Abstract ID-153

DISCOVERING TIME-CONSUMING SNIPPETS IN A MEDICAL IMAGE SEGMENTATION ALGORITHM

Carlos A.S. J. Gulo(1), *Antonio C. Sementille*(2), *João Manuel R. S. Tavares*(3)

(1)**CNPq & FEUP & INEGI, Portugal**
sander@unemat.br

(2)**Universidade Estadual Paulista, Brazil**
semente@fc.unesp.br

(3)**FEUP & INEGI, Portugal**
tavares@fe.up.pt

Keywords: Medical Image Processing, Profiling Tools, Computer Performance Analysis

Summary: Image segmentation is one of the most important operations performed on medical images: it is responsible for delineating structures on images. Briefly, Active Contour Models (ACM) detect the structures whose boundaries are not necessarily associated to high gradient values by minimizing an energy, which can be seen as a particular case of the minimal partition problem. However, the high computation cost required by this type of model, which is commonly used in medical image segmentation, demands optimization strategies in order to reduce their computational runtime. High-performance computing techniques have contributed effectively to reduce the required runtime of many medical image processing algorithms, making them suitable for real-time diagnosis, by fully exploiting all the computational power available in recent computers. This article discusses the use of profiling tools on measuring the computational time demanded by the Chan-Vese's image segmentation algorithm which is based on an ACM. Program profiling is commonly used to measure instruction set use, to evaluate and identify parts of the code that are responsible for excessive resource use. For measuring the performance of functions on the Chan-Vese's algorithm implementation, we focused on the profiling tools: gprof and perf. Gathering profile data was the first step performed, it was responsible for collecting data while monitoring hardware interrupts, operating system calls and performance counters. The collected data was analyzed to extract performance statistics and also to record the arc in the call graph responsible for activating each implemented function. The generated call graph represents time-consuming functions and the number of times the functions were invoked. From the call graph obtained for the Chan-Vese's algorithm, the profile showed that the function responsible for computing locally the signed distance function to its zero level set was the most called and the most time demanded, it required around 75% of the total running time. We have implemented an OpenMP-based parallel version of this costly function and, as result, the runtime was reduced by up to 4 times in comparison with the sequential version. Concluding, computational parallelization assisted by profiling tools increased the application performance and facilitates implementation efforts.