CATEGORY: VISUALIZATION - IN-SITU & SCIENTIFIC - VS02

POSTER
P5116

CONTACT NAME
Ricardo Santos: ricardo.santos@ufpr.br

GPU TECHNOLOGY CONFERENCE

UFPR

MASSIVELY PARALLEL ISOSURFACE EXTRACTION OF CONCRETE SAMPLES

Ricardo César Ribeiro dos Santos (ricardo.santos@ufpr.br), Klaus de Geus, Walmor Cardoso Godoi

PPGMNE
PROGRAMA DE PÓS-GRADUAÇÃO EM MÉTODOS NUMÉRICOS EM ENGENHARIA

INTRODUCTION

In order to preserve concrete samples from destructive testing and allow better experiments to be conducted using simulation, non-destructive methods were developed.

These procedures involve using mathematical concepts such as finite elements or boundary elements to calculate the behavior of the real element in a specific set of conditions.

These methods require a virtual model with as much detail as possible, preserving the geometry of the real body. To extract these characteristics, computer tomography has proven to be a reliable method to extract a density map showing the individual materials of the sample and create a discrete density function for the real body.

For the mesh creation from the tomographic images, an implementation of the Marching Cubes algorithm was used to extract the isosurface. This posed another challenge, as the function to reconstruct the isosurface returned an error related to memory overflow after more than 40 minutes of processing.

To address this issue, an alternative Marching Cubes implementation was developed, based on the bibliography, using the MATLAB suite for prototyping the implemented code and to benchmark the implementation speedup throughout the development cycle and different implementations.

A MATLAB script of the Marching Cubes was initially implemented. As an intermediate step, a vectorized version of the code and finally, a CUDA version of the same algorithm.

Tests were carried out by processing increasingly larger data sets from the whole tomographic image space. The times were computed as a way to measure the performance increase in time and in memory consumption.

CONTEXT

This system was developed as part of a project sponsored within the R&D programme of Agência Nacional de Energia Elétrica (ANEEL) to monitor and ensure the integrity of the dam structure of the Rio Jordão power plant. A part of the project required the extracted sample to be reconstructed in order to undergo non-destructive testing.

For extracting geometric data from the concrete sample, computer tomography was used, generating a dataset composed of 585 tomographic images.

The CPU-based approach was soon dismissed as a not viable option, as it was unable to handle the amount of data to be processed, as the solution implemented would execute for forty minutes and be aborted due to a memory overflow error.

Previous experiments from the project had shown the GPU approach for the problem to be viable, with reduced processing time and memory consumption. As such, a CUDA solution was devised as the path to be followed.

MATERIALS

The concrete samples used in the project are shown in Figure 1. The samples were extracted from the Rio Jordão power plant dam. The tomography procedure to which they were subjected is shown in Figure 2.

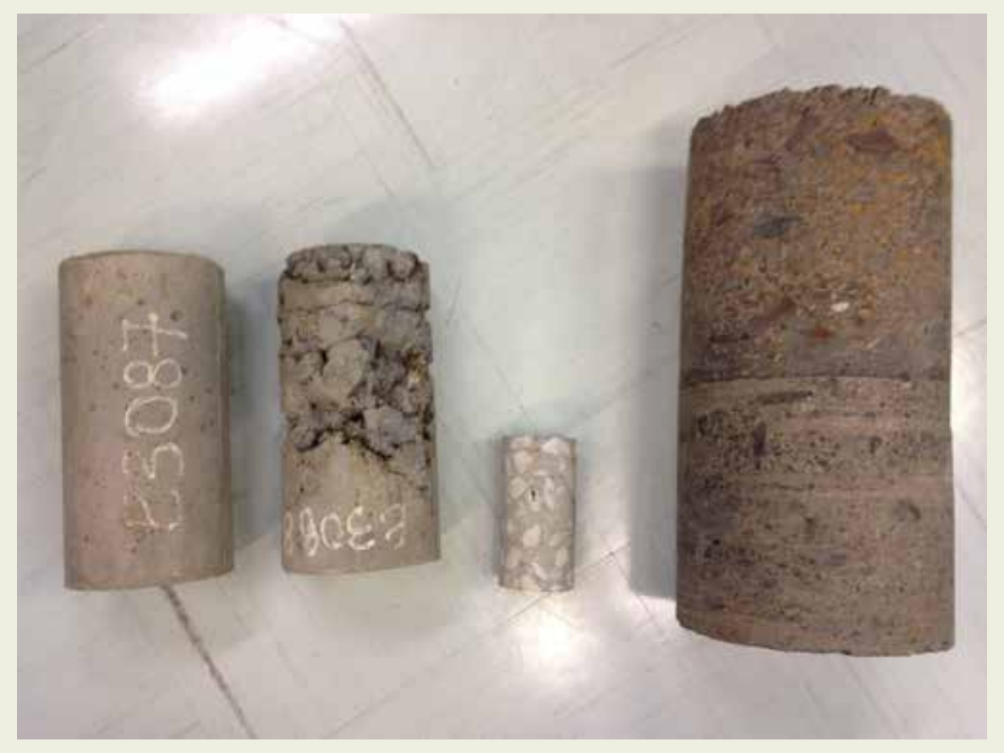


Figure 1: Concrete samples.

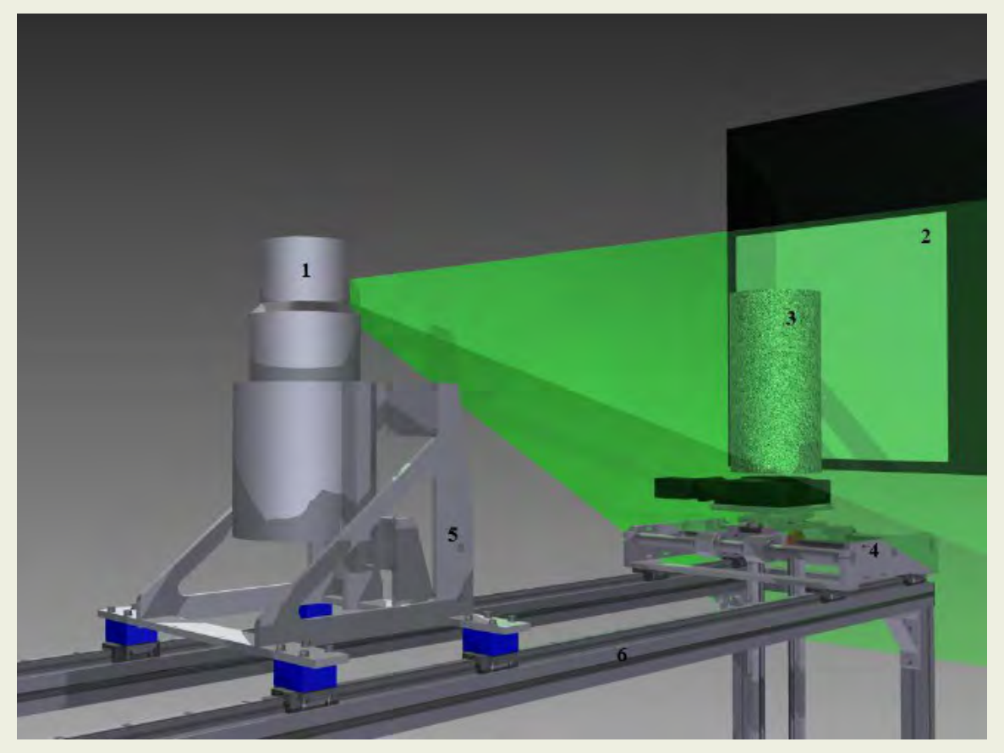
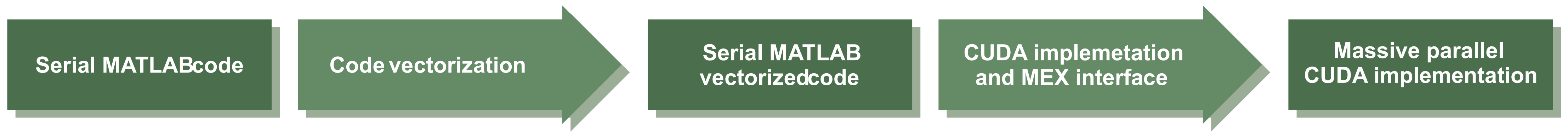


Figure 2: Industrial tomography procedure.

DEVELOPMENT LIFECYCLE

For developing the CUDA application and validating the concept, a prototyping based approach was adopted. The code was initially implemented as a serial MATLAB script and later vectorized. Finally, the MEX interface of the MATLAB suite was used in order to test the CUDA software module as a dynamic library.

The development lifecycle adopted is illustrated in figure 1.



Serial MATLABcode

Code vectorization

Serial MATLAB vectorizedcode

CUDA implemetation and MEX interface

Massive parallel CUDA implementation

Figure 1: Software development lifecycle.

THE MARCHING CUBES ALGORITHM

The Marching Cubes algorithm was first described by Lorensen and Clyne in 1987 [1], and it was described as a divide-and-conquer approach to generate inter-slice connectivity and define triangle topology. In this implementation, the algorithm uses data from the tomographic density maps to create polygonal meshes representing the isosurfaces from the materials that compose the concrete samples.

The Marching Cubes algorithm is based on the conditions in which an isosurface intercepts a cube-like cell. As there are 8 vertices in a cube, this represents 256 (2^8) possibilities. Through rotation and similarities operations, these cases can be reduced to only 15, as shown in figure 2.

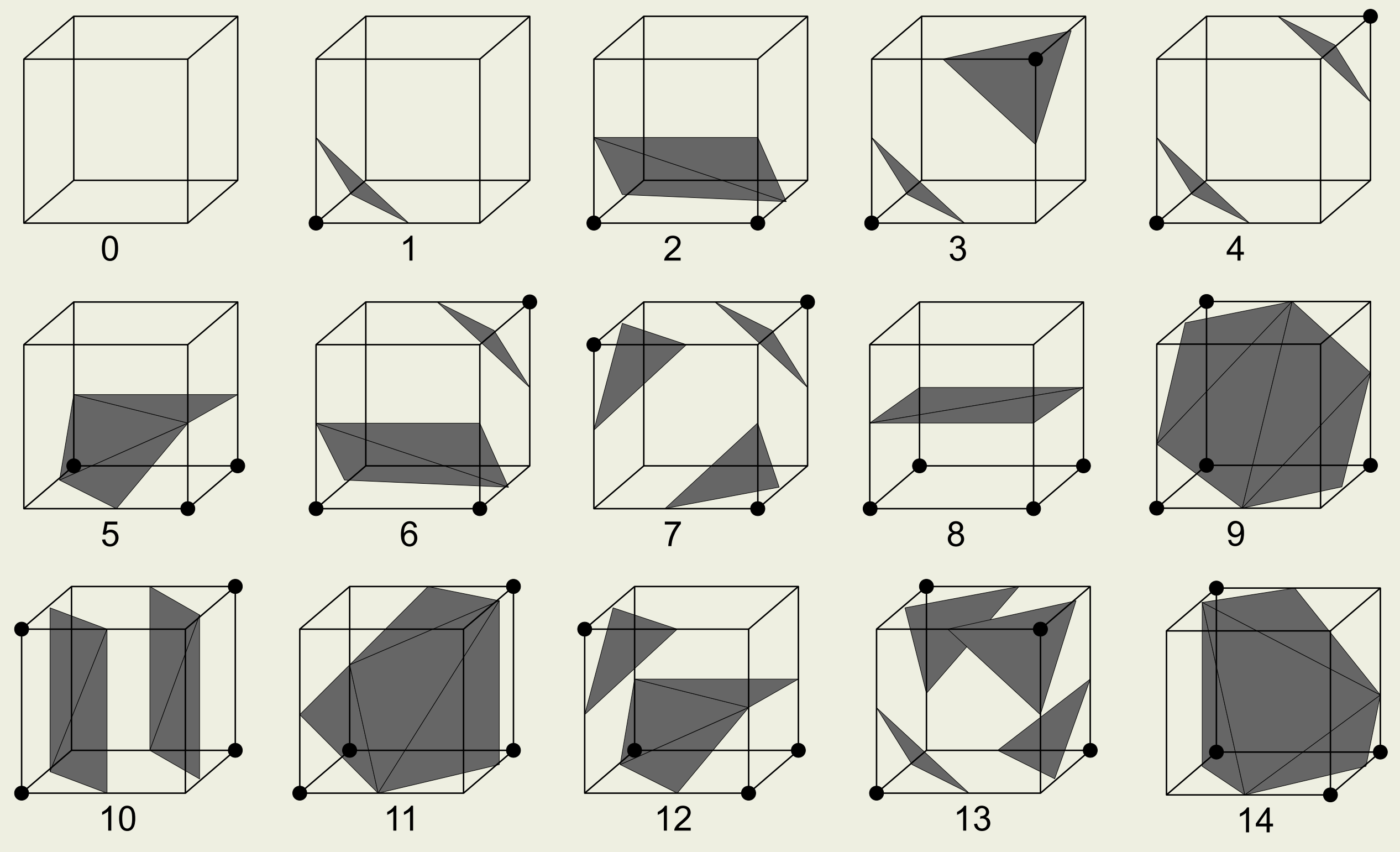


Figure 2: The Marching Cubes algorithm cases in wich an isosurface can intercept a cube.

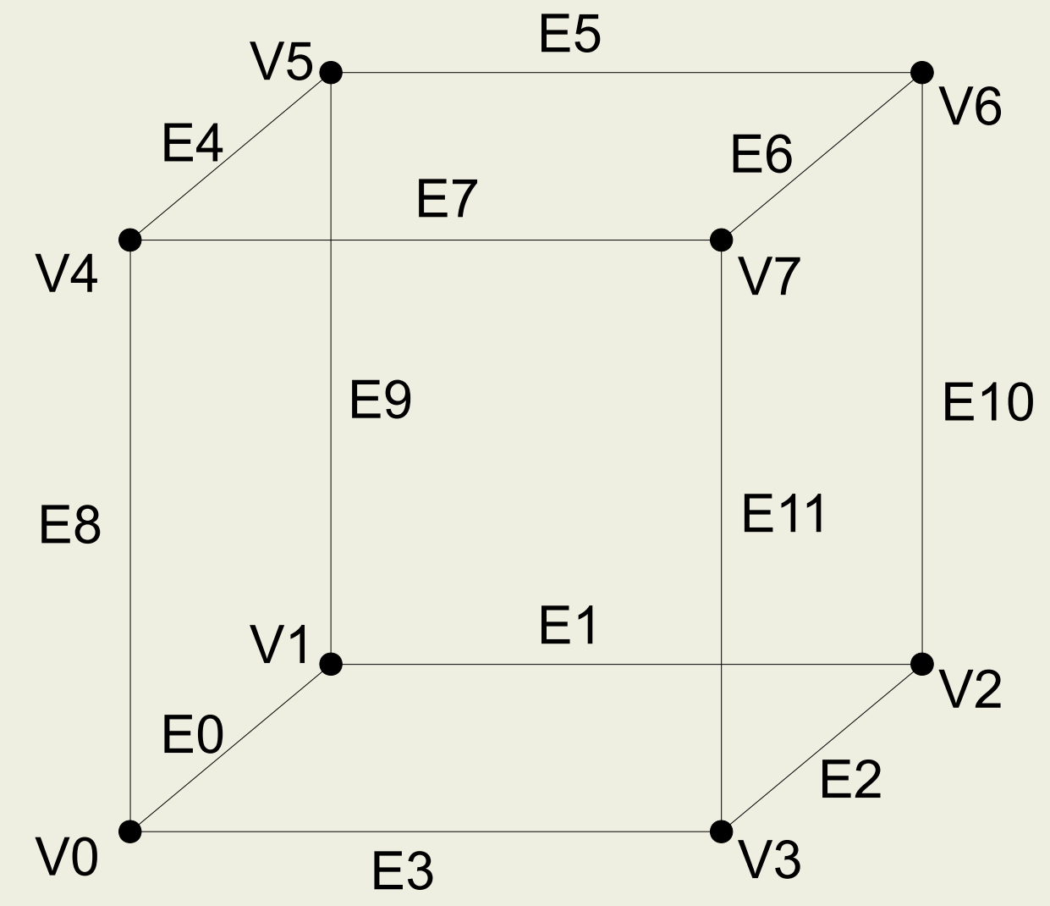


Figure 3: The Marching Cubes algorithm cell, its vertices and edges.

For determining which case can be applied to the isosurface reconstruction, the cell vertices and edges receives indices which are used by the algorithm to access the vertices, edges and normal vectors to the triangles of the relevant isosurface patch being processed, as illustrated in Figure 3.

RECONSTRUCTED SURFACES

The polygonal meshes produced by the developed application are depicted in Figure 4 and showed to be sufficiently detailed for non-destructive testing procedures.

The memory consumption was unexpectedly high, surpassing the available memory in the GPU memory space. To overcome this challenge, a memory management code was implemented. Essentially, it just divided the dataset in equal intervals that were sequentially dispatched for processing. In a subsequent step, the processing results were joined in order to produce the whole reconstructed mesh.

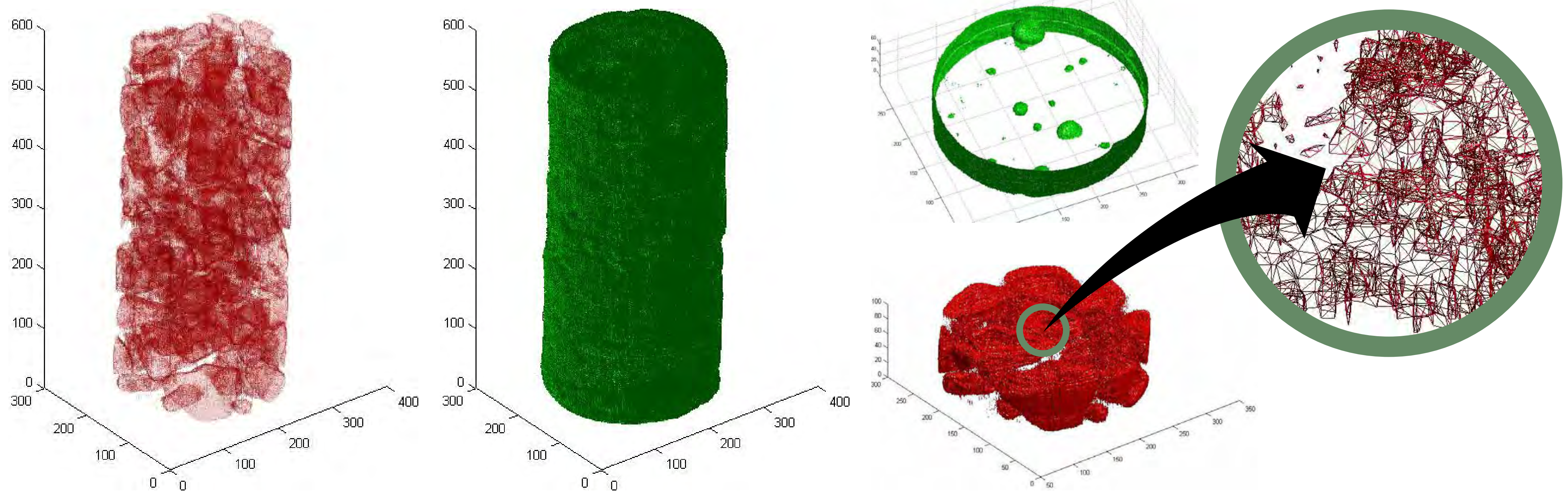


Figure 4: Virtual reconstructon of the concrete samples.

RESULTS

The massively parallel Marching Cubes implementation using GPU reduced the processing time, as shown in the diagrams of Figure 5. Comparing to the serial CPU implementation on MATLAB, the vector implementation, also using MATLAB, achieved a speedup of more than 10 times, whereas the CUDA implemention achieved a speedup of more than 5500 times. For the MATLAB vector implementation, the speedup was more than 10x.

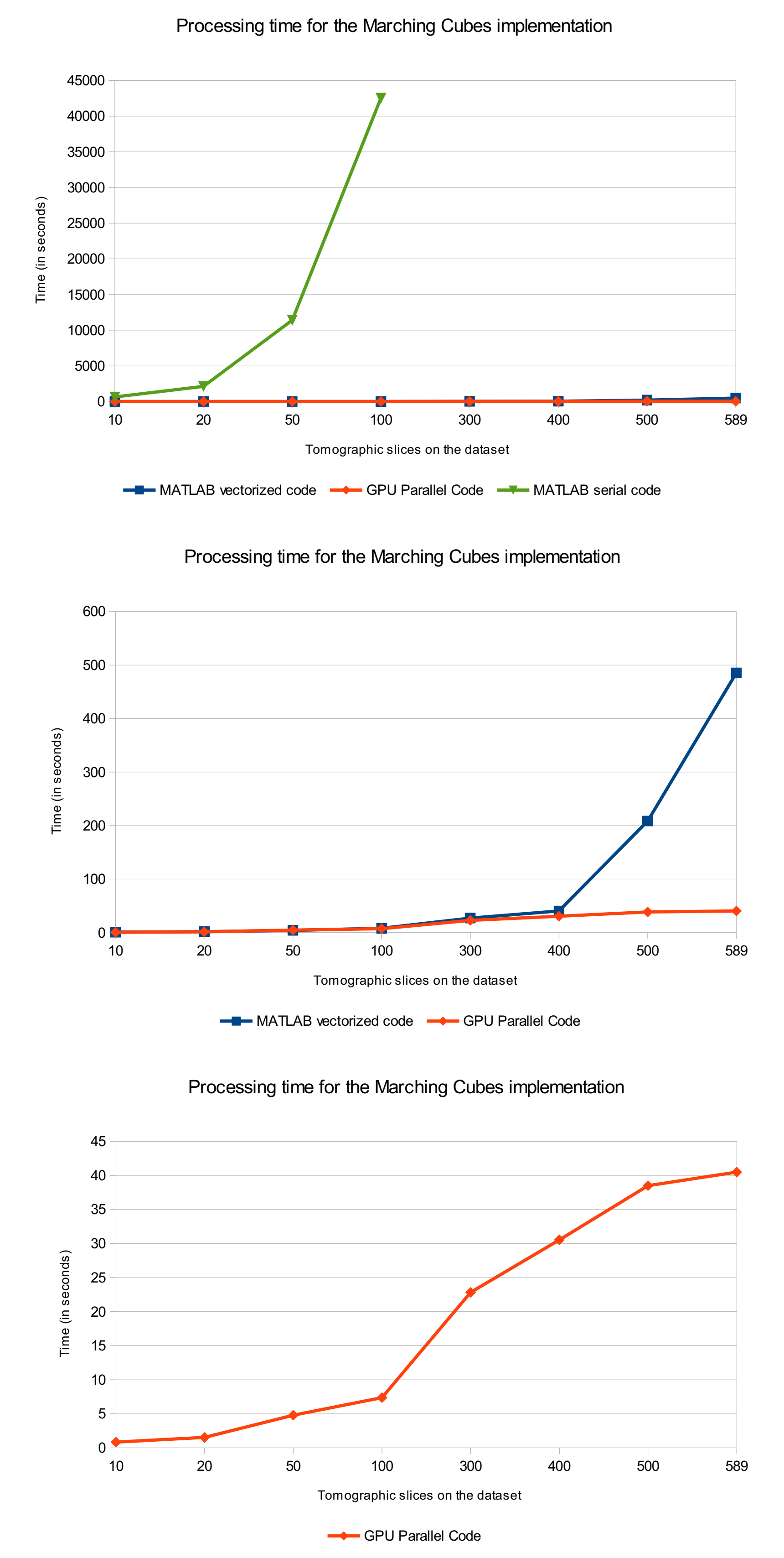


Figure 4: Virtual reconstructon of the concrete samples.

CONCLUSION

It is possible to notice how the GPU processing power can be used in order to decrease time in processing intense tasks, even though there are memory limitations that require some accessory code to handle large data sets, dividing and regrouping portions of the tomographic images space.

The memory transfers, however, must be carefully planned in order to not hinder the application execution, excessively increasing processing time.

REFERENCES

Cook, S., 2013. Cuda Programming a Developer's Guide to Parallel Computing with GPUs. Elsevier.

Suh, J. W., Kim, Y., 2014. Accelerating MATLAB with GPU Computing. Elsevier.

Lorenssen, W. E. & Cline, H. E., 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics, vol. 21, n.4, pp.163-169.

de Geus, K., Godoi, W. C., Junior, S. R., Swinka-Filho, V., 2012. 3D reconstruction of industrial computed tomography applied to concrete samples of power plant dams: CPU versus GPU- based processing. 6th International Symposium on Process Tomography, pp. 1-7, Cape Town, South Africa.

Farber, R., 2011. CUDA Application Design and Development. Elsevier.

Kirk, D. B. & Hwu, F. M-W., 2011. Programando para Processadores Paralelos. Campus.