Original Report

Dino Di Carlo, Ph.D.
University of California
Los Angeles, CA

Strategies for Implementing Hardware-Assisted High-Throughput Cellular Image Analysis

*Henry Tat Kwong Tse,[1,2] Pingfan Meng,[3] Daniel R. Gossett,[1,2] Ali Irturk,[3] Ryan Kastner,[3] and Dino Di Carlo[1,2]**

[1]*Department of Bioengineering, University of California Los Angeles, Los Angeles, CA*
[2]*California NanoSystems Institute, Los Angeles, CA*
[3]*Department of Computer Engineering, University of California San Diego, La Jolla, CA*

Keywords:
high-throughput image analysis, high-speed imaging, GPU image processing, cellular morphology

Recent advances in imaging technology for biomedicine, including high-speed microscopy, automated microscopy, and imaging flow cytometry are poised to have a large impact on clinical diagnostics, drug discovery, and biological research. Enhanced acquisition speed, resolution, and automation of sample handling are enabling researchers to probe biological phenomena at an increasing rate and achieve intuitive image-based results. However, the rich image sets produced by these tools are massive, possessing potentially millions of frames with tremendous depth and complexity. As a result, the tools introduce immense computational requirements, and, more importantly, the fact that image analysis operates at a much lower speed than image acquisition limits its ability to play a role in critical tasks in biomedicine such as real-time decision making. In this work, we present our strategy for high-throughput image analysis on a graphical processing unit platform. We scrutinized our original algorithm for detecting, tracking, and analyzing cell morphology in high-speed images and identified inefficiencies in image filtering and potential shortcut routines in the morphological analysis stage. Using our "grid method" for image enhancements resulted in an 8.54× reduction in total run time, whereas origin centering allowed using a look up table for coordinate transformation, which reduced the total run time by 55.64×. Optimization of parallelization and implementation of specialized image processing hardware will ultimately enable real-time analysis of high-throughput image streams and bring wider adoption of assays based on new imaging technologies. (JALA 2011;■:■–■)

*Correspondence: Dino Di Carlo, Ph.D., Department of Bioengineering, University of California Los Angeles, California NanoSystems Institute, 420 Westwood Plaza, 5121E Engineering V, Los Angeles, CA 90095; Phone: +1.310.983.3235; E-mail: dicarlo@seas.ucla.edu

2211-0682/$36.00

doi:10.1016/j.jala.2011.08.001

Introduction

Imaging is ubiquitous in industrial processing, medicine, environmental science, and cell biology. Given the diverse modes of imaging that exist, an image can contain a wealth of information about an object. Process quality control in semiconductor manufacturing and particle synthesis uses a number of spatial metrics from images from scanning electron microscopy, transmission electron microscopy, atomic force microscopy, and optical microscopy.[1,2] Imaging tools including positron emission tomography, X-ray, magnetic resonance imaging, and computed tomography are widely used in medicine for diagnostic and prognostic purposes. Ocean and waterway monitoring, a critical charge of environmental science, can be performed with high-speed

camera-coupled flow cytometry whereby the diversity and density of microscopic organisms, key indicators of ecosystem health, can be identified.[3,4] In cell biology, for example, cell size, morphology, and location can be extracted from bright-field or phase-contrast images. And, the presence or location of biomolecules within cells can be obtained from fluorescence images of chemically labeled cells, which has recently been implemented with automated fluid handling and imagers for high-content analysis.[5] As technology improves, imaging rates and resolution increase and the cost of acquiring image sets decreases but this can burden the end user or associated analysis or sorting systems with large image backlogs. Ultimately, both extreme high-speed bright-field imaging and high-content analysis systems based on fluorescence imaging now have the propensity to generate truly massive image-based data sets and will require a method to accommodate the time requirements of the user or system (e.g., real-time results will be required for cell sorting in medicine and cell biology). Only if automated image analysis can extract useful information and operate at meaningful rates will emerging image-based technologies find utility.

There are numerous applications where images have several advantages over other types of signal outputs and are often the preferred method of analysis. Qualitatively, images are most effective in conveying certain types of messages. They may also confer some measurable advantages. For example, flow cytometry measures scattered light to assess cell size and granularity, but theses values are only relative.[6] Analysis of microscopic images, on the other hand, yields an exact value of size. Image analysis can also be used to distinguish between cells, debris, and clusters of cells where flow cytometry would yield erroneous results. Spectroscopic readings of biochemically labeled cells in microtiter plates lack the sensitivity to detect rare cell populations[7] and can vary greatly with cell seeding density. In contrast, high-throughput automated microscopy coupled with automated image analysis can be used to identify and measure properties of single cells with multiple spectra, high spatial and temporal resolution for measuring dynamic processes, and with high bit depth. This is a powerful tool for studying complex biological pathways or measuring heterogeneous response to stimuli. Further, new tools are being introduced to take images of cells in flow. High-speed CMOS cameras, with high frame rates, fast shutter speeds, and high sensitivities, have recently enabled novel studies of highly dynamic events such as bubble rupture and microscale phenomena in particle-laden flow.[8–11] Other recent advances in computing, optics, and electronics have enabled imaging flow cytometry. This technology has shared roots in flow cytometry and microscopy, and can be applied to problems that would traditionally require multiple pieces of equipment and users trained in both. The ImageStream system (Amnis Corporation, Seattle, WA) acquires multispectral/multimodal images at a rate of 1000 cells per second with sufficient resolution to extract features such as fluorescence intensity, morphology, and signal localization.[12] It will be a useful tool for complex problems such as detecting tumor cells in body fluids and studying hematopoietic cells.[12,13]

Cell morphology has been studied extensively and is a clinically useful biomarker for several diseases. In this Article, a high-throughput imaging cytometry system examines morphological features of cells. The dramatic morphological change that occurs following stimulation of granulocytes is one example where imaging cytometry will be useful (Fig. 1). The Amnis ImageStream system has been used to characterize morphological changes associated with the maturation processes of erythroid cells; traditional biomolecular labels were correlated with morphological changes in size, shape, texture, and nucleus to cytoplasm ratios.[13] Additionally, multispectral imaging flow cytometry has been used for the identification of morphological changes such as ruffle formations on HIV-1 infection T-cells.[14] In the clinic, cell blocks and smears are scrutinized for cells with shape and size characteristics that are indicative of infections and cancer. These examples illustrate the importance of image analysis of morphology. However, the current stage of morphological analysis relegates it to the role of cell classification, whereas biochemical biomarkers are used for critical applications in cell sorting. The ability to sort cells (requiring real-time measurement and analysis) would greatly enhance the utility of morphology, and this ability will only be obtained by improvements to image analysis algorithms and processes.

However, these new technologies introduce challenges, especially in image processing and informatics.[15] Images can be very complex signals, with blurring occurring if an object's motion is faster than the shutter speed of the camera, and nonuniform illumination in wide field-of-view (FOV) images. Filtering, measuring, tracking, and fitting data from, for example, 3D confocal images[16] to complex multivariable models requires lengthy computation. Further, imaging flow cytometry and high-speed microscopy can produce hundreds of thousands of images per second. The qualities of these images (multispectral, high spatial and temporal resolution, and high bit depth) make the files tremendously large[15] adding to computational requirements. In general, these technologies require offline image analysis. Gradually, increased computing power will increase image analysis speed and methods such as cloud computing image analysis[17] will help share computing power with users on demand. But, for these technologies to be immediately accessible a new strategy that enhances the efficiency of image analysis algorithms is required.

Here we present a high-throughput image analysis strategy for a bright-field flow cytometry application with an image acquisition rate of over 140,000 frames per second. Analysis of bright-field images is in some respects more difficult than fluorescence images because of the more complex variation in intensities that are observed, such that our described strategies should also be compatible with fluorescence-based or other image sets. We first examine bottlenecks in the original detection and analysis algorithm, then design an alternative, less computationally demanding filtering algorithm, put in place a new detection strategy, and implement the algorithm

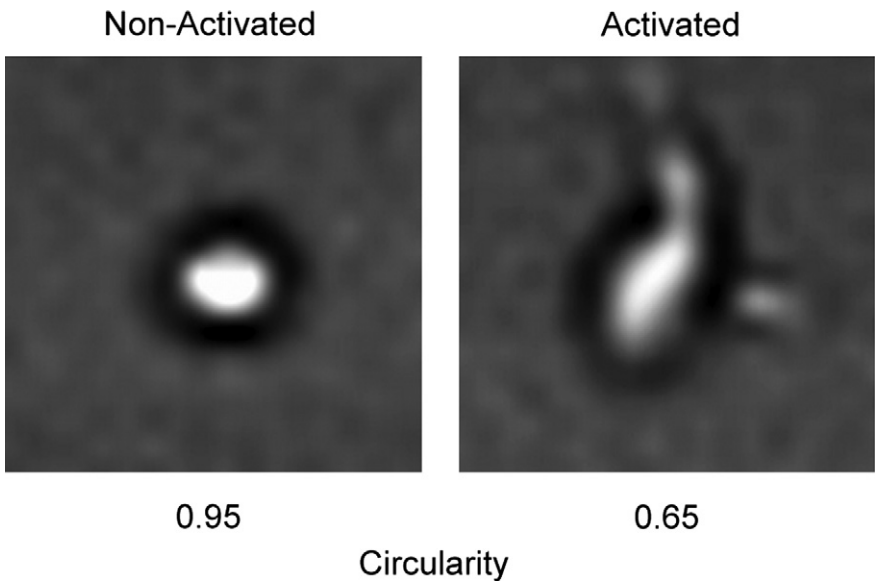

Figure 1. An example of cellular morphology differences from changes in cell states. High-speed microscopic images of a granulocyte (left) and a granulocyte activated by a N-formyl-methionine-leucine-phenylalanine (fMLP) chemical treatment (right) with associated measure of circularity as a characterization of the morphological differences of each cell.

on a graphical processing unit (GPU). Realization of real-time or near real-time analysis will enable systems like this one to be clinically relevant. The strategy that we describe here should be considered for future implementations of high-speed image analysis in hardware.

Algorithm description

Image Enhancement and Cell Detection Method

The automated analysis script was built in MATLAB v2009a (MathWorks, Natick, MA). The algorithm examines high-speed bright-field microscopic images (208×32 pixels $\times$ 142,857 frames/s) of cells in a high-throughput flow-through system. The algorithm (1) performs image adjustments, (2) *detects* the presence of an object, (3) *tracks* its motion, and (4) performs a measurement of *morphology*. The analysis is implemented frame by frame at a FOV of 23×20 pixels at an upstream location. The cell image signature is enhanced by bottom-hat filtering, which highlights intensity differences in a local area to allow for measurement extractions. When a cell is detected in the first FOV, the morphological analysis algorithm extracts morphological features. The same cell is then tracked downstream to examine changes in morphology. Analysis continues by moving the FOV downstream stepwise until the cell is within the FOV. Analysis continues until the cell is tracked out of the field where then the FOV resets to the upstream location for the detection of the next cell.

Morphological Analysis Routine

On detection of a cell in the FOV, the image is interpolated and resized $10\times$ (230×200 pixels) to enhance the accuracy of centroid analysis and the extraction of morphological features of interest: area, diameter, and topography. The centroid method examines the bottom-hat filtered image using a 70-pixel disk (the expected average cell size) to locate the centroid followed by a polar to Cartesian image transformation. The cell walls are highlighted by locating the local maximums in pixel intensity (Fig. 2). Final data outputs are radius measurements at intervals of 4° per cell at each frame.

Hardware acceleration

Recent work in hardware-assisted image analysis acceleration platforms have been implemented for various high-throughput applications.[18] In this section, first, we discuss acceleration platform modalities suited for high-throughput image analysis and their advantages and disadvantages. However, to fully use these hardware platforms for image analysis of cells or particles, the underlying algorithms for (1) image enhancement, (2) detection, (2) tracking, and (3) morphological analysis must be optimized for the specific platform. A systematic review of algorithm bottlenecks in the MATLAB-based CPU-run algorithms is first performed and algorithm components with the most intensive computing requirements are redesigned to optimize performance for GPU implementation. Converting CPU to GPU processing alone will not be sufficient to achieve the significant increase in throughput for real-time analysis, thus we have designed unique GPU-efficient image filtering and GPU-implemented transformation algorithms that will enable image signature-based high-throughput analysis for sorting of biological and biomedical applications. Lastly, we compare the performance enhancements obtained with the GPU over the CPU algorithm.

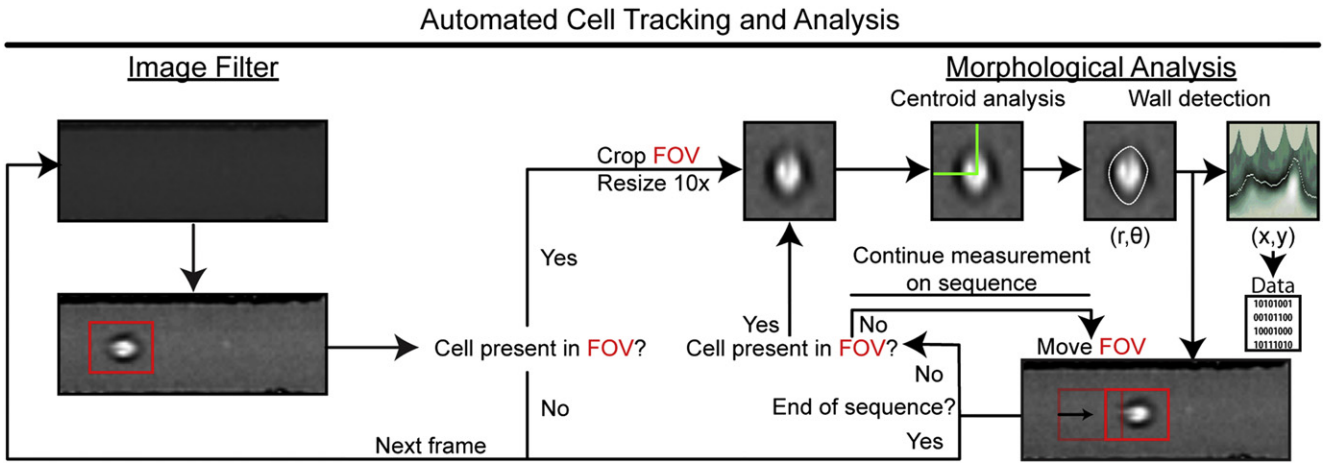

Figure 2. Automated cell tracking and analysis algorithm framework. The raw video frame is first contrast enhanced by the image filter algorithm, and the cell detection algorithm examines the field of view (FOV) (in red) for presence of a cell. When a cell is located in the FOV, the frame is cropped and resized by 10×. The morphological analysis algorithm starts by performing centroid analysis to determine origin of the polar to Cartesian coordinate transformation. The diameters are then extracted from the mapped image. The analysis is then iterated on the remaining cell image series until the FOV tracks out of frame.

Hardware-Assisted Image Analysis Acceleration Platforms

GPUs provide a high-performance, low-cost platform for hardware-assisted image analysis. GPUs use a task parallel architecture that is primarily targeted toward accelerating graphics applications. However, they are being used more widely across other domains including databases,[19] weather forecasting[20] and cryptography.[21] They are also broadly used in digital image processing and the medical imaging field.[22–24] GPUs use a pipeline designed for efficient independent processing of data, and multiple pipelines are used to exploit task level parallelism. GPUs outperform the CPUs by one to two orders of magnitude. They tend to perform well on highly parallel applications with limited number of memory accesses. Furthermore, they are ill-suited for applications with substantial amount of control flow, for example, branching and looping. GPUs are programmed using custom languages and application programming interfaces (APIs); for example, the compute unified device architecture (CUDA) is an extension of the C programming language developed by NVIDIA for programming their GPUs. This and other GPU programming languages work using a stream model of computation where the application is divided into a set of parallel threads, which explicitly defines parallelism and communication.

Field-programmable gate arrays (FPGAs) are another option for hardware acceleration, which is also used in a wide variety of applications. FPGAs use fine-grained programmable logic elements that implement basic Boolean logic functions (e.g., AND, OR, XOR). This creates the opportunity for substantial customization leading to significant performance increases over both CPUs and GPUs. As a concrete example, an FPGA implementation of the Viola-Jones object detection algorithm is about four times faster and uses an order of magnitude less power than the same algorithm running on a GPU.[25] The high degree of customization afforded by FPGAs is both a blessing and a curse. It provides great flexibility in terms of the design of an application, yet it substantially increases the programming complexity. FPGA design tools require a significant amount of hardware design expertize. These design environments are a far cry from those used to program microcontrollers, microprocessors, digital signal processors, and even GPUs. Therefore, the fundamental tradeoff when using an FPGA boils down to the need for high performance versus the ease of development.

In summary, FPGAs are typically the best option to create a high-performance hardware accelerator. However, GPUs tend to be easier to program and more readily allow for

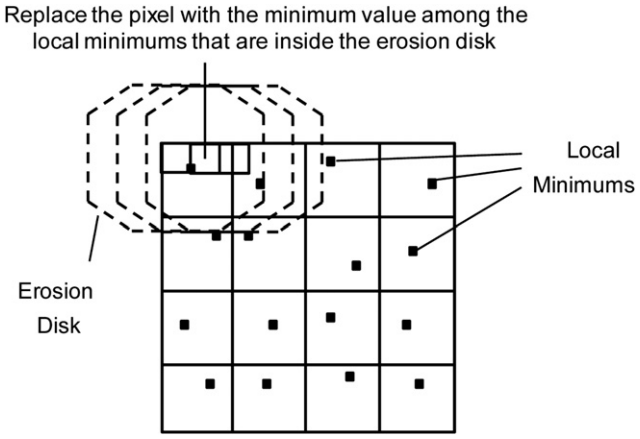

Figure 3. The grid method filter algorithm. The image is divided into several grids and each grid is analyzed for the local minimums or maximums (dots). For each pixel of the erosion or dilated image, the pixel value is determined by searching the grid for local minimums and maximums, respectively, within the disk mask.

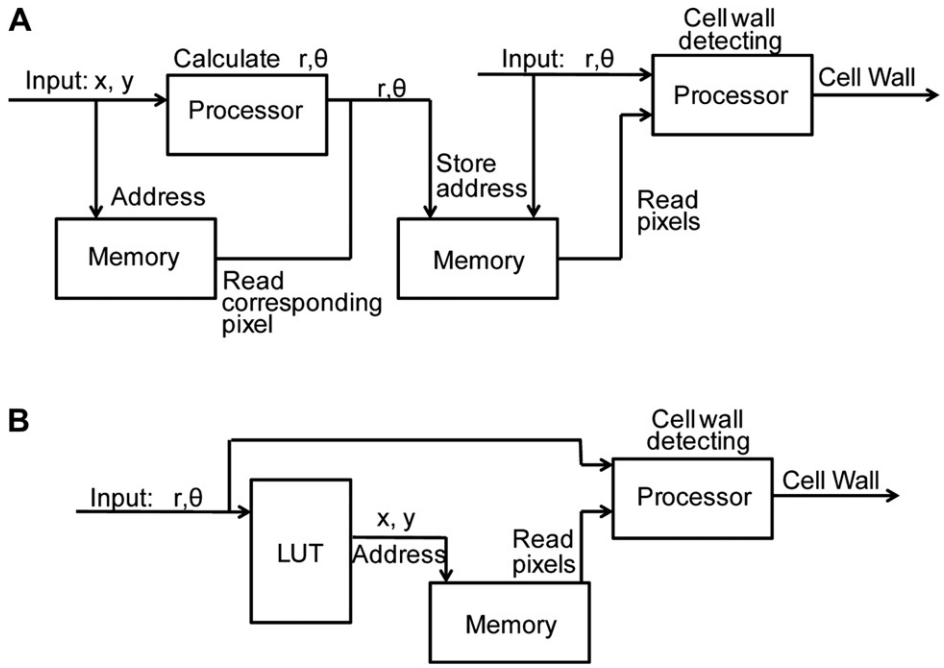

Figure 4. (A) Original morphological analysis algorithm. (B) Modified morphological analysis algorithm, where a LUT lowers computation demands for the (r, θ) → (x, y) transformation. Additionally, memory access is reduced as the polar coordinate image is not used.

experimentation in parallelized algorithm design. Our initial analysis presented in the remainder of this article was done on GPUs because of reduced development complexity. The transition to FPGA implementation will build on the work here emphasizing the design of novel efficient parallelization routines.

Bottleneck Analysis

To identify components of the algorithm that were most computationally intensive, we used MATLAB v2009a and its profiler to obtain the run times of three parts of the algorithm: image filter for image enhancement, morphological analysis, and other minor operations (detection and tracking). The run time associated for each part breaks down to 33.02%, 59.55%, and 7.42%, respectively. These results show that the image filter and morphological analysis constitute most of the computing demands. Therefore, we focused on accelerating these two parts of the algorithms on the GPU platform.

These two computationally intensive algorithms are naturally suitable for GPU parallelization because they call on

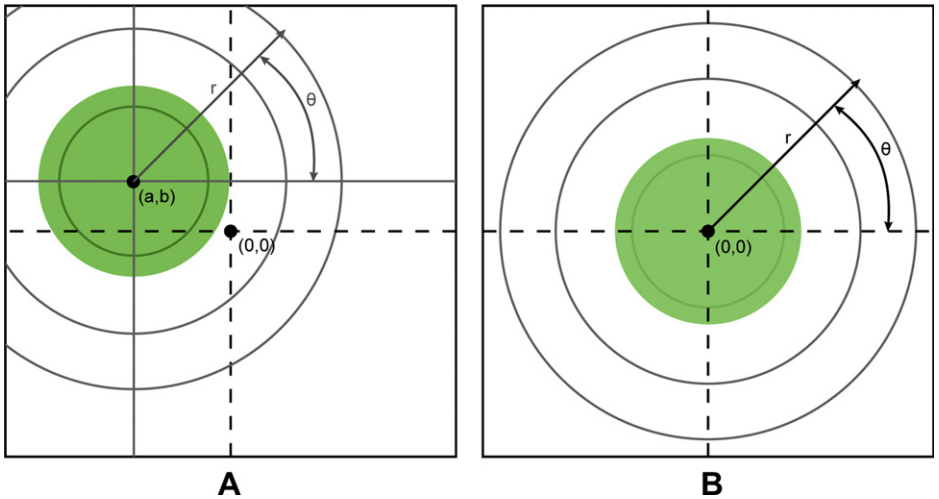

Figure 5. Recentering field of view (FOV) before coordinate transformation. (A) In the original algorithm, the cell center is not at the coordinate origin, thus requiring computing time to map each unique coordinate for the polar to Cartesian transformation from the (a, b) origin. (B) When a recropping event for a new FOV based on the centroid location, this enables the use of the (r, θ) → (x, y) transformation look up tables per mapping event.

high repetition of the same operations. For both image filter and morphological analysis, the algorithm can be divided into many identical operations (e.g., (1) local search operations that find minimum values; (2) repetition of simple transformation algebraic expressions). In a serial CPU program, these operations execute sequentially on the pixels. Because these operations execute the same computation on similar data types and without data dependency (e.g., one operation does not affect the others in a sequence), the image filter and morphological analysis algorithms can be implemented with GPUs very well and significant speedups can be expected.

Algorithm Modification

The architecture of GPUs and CPUs are significantly different in terms of thread scheduling[26] and on-chip memory.[27] If the CUDA program has inefficient thread scheduling or memory access, GPU platforms will be no better than CPUs. This fact makes it necessary to modify algorithms for efficiency and minimal memory access before implementing it with GPUs.

In the image filter algorithm, the computing usage is dominated by image adjustment techniques. The main filtering algorithm is the bottom-hat filter, which uses dilation and erosion operations. These two operations use a mask to search either for maximum or minimum values, respectively, in a local area of the image. The pixel corresponding to the center of the mask is replaced by the maximum or minimum value that is found in the local area.[28] In the algorithm, the mask used is a disk area with an area of 16,357 pixels. This large size is required to ensure higher fidelity in the centroid analysis process. In the dilation and erosion operations, updating every pixel requires a maximum or minimum value search among all 16,357 elements at each pixel of the FOV. Resulting in over 46,000 calls per image (FOV: 230×200 pixels).

In the conventional dilation and erosion operations, each mask area shares an overlapping region with the next mask. To minimize researching of these sections, we developed the "grid method," which organizes the image on to a grid system then reuses the minimum or maximum search result of this common area by recording minima and maxima of each grid (Fig. 3). The pixels on the eroded or dilated image are then replaced by the corresponding value among the local minima or maxima in each grid. Further adjustments are possible by tuning the grid size to manage computing load for balancing resolution and efficiency. For a grid size of 40×40, the minimum or maximum search operation for each image will be 16 elements instead of 16,357 elements.

In the morphological analysis algorithm, the morphological features of the cell (e.g., diameters) are extracted to characterize the shape of the cell. The main operation of this algorithm is a centroid analysis of the cell to establish the origin for a polar to Cartesian coordinate transformation. The mapped image is then examined by a threshold analysis to outline cell walls and output the diameter at intervals of θ. In this original state, the algorithm is not suitable for GPU implementation for two reasons: (1) the centroid defined at each iteration is not located at the coordinate origin, thus creating scheduling inefficiencies for each GPU thread; and (2) the polar and Cartesian coordinate systems have different memory storage patterns, which increase the memory access latency thus limiting run-time speedups (Fig. 4A).

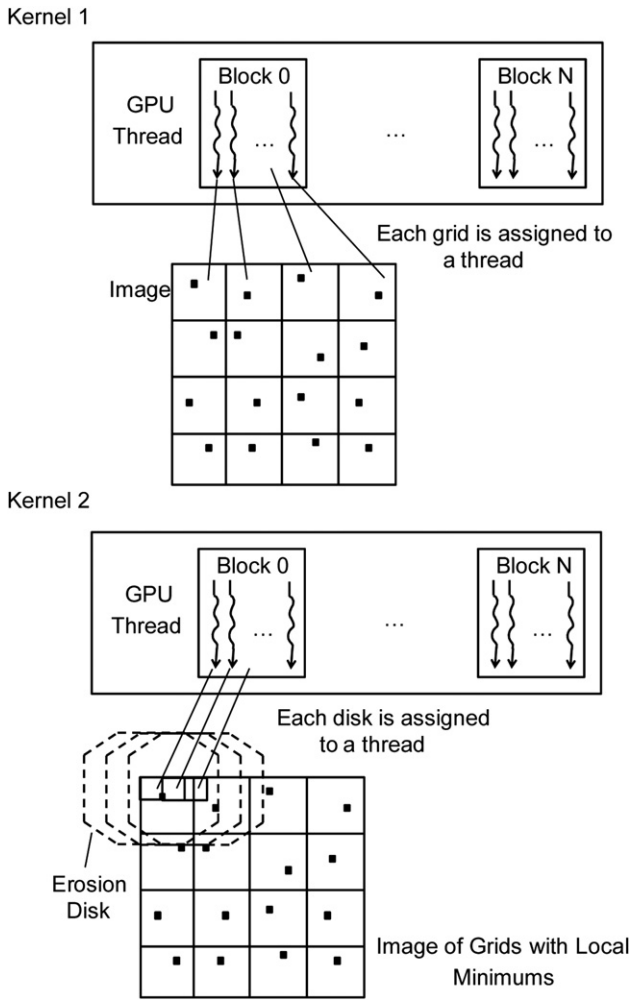

Figure 6. Graphical processing unit (GPU) programming model for grid erosion algorithm. In kernel 1, the local minimum (dots) search operations are executed in parallel by assigning each grid to a thread. In kernel 2, the eroded pixel value is found in parallel by assigning each disk to a thread.

To address these parallelization limitations, we made adjustments to both the centroid algorithm and data storage and access strategies. First, on finding the centroid, the image is adjusted such that the centroid coordinate is also the origin of the polar coordinate (Fig. 5). Because the transformation from $(r, \theta) \rightarrow (x, y)$ are the same for all frames, the computation load for each GPU thread is predictable making it possible to assign threads efficiently in the GPU programming model. We next changed the memory storage and access strategy (Fig. 4B). Instead of searching along all possible (r, θ) values,

the new method accesses a $(r, \theta) \rightarrow (x, y)$ look-up-table (LUT) memory to find a corresponding pixel in the polar coordinate instead of algebraically converting the entire image into a Cartesian coordinate system. This strategy will result in quicker access times for mapping $(r, \theta) \rightarrow (x, y)$.

GPU Programming Model

The GPU program for the grid method bottom-hat filter consists of two modules: (1) erosion and (2) dilation. These two modules are very similar to each other erosion being a search for minimum values and dilation being a search for maximum values. Otherwise, the GPU programming models are the same for both modules. Here, we describe the GPU programming model for erosion.

The grid erosion algorithm consists of two phases: (1) finding a local minimum for each grid and (2) replacing the pixel on the erosion image with the minimum value among the local minimums of all grids inside or intersecting with the disk mask. The GPU grid erosion program model can parallelize these two phases using two kernels (Fig. 6). In the first kernel, we assigned a thread to each grid. The thread executes the local minimum value searching for the correspondent grid. In the second kernel, we assigned a thread to each pixel that is to be updated. The thread searches for the minimum value among the local minimums found in the first phase.

The morphological analysis GPU programming model uses a parallel threading scheme. Each thread is scheduled to operate at a defined θ, for the $(r, \theta) \rightarrow (x, y)$ transformation. Further parallelization can be achieved by a reduction of GPU thread arrangement. The reduction method is not implementable for this version of the program, but in future revisions this process will aid in further decreasing run times. At this state, we have kept the minimum gray-scale pixel searching as a serial process within each GPU thread (Fig. 4B).

Results

Image filter algorithm optimization using the grid erosion bottom-hat method results in distortion at the peripheries of the image (Fig. 7A). The interior of the image displays minimal differences against the conventional bottom-hat method (Fig. 7B). Error analysis of the grid erosion method highlights the top left corner in the final bottom-hat image with higher error when comparing between the grid method and conventional method. This artifact is due in part to the grid erosion disk mask starting the grid analysis method at the upper left corner. These errors are compounded with the dilation operation to produce the high local error in final filter output. The final total error of the top left region is approximately 30% difference compared with the conventional method. Additionally, by using the grid method, high local intensity variations are completely smoothed out. These errors are not expected to interfere significantly with final morphological output as morphological analysis routine uses maxima threshold rules to extract diameter measurements. However, one can decrease the filtering error by adjusting to a smaller grid size.

Lastly, we compared the optimized image filter and morphological algorithm for GPU processing to the native

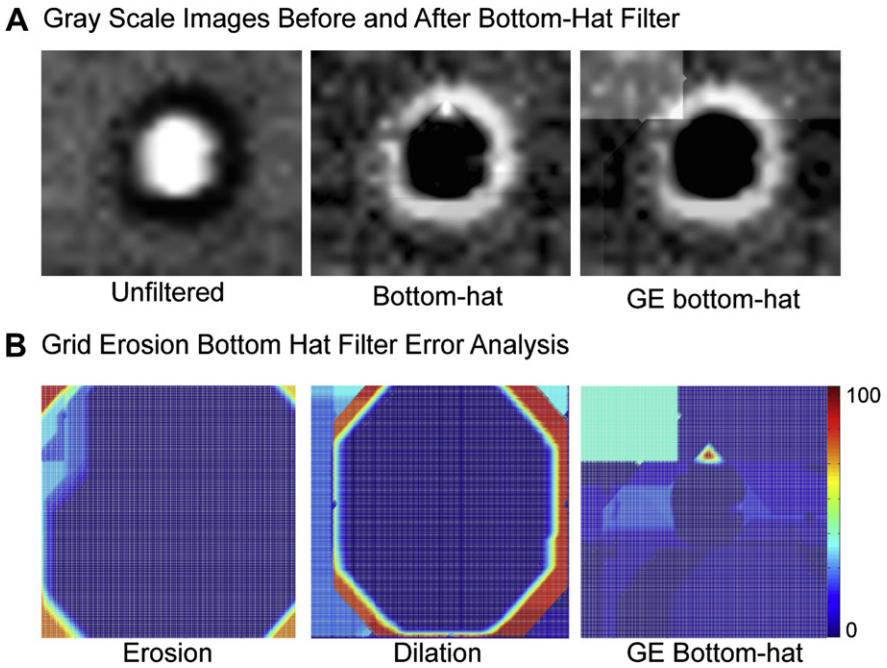

Figure 7. Results of bottom-hat filter using the grid erosion method. (A) Visual comparison of the gray-scale images of original bottom-hat filtered image (middle) versus the grid erosion method bottom-hat filtered image (right). (B) Error analysis of the grid erosion bottom-hat filter method against the naïve bottom-hat filter method illustrating that the highest errors are along the edges of the image, while interior pixel values are conserved.

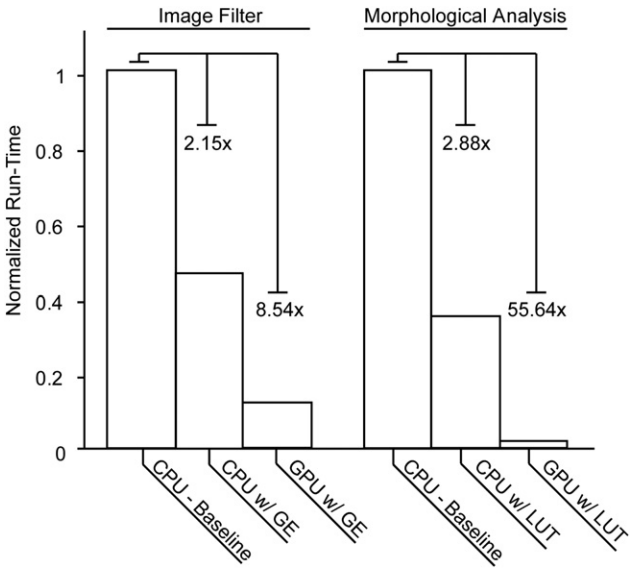

Figure 8. Run-time improvements against baseline algorithm. The grid erosion (GE) routine for the bottom-hat filter method achieves a 2.15× run-time improvement on a CPU, and an 8.54× run-time improvement when implemented on a graphical processing unit (GPU). The coordinate transformation look-up-table (LUT) routine achieves a 2.88× run-time improvement on a CPU, and a 55.64× run-time improvement when implemented on a GPU.

CPU code. The testing was done on a 64-bit Intel Core i7 1.60 GHz machine with 6 GB RAM and NVIDIA GT230M GPU. Our first benchmark compared the conventional bottom-hat filter against the grid method. The grid method bottom-hat filter algorithm resulted in a 2.15× speedup against the baseline algorithm (Fig. 8). When this is implemented on GPU hardware, we see a total decrease of 8.54× speedup.

The morphological analysis algorithm when modified with the transformation LUT also decreased the run time dramatically. When compared with the original method without coordinate correction and the use of the transformation LUT, there is a 2.88× decrease in run time on a CPU, and when implemented on GPU, there is a 55.64× decrease in run time. As discussed in the previous section, we can expect further decreases in run time as we adjust the degree of batch-image processes. Determining the optimal batch rate is an on-going investigation.

Conclusion

In our approach to hardware-assisted high-throughput image analysis, we have shown that through improvements to algorithm design, execution of data management shortcuts, and optimization of image filters for implementation in parallelization schemes, the computational requirements for image analysis routines can be diminished dramatically. The baseline algorithm processes 260,800 frames in approximately 200 min. To date, after algorithm adjustments and processing done on the GPU, processing time is reduced to under 10 min on a single processing unit (a 20.30× reduction). Parallel jobs will allow for even greater improvement. However, to achieve real-time analysis capabilities work remains to further improve the algorithms and eventually transition from a GPU to an FPGA. This shift often imparts an additional speedup that will ultimately be necessary to elicit wider adoption of biomedical technologies, which require computationally demanding image analysis.

Acknowledgments

Competing Interests Statement: The authors certify that they have no relevant financial interests in this article and that any/all financial and material support for this research and work are clearly identified in the Acknowledgments section of this article.

References

1. Witt, W.; Kohler, U.; List, J. Direct imaging of very fast particles opens the application of the powerful (dry) dispersino for size and shape characterization. *International Conference for Particle Technology. Nuremberg, Germany*. **2004**.
2. Provder, T. Challenges in particle size distribution measurement past, present and for the 21st century. *Prog. Org. Coatings* **1997,** *32*, 143–153.
3. Culverhouse, P. F.; et al. Automatic image analysis of plankton: future perspectives. *Mar. Ecol. Prog. Ser* **2006,** *312*, 297–309.
4. Sieracki, C. K.; Sieracki, M. E.; Yentsch, C. S. An imaging-in-flow system for automated analysis of marine microplankton. *Mar. Ecol. Prog. Ser* **1998,** *168*, 285–296.
5. Starkuviene, V.; et al. High-content screening microscopy identifies novel proteins with a putative role in secretory membrane traffic. *Genome Res* **2004,** *14*, 1948–1956.
6. Shapiro, H. M. *Practical Flow Cytometry*. New York: Wiley-Liss; 2003.
7. Di Carlo, D.; Lee, L. P. Dynamic single-cell analysis for quantitative biology. *Anal. Chem* **2006,** *78*, 7918–7925.
8. Di Carlo, D.; Irimia, D.; Tompkins, R. G.; Toner, M. Continuous inertial focusing, ordering, and separation of particles in microchannels. *Proc. Natl. Acad. Sci* **2007,** *104*, 18892–18897.
9. Gossett, D. R.; Di Carlo, D. Particle focusing mechanisms in curving confined flows. *Anal. Chem* **2009,** *81*, 8459–8465.
10. Hur, S. C.; Tse, H. T. K.; Di Carlo, D. Sheathless inertial cell ordering for extreme throughput flow cytometry. *Lab Chip* **2010,** *10*, 274.
11. Bird, J. C.; de Ruiter, R.; Courbin, L.; Stone, H. A. Daughter bubble cascades produced by folding of ruptured thin films. *Nature* **2010,** *465*, 759–762.
12. Basiji, D. A.; Ortyn, W. E.; Liang, L.; Venkatachalam, V.; Morrissey, P. Cellular image analysis and imaging by flow cytometry. *Clin. Lab. Med* **2007,** *27*, 653–670.
13. McGrath, K. E.; Bushnell, T. P.; Palis, J. Multispectral imaging of hematopoietic cells: where flow meets morphology. *J. Immunol. Methods* **2008,** *336*, 91–97.
14. Nobile, C.; et al. HIV-1 Nef inhibits ruffles, induces filopodia, and modulates migration of infected lymphocytes. *J. Virol* **2010,** *84*, 2282–2293.
15. Pepperkok, R.; Ellenberg, J. High-throughput fluorescence microscopy for systems biology. *Nat. Rev. Mol. Cell. Biol* **2006,** *7*, 690–696.

16. Lin, G.; et al. Automated image analysis methods for 3-D quantification of the neurovascular unit from multichannel confocal microscope images. *Cytometry A* **2005,** *66*, 9−23.
17. Simagis Live Technology use cases—simagis live. at <http://live.simagis.com/cases> (accessed May, 2011).
18. Owens, J. D.; et al. A survey of general-purpose computation on graphics hardware. *Comput. Graph. Forum* **2007,** *26*, 80−113.
19. Govindaraju, N. K.; Lloyd, B.; Wang, W.; Lin, M.; Manocha, D. *Fast computation of database operations using graphics processors*. Proceedings of the 2004 ACM SIGMOD international conference on Management of data, 2004; Paris, France: pp 215−226. doi:10.1145/1007568.1007594.
20. Michalakes, J.; Vachharajani, M. *GPU acceleration of numerical weather prediction. Parallel and Distributed Processing, 2008*. IPDPS 2008. IEEE International Symposium, 2008; Miami, Florida, USA: pp 1−7. doi:10.1109/IPDPS.2008.4536351.
21. Manavski, S. A. *CUDA Compatible GPU as an Efficient Hardware Accelerator for AES Cryptography. Signal Processing and Communications, 2007*. ICSPC 2007. IEEE International Conference, 2007; Dubai, United Arab Emirates: pp 65−68. doi:10.1109/ICSPC.2007.4728256.
22. Stone, S. S.; et al. Accelerating advanced MRI reconstructions on GPUs. *J. Parallel Distr. Com.* **2008,** *68*, 1307−1318.
23. Fung, J.; Mann, S. *Using graphics devices in reverse: GPU-based Image Processing and Computer Vision*. Multimedia and Expo, 2008 IEEE International Conference, 2008; Hannover, Germany: pp 9−12. doi:10.1109/ICME.2008.4607358.
24. Mueller, K.; Fang Xu. *Practical considerations for GPU-accelerated CT. Biomedical Imaging: Nano to Macro, 2006*. 3rd IEEE International Symposium, 2006; Arlington, Virginia, USA: pp 1184−1187. doi:10.1109/ISBI.2006.1625135.
25. Hefenbrock, D.; Oberg, J.; Nhat Thanh; Kastner, R.; Baden, S. B. *Accelerating Viola-Jones Face Detection to FPGA-Level Using GPUs*. Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium, 2010; Charlotte, North Carolina, USA: pp 11−18. doi:10.1109/FCCM.2010.12.
26. Ryoo, S.; et al. *Optimization principles and application performance evaluation of a multithreaded GPU using CUDA*. Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming, 2008; Salt Lake City, Utah, USA: pp 73−82. doi:10.1145/1345206.1345220.
27. Ryoo, S.; et al. *Program optimization space pruning for a multithreaded GPU*. Proceedings of the 6th annual IEEE/ACM international symposium on Code generation and optimization, 2008; Boston, Massachusetts, USA: pp 195−204. doi:10.1145/1356058.1356084.
28. Sonka, M.; Hlavac, V.; Boyle, R. Image processing, analysis, and machine vision. Stamford, CT: *Thompson Learning*. **2008**.