Implementation of a well vectorized Marching Cube Algorithm in a Parallel Machine

Prof. Dr. André Luiz Battaiola (1)
Dr. William Hibbard (2)

DC / UFSCar (1)
Rod. Washington Luiz, km 235
São Carlos - SP - 13565-905
andre@dc.ufscar.br

SSEC/University of Wisconsin (2)
1225 W. Dayton St
Madison, WI 53706
whibbard@mscc.wisc

Abstract: This paper describes the results of the implementation of a well vectorized version of the marching cube algorithm, used to generate three-dimensional isosurfaces in scientific visualization applications, in a massive parallel machine called CM-5 (Connection Machine). The implementation was made in the message passing and data parallel programming methods.

Key words: Scientific Visualization, Parallel Processing, Isosurfaces

1- Introduction

Isosurfaces are an important tool in many scientific visualization systems. In order to increase interactivity, a fast version of marching cube algorithm [LOREN87] was used to run on computers with vector facilities was developed.

In an experiment with a virtual reality environment, a parallel machine was used to obtain performance compatible with the interactivity required by the application. The vectorized version of the algorithm was ported to a CM-5 parallel machine. This paper describes the main ideas of the vectorized version of the algorithm and the changes required to port it to the CM-5.

It will be used in this text the following convention:

Dgrid: is the data grid supplied by the user (data grid)
Dsubgrid: is a subset of this data grid (data subgrid)

This convention will avoid confusions with the term subgrid used to indicate a subset of data mapped in the CM-5 nodes with vector facilities. In the same way, we use the term node or processor to mean a CM-5 processor. This will avoid confusions with vector processor, the resource that some nodes have and that is responsible for vector processing.

2- The Algorithm

The marching cube algorithm interpolates a surface through a 3D grid (Fig. 1) defined by all points at a specified value called an isovalue (Fig. 2).

The marching cube algorithm analyzes each grid cube and marks each cube vertex with a flag. The flag is 1 if the vertex has associated a value bigger than the isovalue, and the flag is 0 if the value is equal or lower than the isovalue (Fig. 2). Ordering the cube vertices, it is possible to associate a byte to each cube configuration (Fig. 3).

As the cube has eight vertices, there are 256 configurations. Using rotations of the cube, it is possible to reduce the number of configurations to 30 basic configurations (Fig. 4). Considering negative configurations as equivalent, the number can be reduced to 15 basic configurations. A negative configuration occurs when for each vertex of a cube A, the same vertex of a cube B has a negative logical value. In this implementation we kept the distinction between positive and negative configurations to avoid problems with the calculation of the normals of the polygons.