Contextual Spaces Algorithm Acceleration on APUs

Flávia Pisani[1], **Daniel C. G. Pedronette**[2], **Ricardo da S. Torres**[1], **Edson Borin**[1]

[1]Institute of Computing (IC) – University of Campinas (UNICAMP)
Campinas – SP – Brazil

[2]Institute of Geosciences and Exact Sciences (IGCE) – São Paulo State University (UNESP)
Rio Claro – SP – Brazil

flavia.pisani@students.ic.unicamp.br,

{rtorres,edson}@ic.unicamp.br, daniel@rc.unesp.br

Abstract. *Re-ranking algorithms have been proposed to improve the effectiveness of content-based image retrieval systems by exploiting contextual information encoded in distance measures and ranked lists. This paper describes our efforts to improve the efficiency of these algorithms. We introduce novel implementations for the recently proposed Contextual Spaces re-ranking algorithm that use parallel architectures based on Accelerated Processing Units. Performed experiments demonstrate that our solutions speed up the execution of this algorithm significantly.*

1. Introduction

As the cost of storage devices falls off and new technological advances are made in the field of data acquisition and sharing, larger image collections have been created. In this scenario, search systems are of great importance. The widespread retrieval approaches, such as the ones based on keywords and textual metadata, face serious challenges due to the inherent difficulty in describing an image in words [Datta et al. 2008].

Content-Based Image Retrieval (CBIR) is a technology that mitigates this problem by providing automatic mechanisms for searching based on image visual properties (e.g., color or texture). Given a query image, a CBIR system intends to retrieve similar images in a collection by using one or more content descriptors, which encode image visual properties like shape, color, and texture. A CBIR system ranks collection images by decreasing order of similarity, and since users consider mostly top-ranked results, it is imperative that the rank is as accurate as possible.

In recent years, several successful attempts to increase the effectiveness (quality of results) of CBIR systems have been made [Pedronette and da S. Torres 2011, Yang and Latecki 2011, Datta et al. 2008]. In special, re-ranking methods have been used to improve the effectiveness of CBIR systems by exploiting contextual information encoded in distance scores and ranked lists. These methods are, on the other hand, very costly as they are based on comparing collection images multiple times.

Central Processing Units (CPU) no longer have just one core and Graphics Processing Units (GPU) are now being used as general purpose processors (GPGPU) due to the fact that they have evolved into massive parallel architectures capable of executing hundreds of operations per cycle [Pedronette et al. 2012], so alternatives to increase

performance through parallelization seem like a possible fit for the situation. Good speedups have been achieved using parallelization techniques [Pedronette et al. 2012, Teodoro et al. 2011], leading us to work on the parallel implementation of the Contextual Spaces re-ranking algorithm, which showed to be very effective but lacked in performance [Pedronette and da S. Torres 2011].

2. Contextual Spaces Image Re-Ranking Algorithm

The *Contextual Spaces re-ranking* algorithm [Pedronette and da S. Torres 2011] aims to redefine the relationships between images in a collection by answering the question "*What information can my nearest neighbors give about other collection images?*". In this manner, not only is the pairwise correlation between images analyzed, but also the information contained in the context of the query. A *contextual space* is a bidimensional space in which all collection images are represented according to their distances to an arbitrary image and each one of its K-nearest neighbors.

The algorithm is divided into two main steps. The first one is the *Distance Computation* step, which constructs for each of the collection images a contextual space in which the dimensions are the distance from the image to all collection images and the weighted average of the distances from each of its K-nearest neighbors to all collection images. The closer an image is to the origin of the contextual space, the more similar it is to the image being analyzed and therefore it can be moved towards the top of the ranked list. This change is performed by the second step, *Sorting Ranked Lists*, which rearranges the order of the images in the ranked list to make it correspond to the new distances calculated in the previous step. The process is then executed several times throughout a certain number of iterations with increasing K.

3. APU Acceleration of Contextual Spaces Algorithm

3.1. OpenCL

OpenCL is a new industry standard for task-parallel and data-parallel heterogeneous computing on a variety of modern CPUs, GPUs, and other processors [Pedronette et al. 2012]. An OpenCL program runs on computational *devices* such as CPUs and GPUs. Devices like GPUs usually have one or more *compute units* (processor cores) consisting of one or more single-instruction multiple-data (SIMD) *processing elements* (PE) that execute instructions in lockstep. The program is divided into *kernels*, which are dynamically compiled OpenCL functions. A kernel's execution on a device is scheduled by a C runtime library. Each SIMD kernel instance is called a *work-item* and executes on a single PE [Inc 2012].

3.2. Parallel Contextual Spaces Algorithm

Both steps of the Contextual Spaces algorithm described in Section 2 have inherent potential for parallelization. The *Distance Computation* step can be performed concurrently for each pair of images since all computations are independent, and the *Sorting Ranked Lists* step can be executed simultaneously for each ranked list, as there are no dependencies among them. To guarantee that the synchronization requirement between steps is met, an OpenCL kernel was created for each of them:

- **Distance Computation:** updates the distance between a pair of images considering the contextual space formed by the query image's K-nearest neighbors.
- **Sorting Ranked Lists:** re-orders the ranked list of each image given the new distances calculated. The more similar two images are, the smaller is the distance between them, making more relevant images closer to the top of the list.

Serial code was added to provide the necessary data input for the kernels, creating the design depicted in Figure 1.

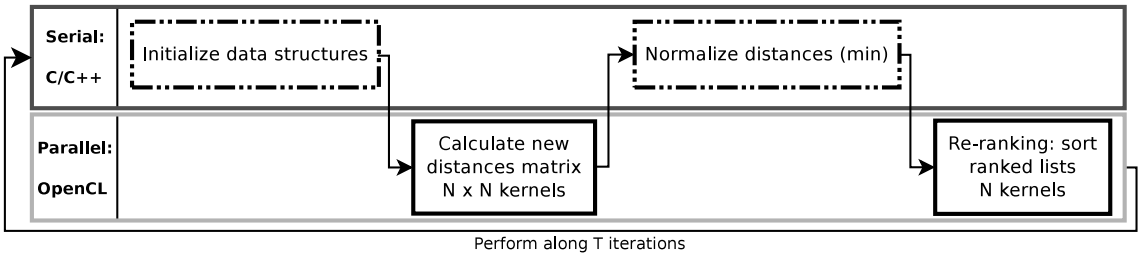

Figure 1. Parallel implementation of the Contextual Spaces re-ranking algorithm

4. Experimental Evaluation

The APU used in our experiments is the AMD A8-3850, which combines 4 CPU cores with an AMD Radeon HD 6550D GPU. We ran the tests on a Linux 3.3.4-5 Fedora 17 environment with AMD OpenCL SDK 2.8 using the MPEG-7 collection and the CFD shape descriptor [Pedronette and da S. Torres 2010].

The performance of our implementation of the Contextual Spaces re-ranking algorithm was measured throughout a series of experiments. We set up a serial implementation in C/C++ to use as a baseline for the comparison, and the results are shown in Table 1.

Table 1. Performance comparison for the Contextual Spaces re-ranking algorithm

Language/Device	*Measure*	**Distance Computation**	**Sorting Ranked Lists**	**Serial Code**
Serial C/C++	**Total Time (s)**	$\mathbf{1.5409 \pm 0.0002}$	3.7015 ± 0.0003	0.2936 ± 0.0001
OpenCL - CPU/CPU	***Execution Time (s)***	0.2285 ± 0.0019	0.2658 ± 0.0031	0.3066 ± 0.0005
	Mem. Transf. Time (s)	0.1432 ± 0.0011	0.1142 ± 0.0019	
	Total Time (s)	$\mathbf{0.3801 \pm 0.0026}$	0.3859 ± 0.0036	
OpenCL - GPU/CPU	***Execution Time (s)***	0.3400 ± 0.0004	0.2660 ± 0.0034	0.3226 ± 0.0032
	Mem. Transf. Time (s)	0.0924 ± 0.0015	0.1241 ± 0.0005	
	Total Time (s)	$\mathbf{0.4353 \pm 0.0015}$	0.3985 ± 0.0037	

Comparing total times for the *Distance Computation* step, we can see a speedup of $\sim 4.1\times$ from the serial implementation to the parallel implementation running on the CPU, and a speedup of $\sim 3.5\times$ for the parallel implementation running on the GPU.

We have also investigated optimization techniques that could help us explore the high-parallelization potential of the GPU. By observing that the work-items executing the *Distance Computation* step concurrently access the i-th position of each image's distance vector, we decided to transpose the distances matrix so that all i-th elements would be in the same line, as opposed to being in the same column. Using this approach, we were able to take better advantage of cached values through coalesced access. Table 2 shows the results.

Table 2. Performance comparison after adding coalesced access

Language/Device	*Measure*	**Distances Computing**	**Sorting Ranked Lists**	**Serial Code**
Serial C/C++	**Total Time (s)**	$\mathbf{1.7114 \pm 0.0003}$	4.2769 ± 0.0006	0.3043 ± 0.0010
OpenCL - CPU/CPU	***Execution Time (s)***	0.2379 ± 0.0021	0.2552 ± 0.0012	0.2884 ± 0.0009
	Mem. Transf. Time (s)	0.1294 ± 0.0032	0.0973 ± 0.0038	
	Total Time (s)	$\mathbf{0.3717 \pm 0.0023}$	0.3563 ± 0.0048	
OpenCL - GPU/CPU	***Execution Time (s)***	0.1340 ± 0.0002	0.2619 ± 0.0022	0.3200 ± 0.0027
	Mem. Transf. Time (s)	0.0891 ± 0.0016	0.1097 ± 0.0036	
	Total Time (s)	$\mathbf{0.2260 \pm 0.0017}$	0.3761 ± 0.0032	

We now see a speedup of $\sim 4.6\times$ from the serial implementation to the parallel implementation running on the CPU, and a speedup of $\sim 7.6\times$ for the parallel implementation running on the GPU, confirming that the change substantially improved our results.

5. Conclusions and Future Work

In this work, we used OpenCL to accelerate the Contextual Spaces re-ranking algorithm. We focused our analysis on the *Distance Computation* step, achieving an initial speedup of $\sim 4.1\times$ running it on a CPU and $\sim 3.5\times$ on a GPU. Further studies of optimization techniques led us to explore the use of coalesced access, improving our results to $\sim 4.6\times$ and $\sim 7.6\times$ on CPU and GPU, respectively. Future work includes the use of larger image collections to test the scalability of the approach, as well as the study of compromises between effectiveness and efficiency.

Acknowledgements

The authors thank AMD, FAPESP, CAPES, and CNPq (grants 306580/2012-8, 484254/2012-0) for their financial support.

References

Datta, R., Joshi, D., Li, J., and Wang, J. Z. (2008). Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):5:1–5:60.

Inc, A. M. D. (2012). Programming guide - AMD Accelerated Parallel Processing OpenCL™.

Pedronette, D. C. G. and da S. Torres, R. (2010). Shape retrieval using contour features and distance optimization. VISAPP, pages 197–202.

Pedronette, D. C. G. and da S. Torres, R. (2011). Exploiting contextual spaces for image re-ranking and rank aggregation. ICMR, pages 13:1–13:8.

Pedronette, D. C. G., da S. Torres, R., Borin, E., and Breternitz, M. (2012). Efficient image re-ranking computation on GPUs. ISPA, pages 95–102.

Teodoro, G., Valle, E., Mariano, N., da S. Torres, R., and Meira Jr., W. (2011). Adaptive parallel approximate similarity search for responsive multimedia retrieval. CIKM, pages 495–504.

Yang, X. and Latecki, L. J. (2011). Affinity learning on a tensor product graph with applications to shape and image retrieval. CVPR, pages 2369–2376.