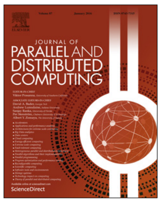
Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc



A hardware accelerated system for high throughput cellular image analysis

Dajung Lee [*,1], Nirja Mehta [1], Alexandria Shearer [2], Ryan Kastner [2]

University of California, San Diego, 9500 Gilman Dr, La Jolla, CA 92122, USA

HIGHLIGHTS

- A scalable, high speed image analysis algorithm for cell morphological analysis.
- A hardware accelerated system to achieve a high throughput and low latency constraint.
- A demonstration of a proposed system in an end-to-end (CPU- FPGA) machine.
- A flexible hardware design for an FPGA using a high-level synthesis tool.

ARTICLE INFO

Article history:
Received 2 July 2016
Received in revised form 7 November 2017
Accepted 15 November 2017
Available online 24 November 2017

Keywords:
Hardware acceleration
Biomedical image analysis
Cytometry
Reconfigurable hardware
FPGA
High throughput

ABSTRACT

Imaging flow cytometry and high speed microscopy have shown immense promise for clinical diagnostics, biological research, and drug discovery. They enable high throughput screening and sorting using biological, chemical, or mechanical properties of cells. These techniques can separate mature cells from immature ones, determine the presence of cancerous cells, classify stem cells during differentiation, and screen drugs based upon how they affect cellular architecture. The process works by imaging cells at a high rate, extracting features of the cell (e.g., size, location, circularity, deformation), and using those features to classify the cell. Modern systems have a target throughput of thousands of cells per second, which requires imaging at rates of more than 60,000 frames per second. The cellular features must be calculated in less than a millisecond to enable real-time sorting. This creates challenging computing performance constraints in terms of both throughput and latency. In this paper, we present a hardware accelerated system for high throughput cellular image analysis. We carefully developed algorithms and their corresponding hardware implementations to meet the strict computational demands. Our algorithm analyzes and extracts cellular morphological features from low resolution microscopic images. Our hardware accelerated system operates at over 60,000 frames per second with 0.068 ms latency. This is almost 1400× faster in throughput than similar software based analysis and 335× better in terms of latency.

1. Introduction

Quantitative analysis of cellular properties, such as size, shape, structure, life span, and molecular contents, can characterize cell function, give insight into how it behaves, and provide a technique for cell screening and/or sorting. This is useful for diagnosing disease, monitoring immune systems, screening drugs, and developing regenerative medicine [3,28,23,18]. However, there are strict performance constraints to achieve real-time cellular analysis; the system must have enough processing power to handle a very high cell throughput, and must perform cell feature analysis with a sub-millisecond latency to facilitate sorting.

Our system is capable of analyzing thousand of cells per second based on image based technology, which corresponds to work at over 60,000 frames per second; this is a common goal in imaging flow cytometry [13,34,9]. In our system, a high speed camera images cells at very high frame rate on a specialized experimental setup to catch fast moving target cell features. Camera at such high frame rates have lower resolutions, in our case 64 × 64 pixels. These images may have low contrast; they are sensitive to even a single pixel noise; and the fast movement of the cells within a tiny field of view causes blurring and other optical effects. Furthermore, there are strict latency constraints—real-time cell sorting requires decisions in under 10 ms [25], and ideally under one millisecond.

[*] Corresponding author.
E-mail addresses: dal064@eng.ucsd.edu (D. Lee), njmehta@eng.ucsd.edu (N. Mehta), ashearer@ucsd.edu (A. Shearer), kastner@ucsd.edu (R. Kastner).
[1] Department of Electrical and Computer Engineering.
[2] Department of Computer Science and Engineering.

https://doi.org/10.1016/j.jpdc.2017.11.013
0743-7315/© 2017 Elsevier Inc. All rights reserved.

Our system processes these noisy, low-contrast, blurry images with very high throughput and minimal latency.

Cytometry systems in such high performance use a specialized sensor, such as a laser, or focus on observing simple properties. However, our cell analysis system computes cellular morphological feature using bright field imaging on a microfluidic device, which gives more sophisticated information of cellular mechanical properties. The system images cells using a high speed optical image sensor, analyzes the resulting video streams, extracts features from the images, and classifies cells based on those observed features. It does not require labeling the cells with fluorescent chromes, which minimizes the preparation time and effort. However, the bright field based images have limited resolution; they are sensitive to variations in lighting; and they easily become blurred or noisy. This makes the accurate extraction of the morphological features difficult, which hinders high throughput massive cell analysis inspite of many other benefits of image based cell analysis system.

In this work, we carefully develop a cellular analysis algorithm to extract their morphological features from microscopic images and build a real-time system using an FPGA device. There are various image analysis approaches for feature detection over low contrast images. However, these approaches are too computationally intensive and hard to achieve such high performance even on a hardware implementation. Most of them have iterative solutions to refine analysis results or use spatial and temporal signatures to estimate target features accurately, which causes longer latency and lower throughput. Our method does not have an iterative process to find a solution and minimizes data dependency for independent operations. It processes input and intermediate data in streaming way, which is intended for an efficient hardware implementation in terms of performance and resources.

The major contributions of our work are:

- Accurate image analysis algorithms for high speed cell morphological analysis.
- Hardware architectural optimizations using high-level synthesis (HLS) code.
- Developing a hardware accelerated system for microfluidic deformability cytometry.
- An in-depth evaluation and end-to-end demonstration of our system using a heterogeneous (CPU–FPGA) compute platform.

The remainder of the paper is organized as follows. We describe background and introduce our target system in Section 2. Section 3 overviews and discusses related works. Then, we explain detailed image analysis algorithms and hardware architecture optimization methods in Section 4. Section 5 presents the system description and experimental results in terms of accuracy and performance. We conclude in Section 6.

2. Background

Cytometry assesses biological, physical, or chemical characteristics of cells using specialized instruments or micro-devices. Imaging cytometry and flow cytometry are the most well known methods of cell analysis.

Imaging cytometry is the oldest and most basic method. It observes cells using a microscope which results in high contrast and high resolution images, yet cannot be performed in a high throughput manner. Imaging cells at the microscopic level commonly requires staining them with a fluorochrome, which binds to a structure within the cell [17,36]. This labeling process highlights particular molecules or cellular structures. For example, it can separate out individual cell features (like the cell membrane or nuclei) and determine interactions between multiple cells [14]. Accurately extracting cell parameters demands significant effort making it difficult to perform high throughput analysis [8,32].

Flow cytometry uses a laser [31,30], an optical device [26,16], or an electrical impedance device [6,10] to extract course features from cells suspended in a fluid. For example, cells are labeled with fluorochromes, which activate when targeted with a particular wavelength of light. A cytometer reads these tagged response signals and determines the cell types or properties. This method is capable of providing high throughput cell analysis, but is not capable of extracting sophisticated cell parameters.

Imaging flow cytometry combines the strengths of flow cytometry (high throughput) and imaging cytometry (high sensitivity) [4,5,1]. Take, for example, the ImageStream by Amnis [34] — a commercial imaging flow cytometer capable of processing 5000 cells per second. It produces 12 images: 10 fluorescent markers in addition to darkfield and lightfield images. The fluorescent images provide higher contrast but require a pre-processing step to add the fluorochromes. On the other hand, the lightfield and darkfield images have no pre-processing requirement, but have reduced image clarity.

Our system targets a particular type of imaging flow cytometry that analyzes cellular mechanical properties using bright-field images. The main idea is to generate a force on a cell in a flowing fluid and determine its physical response. We can analyze the high speed images to determine mechanical properties based upon the cell's shape, size, circularity, and deformability. And then we can use those features to classify the cell.

For example, we can determine the deformability of a cell by analyzing the image. Different cells will deform in different ways. A pluripotent stem cell deforms more than its differentiated progeny; pleural fluid with metastatic cells will deform more than fluid with normal cells [13]; cells susceptible to tumor cell invasion have a changing mechanical behavior [24]; cancerous cells with the highest invasive potential are stiffer than those with lower migrations [33]; and older cells deform differently than younger ones [38]. More generally, recent research states that cells' mechanical properties "play important roles in the regulation of various biological activities at the molecular and cellular level" [39]. Thus, a cellular image analysis system for microfluidic deformability cytometry provides an attractive approach for high throughput cell screening and sorting.

In this paper, we focus on developing a high speed cellular analysis system for microfluidic deformability cytometry. This uses a microfluidic channel to deliver a cell into the center of a stretching extensional flow, which generates a uniform stress on the cell causing a deformation. The cells flow quickly through the microfluidic channel enabling high throughput processing. By imaging the cell in the extensional flow with a high speed image sensor, we can observe the deformation of large population of cells with a high throughput.

Fig. 1 shows the target system that is designed to produce cell stretching in an extensional microfluidic channel. The target system uses a high speed camera to observe cellular deformation. It applies uniform hydrodynamic force to a single cell on the channel, while the fast flowing fluid enters the field of view of the camera. High speed microscopy focuses on the center point imaging cell's movement and its deformation. For example, Fig. 1(d) shows a sequence of a single cell events from entering the field of view to exiting into an outlet for sorting. The resolution of this microscopic image is 64×64 and one cell stays in this view only few microseconds or few frames. Because of the fluid speed in the channel, this hydrodynamic approach is able to assess a large number of cells efficiently. This technique has the potential to process up to 20,000 cells per second. However, it comes with the critical bottleneck of handling the generated image data.

Our target performance is to analyze 2000 cells per second while assuming (1) any frame has no more than one cell (2) one

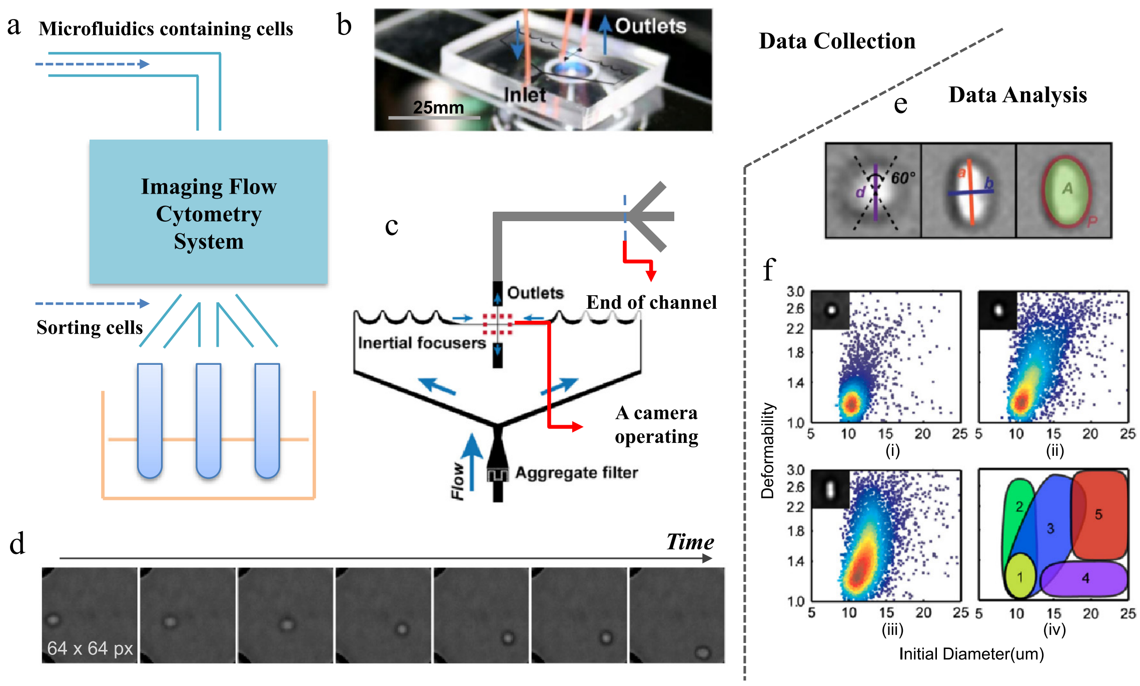


Fig. 1. (a) An imaging flow cytometry system can be used to sort cells by imaging micro-fluid having cells in a device. It consists of two big processes, data collection and data analysis. Data collection: (b) this system has a microfluidic device where the fluid flows (c) the device is designed to generate uniform pressure to stretch cells. (d) raw images for one cell event. A cell can appear on 15 frames at most from entering to exiting a field of view. Data analysis [13]: (e) it analyzes cellular morphological features, such as initial diameter, circularity, or size. (f) examples of cellular mechanical property analysis, (i), (ii) and (iii) density scatter plots of the size and deformability of different cells. (iv) different patterns for different cells in size-deformability map.

cell event appears in 7–15 consecutive frames, and (3) only 50% of frames have valid cell feature. That means the system must process 28,000–60,000 frames in one second. Also, for the practical usage of cell sorting, it should analyze one cell within a predetermined latency, which is dictated by the time it takes the cell to arrive at the sorting point or the end of the extensional channel.

3. Related work

In this section, we review other literature pertaining to high-performance hardware-accelerated cytometry research. High-throughput cell analysis systems are mostly based on an FPGA, DSP or embedded-sensor system. As we discussed in Section 2, the most conventional cytometry research for high throughput analysis is flow cytometry, which reads a particular reaction signal that varies depending on the experimental setup and cellular properties. This specialized hardware has been commonly employed in various signal-processing research and provides an optimized system to meet high throughput and low latency constraints. Thus, flow cytometers for high-throughput cell analysis are, in general, implemented on an embedded system to maximize their performance.

There are several imaging flow cytometry approaches, but mostly they are limited to a particular type of image input that still requires the researcher to biotag or attach fluorescence on a cell or risk limited performance. [4,11,12] Our approach is based on a bright-field image taken by a high-speed phantom camera, which requires no tags on a cell. We measure cellular morphological features based on this image input enabled by a high-performance system using an FPGA.

Buschke et al. [4] present a cell analysis system for image cytometry using a DSP and FPGA. Their system attempts to detect the cell and automatically process the image in the same way we do. However, our input data are bright field images, which require more complex processing than their fluorescence images. Moreover, our requirements are more challenging in terms of throughput and latency compared to their system, which operates at 2.33 frames per second.

Goda et al. [11,12] developed a heterogeneous hardware accelerated (FPGA and CPU) image cytometer for cancer cell detection. For this application, the FPGA is used for data capture and performs only simple low-level processing, which amounts to coarse size-based classification. The CPU performs the bulk of the analysis on the filtered images. Our system requires significantly more advanced morphological feature detection to be performed in real-time on the FPGA.

GPUs are frequently used to accelerate image processing and computer vision applications. GPUs achieve speedups by processing and computing upon data using thread level parallelism. Tse et al. [35] present a GPU-based approach to analyze cellular features in image. It parallelizes two bottleneck modules in a system pipeline: an image transform function from one image coordinate to another one and a bottomhat-filter module for contrast enhancement. This work accelerates only partial modules of the entire image analysis process, but shows substantial improvement over a CPU-based implementation.

However, a GPU implementation must access DDR memory frequently to load/store data, which inevitably causes high latency. This inherent restriction of GPU architecture prevents our application from achieving its strict latency goal. Our previous work [25] presents an FPGA-based approach for imaging flow cytometry and compares the accelerated result in different platforms including a GPU. The FPGA based architecture for cell analysis achieves 2262 FPS throughput and 1.4 ms/frame latency. This result is better than a GPU implementation in terms of throughput, and it shows an FPGA can meet the strict latency constraints that are not achievable on the GPU.

In this article, we demonstrate an advanced FPGA acceleration work for various datasets with better accuracy in higher performance than the previous work [25]. Imaging cells in a microscope highly depends on the camera performance. Bright field based image is easily blurred and sensitive to the light level, so noise level or intensity level of input image varies and image quality is not consistent. The previous work presents a prototyped design only for a single data set, so it is not adjustable for other dataset in different conditions. We develop more generic algorithm for more difficult datasets using experimental setups. The noise level in the new datasets has small particles that can be easily misread as a valid cell. The algorithms described in this research work on a broader set of data with higher accuracy.

Thus, the system pipeline is deeper and has more stages, which may increase the computation complexity if it is not designed carefully. We partition the analysis into a detection module and an analysis module. The separate detection module screens empty frames at a higher frame rate and only passes valid cell frames to the analysis module. The analysis stage analyzes cell morphology accurately on a subset of the original frames. The prefiltering detection operation reduces the frame rate requirements for the analysis, which generally contains more complex functions that end up being the bottleneck. We parallelize these operations when necessary to achieve higher framerates.

To achieve our performance goals in both latency and throughput, we should minimize memory access and parallelize data and tasks intensively. Our analysis algorithm is carefully designed to feed data in a feed-forward streaming manner. We leverage on-chip memories and our hardware design is customized to implement pipeline, data, task level parallelism in our system in various level. These optimizations enable a design that is 30 times faster speed up than our previous implementation.

Using hardware acceleration for image processing is well known in many other domains. However, the target throughput is generally less demanding (under 1000 frames per second) and focused on processing the larger resolution images [21,15]. Greisen et al. present a video processing pipeline for high definition stereo video in [15]. It utilizes a FPGA–GPU–CPU system for high speed stereo vision and processes video streams up to the resolution 1920×1080 at 30 frames per second.

There are several works accelerating a medical imaging system on an FPGA [37,7]. Coric et al. [7] present a hardware accelerated parallel-beam backprojection algorithm used in computerized tomography (CT). Xu et al. [37] developed a medical imaging system for a CT filtered backprojection algorithm. They compare and investigate different hardware designs using C, Impulse C and VHDL. Their work is limited in that they show their manual design has better performance than Impulse C. Our work targets a different imaging system and achieves the high performance requirements.

We can find several other image processing systems running at higher frame rates (thousands of frames per second) [22,19,12]. Kagami et al. shows a networked vision system of transferring visual features using ethernet at 1000 fps in [22]. It handles a 64×64 image on an FPGA attached to a CMOS vision chip, but it does not analyze the image itself and only performs on preprocessed vision features and transfers them through a network. IDP Express is a high speed vision system that uses an FPGA to record 512×512 images and operates at 2000 frames per second [19]. Compared to these works, our target system has more challenging requirements as it needs to perform more complex image analysis to extract the cellular morphology.

4. Methods

In this section, we introduce our cell image analysis algorithm and its hardware accelerated architecture on an FPGA. To ease the design space exploration process, we design and optimize the hardware architecture using a high level synthesis tool, Xilinx Vivado HLS 2015.2. It allows us to focus on behavior level synthesis, data access patterns, pipelining, connections between modules, and so on, rather than low-level hardware debugging.

4.1. System goal

We aimed to extract cellular morphological features from given cell images, such as cell shape, size or radius, circularity, and deformability. To measure cell shape or roundness, we convert a cartesian coordinate of the cell image into a polar coordinate and trace the cellular wall. The basic idea is that, if a cell is a closed convex shape, the converted image can be used to describe a cell morphology. For example, if a cell is a perfect circle, the distances between the cell center and the cellular wall in every angle will be the same. However, if the cell is stretched, then they will not.

One cell is captured in multiple frames as it flows across the field of view of the camera. The number of consecutive images depends upon the flow rate. In our system, a cell can be present in up to 15 frames. By integrating morphological feature results of these frames, we can acquire the deformability or the shape changing pattern of cell.

4.2. Overall flow

The imaging flow cytometry system is not only computationally intensive for image analysis algorithm, but it is highly data intensive. Our method processes these image data in streaming and fully pipelined manner in finer level as well as functional level for high performance and low resource usage. It has to handle massive amount of cellular images that are coming in the analysis pipeline in streaming way pixel by pixel initially. It can process incoming data when it is ready and forwarding it to next module right away with no storage required. It does not have iteration or feed data backward in its data flow, which prevents a faster pipelining operation. Our streaming data processing approach minimizes delay of analysis results and on-chip memory for caching data.

Also, modules do not have iterations. All functional modules can be implemented in one-pass processing, and input and intermediate data in the algorithm pipeline are delivered only forward. That enables us to create a hardware architecture that works on streaming images with a pipelined structure. The cell image analysis algorithm has two major parts: *cell detection* and *cell analysis*. Fig. 2 provides an overview of the process.

Fig. 3 shows a cell analysis core architecture at a high level, corresponding to the algorithm flow. The *averaging* module generates a background image as a preprocessing step. It takes the first 256 frames and averages them. The averaged background is stored to a BRAM and persists in memory while the system is running. All incoming input images after the first 256 frames are directly connected to the *detection* module. The *detection* module is relatively smaller and simpler than the *analysis* module. It quickly checks if the current frame has a cell or not. The background image will be used in this process. Only valid cell frames are passed to the *analysis* module. The *analysis* module processes the rest of the major operations; deformation/morphological analysis. It consists of three stages; *find cell*, *find center*, and *trace cellular wall.* We synthesize these big modules separately and integrate them manually to generate a bitstream in Vivado 2015.2.

4.3. Image analysis pipeline

Fig. 4 shows a detailed block diagram of our hardware design including all modules and the connections between them. The box (a) in the figure is *cell detection*, and the rest of them, (b), (c), and(d), are substages of *cell analysis*, *find cell*, *find center*, and *trace cellular wall*, respectively.

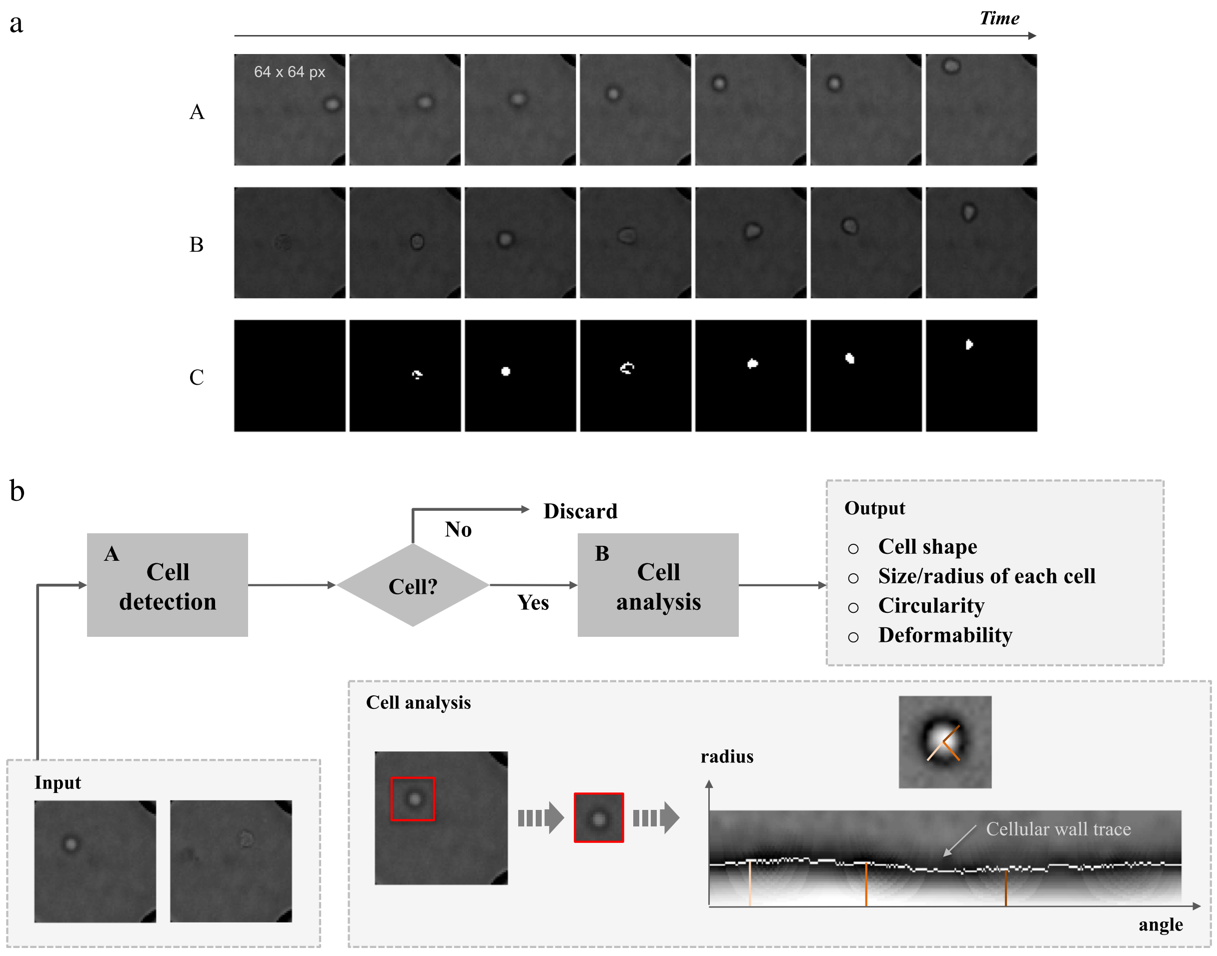


Fig. 2. Cell image dataset and a cellular morphology feature analysis flow in high level; (a) A and B are raw input data for cell events. They have 64 × 64 resolution. White pixels in C represent cell area of the cell event B, which is intermediate data in processing module. (b) The algorithm flow consists of two main modules; cell detection and cell analysis. The detection module checks if a current frame has a cell and passes only cell frames to the analysis module. The analysis module converts cell-focused areas into polar coordinate images and traces cellular walls on the image. This shape information can be used to estimate radius, circularity, and deformability; Note that frames in the C row of (a) is from the cell detection module in (b).

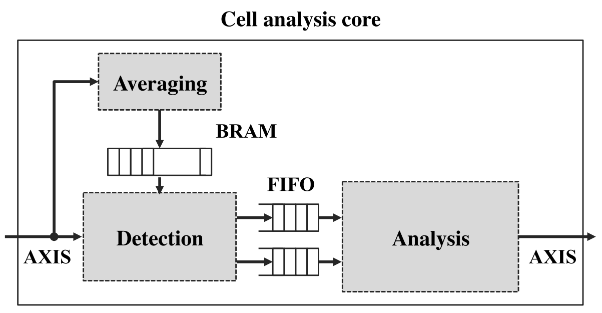


Fig. 3. Cell analysis core; *Averaging, detection, and analysis*. The averaging module takes the first 256 frames to generate a background image. The detection and analysis modules start running after that. The averaging module generates a background image and stores it using a BRAM, which is read by the detection module. The detection module passes intermediate images to the analysis module using FIFOs, and, if a current frame has no cell, it discards it.

4.3.1. *Detection module*

The *detection* module detects the presence of a cell quickly from incoming frames and passes only valid cell frames to the next stages, rejecting empty ones. We minimize the complexity of the *detection* module for a fast detection process. This rejecting process is based on a binary image, where it represents the cell area as white (or 1 in binary) pixel values (see in Fig. 5(a)).

In Fig. 4(a), when an input frame (C) comes in, it subtracts (B) the background image acquired by averaging the first 256 input frames in the *averaging* module. Then, it considers the bins in its histogram with the lowest intensity as background and selects a gray level value to separate the cell area from background. The binary image generated from this process may have noise in the background, so it applies a binary morphology operation, *erosion-only*, to leave only big particles, which is likely to be a cell. The number of valid true pixels in this frame is used to determine frames with a cell. A one-bit *iscell* flag indicates this frame has a cell (A).

4.3.2. *Find cell stage*

The find cell stage needs a more accurate cell location than the cell detection stage. The input is the background subtracted image (B) from the *detection* module. Then, a Gaussian filter is used to denoise the input (E), and the thresholding module converts it (E) into a binary image. To remove extra particles from the background, it does a binary morphology operation, *opening*, i.e. *dilation* after *erosion*. The resulting binary image has a white blob on a plain black background representing the cell area as shown in Fig. 5(b). Averaging the number of these white pixels in each row and column gives an exact location of cell (D) (See Fig. 7).

4.3.3. *Find center stage*

Based on the location of the cell (D), it crops a 24 × 24 cell area from three images (B, C, and E). The *resizing* module interpolates

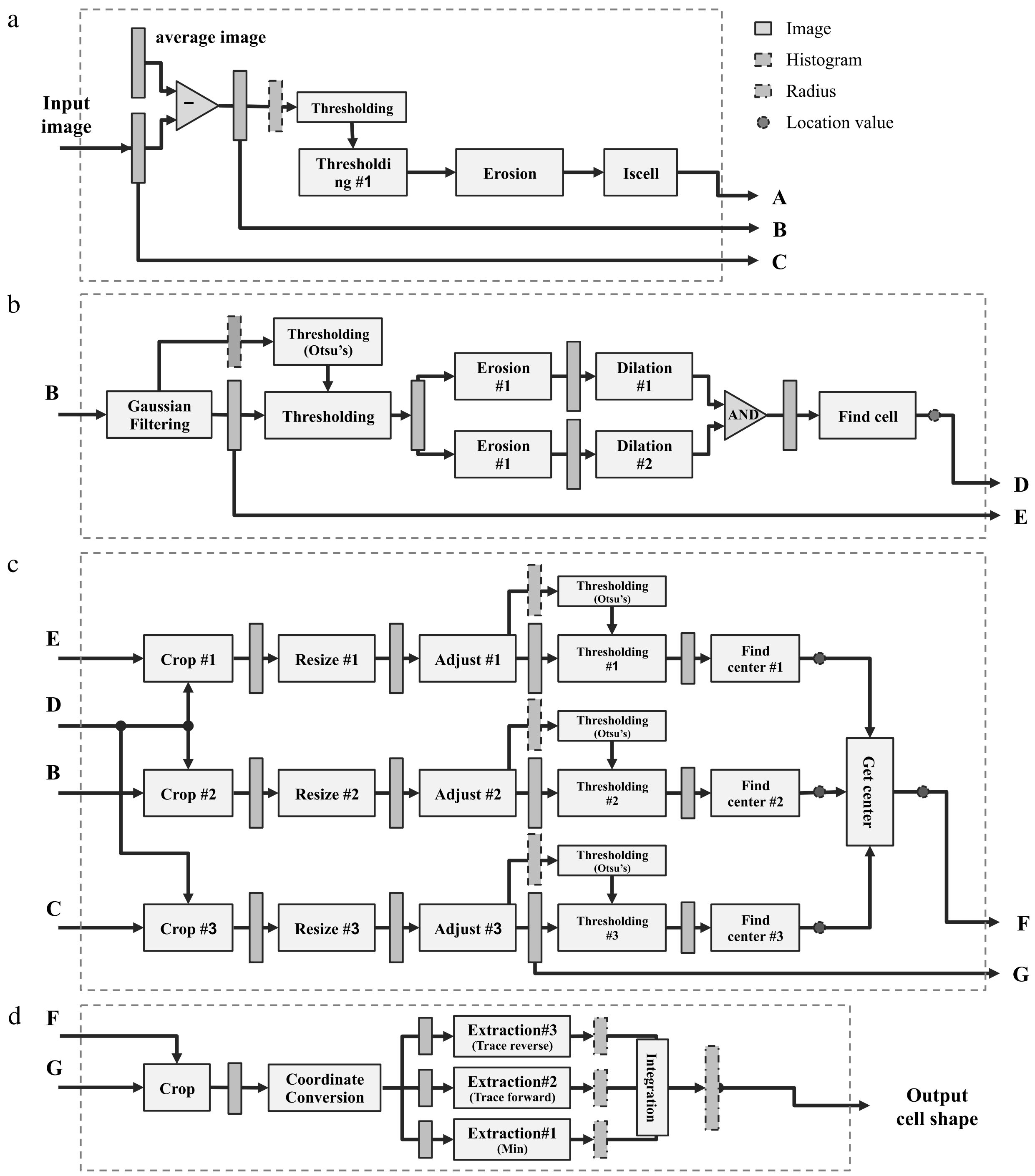


Fig. 4. Cell analysis core pipeline block diagram (a) cell detection module (b)(c)(d) cell analysis module; (b) find cell, (c) find center, (d) trace cellular wall. The connections between these stages are noted alphabetically.

them 5 times and the *adjusting* module enhances its contrast. It converts the contrast-enhanced images to binary images to find the center point of a cell. In the binary images, white pixels represent the inner cell area or cellular walls as shown in Fig. 6. It finds a center point (F) by averaging the number of these pixels in each row and column similarly to the *find cell* module. Averaging the 3 images also confers the benefit of reducing the noise since the errors in one are compensated for in the other images. The center point will be the input for the next stage.

4.3.4. *Trace cellular wall stage*

In this stage, the *conversion* module converts cartesian coordinate cell images into polar coordinate images based on the center point (F). The horizontal axis represents the angle from the x axis in the original image, 0 to 360 degrees, and the vertical axis represents the distance from the center of the cell, or the radius. It uses the contrast-enhanced input image (G) as the module input and the darkest pixel in every single angle is considered the cellular wall. Finding the minimum intensity value at a particular angle is

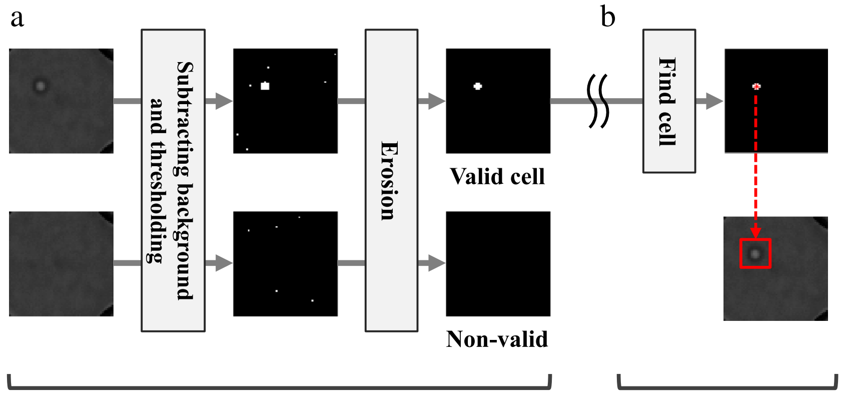


Fig. 5. The cell detection process and the find cell stage in the cell analysis module (a) *cell detection*; it subtracts background from a given input image using thresholding. Then based on a converted binary image, it determines valid/non-valid cell frames.(b) *finding cell*; similar to *cell detection*, it finds a location of the cell from a denoised binary image, which is more accurate.

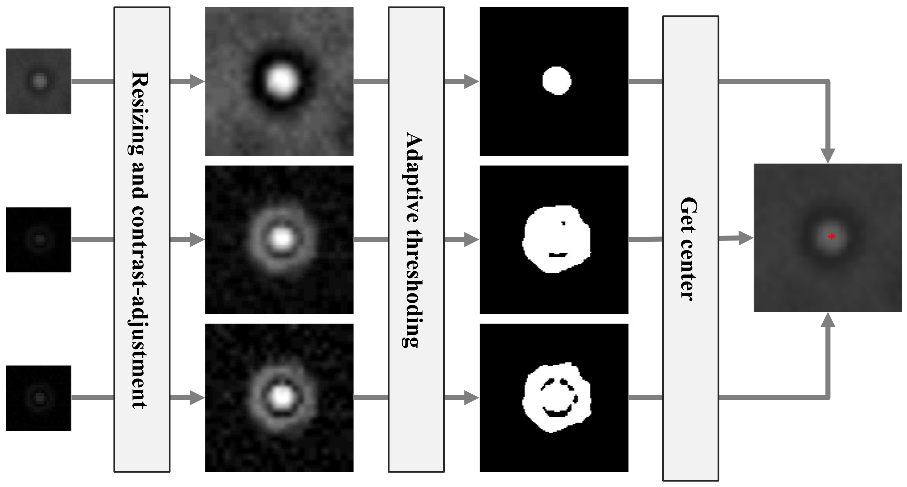


Fig. 6. Find center stage; this stage resizes the cropped cell area from the three images based on the cell location found in the find cell module. Then, it resizes the cropped images 5 times and enhances their contrast. Adaptive thresholding converts the adjusted images to binary images. A center point is found by averaging the number of white pixels in each row and column.

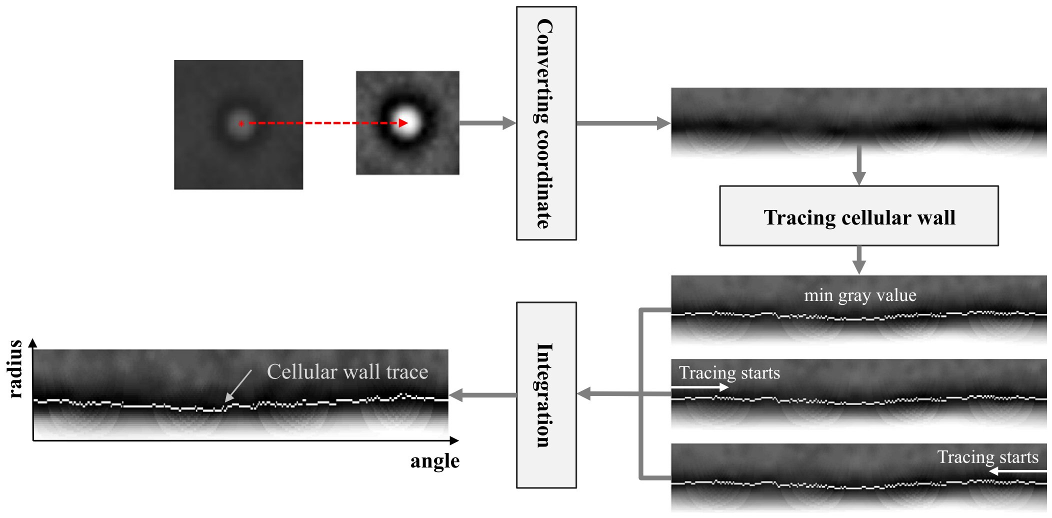


Fig. 7. Trace cellular wall; The input to this module is the contrast-enhanced input image and the center point from the previous module. Based on the center point, it converts the cell image into a polar coordinate image and traces the cellular wall. The horizontal axis represents 0 to 360 degree angles. The vertical axis represents the radius, the distance from the center point to cellular wall in terms of the number of pixels. The lowest intensity values are considered to be cellular wall.

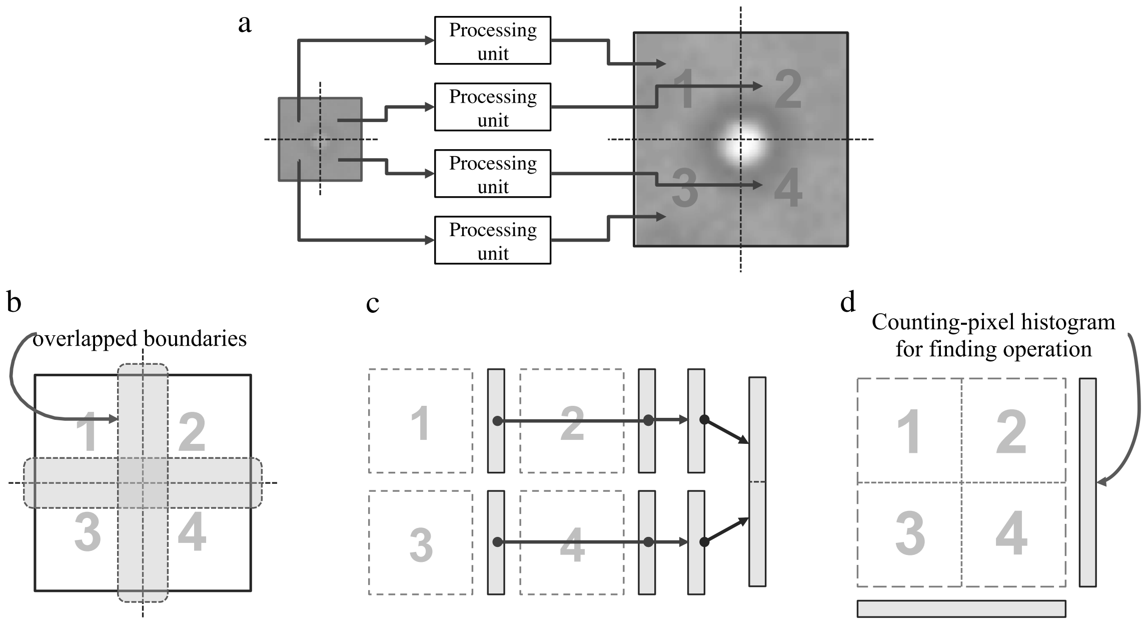


Fig. 8. Hardware optimization for bottleneck modules; *Resizing, adjusting*, and *get center* are the main bottlenecks because they handle the largest size images. (a) To balance the overall performance, they are partitioned into four quadrants and parallelized. (b) The *bicubic interpolation* operation has weak data dependency on the overlapped boundaries region. If we ignore this, it can cause artifacts at the partitioned edges. (c) Generating a histogram has a data dependency on the entire scanned image. It generates a histogram for each partitioned image and reduces them. (d) The *get center* module also needs to scan the image by rows and columns. This process is also partitioned and reduced similar to (c).

the simplest way to determine the distance to the cellular wall, but this method is likely to produce noisy results. So the conversion module extracts the distance to the cellular wall using several different methods and takes the median value of the results for each angle.

4.4. Hardware modules

Some modules in Fig. 4 share some common patterns of computation or data access, like *templates* [27,2]. While [27,2] identify general computation or data access patterns, our work is more specific to image processing. The *sliding window* pattern is the most frequently used pattern in this domain. Gaussian filtering, erosion, and dilation are sliding window kernel operations. Scaling is similar except it changes the size of output image. Thresholding and adjusting are matching a pixel value to another pixel value, and these are not dependent on neighboring pixels. The find cell and get center modules are group detection operations, which find a point by averaging the number of white pixels in a binary image in each row and column. Coordinate conversion transforms a geometrical shape of image, like warping, but also changes the size. Modules in each group are optimized in a similar way, and optimized modules are used to compose larger modules, such as *cell detection, find cell, find center,* or *trace cellular wall* in Fig. 4.

Our hardware implementation is fully pipelined at a fine-grained level (intra-modules) as well as a coarse-grained level (inter-modules). Most of the optimized modules achieve Initiation Interval *(II)* in one clock cycle using the *pipeline* directive. All submodules inside of two big modules run in a functionally pipelined way by applying the *dataflow* directive in HLS (See Table 1).

This hardware design method combined with image analysis algorithm flow let us estimate a deterministic performance result overall and gives a stable throughput in spite of probable algorithm flow update. Because they are divided into small modules and fully pipelined, only a bottleneck module decides the entire throughput. That means, even if the algorithm flow adds more modules for further extensive analysis, it does not affect latency or throughput

Table 1
Modules grouped based on their computation patterns.

Groups		Modules
Sliding window	Kernel operation	Gaussian filtering, erosion, dilation.
	Scaling	Bicubic interpolation, cropping
	Pixel-matching	Adjusting, thresholding
Detection		Find cell, get center
Geometry transformation		Coordinate conversion
Others		Generating histogram, Otsu's method

performance that much as long as additional modules are optimized to meet the similar condition within similar patterns, II in one clock cycle with a *pipeline* directive.

4.5. Bottleneck modules

In a pipelined design, it is important to balance throughput performance by improving main bottlenecks since they are the critical path of the entire system. In our hardware design, the latency of each module depends on the size of image. The main bottleneck modules in the system pipeline are *find center* since it handles the largest frame after resizing; *resizing (bicubic interpolation), adjusting, thresholding,* and *get center*.

To achieve the higher performance, these modules should balance their performance with different modules. There are two ways to achieve that: scaling and partitioning. Scaling is simply replicating bottleneck modules multiple times and maximizing bandwidth, and partitioning breaks a bottleneck module down into submodules. Scaling is relatively simple to implement but it uses more resources to hold data for multiple frames. Partitioning should be done carefully considering data dependencies between submodules, which can introduce more complexity. But processing units in partitioning are smaller and use less resources. So we partition the bottleneck modules, *resizing (bicubic interpolation), adjusting, thresholding,* and *get center*, to multiple submodules that can run in parallel (see Fig. 8(a)). Each module operation and its data are divided and sent to four quadrants.

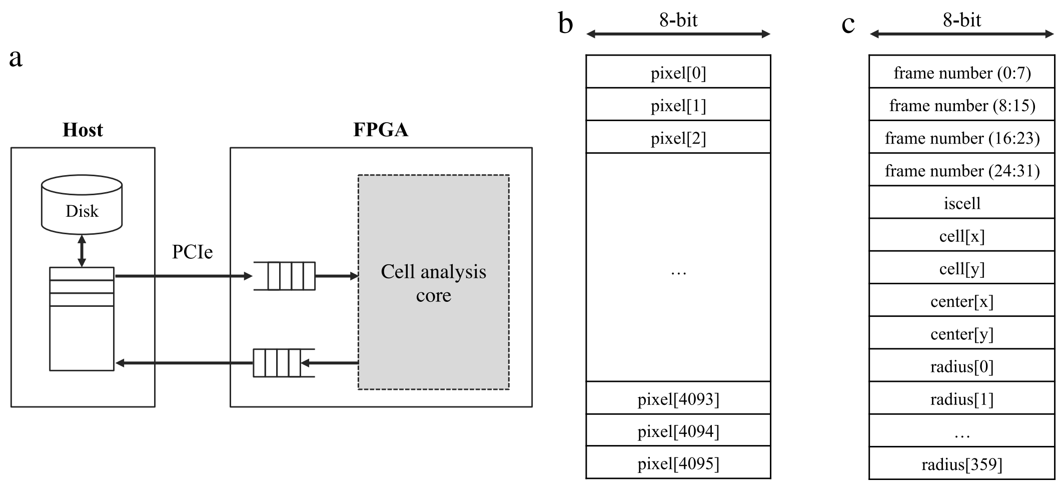


Fig. 9. System description (a) offline cell analysis system connecting a host computer and an FPGA. The host reads raw data from a disk and sends them to the hardware using RIFFA for PCIe. All image analysis is processed on the FPGA side. (b)(c) input and output data format. Input data are streamed as 4096 pixel values and output data for a single frame consists of the frame number, valid cell flag (iscell), cell location from raw data, center point found, and 360 radius values in every angle.

4.5.1. *Bicubic interpolation*

The *interpolation* module is basically a sliding window operation. Line buffers hold pixel data while a sliding window moves over them. It uses a 4×4 sized window for the bicubic operation. No temporal dependency between frames is needed, but there is a spatial dependency in one frame for neighboring window pixels. Ignoring this dependency can cause aliasing at cut boundaries (see in Fig. 8(b)). To minimize this error, line buffers should contain pixel values from the overlapping region. Cropped images are delivered to BRAMs and the line buffers can be filled from the cropped images.

4.5.2. *Image adjustment*

The *image adjustment* operation stretches an image histogram and matches each pixel to another pixel value. Generating the histogram requires checking every pixel value in a given image and therefore there exists a strong data dependency. To generate the histogram of the partitioned image without sacrificing throughput performance, it calculates small histograms for each quadrant first and then reduces them into one later (Fig. 8(c)). The histogram reduction is a one pass operation with a small input size, so it does not decrease the throughput performance.

4.5.3. *Get center*

The *get center* module is similar to the image adjustment operation. It also needs to scan the entire image *counting-pixel histograms* by rows and columns and has a strong data dependency (see in Fig. 8(d)). For partitioned images, each processing unit generates multiple histograms independently, then reduces them into one.

5. Experimental results

In this section, we describe our experimental system and present the accuracy and performance results that we achieved.

5.1. *System description*

We tested our method on 3 different platforms: Matlab, C, and FPGA. The software implementations used a 2.3 GHz Intel Core i5 with 8 GB DDR3. The target FPGA board for hardware implementation is the Xilinx VC707, which has a Xilinx Virtex 7 FPGA device, xc7cx485tffg1761-2. We demonstrate an end-to-end system by connecting the FPGA board with a host computer communicating through PCIe. The control PC has Windows 7 or Ubuntu Linux and runs with the latest RIFFA 2.1 driver for FPGA communication [20]. On the software side, the host is responsible for reading data, streaming the data to the FPGA, and receiving the results (see Fig. 9(a)). The analysis core of the FPGA design uses the AXI-stream interface. This system is for offline analysis, and the connection can be replaced with any streaming interface for an online system.

Table 2
Test video set for accuracy; the number of valid cell frames for *cell detection* results. The first 1000 frames are taken from each video data. Note that the concentration of cells can be controlled by diluting the fluid.

	Set 1	Set 2	Set 3	Set 4
Cell	135	170	148	150
No-cell	865	830	852	850

5.2. *Test dataset*

The test data are organized in four different sets, each of which has 5000 frames. The frames are 64×64 in resolution, 4096-pixels, and stored in an unsigned 8-bit data type. The raw data are taken by a phantom camera [29]. Since this setup is very sensitive to light-level, the contrast level of the test video varies slightly. For accuracy testing, we take the first 1000 frames and generate ground truth data. Then we compare our results to them (see Table 2.) The hit/miss ground truth data are manually produced, but others are from an initial work, which are also estimated results.

The format of the input test data sent to the FPGA is in Fig. 9(b). The output data consists of the frame number, *iscell* signal, the location of the cell, the location of the cell center in the resized image, and 360-radius values (see in Fig. 9(c)). If the *detection* module decides a current frame has no cell, the *analysis* module does not start and generates no output. The first four bytes of the output, i.e. the frame number, help to synchronize an input frame and the output data by counting the number of frames sent to the FPGA.

5.3. *Target throughput performance*

Our initial target performance is to analyze 2000 cells per second. If one cell event appears in 7–15 frames and the event happens in 50% of frames, the system has to be capable of processing 28–60k frames/sec at the front end.

$$\frac{2000\ (\text{cells/sec}) \times 7\text{–}15\ (\text{frames/cell})}{0.5\ (\text{valid frames/total frames})} = 28\text{–}60\ \text{k fps}.$$

Table 3
Detection results with *sensitivity* (true positive rate), *specificity* (true negative rate), *precision* (ratio of true positives to number of positive predictions), and *accuracy*.

	Set 1	Set 2	Set 3	Set 4
Sensitivity (%)	55.31	66.47	64.86	75.33
Specificity (%)	99.88	99.64	100	99.29
Precision (%)	98.73	97.41	100	94.96
Accuracy (%)	93.60	94.00	94.8	95.7

Table 4
Find cell results representing hit/miss rate within a fixed distance from a true cell position.

	Set 1	Set 2	Set 3	Set 4
Cell location (%)	97.36	100	97.92	98.23

Table 5
Accuracy results in mean absolute error (MAE) and statistical distributions of test and ground truth data in terms of mean (μ) and standard deviation (σ).

			Set 1	Set 2	Set 3	Set 4
Size	MAE		1.31	1.37	6.88	1.74
	True	μ	15.03	14.73	14.20	12.43
		σ	2.52	2.71	3.26	2.98
	Test	μ	16.11	15.72	20.47	13.25
		σ	2.24	1.99	4.13	4.03
Ratio	MAE		0.22	0.17	0.26	0.21
	True	μ	1.04	1.03	1.19	1.00
		σ	0.29	0.29	0.36	0.35
	Test	μ	1.06	0.98	1.21	0.84
		σ	0.30	0.25	0.18	0.36

5.4. Accuracy results

For accuracy results, we use several metrics: (1) detection result (hit/miss) (2) find cell result (hit/miss) (3) size of cell (average radius) (4) respective ratio (ratio of long/short axis of cell.)

The *detection* output represents the result of determining cell/no-cell frames. The results in Table 3 show the true positive rate (*sensitivity*), true negative rate (*specificity*), ratio of true positives to number of positive predictions (*precision*), and true value rate (*accuracy*). They can be calculated as below.

- *Sensitivity* = (true positive)/(condition positive, or true positive + false negative)
- *Specificity* = (true negative)/(condition negative, or false positive + true negative)
- *Precision* = (true positive)/(test outcome positive)
- *Accuracy* = (true positive + true negative)/(total).

Sensitivity means how many valid frames the system can detect out of true valid cell frames, and *specificity* means the number of empty frames the system can find out of true no-cell frames. And high *precision* represents that there is a high probability that most of the correctly predicted frames have real cell features. Our system ignores some frames at the edges or cell blobs that are too blurry, and effectively screens unnecessary frames and assures the validity of cell frames. A high *precision* result here indicates more efficient performance with fewer wasted operations.

The Find cell result presents the correctness of cell location found. The next stage, cropping, cuts off cell focusing area based on this result. Table 4 shows the findcell result. It decides hit if a cell location is found within a certain boundary from a ground truth point, which is 5-pixels in Euclidean distance in this test.

We check the size of cells and respective ratios to evaluate tracing cellular wall results. Fig. 10 visualizes the tracing result examples using test cell images, and Table 5 shows mean absolute error (MAE), mean (μ) and standard deviation (σ) of cell size and respective ratios in each dataset.

Table 6
Throughput performance in detection and analysis modules of the hardware design pipeline.

	Detection	Analysis
Bottleneck latency (cycles)	4102	8287
Max clock frequency (MHz)	268	255
Max frame rate (FPS)	65.3k	30.8k

Table 7
Performance comparison in terms of throughput and latency for different platforms: Matlab, C, and FPGA.

	Matlab	C	FPGA
Throughput (FPS)	43.73	230.76	60.9k
Speed up	×1392	×263.9	N/A
Latency (ms)	22.87	4.33	0.068
Speed up	×335.6	×63.6	N/A

5.5. Performance results

In this section, we show our performance results on an FPGA and compare them to other design platforms. We have three different test platforms: Matlab, C, and FPGA.

In the FPGA design, we build two main processes which run independently: *detection* and *analysis*. Table 6 shows the maximum achievable frame rate of each module. The *detection* module deals with smaller images, so it spends less clock cycles than a bottleneck module and runs at a higher clock frequency. The *analysis* module consists of more submodules and processes more complex operations. The image size is larger in this module and the critical path is longer. Even though the maximum frame rate is less than *detection*, it does not affect the system performance since it is handling less frame data.

The performance of the entire design in terms of throughput and latency is represented in Table 7 across platforms. Latency is an average time to process one frame. Throughput is an inverse of latency and means the amount of frames processed in a second. The performance result of hardware is deterministic and predicable, which depends on the size of image, but software is not. Performance can differ across varying input. Our result is evaluated using the dataset introduced in Section 5.2.

When the FPGA design runs at a 250 MHz unified clock frequency, it is able to process 60.9k frames per second at the front end. This is a ×1392 speed up when compared to the Matlab design and is ×263.9 faster than the C-based software design. The latency result also shows a significant speed up. The hardware design takes 0.068 ms to process one frame on average, which is ×335.6 faster than Matlab and ×63.6 than the C-design.

We compare our performance achievement with our previous work [25] in Table 8. Even though the new image analysis process is enhanced and more complicated covering broader cellular image sets, it presents much faster throughput and latency performance against both FPGA and GPU implementations. Our new architecture filters valid cell frames quickly at the front end, and all bottleneck modules are balanced to achieve the target throughput as in Section 4.5. The new design results in about 30 times faster than an FPGA design in [25]. GPU design has a critical bottleneck in terms of latency because of data load operation. We utilize only on-chip memories for minimal latency, which is tiny but provides great memory bandwidth.

5.6. FPGA resource utilization

Table 9 shows the resource utilization results of our hardware design. The *detection* module consumes less than 1% of FPGA resources; 0.8% of LUTs and 0.3% of FFs. The *analysis* module uses 16.6% of BRAMs, 18.8% of LUTs, 7.72% of FFs, and 10.14% of DSPs.

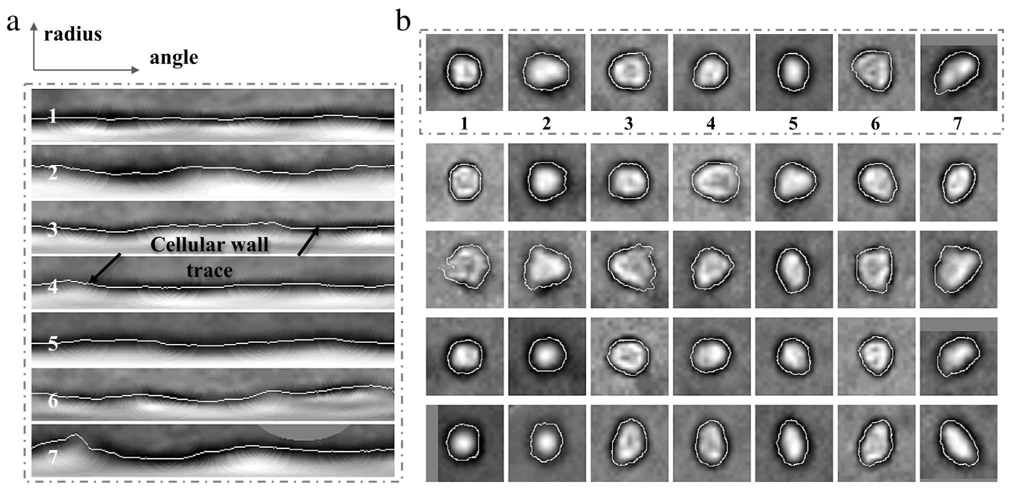


Fig. 10. Trace cellular wall results example (a) polar coordinate images with the trace of the cellular wall in white lines (b) cell images with corresponding trace lines.

Table 8
Performance comparison in terms of throughput and latency with our previous work in [25].

	Previous works [25]		This work
	GPU	FPGA	
Throughput (FPS)	1.32k	2.26k	60.9k
Latency (ms)	151.7	1.4	0.068
Frequency (MHz)	–	100	250

Table 9
Resource utilization in hardware design pipeline analysis. Utilization of *detection* and *analysis* modules and total utilization including PCIe connection.

	Detection	Analysis	Total
BRAM	0	328	390 (19.0%)
LUT	2653	45 242	58 533 (19.3%)
FF	1683	45 338	60 587 (9.98%)
DSP	0	299	299 (10.7%)

The entire design, including the PCIe API, consumes less than 20% of the resources overall.

Since the image analysis process runs with no dependency between frames, it can be scaled as much as the resources are available. The current design uses only 20% of resource on a target FPGA device, it is possible to add more processing element pipeline. If it scales the procedure multiple times, the frame rate could be more than the current 60k FPS up to few hundreds thousand frames per second. Our current goal in the algorithm is estimating the cellular feature accurately, but if the rest of resources could be also used for post processing after measuring cellular feature if necessary.

6. Conclusion

In this work, we developed and demonstrated a hardware accelerated cellular image analysis system. We designed an algorithm to analyze low resolution microscopic videos, and we created a custom hardware architecture to implement the algorithm on an FPGA. Our target setup is designed to extract cellular morphological features from a high speed camera. Our system meets the challenging performance requirements in terms of throughput as well as latency. Our system can handle video streams up to 60,900 frames per second and process each image with a 0.068 ms average latency. This provides the capability to perform real-time analysis and sorting of 2000 cells per second.

References

[1] M. Abkarian, M. Faivre, H.A. Stone, High-speed microfluidic differential manometer for cellular-scale hydrodynamics, Proc. Natl. Acad. Sci. USA 103 (3) (2006) 538–542.

[2] K. Asanovic, R. Bodik, B.C. Catanzaro, J.J. Gebis, P. Husbands, K. Keutzer, D.A. Patterson, W.L. Plishker, J. Shalf, S.W. Williams, et al., The landscape of parallel computing research: A view from berkeley, Tech. rep., Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.

[3] S.C. Bendall, E.F. Simonds, P. Qiu, D.A. El-ad, P.O. Krutzik, R. Finck, R.V. Bruggner, R. Melamed, A. Trejo, O.I. Ornatsky, et al., Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum, Science 332 (6030) (2011) 687–696.

[4] D.G. Buschke, J.M. Squirrell, H. Ansari, M.A. Smith, C.T. Rueden, J.C. Williams, G.E. Lyons, T.J. Kamp, K.W. Eliceiri, B.M. Ogle, Multiphoton flow cytometry to assess intrinsic and extrinsic fluorescence in cellular aggregates: applications to stem cells, Microsc. Microanal. 17 (04) (2011) 540–554.

[5] J. Che, V. Yu, M. Dhar, C. Renier, M. Matsumoto, K. Heirich, E.B. Garon, J. Goldman, J. Rao, G.W. Sledge, et al., Classification of large circulating tumor cells isolated with ultra-high throughput microfluidic Vortex technology, Oncotarget 7 (11) (2016) 12748–12760.

[6] K. Cheung, S. Gawad, P. Renaud, Impedance spectroscopy flow cytometry: On-chip label-free cell differentiation, Cytometry A 65 (2) (2005) 124–132.

[7] S. Coric, M. Leeser, E. Miller, M. Trepanier, Parallel-beam backprojection: an FPGA implementation optimized for medical imaging, in: Proceedings of the 2002 ACM/SIGDA Tenth International Symposium on Field-Programmable Gate Arrays, ACM, 2002, pp. 217–226.

[8] A.A. Dima, J.T. Elliott, J.J. Filliben, M. Halter, A. Peskin, J. Bernal, M. Kociolek, M.C. Brady, H.C. Tang, A.L. Plant, Comparison of segmentation algorithms for fluorescence microscopy images of cells, Cytometry A 79 (7) (2011) 545–559.

[9] J.S. Dudani, D.R. Gossett, T. Henry, D. Di Carlo, Pinched-flow hydrodynamic stretching of single-cells, Lab Chip 13 (18) (2013) 3728–3734.

[10] S. Gawad, L. Schild, P. Renaud, Micromachined impedance spectroscopy flow cytometer for cell analysis and particle sizing, Lab Chip 1 (1) (2001) 76–82.

[11] K. Goda, A. Ayazi, D.R. Gossett, J. Sadasivam, C.K. Lonappan, E. Sollier, A.M. Fard, S.C. Hur, J. Adam, C. Murray, et al., High-throughput single-microparticle imaging flow analyzer, Proc. Natl. Acad. Sci. 109 (29) (2012) 11630–11635.

[12] K. Goda, D. Di Carlo, B. Jalali, Ultrafast automated image cytometry for cancer detection, in: Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE, IEEE, 2013, pp. 129–132.

[13] D.R. Gossett, T. Henry, S.A. Lee, Y. Ying, A.G. Lindgren, O.O. Yang, J. Rao, A.T. Clark, D. Di Carlo, Hydrodynamic stretching of single cells for large population mechanical phenotyping, Proc. Natl. Acad. Sci. 109 (20) (2012) 7630–7635.

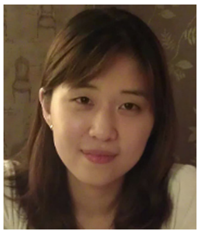
[14] H.E. Grecco, S. Imtiaz, E. Zamir, Multiplexed imaging of intracellular protein networks, Cytometry A (2016).

[15] P. Greisen, S. Heinzle, M. Gross, A.P. Burg, An FPGA-based processing pipeline for high-definition stereo video, EURASIP J. Image Video Process. 2011 (1) (2011) 1–13.

[16] J. Guck, S. Schinkinger, B. Lincoln, F. Wottawah, S. Ebert, M. Romeyke, D. Lenz, H.M. Erickson, R. Ananthakrishnan, D. Mitchell, et al., Optical deformability as an inherent cell marker for testing malignant transformation and metastatic competence, Biophys. J. 88 (5) (2005) 3689–3698.

[17] J.E. Hobbie, R.J. Daley, S. Jasper, Use of nuclepore filters for counting bacteria by fluorescence microscopy., Appl. Environ. Microbiol. 33 (5) (1977) 1225–1228.

[18] L. Hood, J.R. Heath, M.E. Phelps, B. Lin, Systems biology and new technologies enable predictive and preventative medicine, Science 306 (5696) (2004) 640–643.
[19] I. Ishii, T. Tatebe, Q. Gu, Y. Moriue, T. Takaki, K. Tajima, 2000 fps real-time vision system with high-frame-rate video recording, in: 2010 IEEE International Conference on Robotics and Automation, (ICRA), IEEE, 2010, pp. 1536–1541.
[20] M. Jacobsen, D. Richmond, M. Hogains, R. Kastner, RIFFA 2.1: A Reusable Integration Framework for FPGA Accelerators, ACM Trans. Reconfigurable Technol. Syst. (TRETS) 8 (4) (2015) 22.
[21] S. Jin, J. Cho, X. Dai Pham, K.M. Lee, S.-K. Park, M. Kim, J.W. Jeon, FPGA design and implementation of a real-time stereo vision system, IEEE Trans. Circuits Syst. Video Technol. 20 (1) (2010) 15–26.
[22] S. Kagami, S. Saito, T. Komuro, M. Ishikawa, A networked high-speed vision system for 1000-fps visual feature communication, in: 2007 First ACM/IEEE International Conference on Distributed Smart Cameras, IEEE, 2007, pp. 95–100.
[23] M.J. Kim, S.C. Lee, S. Pal, E. Han, J.M. Song, High-content screening of drug-induced cardiotoxicity using quantitative single cell imaging cytometry on microfluidic device, Lab Chip 11 (1) (2011) 104–114.
[24] S. Kumar, V.M. Weaver, Mechanics, malignancy, and metastasis: the force journey of a tumor cell, Cancer Metastasis Rev. 28 (1–2) (2009) 113–127.
[25] D. Lee, P. Meng, M. Jacobsen, H. Tse, D. Di Carlo, R. Kastner, A hardware accelerated approach for imaging flow cytometry, in: 2013 23rd International Conference on Field Programmable Logic and Applications, (FPL), IEEE, 2013, pp. 1–8.
[26] Y.-H. Lin, G.-B. Lee, Optically induced flow cytometry for continuous microparticle counting and sorting, Biosens. Bioelectron. 24 (4) (2008) 572–578.
[27] J. Matai, D. Lee, A. Althoff, R. Kastner, Composable, parameterizable templates for high Level Synthesis, in: Proceedings of the Conference on Design, Automation and Test in Europe, EDA Consortium, 2016.
[28] S.P. Perfetto, P.K. Chattopadhyay, M. Roederer, Seventeen-colour flow cytometry: unravelling the immune system, Nat. Rev. Immunol. 4 (8) (2004) 648–655.
[29] PhantomCamera [cited June 28, 2016]. URL https://www.phantomhighspeed.com/Products/Phantom-Camera-Products.
[30] I. Schmid, W.J. Krall, C.H. Uittenbogaart, J. Braun, J.V. Giorgi, Dead cell discrimination with 7-amino-actinomcin D in combination with dual color immunofluorescence in single laser flow cytometry, Cytometry 13 (2) (1992) 204–208.
[31] I. Schmid, C.H. Uittenbogaart, B. Keld, J.V. Giorgi, A rapid method for measuring apoptosis and dual-color immunofluorescence by single laser flow cytometry, J. Immunol. Methods 170 (2) (1994) 145–157.
[32] R.A. Schultz, T. Nielsen, J.R. Zavaleta, R. Ruch, R. Wyatt, H.R. Garner, , Hyperspectral imaging: a novel approach for microscopic analysis, Cytometry 43 (4) (2001) 239–247.
[33] V. Swaminathan, K. Mythreye, E.T. O'Brien, A. Berchuck, G.C. Blobe, R. Superfine, Mechanical stiffness grades metastatic potential in patient tumor cells and in cancer cell lines, Cancer Res. 71 (15) (2011) 5075–5080.
[34] The imagestreamx by amnis [cited June 28, 2016]. URL https://www.amnis.com/documents/brochures/ISX-MKII%20Brochure_Final_Web.pdf.
[35] H.T.K. Tse, P. Meng, D.R. Gossett, A. Irturk, R. Kastner, D. Di Carlo, Strategies for implementing hardware-assisted high-throughput cellular image analysis, J. Assoc. Lab. Autom. 16 (6) (2011) 422–430.
[36] R. Tsien, T. Rink, M. Poenie, Measurement of cytosolic free Ca 2+ in individual small cells using fluorescence microscopy with dual excitation wavelengths, Cell Calcium 6 (1) (1985) 145–157.
[37] J. Xu, N. Subramanian, A. Alessio, S. Hauck, Impulse C vs. VHDL for accelerating tomographic reconstruction, in: 2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, (FCCM), IEEE, 2010, pp. 171–174.
[38] F. Xue, A.B. Lennon, K.K. McKayed, V.A. Campbell, P.J. Prendergast, Effect of membrane stiffness and cytoskeletal element density on mechanical stimuli within cells: an analysis of the consequences of ageing in cells, Comput. Meth. Biomech. Biomed. Eng. 18 (5) (2015) 468–476.
[39] Y. Zheng, J. Nguyen, Y. Wei, Y. Sun, Recent advances in microfluidic techniques for single-cell biophysical characterization, Lab Chip 13 (13) (2013) 2464–2483.


Dajung Lee is currently a Ph.D. candidate in the University of California San Diego (UCSD) and is a system engineer in Intel since 2017 April. She received her bachelor degree (B.S.) in Electronic Engineering from Sogang University, South Korea in 2010 and her master degree (M.S.) in Electrical and Computer Engineering from UCSD in 2013. She was an engineer in Samsung Electronics before her graduate study and worked in IBM T.J. Watson Research Center as an intern in 2015. Her primary research interests are in hardware acceleration and system design at image processing, computer vision and machine learning.

Nirja Mehta joined UC San Diego in 2014, as a Graduate Student in Electrical and Computer Engineering Department. She received her B.Tech. in Electronics and Communication Engineering from Nirma University, Gujarat, India in 2013 and M.S. in Electrical and Computer Engineering with concentration in Signal and Image Processing under the advisement of Professor Ryan Kastner from UC San Diego in 2016. She currently works in Engineering Development Group at MathWorks.

Alexandria Shearer received her Bachelors in Computer Science and Engineering and a minor in Mathematics from Santa Clara University in 2013, was admitted to the UCSD Computer Science Ph.D. program, and recently earned her masters degree in Computer Science from UCSD. She was recognized with the Outstanding Computer Science and Engineering Senior Award in undergrad (2013). She is a Google Anita Borg Memorial Scholar (2012), a Eugene Cota Robles Fellow, and a National Science Foundation Graduate Research Fellow. She is now a Robotics Technologist at the Jet Propulsion Laboratory in Pasadena working on deep learning classifiers for vessel types on water.

Ryan Kastner is currently a Professor in the Department of Computer Science and Engineering at the University of California, San Diego. He received a Ph.D. in Computer Science from UCLA, a masters degree (M.S.) in Engineering and bachelor degrees (B.S) in both Electrical Engineering and Computer Engineering, all from Northwestern University. He is the Co-director of the Wireless Embedded Systems Master of Advanced Studies Program. He also co-directs the Engineers for Exploration Program. His current research interests reside in three areas: hardware acceleration, hardware security, and remote sensing.