* x_i not = 1)
      draw a circle filled with the value of 1 around each sounding in **Buffer_A** at the correct location for that sounding.
      output x_i
  END

By far the most expensive step of this algorithm is the sorting by depth. We use a sorting utility provided by the operating system to accomplish this.

Space and Time Performance Characteristics of the Algorithms

These algorithms require large amounts of main memory. Just how much depends primarily on the resolution required. For example, a 1024×1024 array will require 4 Mbyte of storage for each of the Buffer variables listed, assuming each variable is a 32 bit floating point array. We generally use a 512×512 array which requires 1 Mbyte per Buffer variable. Ideally this memory either should be in the form of main memory, or a frame buffer which has fast pixel access. With this resolution the ANOVA algorithm requires 7 Mbyte just for the Buffer arrays. The reason why these algorithms are useful is that this amount of memory is not unusual on even low cost workstations, also the implementation of virtual memory can make them work even with smaller amounts of memory. Nevertheless, working with these algorithms is similar to a return to an older style of programming where the memory used by every variable is important and variables must be reused wherever possible.

To give an idea of the speed of these algorithms some figures are given in Table 1 for the time taken to run each of the algorithms on a Sun 4 workstation with TAAC_1 application accelerator used as a frame buffer in Algorithms (1) and (5). For the test examples 40,686 soundings were used. The Buffer array size used was 540×540 and the weight field diameter used was 11 pixels. The number of lines was 29.

Table 1. Algorithm's speed

Algorithm	Time (sec)
1 Coverage Estimation	22
2 Weighted Moving Average	35
3 Weighted Standard Deviation	66
4 Weighted Variance Surfaces	240
5 Data Thinning	
sorting step	75
sounding selection	13

The weight field diameter is critical to the running time because the algorithm's time complexity is linear with the number of soundings but proportional to the square of the diameter of the weight field. The size of the Buffer variables is not significant particularly because it only adds a constant amount to the computation for Algorithms (2), (3), and (5), whereas it adds time proportional to the number of lines for Algorithms (1) and (4).

VISUALIZATION

One of the primary uses for this set of algorithms is for data visualization (Bishop, Monger, and Ramsey, 1990). We use a combination of shading (Ware, 1989) and pseudocoloring for this (Ware, 1988; Robertson and O'Callaghan, 1985) using software which has been developed in our laboratory. When we wish to view a bivariate choropleth map, for example, if we wish to see how height and standard deviation interrelate then we usually map one of the variables to the height of the map for the purpose of rendering and shading and we map the second variable to the color of the surface. Figure 7 shows an example where standard deviation is represented as a color sequence, whereas the running average depth is represented by the rendered surface.

Algorithm (2) also can be applied to produce a contour map. A simple way of producing a form of contour map is to use the color look up table (LUT) and color every nth value white and the rest black. This produces a plot as in Figure 8. The nice thing about this method is that given that the digital terrain representation already has been computed the contour representation is almost instantaneous (it only requires loading a 256 entry LUT) and the contour spacing can be changed as quickly. There are also a number of algorithms available for extracting a vector representation of contours from regularly gridded data (see for example Dayhoff, 1968) should these be required.

The algorithms described here are suited ideally for vector or parallel computation, in so far as all the additions of the weight field array elements to the Buffer array elements can be done in parallel. Also, all the operations which are done between Buffer variables can be done in parallel. Thus, for example, if the weight field is a 10×10 array, close to a 100-fold speedup can be expected given the appropriate processor.

Acknowledgments—This work was funded partly by a grant from the Natural Sciences and Engineering Research Council of Canada, and partly through a contract from the Canadian Hydrographic Service. The display programs used to create the figures were written by Sean Riley and K. Wing Wong. System support was provided by Ken Doucette and Mike MacDonald and the raw data are courtesy of the Canadian Hydrographic Service.

REFERENCES

Bishop, G., Monger, M., and Ramsey, P., 1990, A visualization programming environment for multicomputers: IEEE Computer Graphics and Applications, v. 10, no. 4, p. 50–58.

Dayhoff, M. O., 1963, A contour map program for X-ray crystallography: Assoc. Computing Machinery, Communications, v. 6, no. 10, p. 620–622.

Herley, D. R., Matula, S. P., and Andreasen, C., 1988, Swath mapping data management within the National Oceanic and Atmospheric Administration: Intern. Hydrographic Review, v. 65, no. 2, p. 55–73.

Midthassel, A., Solvberg, E., and Pohner, F., 1988, Data management of swath sounding systems: Intern. Hydrographic Review, v. 65, no. 2, p. 91–115.

de Moustier, C., 1988, State of the art in swath bathymetry survey systems: Intern. Hydrographic Review, v. 65, no. 2, p. 25–54.

Olea, R. A., 1974, Optimal contour mapping using universal kriging: Jour. Geophys. Research, v. 79, no. 5, p. 695–702.

Orass, S. R., 1978, Automated sounding selection: Intern. Hydrographic Review, v. 52, no. 2, p. 109–119.

Robertson, P. K., and O'Callaghan, J. F., 1985, The application of scene synthesis techniques to multidimensional image data: ACM Trans. on Graphics, v. 4, no. 4, p. 247–275.

Scheffé, H., 1959, The analysis of variance: John Wiley & Sons, New York, 477 p.

Varma, H., Boudreau, H., McConnel, M., O'Brien, and Piccott, A., 1989, Probability of detecting errors in dense digital bathymetric data sets by using 3D graphics combined with statistical techniques: Lighthouse, v. 40, p. 31–36.

Ware, C., 1988, Color sequences for univariate maps: theory, experiments and principles: IEEE Computer Graphics and Applications, v. 8, no. 5, p. 41–49.

Ware, C., 1989, Fast hill shading with cast shadows: Computers & Geosciences, v. 15, no. 8, p. 1327–1334.