F. Solina, W. G. Kropatsch (Eds.): *Visual Modules, Proc. 19th ÖAGM and 1st SDRV Workshop*
Schriftenreihe der OCG, Band 81. R. Oldenburg, Wien-München 1995, pp. 41–49

Parallelization Experiments for Radar Image Shape-from-Shading

A. Goller, M. Gelautz, F. Leberl

Institute for Computer Graphics, Technical University Graz
Münzgrabenstraße 11, A-8010 Graz e-Mail: gelautz@icg.tu-graz.ac.at

Abstract: *We start with a review of the Shape-from-Shading algorithm in its application to radar images. A given serial version of this algorithm was parallelized and improved to handle large images. It was implemented on a cluster of Silicon Graphics workstations and a multi-processor graphics computer. Important results concerning the performance of the parallel Shape-from Shading algorithm in different configurations are presented and discussed.*

1 Introduction

Raw radar data are subject of signal processing. In an imaging radar system, such signal processing will result in conventional image pixel arrays. Radar image processing will then be applied to improve the images, or to extract information about the objects appearing in the images. Due to the large quantity and high rate of raw radar signals it is customary to configure parallel processing systems for radar signal processing [1]. In contrast, radar image processing has hardly been a topic of parallel processing research, with the exception of early work with radar images from the Space Shuttle [8]. However, parallelization of radar image processing is becoming a strategy of increasing importance, partly due to the large amounts of data that need to be processed.

Data rates from current Earth-orbiting satellites carrying radar imaging sensors are high. The European Space Agency's (ESA's) ERS-1 sensor produces 8000 x 560 pixels per second, however with 32 bits per pixel or at 18 Mbytes per second, day and night and through clouds [10]. Similar data quantities can be reported from the Japanese JERS-1 radar sensor, NASA's Space Shuttle Imaging Radar "SIR-C", ESA's ERS-2, and from sensors to be put into orbit, such as the Canadian Radarsat (to be launched in 1995), the Russian Almaz, and Envisat (1998). The need to cope with extraordinary quantities of image data is being demonstrated with the radar image coverage of planet Venus which was obtained by NASA in its Magellan mission in 1990–1992 (e.g. Leberl [6]). Processing of the raw data signals acquired during the mission resulted in 400 Gbytes of image pixels. Due to the restrictions of this particular mission each pixel is being represented by only 8 bits. Radar image processing aims at organizing the image pixels into a map-like format, mainly so that future users can access, study and interact with the data. In addition, the images serve to extract information about the imaged surface, such as topographic shape, surface roughness, types of materials on the surface, or changes which might occur there over time. Such information extraction often requires the use of multiple images and of elaborate algorithms. Parallel radar image processing is therefore motivated not only by data quantities, but also by the need to cope with complex algorithms.

We begin with a review of the Shape-from-Shading algorithm, which is the subject of our parallelization studies. Its parallelization leads to numerical experiments on a cluster of Silicon Graphics workstations and a multi-processor graphics computer. We report the performance of this parallel implementation in the application to test data taken from the 400 Gbyte image data set produced in NASA's Magellan mission of planet Venus.

2 Shape-from-Shading

Shape-from-Shading (SfS) is based on the idea that the variation of brightness, or *shading*, in a radar image is a result of the terrain shape responding to the illumination by the radar sensor. Therefore an inversion of the process is feasible to extract the three-dimensional shape of an object from its two-dimensional image. Horn and Brooks [3] collected ideas about this subject as they existed in 1989. Examples of recent research on SfS include a study on local SfS presented by Sara [9] and an approach using neural networks reported by Wei and Hirzinger [12]. However, radar images were not considered in these works. Wildey [14], Kirk [4], Frankot and Chellappa [2], and Thomas et. al. [11] pioneered the field's application to radar images. Kirk [4] points out the computational efficiency of his method, which uses finite elements, as opposed to the calculus of variations approach adopted by the other authors. However, no experiments are reported on images larger than 200 $\times$ 200. The algorithm presented by Thomas et al. [11] is an extension of the approach suggested by Frankot and Chellappa [2]: Due to the use of multiple images, the simplified assumption of constant reflectance properties is no longer necessary. Furthermore, this technique proved to be more robust to noise.

Generally the assumption is made that the amount of light reflected by a particular part of the terrain surface is determined by its orientation towards the light source and the viewer, as well as by its reflecting properties σ_0. If the reflecting properties were known, then the brightness of a pixel corresponding to that surface patch would only be a function of its orientation: $I_{pixel} = R(\theta, \sigma_0(\theta))$. If we know the pixel's brightness, I_{pixel}, and its reflective behavior, σ_0, then we should be able to compute the slope θ of the surface patch with respect to the radar's antenna position. Slope values θ_{ij} in each pixel ij need to be integrated into a continuous terrain surface in such manner that it is consistent with the observed slant ranges as well with the gray values $I_{ij}(1)$, $I_{ij}(2)$ in the two input images (1) and (2). But even with the knowledge of $\sigma_0(\theta)$, the problem of height reconstruction from shading remains mathematically underdetermined. Thus, some constraints have to be introduced in order to obtain a unique solution.

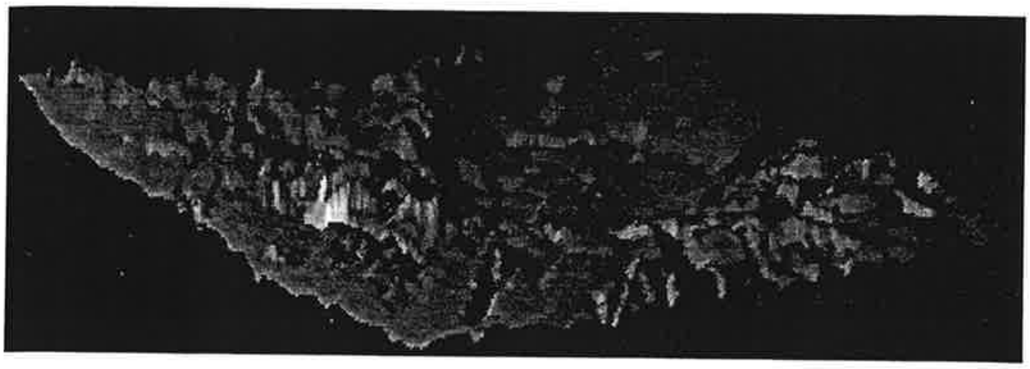

Figure 1: The area covers 176 km $\times$ 152 km. The DEM was produced from Magellan radar images with incidence angle disparities of 19°. Elevation differences are about 3200 m.

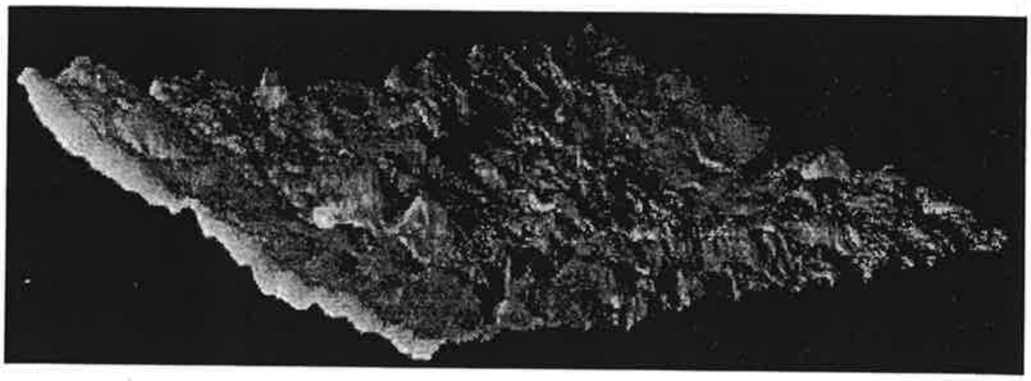

Figure 2: The DEM combines stereo and SfS data. The area is the same as in *Figure 1.* This data set is being used to assess parallelization of SfS (originally processed sequentially.)

Figures 1 and *2* illustrate the use of SfS as a refinement of a DEM produced by stereoscopy. The reconstruction problem is mathematically formulated as a calculus of variations problem. The solution is obtained by iteration. At each step of the iteration, first an improved estimate of the terrain is computed. These terrain slopes are then integrated to calculate heights. This integration is best performed in the spectral domain, hence the massive use of Fast Fourier Transforms (FFTs) in SfS. The algorithm itself is fairly complex, but extensively described in the literature [5, 11]. We therefore abstain from repeating its description here. However, there are several reasons why the recovery of SfS is by no means trivial:

(a) The slope, θ, is only determined with respect to the antenna's position (normal to the direction of the flight line of the satellite) and the slope in the along-track direction remains undetermined.

(b) The reflecting properties, $\sigma_0(\theta)$ normally remain the subject of speculation; various surface materials such as lava or wind-blown sand may follow vastly different reflection laws.

(c) Radar images are very noisy. Thus, different gray values of neighboring pixels caused by noise are also interpreted as a change in terrain height which amplifies the tendency to oscillate.

Problem (a) is solved by means of boundary conditions that are imposed on the data. Problem (b) is solved by classifying the surface into regions of homogeneous materials, and by using more than one image: using multiple I_{pixel}-values reduces the ambiguity if the imaging geometry varies widely, i.e. we have multiple orbits at different θ-values (see also [11]). Problem (c) is solved by appropriate filtering in the frequency domain and, again, by using more than one image. In fact, the stereo data serve to combine two images by geocoding them both. As a result one obtains a data set consisting of several values at each surface point xy which are input to the SfS computation:

- elevation (or slope))
- σ_0 (assumed)
- digital number (image 1)
- digital number (image 2)

Experience has shown that SfS's application should be to refine the shape information obtained from other techniques, not, however, to be the sole source of shape information [6].

3 Parallelization of Radar Shape-from-Shading

One of the reasons why parallelization is by no means trivial is that the available tools are very specific and mostly bound to computer architectures. Thus, the computing environment must be predetermined prior to the design of the parallel algorithm. Further problems occur if the parallelized algorithm should be scaleable, meaning that it can be distributed across any given number of processors. Data decomposition distributes equally sized parts of the whole data set amongst all processors. Due to the 2-dimensional structure of SfS, this approach seems to be better than program decomposition which suffers from the fact that the code cannot be split up into an arbitrary number of equally sized parts. Another major problem, which has not been completely solved until now, is the question how to handle large data sets:

3.1 Problems with large data sets

Whilst the FFT works well for small images, the time needed by Fourier- and Inverse Fourier Transforms rises according to their time complexity of $O(n^2 \cdot log(n))$, whereas n is the number of pixels in one line of the image. Calculating the FFT for the whole image would last too long and is not necessary because the SfS algorithm is just designed to do local slope and terrain refinements. The overall shape of the terrain can be determined from the initial DEM, which is generally derived from stereo processing or, in the case of new satellite missions like ERS-1 and ERS-2, from interferometry.

Using just local FFTs has two impacts: Firstly, execution time is reduced to $O(n^2)$, because the FFT is applied to equally sized, small parts, regardless of the total problem size. The time complexity of the whole algorithm thus is reduced to $O(i \cdot n^2)$, where i is the number of refinement iterations. Secondly, the lower frequencies, which are obtained by the FFT in the spectral domain, must be replaced by the low frequencies of the initial reference DEM. This guarantees that the small terrain patches which can then be calculated independently fit together after the refinement operation.

SfS then merely serves to refine high-frequency, local surface shape, but leaves the low-frequency, coarse shapes obtained form stereo or interferometry unaffected. However, the low frequency enforcement guarantees that the patches do not differ with respect to medium height and average slope. Local disparities occurring at the seams must be removed by an extra procedure called feather. As shown in *Figure 3,* the patches overlap each other. To obtain satisfying results in conjunction with an acceptable amount of overhead calculation, the overlapping region was found to be about one quarter of the patch size. Due to the fixed size of all patches, the overlapping region is larger at the rightmost and bottom patches.

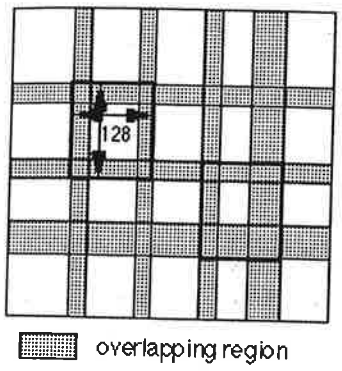

Figure 3: Data decomposition of the whole input DEM.

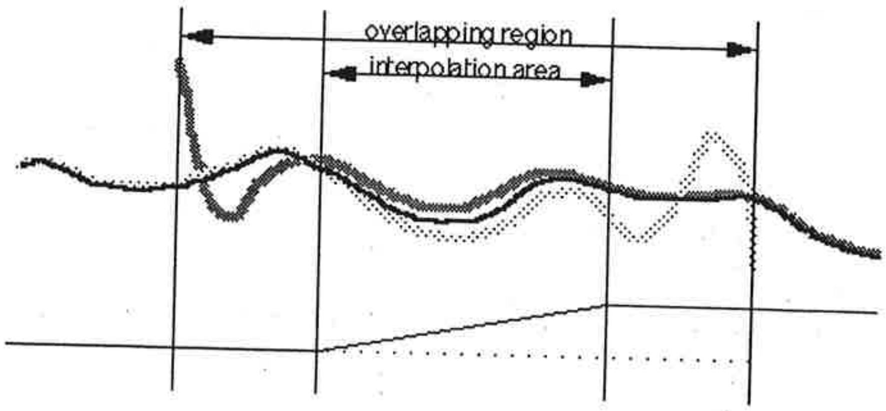

Figure 4: Interpolation between two DEM patches *(feather).* The gray lines represent the profile of the patches and the black line the profile of the resulting DEM.

Another reason for using small patches is that SfS tends to produce unstable solutions from large pixel matrices. This results in very high-frequency and high-amplitude oscillations, especially at the rim of the patches. Using larger patches than 128 $\times$ 128 pixels would result in an additional oscillation over the whole patch. This is mostly caused by the form of the cost function and the implementation of the constraints, which are imposed in order to guarantee a unique solution. If the parameters passed to SfS are not set properly, both types of oscillations will influence a broader seam with even higher amplitudes at the rim. This results in images where a curtain-like effect will be seen.

However, this effect cannot be removed completely until now. Thus, the overlapping region, and the percentage of it used as interpolation area, must be set in such a way that there are no artifacts in the resulting DEM. About 10 pixels are cut off and replaced by the neighboring patch. In order to provide a smooth transition between the patches, interpolation is done in the middle part of the overlapping region, where the terrain is linearly interpolated from one patch to the other one. *Figure 4* shows a sketch of how the feather routine works. However, this interpolation is not always satisfying and will be subject to further research.

3.2 Iteration Control

As mentioned in *section 2,* SfS refines the DEM iteratively. When executed on a single processor, SfS runs interactively and the user has control over several parameters and may react to statistics and status information produced by SfS. However, parallelization only works if the calculation of all the patches can be controlled by a single master process. This master process may be able to report some status information to the user, but the parameters for the calculation must be produced by the program itself. This means that the parallelized version runs in a batch-like mode and provides less feedback to the user than the serial one, which prevents one from tuning parameters between different iterations.

Particularly, it is difficult to determine the necessary number of iterations, because sometimes the cost function will pass a local minimum to get a better one. Finishing the iteration loop at this local minimum may result in non-satisfying results. Therefore, sometimes continuing the search for another minimum may produce better results, but this is, of course, not guaranteed at all. Hence, user interaction at the end of each iteration will be highly desirable, because the cost function is not the only criterion for termination. Iteration control in the parallel version of SfS is done by a fixed number specified by the user before the calculation starts. Further work will be necessary to find better and computer-decideable iteration termination conditions.

3.3 Parallelization by Data Decomposition

The algorithm has four distinct parts internally denoted as: *User Interface, Breakup, Hagfors and Feather.* Since we need to cope with long execution times, the *user interface* serves to input some initial processing parameters to start and control SfS as a batch job. Even on a graphics workstation by Silicon Graphics Inc. (SGI) with a MIPS-R4000 processor on board, it will take more than two hours for a 1000 × 1000 pixel DEM to complete about 25 iterations for successfully computing an SfS result.

Breakup, located at the master processor as shown in *Figure 5,* divides the original DEM, generated by stereo matching, and the two corresponding orthorectified SAR images into subimages (patches) with a constant size of 128 lines × 128 pixels. These patches are distributed to the working processors. *Hagfors* is the actual SfS shape refinement as discussed in *section 2.* It is applied to each patch. Finally, *Feather* gathers all subimages and merges them into a seamless array. This requires the removal of the errors at the seams of the subimages. The interpolation algorithm mentioned above is applied to calculate a smooth transition from one DEM patch to its neighbors.

From the theoretical aspect, the data decomposition approach is the best one for the SfS algorithm, too. Sometimes, especially if high user interaction is demanded, theoretical results and applicable implementations differ. As the so-called "grain size lemma" states, the parallelization should start at the outermost loop [13]. In our case, it is the loop over all subimages. Each process is able to refine "its own" DEM patch without looking around or waiting for synchronization signals.

Breakup and *Feather* are not very suitable for parallelization, because the serial part of the algorithms is too large. Both have to read or write a single image and to calculate indices and border values for all subimages. The percentage of serial code f should not exceed $f = 1/(p-1)$, where p is the number of processors [7]. Thus, since $f > 50\%$ in these components, there is no use for more than two processors. Furthermore, it does not make sense to parallelize these two routines since execution times are far less than in *Hagfors*, and execution is I/O bound.

3.4 Computing Environment

The concurrentized algorithm is designed to run on a workstation cluster. In our case the cluster consists of SGI workstations, namely Indigos, Indies and one SGI Power Challenge with two processors. They are interconnected via Ethernet. Data is transferred using the network file system NFS, whilst communication is done by BSD sockets. To process an input data set of about 1000 × 1000 pixels, we need to handle about hundred patches and manage approximately one or two dozen computers. Therefore, the problem size is much larger than the number of processors. This is typical of multiprocessor architectures.

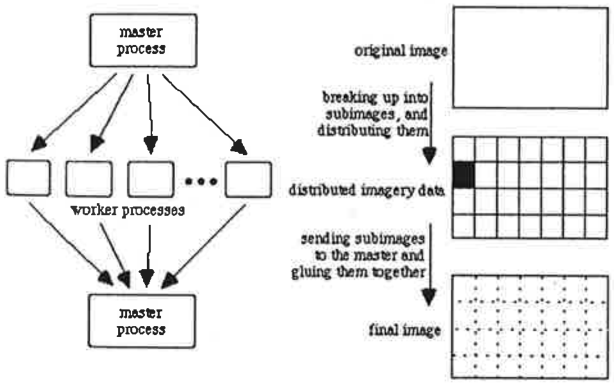
master process
worker processes
master process
original image
breaking up into subimages, and distributing them
distributed imagery data
sending subimages to the master and gluing them together
final image

Figure 5: Logical sequence of data decomposition and feathering.

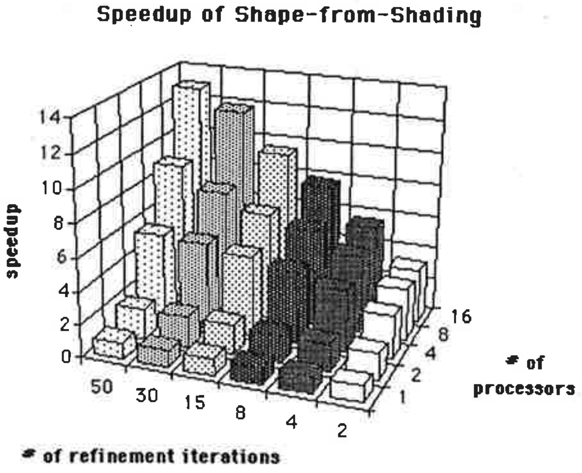

Figure 6: Speed-up obtained by refining a stereo-derived DEM (see *Figures 1* and *2*). Problem size is 1000 × 1000 pixels.

4 Performance Assessment

Before measuring performance, one should discuss all parameters that can be varied. An obvious parameter is the number of processors p that is only the overall sum of processors in the cluster, regardless of the individual speed of a workstation. Another important performance parameter relates to image size n^2. In our experiments the number of iterations i, which is a further parameter, is varied between 2 and 50. The size of a single subimage may be seen as another parameter. However, changes influence the resulting DEM. We therefore abstain from treating it as an independent parameter.

Figure 6 presents some results of processing two images and a stereo-derived DEM which is interpolated to the same size. The speedup of one processor is scaled to 1 in each column, although the time needed for 50 iterations is about 50 times longer than that for a single iteration step. Due to the inhomogeneity of our cluster, some strange results appear. Looking at the row with four processors invoked, a speedup of 5.2 is displayed at the leftmost bar. The reference machine, in the first row, was a relatively slow Indy, and as the number of processors increased, some more powerful Indigos and the Power Challenge were added to the cluster. An Indy was used as reference because nearly all computers in our cluster are Indies. Additionally, it is to be noted that our workstation cluster is not a stand-alone facility. All computers are normally used by multiple users. Other jobs may therefore run at the same time while we attempt to measure the performance. Thus, some perturbation may be seen in our results due to the interference with ongoing and unrelated use of the workstations.

Given these constraints we note from *Figure 6* that, using 16 processors, a speedup of 13.4 could be achieved. It also can be seen in the upper right corner that the size of the tasks should not be too small when using many processors. Ideally, the speedup should be constant along one row, which is true for the first two rows. Due to the bus architecture of Ethernet, only one sender is allowed any time. This causes, especially if many processors are involved, communication hot spots and collisions, and communication becomes the bottleneck. It shows that Ethernet performs quite poor as the underlying network in parallel computing. Another problem arises as image size n increases. *Hagfors* needs at least five files – two images, one DEM and one parameter file for input and the output DEM. Refining, for example, a 3000 × 3000 pixel image, this results in more than 5000 files that must be handled concurrently by the file system. We noticed wait cycles caused by the operating system.

5 Conclusions and Outlook

We have begun work to create parallelized radar image processing software. Initially we implemented a parallelized version of an existing system for radar image SfS and studied its performance on a cluster of graphics workstations by Silicon Graphics Inc. The process for a single image with 1000 × 1000 pixels on a single SGI computer with an R4000 processor may consume more than two hours. Executing the same mathematical process on 16 workstations reduces the computing time to about ten minutes.

Based on the knowledge obtained from this implementation, we intend to proceed with the parallelization of other time-consuming algorithms used to process Magellan's massive 400 Gbytes of image data, as well as signal processing code to compute pixels from raw radar echoes. Simultaneously, of course, we need to address the issue of the mathematical methods used for image analysis. As we proceed we will certainly also deal with improvements in these methods.

References

[1] Curlander, J., R. McDonough (1991) *Synthetic Aperture Radar: Systems and Signal Processing*; Wiley

[2] Frankot, R., R. Chellappa (1989) A Method for Enforcing Integrability in Shape-from-Shading Algorithms, in B. Horn, M. Brooks (eds.), *Shape-from-Shading*, MIT Press, Cambridge

[3] Horn, B., M. Brooks (1989) *Shape-from-Shading*; MIT Press, Cambridge, Massachusetts

[4] Kirk, R. (1987) *A Fast Finite-Element Algorithm for Two-Dimensional Photoclinometry*, Thesis, California Institute of Technology, Pasadena, California

[5] Leberl, F. (1989) *Radargrammetric Image Processing*, Artech House, Norwood, Massachusetts

[6] Leberl, F. W., J. K. Thomas, K. E. Maurice (1992) *Initial Results From the Magellan Stereo-Experiment*. J. Geophysical Research, 13675-13687

[7] Quinn, M. (1988) *Algorithmenbau und Parallelcomputer*; Mc-Graw-Hill

[8] Ramapriyan, H. et al. (1986) Automated Matching of SIR-B Images for Elevation Mapping, *IEEE Trans. Geoscience and Remote Sensing*, vol. GE-24 no. 4

[9] Sara R. (1994) Physical Correctness of Local Shading Analysis; in "Mustererkennung 1994", eds. W.G. Kropatsch and H. Bischof, Informatik Xpress

[10] Schreier, G. (1993) *SAR Geocoding: Data and Systems*, Wichmann Verlag, Karlsruhe

[11] Thomas, J., W. Kober, F. Leberl (1991) Multiple Image SAR Shape-from-Shading; *Photogrammetric Engineering and Remote Sensing*, vol. 57, no. 1

[12] Wei, G., G. Hirzinger (1994) Learning Shape-from-Shading by Neural Networks; in "Mustererkennung 1994", eds. W.G. Kropatsch and H. Bischof, Informatik Xpress

[13] Wenzel, L. (1991) *Parallele Programmierkonzepte*; Franzis Verlag, Muenchen

[14] Wildey, R. (1986) Radarclinometry for the Venus Radar Mapper; *Photogrammetric Engineering and Remote Sensing*, vol. 52, no. 1