# TNCG14 — Advanced Computer Graphics Programming
# 3. Parallel Programming

March 16, 2009

The task is to implement a simple direct ray-casting program to render a volume of data and then to parallelize the application for faster execution. Data can be downloaded from the website volvis.org. The selected data should be of a resolution of at least $256^3$ to make the parallel computing worthwhile.

The direct ray casting algorithm can be found in any standard volume graphics textbook. The basic idea is that for each pixel in the image a ray is cast through the pixel and into the volume of data. As the ray passes through the data, the values of each voxel are combined to define the colour and intensity for the pixel. This enables a display which the user can use to see the interior of 'solid' objects present in the data. The individual rays cast through the data are, of course, completely independent and so the rays (and hence the pixels) can be calculated in parallel.

To make the implementation rather less complex only a very simple implementation is required for this lab: perspective is not needed and there is no requirement that the user should be able to rotate the camera or data though it would be fun if these things were available. This reduces the problem to a plane ray casting, each ray being orthogonal to the screen and to the data (unless you rotate it) so it should be quite easy to implement.

Once you have a working serial program you can parallelize it using pthreads or OpenMP directives (both available through 'cc') on the 8 processor SGI Onyx2 machine, 'matrix.itn.liu.se'. For OpenMP, include omp.h and use the -mp compiler flag. You should be able to log into that computer system using ssh. Think about the way in which the task should be partitioned between the processors and how the data should be partitioned so that the different processors can operate on different data segments without conflict. You may also use OpenMPI on the computers in the Linux-lab, but that is a more complicated approach.