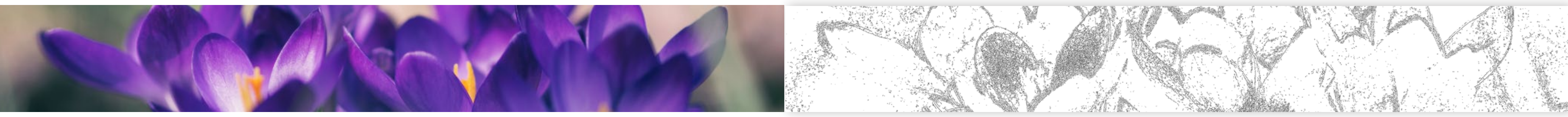
Image processing in lightning speed from smartphones to supercomputers: the story of Canny Edge Detection and Parallelware Analyzer

July 7th, 2021



Executive Summary

Canny image edge detection algorithm is used to detect edges in an image and it has a huge importance for feature extraction in machine learning. It has found application in many industries: optical character recognition, facial recognition, computer aided diagnosis, autonomous vehicles, space exploration, and many others. The performance of the algorithm is essential, since it is often used in environments where computational resources are scarce, such as mobile phones or autonomous vehicles, or where the input is huge, such as high-performance systems.

At Appentra, we are committed to helping developers deliver optimum performance in their software. Our Parallelware Analyzer is the first static code analyzer specializing in performance. It provides actionable insights through optimization reports that help ensure best practices to speedup the code through parallel computing in vector, multicore and accelerator processors.

By detecting optimization opportunities and offering recommendations, Paralleware Analyzer helped developers reduce the execution time of the Canny algorithm by 32 % on embedded and desktop systems and **50 %** on high-performance systems.

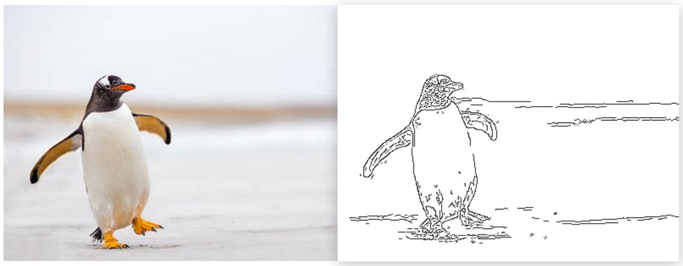
The Image Edge Detection challenge

Image edge detection includes a variety of mathematical methods that aim at identifying points in a digital image at which the image brightness changes sharply, which are typically organized into a set of curved line segments called edges.

Image edge detection algorithms are very important in a variety of domains: the output of edge detection is used as an input for other algorithms in areas such as image processing, computer vision, and machine vision.

The Canny edge detection algorithm is used widely because of its simplicity and good results. The performance of edge detection is important for several reasons. Among others, they are used in resource limited devices, often with real-time constraints. Also, they are used for machine learning where the number of images can be huge. Or in scientific settings where image size can be huge. We took an opportunity to address the challenge of high-performance image edge detection.

The Canny algorithm consists of four phases: gaussian filtering to reduce the noise, finding intensity gradient of the image in X and Y directions, suppression of non-maximal values and removal of weak edges using hysteresis thresholding. The output of the algorithm is an image with clear, thin edges and a very small amount of false edges.



In all the phases, the input image is processed pixel-by-pixel, either row-wise, column-wise or both. To achieve fast image processing, efficient memory access pattern, vectorization and parallelization are three crucial elements.

The implementation we at Appentra used as a starting point is available in the public domain.

Speeding up the Canny code using Parallelware Analyzer

The optimization process consists of finding the hot spots using a profiler and running Parallelware Analyzer to determine the optimization potential of the hot loops. Profiling showed that the gaussian smoothing phase is the hot spot. Parallelware Analyzer detected parallelization opportunities in the two loop nests that make the gaussian smoothing phase and proposed a combination of multithreading for the outermost loops and SIMD vectorization for the innermost loops to take advantage of all available hardware resources. The tool also suggested a command to automatically rewrite the two loop nests in the gaussian smoothing phase. By following that suggestion, Parallelware Analyzer automatically created a faster version of the code using OpenMP directives.

Benchmarking Canny: towards performance portability

We tested the performance impact of changes done by Parallelware Analyzer on several different systems: embedded (ARM), desktop (AMD and Intel) and high-performance computing systems (ARM, Intel and IBM). The image we used as the algorithm input had a resolution of 15,360 x 8.640 pixels. All programs were compiled with CLANG using '-O3 -ffast-math' optimization options which correspond to high optimization levels typically used in production software.

	Original	Optimized	Improvement
Desktop systems			
AMD Ryzen 7 4800H	8.6 s	5.3 s (8 threads)	**38.3 %**
Intel i5-10210U	15.8 s	11.0 s (8 threads)	**29.7 %**
Embedded systems			
ARM Cortex-A72	60.6 s	41.3 s (4 threads)	**31.8 %**
High-Performance systems			
ARM Fujitsu A64fx (Ookami)	79.5 s	37.5 s (16 threads)	**52.8 %**
ARM Hisilicon Hi1616 (Juawei)	27.7 s	13.4 s (16 threads)	**51.6 %**
IBM POWER9 (8-core SMT4)	9.9 s	5.0 s (16 threads)	**49.5 %**
Intel Xeon Gold 5218 (EPEEC)	10.3 s	6.0 s (16 threads)	**41.7 %**

The reduction of execution time was 31.8 % on an embedded system, between 29.7 % and 38.3 % on desktop systems and between 41.7 % and 52.8 % on high-performance systems.

Thanks to the Jülich Supercomputing Centre, the Open Edge and HPC Initiative, Stony Brook University and the EPEEC project (Horizon 2020 programme, GA No 801051) for providing access to their systems for benchmarking.

appentra
code performance

Appentra is a Deep Tech company that delivers the Parallelware Analyzer software, the first static code analyzer specializing in performance.

www.appentra.com | info@appentra.com | +34 881015556

Unión Europea
Fondo Europeo de Desarrollo Regional
"Una manera de hacer Europa"

FINANCIADA POR ENISA
enisa
MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO