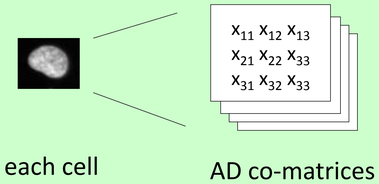
Haralick's Texture Features Computations Accelerated by GPUs in Biological Applications

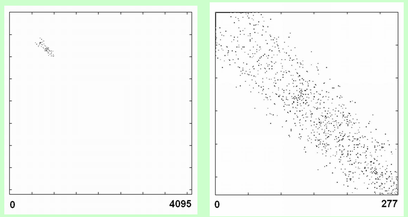
M. Gipp[2], G. Marcus[2], N. Harder[1], A. Suratanee[1], K. Rohr[1], R. König[1], R. Männer[2]
[1]Dept. Bioinformatics and Functional Genomics, BIOQUANT, IPMB, and DKFZ, University of Heidelberg
[2]Institute for Computer Engineering, ZiTi, University of Heidelberg
markus.gipp@ziti.uni-heidelberg.de

Abstract
This poster presents the speedup of the computation of co-occurrence matrices (co-matrices) and Haralick Texture Features (features), as used for analyzing microscope images of biological cells, by general-purpose graphics processing units (GPUs).
In a pipeline of automated image analysis algorithms the feature computation is the most costly computing part. The computing time of the algorithm without acceleration amounts to several months. Hence, a massive speedup is required.
Analyzing the features results in a graph showing the dependency of the feature computations on intermediate results and on other features. With the dependency graph the optimal order of the feature computation could be determined which saved costly double computations.
Analysation of co-matrices showed that they are sparsely filled, and for a highly parallel approach they consume too much memory. We reduced the size of a full co-matrix by removing all rows and columns filled with zeros. This reduction strategy allowed us to keep up to two hundred co-matrices in the memory of an ordinary graphics card with direct memory access.
For each single cell image 20 co-matrices with different orientations are generated. Altogether, the features of 8 cells can be computed in parallel, requiring the calculation of 160 co-matrices. To reduce the complexity of the feature computation 24 kernel functions are used on the GPU and each one maps all co-matrices to the parallel computing architecture of the GPUs.
On a single node of a cluster, a speedup of 500 was obtained compared to an unoptimized software version, and a speedup of 46 was obtained compared to an optimized software version.
The GPU implementation reduces the computational time from half a year to around 9 hours, which opens up a new research application in the field of biological image analysis.

Parallel Feature computation on many Co-Matrices Each Co-Matrix is associated to one CUDA Block.



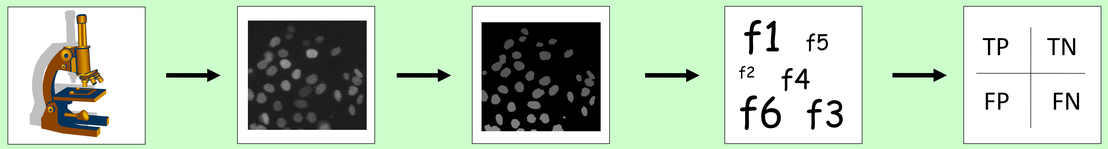
Sparse filled Co-Matrices
Left a full Co-Matrix (64 MBytes) and right a packed Co-Matrix (75 kBytes)



List of parallel computing kernel functions executing on the GPU

Initialisation part	
Function 0A	generate index / gray level lookup tables
Function 0B	clear co-occurrence matrices
Function 0C	compute co-occurrence matrices
Function 0D	normalize co-occurrence matrices
Part 1, read from co-occurrence matrices	
Function 1A	compute f1
Function 1B	compute f5
Function 1C	compute f6
Function 1D	compute P
Function 1E	compute P|x-y|
Function 1F	compute Px+y
Part 2, read from P	
Function 2A	compute mean
Function 2B	compute var
Function 2C	compute H
Part 3, read from P|x-y|	
Function 3A	compute f2
Function 3B	compute f11
Function 3C	compute MacP|x-y|
Function 3D	compute f10
Part 4, read from Px+y	
Function 4A	compute f6
Function 4B	compute f8
Function 4C	compute f7
Part 5, read from co-occurrence matrix	
Function 5A	compute Pij and f3
Function 5B	compute f4
Function 5C	compute HXY1, f12, read from P
Function 5D	compute HXY2, f13, read from P only

Pipeline of automated image analysis.
(Microscope, multi cell image, segment image, feature values, classification)



Pseudo-code of the parallel GPU program structure

C=8, AD=20 -> 160 CUDA blocks

```
loop over all cells, C cells in parallel
    generate matrices, C*AD in parallel
    compute intermediate results and features
    for all matrices C*AD in parallel
    store feature values
    clean matrices
```

Haralick Texture Features (Features) we used and simplified for symmetrical Co-Matrices

$$f_1 = \sum_{i=1}^{Ng}\sum_{j=1}^{Ng} P_{(i,j)}{}^2 \qquad f_7 = \sum_{k=0}^{2Ng-2}(k-f_6)^2\, P_{x+y}(k)$$

$$f_2 = \sum_{k=0}^{Ng-1} k^2 \left(\sum_{i=1}^{Ng}\sum_{j=1}^{Ng} P_{(i,j)} \right)_{|i-j|=k} \qquad f_8 = -\sum_{k=0}^{2Ng-2} P_{x+y}(k)\, \log[P_{x+y}(k)]$$

$$f_9 = -\sum_{i=1}^{Ng}\sum_{j=1}^{Ng} P_{(i,j)}\, \log[P_{(i,j)}]$$

$$f_3 = \frac{1}{\sigma^2}\sum_{i=1}^{Ng}\sum_{j=1}^{Ng}(ij)\, P_{(i,j)} - \mu^2$$

$$f_{10} = \sum_{k=0}^{Ng-1}\left[P_{|x-y|}(k)\,(k - \sum_{l=0}^{Ng-1} l\, P_{|x-y|}(k))^2 \right]$$

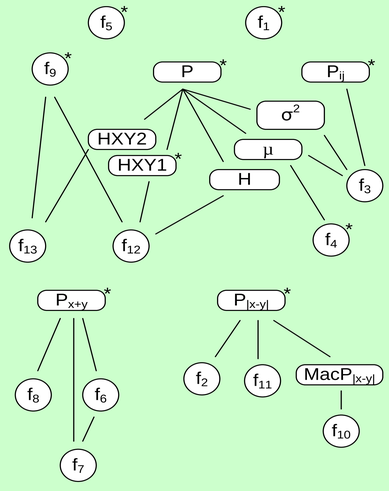
$$f_4 = \sum_{i=1}^{Ng}\sum_{j=1}^{Ng}(i-\mu)^2\, P_{(i,j)}$$

$$f_{11} = -\sum_{k=0}^{Ng-1} P_{|x-y|}(k)\, \log[P_{|x-y|}(k)]$$

$$f_5 = \sum_{i=1}^{Ng}\sum_{j=1}^{Ng}\frac{P_{(i,j)}}{1+(i-j)^2} \qquad f_{12} = \frac{f_9 - HXY1}{H}$$

$$f_6 = \sum_{k=0}^{2Ng-2} k\, P_{x+y}(k) \qquad f_{13} = \sqrt{1-\exp[-2\,|HXY2 - f_9|]}$$

Dependency graph of the Haralick Texture Features used to derive the optimal order of computation



Results of various implementations of the Haralick Texture Feature algorithm with seed up factors

	Computing time (s)	Factor to 1.	Factor to 2.	Factor to 3.
1. Original feature algorithm	2378	1x	-	-
2. Software optimized version	214	11x	1x	-
3. GPU version I (8800 GTX)	11.1	214x	19x	1x
4. GPU version I (GTX 280)	6.6	360x	32x	1.7x
5. GPU version II (GTX 280)	4.65	**511x**	46x	2.4x

Conclusions

The GPU implementation reduced the computational time from half a year to around 9 hours of an un-optimized software and to 4 days of an optimized software version.
Our latest optimizations could further increase the computational speed by a factor of 1.4 compared on the same graphics card. This increase in performance was the result of solving memory bandwidth limiting bottle necks discovered with the profiler tool. We conclude that avoiding the use of shared memory and barrier synchronization in small and simple kernel functions gives better results on the latest architecture.
The speedup of the GPU versions scales with the memory bandwidth.

References

(1) N. Harder, B. Neumann, M. Held, U. Liebel, H. Erfle, J. Ellenberg, R. Eils, and K. Rohr, ''Automated recognition of mitotic patterns in fluorescence microscopy images of human cells'', Proc. IEEE Internat. Symposium on Biomedical Imaging: From Nano to Macro (ISBI'06), Arlington/VA, USA, April 6-9, 2006, 1016-1019
(2) C. Conrad, H. Erfle, P. Warnat, N. Daigle, T. Lörch, J. Ellenberg, R. Pepperkok, and R. Eils, "Automatic identification of subcellular phenotypes on human cell arrays," Genome Research, vol. 14, pp. 130-1136, 2004.
(3) R. M. Haralick and K. Shanmugam, "Computer Classification of Reservoir Sandstones," IEEE Transactions on Geoscience Electronics, vol. 11, pp. 171-177, 1973
(4) R. M. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, vol. 67, pp. 786-804, 1979.
(5) S. Theodoridis and K. Koutroumbas, Pattern Recognition Third Edition. San Diego, CA, USA: Academic Press An imprint of Elsevier, 2006.
(6) Gipp, M., G. Marcus, N. Harder, A. Suratanee, K. Rohr, R. König, and R. Männer. 2009. Haralick's Texture Features Computed by GPUs for Biological Applications. IAENG International Journal of Computer Science, Volume 36 Issue 1. Newswood Limited, International Association of Engineers, London, 17 February.
(7) NVIDIA CUDA Programming Guid Version 2.1