CHAPTER 18 - PARALLEL SURFACE RECONSTRUCTION

Consideration of the proper approach to a parallel implementation of the feature point surface reconstruction algorithm has been done. A minimal partial implementation has been created. The considerations and implementation are discussed in the following section.

18.1 PRINCIPLES

Several portions of the surface reconstruction algorithm are inherently parallel as long as enough data is available. For example, if enough points are available on a processor, neighborhoods can be determined and surface normals calculated independently from work performed on other processors. Once the normals are determined, the points in the neighborhood can be projected and sorted and boundary and variation feature points extracted as long as the data points for all neighbors are available.

Most parts of the algorithm are driven by loops that either examine each point or, in the later stages, examine each edge. If a processor can access all relevant data efficiently, multiple processors can execute part of the loops on a subset of the points or edges.

A shared memory implementation appears natural since the data modification done by the algorithm is almost exclusively performed for one point at a time and in such a way that another process would not be modifying the same point or edge data structure concurrently. There would need to be synchronization points so that a processor would not attempt to use data not yet available. An example of such synchronization is that a processor could not examine feature
points in the neighborhood of a point if another processor that was responsible for detection of the points had not completed its task.

The fact that all data is available to all processors is a large advantage for shared memory parallel algorithms for surface reconstruction. However, shared memory processors, other than SMP (Symmetric Multiple Processor) machines with 2 to 4 processors are not commonly available. Therefore, a distributed memory system was used for the initial investigations of a parallel implementation of this algorithm.

18.2 DIFFERENCES FROM SEQUENTIAL ALGORITHM

The general areas of the feature point surface reconstruction algorithm are discussed below with comments as to the necessary differences to support a parallel implementation.

18.2.1 Neighborhood Processing

The first task to begin neighborhood processing is to distribute the points to the processors. One approach would be to send a subset to each processor consisting of the points to be processed by the given node. In addition, any points within a neighborhood radius of the point set would also need to be available on a given node. This approach would require some sequential processing to determine the points to be sent to each node. In addition, some of the results such as identified feature points would need to be communicated to the root node and back to other nodes.

If sufficient memory is available, an alternate approach would be to store all points on all processors. Each processor would be responsible for a subset of points but would have access to
all potential neighbors. Some additional communication would be required so that a node would know which points outside its subset were identified as feature points. There would be several trade-offs to be considered between need for synchronization and data communications.

18.2.2 Feature Point Extraction

If all neighbors are available, each node could do all feature point extraction for its portion of the data set without interaction with the other nodes. A synchronization point would be necessary after feature point extraction. Boundary and normal variation feature points could be extracted before interaction with other nodes is necessary, but extraction of complementary points should wait until all nodes are finished extracted the initial feature points.

18.2.3 Mesh Generation

Much of the mesh generation process can be done independently on individual nodes as long as data is available for all neighbors of the node's point subset. However, when edges are extended neighborhood boundaries are crossed and more communication will be necessary. Similarly, polygon traversal and triangulation may span multiple point subsets. Therefore, data must be shared either by distributing all data to each node and transmitting updated point information after the various algorithm steps, or by communicating on an as-needed basis. The latter type of communication can be quite time consuming since nodes to which messages must be transmitted need to be determined and several types of messages may be active at a given time. Synchronization would also be more difficult in this environment.
The actual processing involved on polygon traversal and triangulation would need to be divided between the processors in a different manner. In previous steps, a node could be responsible for data points assigned to it if appropriate information about other data points was available locally or upon request. For polygon and triangle manipulation, a processor would need to modify data structures that correspond to points for which it was not originally responsible.

18.3 PARTIAL IMPLEMENTATION

Based on the above concepts, a implementation has been begun but is very incomplete. The present results are described below after a description of the hardware and software utilized.

18.3.1 Hardware and Software

The distributed memory system utilized is a Parsytec PowerXPlorer system which consists of 8 nodes each of which is an 80 MHz PowerPC processor coupled with a Transputer dedicated to communications between nodes. The system is configured as 2 groups of 4 nodes. Each group is housed in a cabinet and the nodes are connected in a square configuration. In addition, the two groups are connected by an external cable so that a hypercube structure is created. Communication between the 2 physical boxes, however, is slower than communication between nodes in the same box.

The Parsytec system runs the PARIX operating system that is a parallel extension of UNIX and relies on an external host processor for network and disk communication. The connection to the host processor, a Sparc running SunOS 4, is a SCSI controller. Although native communication
primitives are available in PARIX, a version of PVM has been utilized that understands the native communication channels of the Parsytec hardware.

One of the limitations of the available configuration is that each node has only 8 MB of memory. The system will be upgraded soon to 32 MB per node that will make further exploration practical for large data sets.

18.3.2 Results

The current implementation reads data on a root node and broadcasts all the data, along with necessary parameters, to each node. Each node, including the root node, is responsible for determining the subset for which it is responsible, finding the maximum and minimum values of the data for x, y, and a, and constructing a grid to assist in neighbor identification.

Each node then establishes neighborhoods, calculates neighborhood normals, and transmits its modified data to the root node. The root node than broadcasts all data to each node.

Individual nodes then project and sort neighbors and check points to determine if they are boundary feature points. The next step would be to calculate variation feature points and transmit the feature point information back to the root node.