High-speed FPGA-based Flow Detection for Microfluidic Lab-on-Chip

Liberis Voudouris, Calliope-Louisa Sotiropoulou, Nikolaos Vassiliadis, Athanasios Demiris and Spyridon Nikolaidis

Abstract—**Machine vision algorithms provide significant benefits for Lab-on-Chip (LoC) systems by automating the experimental process. This paper presents an FPGA-based machine vision flow detection implementation for microfluidic Lab-on-Chip (LoC) experiments. We propose and implement a novel architecture that exploits modern FPGA parallelism capabilities and makes efficient use of device resources to achieve real-time data collection in megapixel resolutions, at rates exceeding 30000 frames per second.**

I. Introduction

MICROFLUIDIC Labs-on-Chip offer significant advantages to clinical diagnostics by bringing the benefits of miniaturization and automation and enabling the execution of complicated biological experiments in an integrated environment at the Point-of-Care (PoC). Control of biological fluids through motion detection, often makes use of point sensors (light barrier, electromechanical, etc.), which have several disadvantages like reduced measurement resolution, limited information capabilities and higher cost. In recent years, various research groups have used machine vision solutions for information extraction and actuator control, which offer significant benefits compared to previous approaches, like increased precision and real-time response. In [1] Shin and Lee measure the kinetics of biomolecular interactions and perform droplet motion control in digital microfluidics, by using a software approach. Kornaros [2] proposes an FPGA-based multi-core soft-processor system for LoC microarrays, utilizing edge detection techniques, in [3] Dimantala et al. present a microfluidic system for DNA molecular analysis using machine vision for molecule detection and data acquisition, while Sapuppo et al. [4] have utilized an ad-hoc optical system to detect air bubbles in fluids circulating in microfluidic chips.

In this paper, we propose a novel FPGA-based machine vision system for real-time, concurrent detection of multiple flows in microfluidic LoC experiments. For each video frame, the system reports the coordinates of each fluid's front and back, and signals whether a flow passes from any user-defined points of interest. The retrieved data is then passed to the user and/or the LoC control unit, to extract information such as fluid speed and volume or to control the device's actuators.

The flow detection algorithm and software model were introduced by Micro2gen and patented in [5] and subsequently implemented in hardware by Aristotle University of Thessaloniki, after adaptation. The proposed implementation constitutes an integral part of a complete machine vision application presented by the authors in [6]. By efficient resource utilization and image space reduction, our implementation achieves real-time flow information retrieval, in high-frame rate high-resolution videos.

II. Algorithm Fundamentals

A. Video data retrieval

The flow detection system requires a frame-grabbing module to retrieve and store the (uncompressed) grayscale component of each video frame into a high speed external memory. Access to the frame data is done through a common memory interface that handles arbitration and data exchange. A memory word is 32-bits wide which facilitates transfers of 4 pixels/word.

B. Input file and parameters

Before retrieving the first video frame from memory, external initialization data must first be loaded into the system. An input file provides video and chip information, constant for the duration of each experiment, which includes information about video resolution, chip dimensions, flow and alarm point data. In addition, the system is fed with user adjusted parameters that fine-tune the algorithm's detection sensitivity, including threshold values and error margins, which are constant only for the duration of a single frame, giving the user the option to alter their value at run-time.

Manuscript received January 30, 2012. This work was partially supported by Hellenic Funds and by the European Regional Development Fund (ERDF) under the Hellenic National Strategic Reference Framework (ESPA) 2007-2013, according to Contract no. MICRO2-49-project LoC.

Liberis Voudouris is with the Department of Physics, Aristotle University of Thessaloniki, Thessaloniki, Greece (e-mail: lvoudour@auth.gr)

Calliope-Louisa Sotiropoulou is with the Department of Physics, Aristotle University of Thessaloniki, Thessaloniki, Greece (e-mail: lsoti@physics.auth.gr)

Nikolaos Vassiliadis is with Micro2gen Ltd., New Technology Park NCSR Demokritos, Ag. Paraskevi, Athens, Greece (e-mail: nivas@micro2gen.com)

Athanasios Demiris is with Micro2gen Ltd., New Technology Park NCSR Demokritos, Ag. Paraskevi, Athens, Greece (e-mail: dema@micro2gen.com)

Spyridon Nikolaidis is with the Department of Physics, Aristotle University of Thessaloniki, Thessaloniki, Greece (e-mail: snikolaid@physics.auth.gr)

978-1-4673-2529-5/12/$31.00 ©2012 IEEE

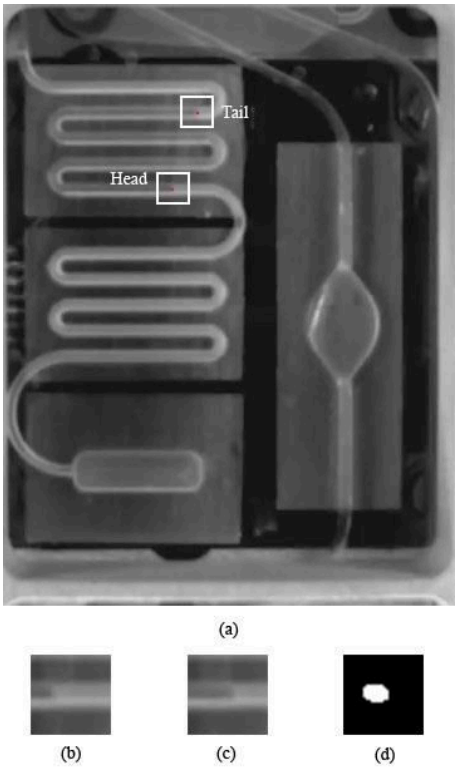

Figure 1. (a) Microfluidic chip (b) Current window frame (c) Previous window frame (d) Absolute difference with binary thresholding. The flow direction is from left to right (the fluid is the

C. Flow characterization and alarm points

Each fluid is characterized by its front and back (henceforth called "head and "tail" respectively - Figure 1a) and their associated coordinates. At each frame, the system must detect and report the current coordinates of each flow's head and tail. The user can pre-define a set of points of interest, called alarm points, which are stored in the input file. Whenever a flow's head or tail crosses an alarm point, a signal is raised informing about the event along with an index indicating the particular alarm point

D. Detection window

The input file holds a value indicating the dimension of a square detection "window". The detection window defines a square area around the flow's head and tail, in which fluid movement is expected on the next frame. Window dimension is a function of the fluids' maximum speed, the microfluidic chip's minimum channel width and distance between channels. Using a detection window eliminates the need to examine the entire frame, minimizing data processing and storage and thus, increasing performance and decreasing area requirements. To avoid detection conflicts from window overlapping, we assume that a flow's tail does not enter until the head has moved a distance at least equal to the window dimension, which is a safe assumption for most applications.

III. Proposed Implementation

A. Current frame loading

At the start of each frame, the pixel data defined by each flow window's area must be loaded from memory for further processing. A detection window is a rectangle defined by its width, height and (x,y) coordinates of its upper-left corner and all window rectangles are stored in two internal RAMs, separate for the heads and tails. Although a window is initially supposed to be square, for flows near the frame's bounds it gets cropped. Furthermore, due to the requirement of 4 pixels/word, the window's width is always adjusted to the next multiple of 4. The window's rectangle is then passed to the memory controller which fetches the respective pixels from the current frame stored in memory.

B. Frame subtraction

Pixel data are stored in an internal BRAM, intended for current frame storage, as they are fetched from memory and are simultaneously compared to the contents of the previous frame. The comparison is done by taking the absolute difference of a pixel's value between the two frames and comparing it against a binary threshold (Figure 1b, 1c, 1d). Thresholding output is a 1-bit value indicating if a difference was found or not. The binary threshold is a user supplied parameter and controls noise tolerance and detection sensitivity. The binary pixels are fed through a pipelined datapath to the flow detection stage.

C. Flow detection

Flow detection is performed on the binary pixels of the subtraction stage, by locating the geometric center (or barycenter) of the "active" pixels:

$$\mathbf{r_c} = \frac{1}{N} \cdot \sum_{N-1}^{0} p_i \cdot \mathbf{r}_i, \quad p_i \in \{0,1\} \tag{1}$$

where N is the total number of pixels in a window rectangle. This can be reduced to:

$$\mathbf{r_c} = \frac{1}{N_{wp}} \cdot \sum_{N_{wp}-1}^{0} \mathbf{r}_i \tag{2}$$

where $N_{wp} = \sum_{N} p_i$, $p_i \in \{0,1\}$ is the number of active pixels. In terms of hardware, since pixels are fed in batches of 4, coordinates are accumulated in parallel. A counter counts the number of active pixels and at the end of the accumulation process, if the number of active pixels exceeds a threshold value $N_{wp} > T_{flow}$, a fixed-point divider performs the final division, otherwise no change in fluid position is indicated. The threshold value (called flow detection threshold) is another user adjusted parameter that enables fine-tuning of detection sensitivity. The barycenter coordinates represent the current flow coordinates, which are output to the LoC

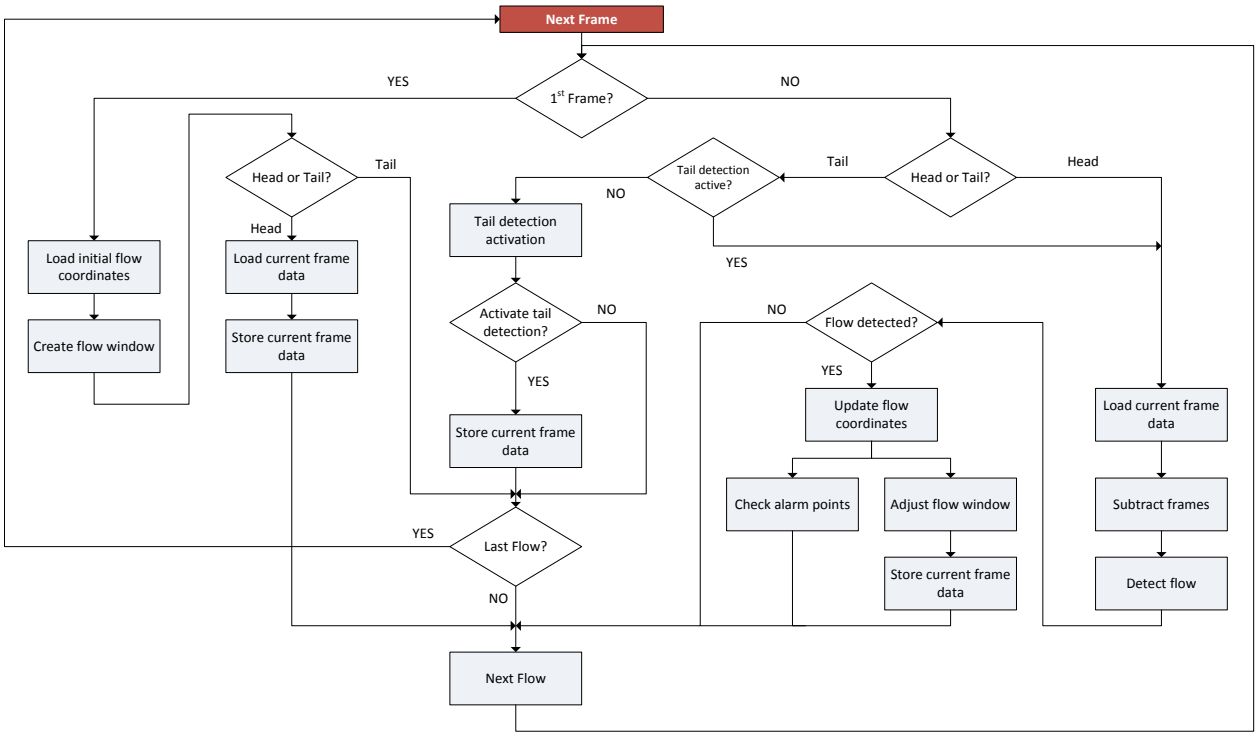

Figure 2. Flowchart of the per-frame operation of the proposed algorithm implementation.

control unit.

D. Alarm point detection

The detected flow coordinates must be checked against the predefined set of alarm points, to determine if any of them is crossed by the liquid. Each alarm point's coordinates are fetched from the input file and compared to the current flow coordinates. If their absolute difference is within a user pre-defined error margin (that controls alarm detection sensitivity), an alarm is signaled.

E. Window rectangle adjustment

If fluid movement was detected, the corresponding window rectangle must be re-adjusted around the new flow coordinates. This is done to ensure that the flow is centered and won't escape from the detection window on a subsequent frame. For the first frame only, the window is adjusted (or rather created) prior to the initial frame loading, using the initial coordinates of a fluid's entry point - which are stored in the input file - and pixels are fetched and stored unconditionally in the previous frame storage BRAM.

F. Current frame storage

After the window rectangle is adjusted, the pixels corresponding to the new window are fetched and stored in an internal BRAM intended to serve as previous frame data for comparison on the next frame. If a flow was not detected (and thus the window rectangle is unaltered) no new data needs to be fetched and stored. This way, minuscule flow of a slow moving fluid that passes undetected, is accumulated over several frames until movement is finally discernible.

The proposed algorithm operation is summarized in the flowchart of Fig. 2.

IV. Performance and Area Analysis

System performance is not only affected by implementation, but is strongly data and parameter dependent. A good metric to assess performance is to identify worst case completion time per frame (t_{cpf}). The worst case completion time appears when fluid movement is detected. Since alarm point detection and window rectangle adjustment (and storage) operate concurrently, we need to identify which path is the most time consuming. The window adjustment / storage path takes

$$t_w = 2 \cdot t_{clk} \cdot n_f \cdot (\frac{d^2}{2} + 2c_{mr} + 46) \text{ [sec]} \quad (3)$$

where d is the detection window dimension, c_{mr} the memory latency, t_{clk} the clock period and n_f the number of simultaneous flows. The alarm point path requires:

$$t_a = 2 \cdot t_{clk} \cdot n_f \cdot (\frac{d^2}{4} + 4n_a + 39) \text{ [sec]} \quad (4)$$

where n_a is the number of alarm points. The number of alarm points beyond which $t_a > t_w$ is

$$n_a > 0.0625 \cdot d^2 + 0.5 \cdot c_{mr} + 1.75 \quad (5)$$

Completion time per frame along with detection window dimension strongly affect the maximum fluid speed that can be detected. For a given completion time and window

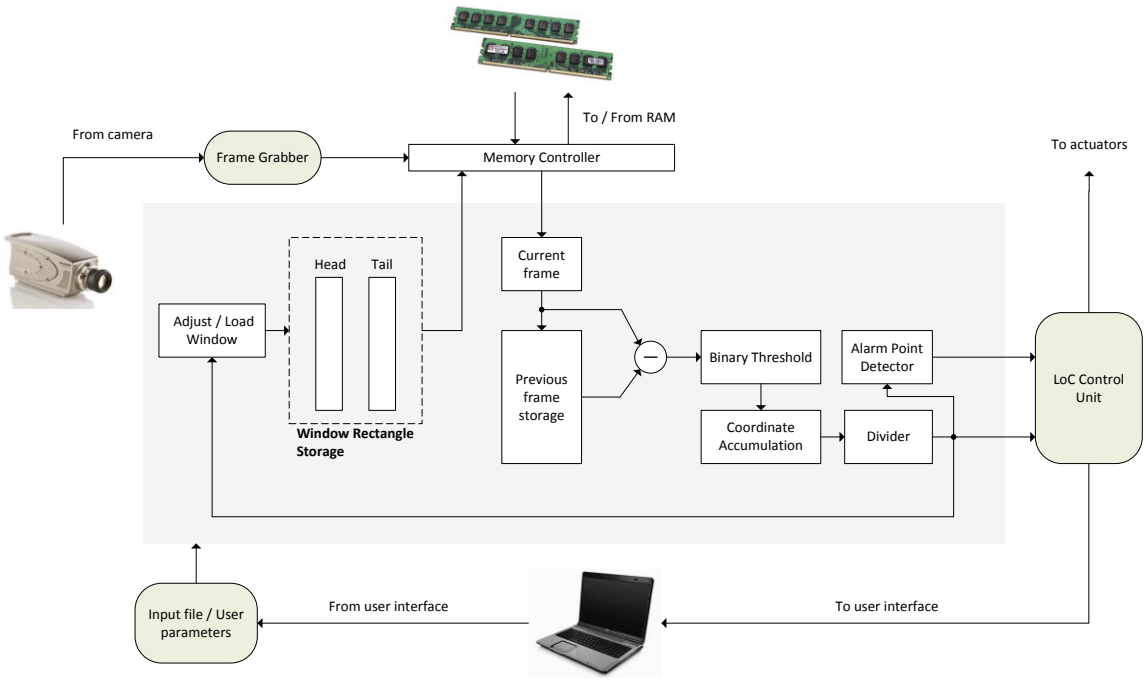

Figure 3. Machine vision flow detection system for microfluidic Lab-on-Chip.

dimension, the maximum fluid speed that can be measured is

$$u_{\max} = \frac{r_p \cdot (d-2)}{2 \cdot t_{cpf}} \left[\frac{mm}{\sec}\right] \quad (6)$$

where r_p is the mm/pixel resolution of the video and t_{cpf} the completion time per frame. Circuit area is mainly affected by the amount of internal storage required. Internal storage is comprised by the two LUT-based RAMs used to hold head and tail detection windows, the BRAM used to hold the current frame window and the BRAM that stores previous frame windows. The size used for each LUT based RAM is

$$M_{LUT} = 2 \cdot n_f \cdot \left(\lfloor \log_2(F_w - 1) \rfloor + \lfloor \log_2(d-1) \rfloor + 2\right) \text{[bits]} \quad (7)$$

where F_w is the video frame width. Current frame window BRAM size is

$$M_{CF} = 8 \cdot d^2 \text{ [bits]} \quad (8)$$

and previous frame window BRAM size is

$$M_{PF} = 16 \cdot n_f \cdot d^2 \text{ [bits]} \quad (9)$$

V. Experimental Results

An overview of the proposed system is depicted in Fig. 3. A frame grabber retrieves video data from the camera and stores them into an external RAM through a memory controller. Through the same controller the flow detection system reads current frame data into the system. For each frame the system outputs flow information to the LoC control unit, which controls the actuators of the system and sends data to the PC-based user interface. The PC interface also serves as a means for the user to control the properties and behavior of the flow detection system, through the input file and user defined parameters.

The proposed system was simulated, and implemented for a Xilinx Virtex-5 (XC5VLX110T-FF1136) device, in a highly parametric fashion covering a variety of input parameters. Simulation testbenches evaluated the system's algorithmic behavior over several video inputs and parameter scenarios, measuring detection accuracy, as well as time to completion per frame.

To verify the correct functionality of the system and obtain performance results, an evaluation setup was implemented using an XUPV5 FPGA development board by Digilent Inc., built around the aforementioned Xilinx FPGA The evaluation setup is depicted in Fig. 4. Starting from a recorded video of an operating microfluidic LoC stored in a PC, individual video frames are extracted, using the free software ffmpeg [7]. The extracted frames are then stored in a compact flash memory card and the card is connected to the FPGA board. An embedded Xilinx Microblaze processor, which serves as the system controller, retrieves the frames from the memory card and stores them in the external RAM of the FPGA board through a Xilinx Multiport Memory Controller (MPMC), emulating the camera/frame-grabber chain. Utilizing the same memory controller through a Video Frame Buffer Controller (VFBC), the implemented flow detection system retrieves and processes the frame data, feeding the results to the processor, which through a UART serial interface sends the processed data back to the PC. A custom on-screen display (OSD) utility created for the purposes of this evaluation setup,

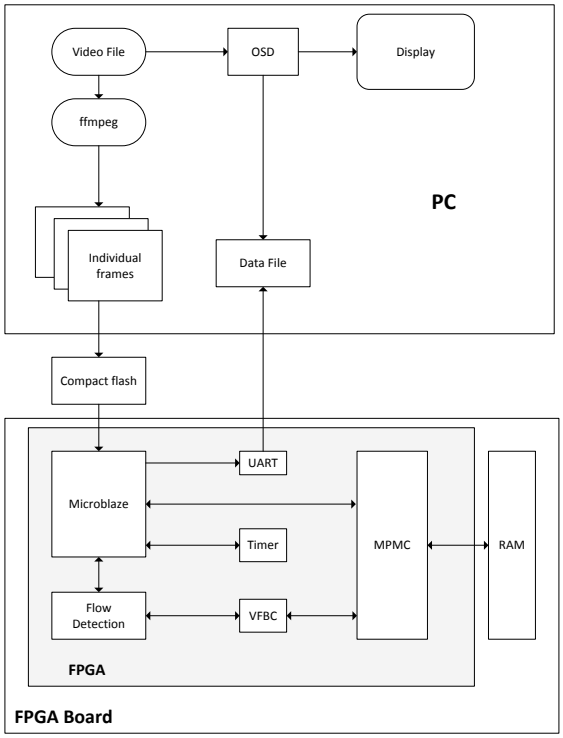

Figure 4. Performance and functionality evaluation setup of the proposed system on an FPGA board..

retrieves the data (flow positions and alarm point crossings) and, for each frame, overlays them on the recorded video file.

To obtain performance metrics a watchdog timer is connected to the embedded processor which keeps track of the flow detection system's completion time. The processor operates at 100MHz and thus a separate, faster clock is needed to drive the flow detection system, for optimal performance. The second clock is fed through an internal clock manager provided by the FPGA chip. The timer operates at processor speed, with a resolution of 10 ns, which is more than adequate to evaluate the performance of our system.

Performance results for various parameter values are presented in Table I. Due to the use of detection windows that render frame size irrelevant, video resolution has no effect on performance and little effect on utilized area. The

TABLE I. PERFORMANCE RESULTS

Parameters			Performance metrics	
d	n_a	n_f	t (μs)	fps
16	128	5	26.94	20254
16	256	5	49.37	10012
16	512	5	99.88	6635
24	128	5	150.70	10612
24	256	5	94.24	5246
24	512	5	190.62	3477
32	128	5	287.64	32843
32	256	5	30.45	16236
32	512	5	61.59	10760

nf: number of flows, **na**: number of alarm points, **d**: detection window dimension, **t**: completion time, **fps**: frames per second

TABLE II. DEVICE UTILIZATION OF A. VIRTEX 5 (XC5VLX110T-FF1136) DEVICE

LUT	FF	BRAM	DSP	Frequency (MHz)
1581	1447	6	1	228.258

number of alarm points (n_a) used exceeds by far the limit in (5) and thus completion time is defined by alarm point detection delay. It may seem extreme to use 512 or even 256 alarm points in normal operation, but these unlikely numbers serve to test and evaluate the capabilities of the proposed system. The number of concurrent flows is set to 5, which is by no means an upper limit to the maximum concurrent flows in microfluidic LoC, but it is more than adequate to extract sufficient results about the robustness of the algorithm.

Table II presents FPGA device utilization for an indicative set of parameters: W=1024, H=1024, d=16, n_a=128 and n_f=5. The implemented system occupies a very small area, making efficient use of the underlying fabric. The number of alarm points does not affect on-chip memory utilization and due to fluid speed limits, the window detection size is unlikely to grow beyond 32x32 pixels. Thus, on-chip memory usage will scale more than often with the maximum number of concurrent flows.

The obtained timing and area results are in accordance to the theoretically derived numbers, something that enables us to accurately pre-calculate the parameters needed to tune the system in compliance to given specifications. Performance-wise, the proposed system achieves very high frame rates for a variety of configurations, even exceeding 30000 frames per second. This means that, for example, at a resolution rp=0.001 mm/pixel, we can accommodate fluid speeds of 300 mm/sec or more.

VI. CONCLUSION

In this paper, an FPGA-based machine vision flow detection system for microfluidic Lab-on-Chip experiments, was proposed and implemented, supporting multiple simultaneous high-speed fluids, a first to our knowledge in relevant literature. An implementation test setup was designed to evaluate the performance and functionality of the system on an XUPV5 FPGA development board, using a Virtex-5 FPGA device. The obtained results clearly demonstrate the high speed detection capabilities and robustness of the system, with frame rates approaching and exceeding 30kfps for common parameter sets, providing significant performance and automation advantages to molecular diagnostics applications. The small area footprint and minimal on-chip memory utilization, provide adequate space for larger systems to be implemented on the same chip.

REFERENCES

[1] Y.-J. Shin, J.-B. Lee, "Machine vision for digital microfluidics," Rev. Sci. Instrum., vol. 81, no. 1, Jan. 2010, doi: 10.1063/1.3274673

[2] G. Kornaros, “A soft multi-core architecture for edge detection and data analysis of microarray images,” Journal of Systems Architecture - Embedded Systems Design, (56) 1: 48-62, Year 2010

[3] E. Dimalanta et al., “ A microfluidic system for large DNA molecule arrays,” Analytical Chemistry, Vol. 76, No. 18, pp. 5293-5301, September 15, 2004

[4] F. Sapuppo, M. Intaglietta, M. Bucolo, "Bio-Microfluidics Real-Time Monitoring Using CNN Technology," Biomedical Circuits and Systems, IEEE Transactions on , vol.2, no.2, pp.78-87, June 2008

[5] A. Demiris, S. Blionas, "Integrated System for the Visual Control, Quantitative and Qualitative Flow Measurement in Microfluidics," Hellenic Industrial Property Organisation Patent 20110100390, July 7, 2011

[6] C.-L. Sotiropoulou et al., "FPGA-based Machine Vision Implementation for Lab-On-Chip Flow Detection," Circuits and Systems (ISCAS), 2012 IEEE International Symposium on, pp.2047-2050, 20-23 May 2012.

[7] FFmpeg [Online], Available: http://www.ffmpeg.org