2017 International Conference on Computer, Communications and Electronics (Comptelix)
Manipal University Jaipur, Malaviya National Institute of Technology Jaipur & IRISWORLD, July 01-02, 2017

Accelerating Real-time Computer Vision Applications using HW/SW Co-design

Anirudh B.K., Vivek Venkatraman, Abhishek Rathan Kumar* and Sumam David S.
Department of Electronics and Communication Engineering
National Institute of Technology Karnataka Surathkal, Mangalore 575025 INDIA
* Email: abhishek.94@hotmail.com

***Abstract*—Video processing applications have become increasingly difficult to implement on hardware, owing to the complex computer vision algorithms involved. This paper presents a real-time video processing architecture based on hardware/software co-design that improves execution speed and reduces the time to market of applications. We have implemented this framework for handwritten digit recognition on the Zybo Zynq-7000 ARM/FPGA SoC using Vivado High Level Synthesis (HLS) and Xillybus tools. Histogram of Oriented Gradients (HOG) feature extraction algorithm has been optimised for hardware execution and acceleration techniques have been applied on Vivado HLS to achieve a speed up of 38.89 for the HOG algorithm and recognition accuracy of 95.6%. Low precision arithmetic along with our approximations for costly functions, produced this significant gain in throughput by reducing 90% of the hardware resources required with just a marginal accuracy reduction by 1%. An overall performance improvement of 77% is obtained through hardware/software co-design over software execution. The framework identified digits seamlessly in a real-time video stream at 30 frames per second and enabled high frame rate video processing.**

***Keywords*—real-time video processing, hardware/software co-design, HOG, Xillybus, Vivado HLS**

I. INTRODUCTION

Embedded computer vision applications are on the rise in all fields, ranging from medical and space technology to multimedia and security. These applications involve large number of computations for digital processing and decision making, making them slow to run on software. However, implementing them completely on hardware requires a lot of resources and power. The advent of lost-cost and power-efficient processors and FPGA has made the realisation of these applications possible through hardware/software co-design [1].

Digit recognition is a simple computer vision problem that involves image processing as well as machine learning. We have used the hardware/software co-design approach to develop a real-time handwritten digit recognition system, to evaluate the benefits of this approach over pure software and pure hardware implementations. LeCun et al. have explored several classification methods [2] such as K-nearest neighbours (k-NN), support vector machines [3], neural networks and convolutional neural networks. The classifiers either work with image intensities or features obtained through Scale Invariant Feature Transform [4] that uses Difference of Gaussians (DoG) operator to identify keypoints, Histogram of Oriented Gradients [5] that uses gradient operator, GIST [6] that employs Gabor filters to obtain the *gist* of the image or several other methods.

Our application computes Histogram of Oriented Gradients (HOG) features and uses a Linear Support Vector Machine (SVM) to classify digits. Gradient, being a difference operator, is unaffected by changes in bias. Since it is a local operator, it is also invariant to geometric transformations, except rotation, and it can be run in parallel on all pixels. It is also simple to implement on hardware. The cost function of the Linear SVM is a dot product and hence easy and quick to compute on hardware. It does not involve non-linear functions as in the case of k-NN or neural networks and does not require large memory to store the function values as a look-up table.

For real-time implementation, the algorithms may require more time for execution than the desired frame interval time for processing. In such cases, software implementation would not produce a seamless video output. There have been several approaches to implement video processing systems on hardware to exploit inherent spatial and temporal parallelism [7]. Image processing algorithms that involve pixel-level operations can be efficiently implemented using GPUs and SIMD architectures [8],[9]. Several efforts have gone into designing vector co-processors with specific instruction sets [10],[11]. However, the full power of these architectures cannot be harnessed when algorithms involve pixel-overlapped operations and decision making. An FPGA, on the other hand, can perform such operations extremely fast [12]. FPGAs also have an edge over co-processors due to their ability to support custom logic and capability to run parallel operations [13]. The rise in machine learning algorithms being run on images and videos to extract useful information has restricted their complete implementation on hardware due to high amount of information caching, hardware resources and power required for complex processing [14],[15]. This necessitates the use of both hardware and software for real-time decision making systems.

In our design approach, the sequential flow of the application is maintained by the software and the parallelizable portions of the application are implemented on hardware. Instruction and data level parallelism is exploited in the Programmable Logic and thread level parallelism is realized on the Processing System. Such a hardware-accelerator system provides a much faster execution, despite the communication overhead between the processor and the hardware logic. It also provides the liberty to develop software and hardware portions of the application concurrently, reducing the development time.

978-1-5090-4708-6/17/$31.00 ©2017 IEEE

Section II of the paper describes the handwritten digit recognition algorithm used – the pre-processing steps, Histogram of Oriented Gradients algorithm, its hardware adaptation and Linear SVM classifier. Section III elaborates on the implementation, system architecture and development flow. Results are elaborated in Section IV.

II. HANDWRITTEN DIGIT RECOGNITION

The steps involved in the application are shown in Figure 1. and described in the following sub-sections.

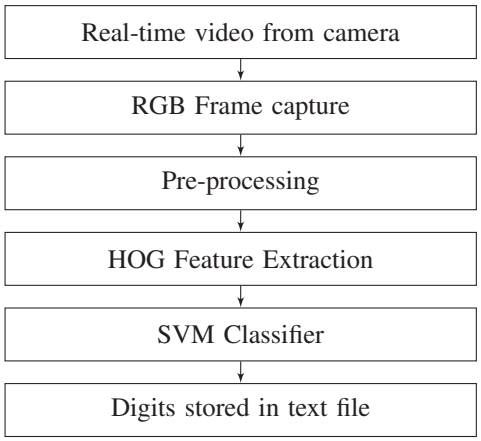

Fig. 1: Handwritten digit recognition application

A. Pre-processing

Pre-processing brings images to a standard format suitable for feature extraction. The pre-processing steps used are shown in Figure 2.

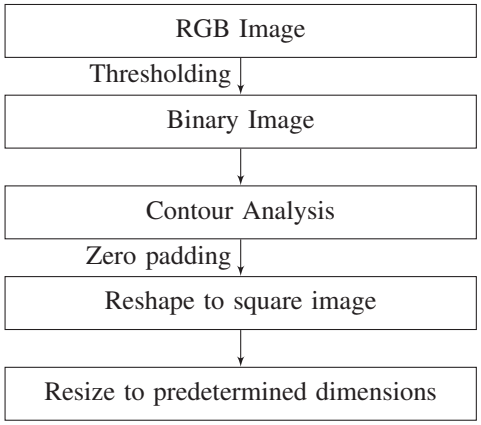

Fig. 2: Pre-processing steps

Multiple contours may exist for digits, in which case the outer contours are considered as they depict the digits better. Each digit's bounding box coordinates are stored so that digits can be individually and independently processed. Zero-padding followed by scaling is chosen over direct resizing to maintain the aspect ratio of the digit. Pre-processing for digit 4 is shown in Figure 4.

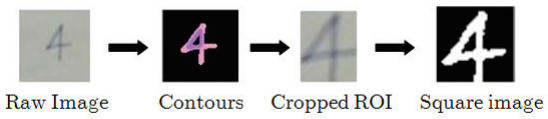

Fig. 4: Pre-processing steps for digit '4'

B. Feature Extraction – Histogram Of Oriented Gradients (HOG)

The HOG features extract gradient magnitude and orientation from images to estimate the strength and direction of edges and recognise shapes. Gradients are used as features because they are invariant to changes in bias (lighting and camera changes). The algorithm [5],[16] is shown in Figure 5.

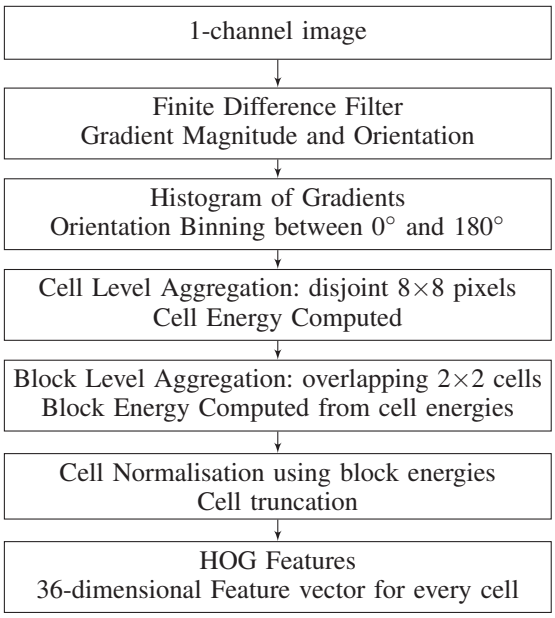

Fig. 5: Histogram of Oriented Gradients Algorithm

The HOG algorithm contains complex calculations that would require specialized hardware and long execution time. The algorithm can be adapted for faster and optimized hardware execution [17] through some approximations as shown in Table I.

- dx and dy represent horizontal and vertical gradients respectively.
- $\theta(x, y)$ represents the gradient orientation and i denotes the orientation bin it falls in. The histogram is computed with the angle bins being multiples of $20°$. So the comparison of the tangents is done for $\theta_i = 10°, 30°, 50°, \ldots, 170°$. Since tangent function is symmetric about $90°$, the *dx* located at the second quadrant can be mapped into the first quadrant by *dx* = *–dx*.
- x_{IEEE754} is the IEEE754 floating point representation of x. Decimal$\{h\}$ is the decimal representation of the hexadecimal h.

TABLE I: Hardware Acceleration of HOG

Operation	Implementation	Hardware Approximation
Gradient Magnitude	Root Mean Square $\sqrt{dx^2 + dy^2}$	$max\Big(a - (a >> 3) + (b >> 1),\ a\Big)$ $a = max(dx, dy),\ b = min(dx, dy)$
Orientation Binning	$\theta(x, y) = tan^{-1}\dfrac{dy}{dx}$	$dx \times \tan\theta_{\mathrm{i}} \leq dy \leq dx \times \tan\theta_{\mathrm{i+1}}$
Block Normalisation	Inverse square root $\dfrac{1}{\sqrt{x}}$	$y_d\left(\dfrac{3\ -\ x\ {y_d}^2}{2}\right)$ $y_d = Decimal\{\ (x_{\_\mathrm{IEEE754}} >> 1) - 0x5F3759DF\ \}$

C. Classification -– Linear SVM

A linear SVM classifier uses utmost a *N-1* dimensional hyperplane to separate *N* dimensional data points. Digit classification is a multi-class SVM classification problem that can be implemented using a one-versus-all approach. The image is subjected to ten two-class SVM classifiers, one for each digit. The digit for which the classification provides the lowest cost is considered as the predicted label.

III. IMPLEMENTATION

A. Software Implementation

The MNIST Database is used for training and testing samples of single handwritten digit data [18]. We have used 1000 random handwritten images of each digit (out of 6000 available). Each image is of size 28×28 in grayscale format. They are of various orientations, thickness and some images are incompletely cropped.

The handwritten digit recognition application is developed using OpenCV. OpenCV Library provides a class called 'HOGDescriptor' that computes the HOG features [21]. The class contains the 'compute' function that takes a 2-dimensional image as input and gives the HOG feature vectors as output. The HOG window size is set to 128×128, block size is 16×16, cell size chosen as 8×8 and 9 contrast-insensitive bins are taken for our application. OpenCV SVM classifier is trained with MNIST images and support vectors are stored in an XML file. Training is done on an independent machine and the classifier is loaded into the SD card and can be used during testing and for the real-time application on the Zybo board.

Detection of a user writing on a paper is done by identifying the hand through blob analysis. When the digits are no longer occluded by the hand, the camera captures a sequence of frames and passes the averaged image for pre-processing. The individually cropped digits are tagged based on their position in the paper and stored in the order in which they appear. Spaces in between the numbers is identified and they are stored as separate words. Numbers written in different lines are stored in separate lines to keep the structure of the writing intact.

Code profiling is done to analyse the time spent in executing and memory used by different parts of the program. The profiling results indicated that the Histogram of Oriented Gradients function has great scope for improvement in terms of execution time. Thus, a hardware/software co-design approach is explored to achieve this speedup.

B. Xillinux

Xillinux is a Linux-based operating system designed by Xillybus for the Xilinx Zynq-7000 based SoC boards [22]. It has special communication modules like drivers that run on the ARM Cortex A7 processor to interface peripherals like Camera (through USB) and Monitor (through HDMI and VGA). It instantiates Xillybus IP Core on the Zynq FPGA to link the Processing system (PS) and the Programmable

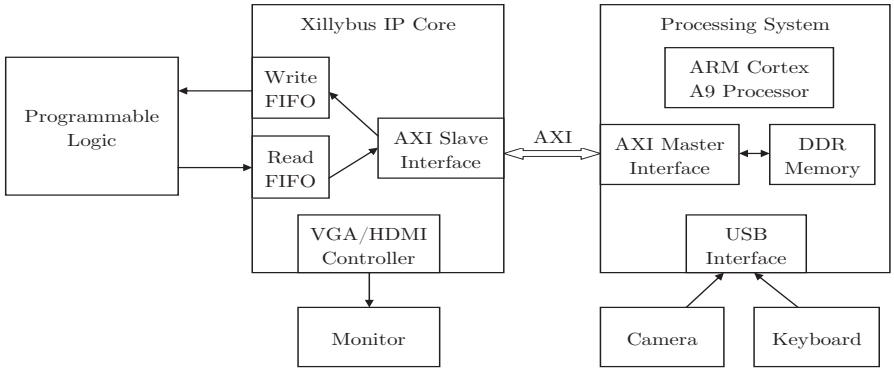

Fig. 3: System Architecture

Fig. 6: Xillinux setup

Logic (PL) through the AXI Interface. Data can be read and written byte-wise and word-wise from the FPGA to the AXI interface through a pair of FIFOs [23]. This hardware/software co-design system is simple to setup and use, leading to much faster hardware-based application development. The architecture flow is shown in Figure 3.

C. Development Flow

Xilinx provides the Vivado High-Level Synthesis(HLS) tool that enables C and C++ programs to be directly converted to synthesizable HDL files without the need to manually create RTL [19]. The development flow is shown in Figure 7. The algorithm to be implemented on hardware is written in C++. Vivado HLS converts this code into a set of Verilog/VHDL files. The new bit file generated by the tool is written into the SD card. The board is then booted up with the SD Card and the application is run on the processor which can call the synthesized hardware when required [23].

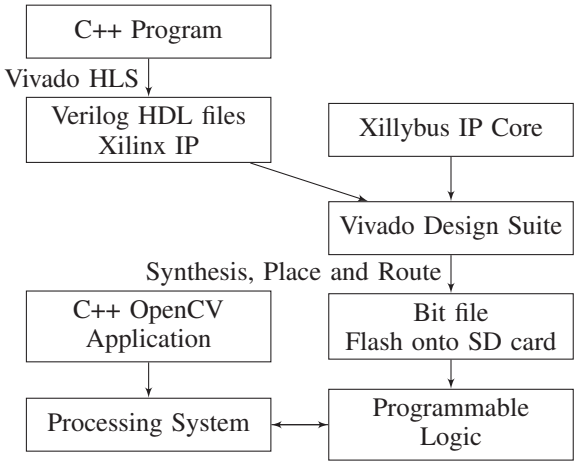

Fig. 7: Development Flow

D. Hardware Acceleration

The following acceleration methods are also used to improve execution speed:

- Loop unrolling: Exploits instruction level parallelism to reduce control hazard stalls. *#pragma HLS unroll factor* directive[24] is used to unroll the loop and reduce branch mispredictions and stalls.
- Pipelining: Increases throughput by running all of the existing hardware path in parallel. Vivado HLS provides the *#pragma HLS PIPELINE* directive [24] to pipeline concurrent statements a sequential block, such as performing pixel level operations, that can be done in parallel.
- Bit Packing: Four 8-bit numbers are packed into 32-bit to reduce communication overhead by a factor of 4.
- Multi-threading: The speedup is also increased by using two threads on the dual core ARM Cortex A9 processor present on the Zybo Board. One thread initiates performs pre-processing, sends the pixel values to the hardware for HOG extraction and then starts the same procedure on the next image. The other thread waits for the hardware to return the features and classifies the digit.

IV. RESULTS

The handwritten digit recognition application is implemented completely on OpenCV and run on the ARM processor on the Zybo. This is compared with the hardware/software co-design implementation on the Zybo.

A. Execution Time

We have used the GNU gprof profiler [25] to measure execution time. The time resolution of the profiler is 0.01 seconds. The profiling results obtained when the same set of 20 digits are classified using the two approaches is shown in Table II.

TABLE II: Performance of HW/SW Co-design

Application component	Execution Time (seconds)	
	SW implementation	HW/SW co-design
HOG feature extraction	0.56	0.02
Pre-processing	0.08	0.08
SVM classification	0.06	0.06
Total execution time	0.70	0.16

The timing parameters of the Vivado HLS synthesized design subject to device constraints are shown in Table III.

TABLE III: Timing results

Timing Parameters	Values
External clock frequency of Zybo	125 MHz
Maximum frequency of operation	126.9 MHz
Initiation interval	90672 clock cycles
Processing time for one digit	0.72 ms

For 20 images, the execution time would amount to 0.014 seconds. The profiler result of 0.02 seconds includes

the communication overhead, which would be about 0.006 seconds.
HOG execution on software $= 560ms$
HOG execution on hardware $= 0.72ms \times 20 = 14.4ms$
HOG speedup $= 38.89$
Overall speedup $= 0.70/0.16 = 4.375$ (77.14% faster)
According to Amdahl's Law, the overall theoretical speedup *S* due to a speedup *s* in a fraction *f* of the algorithm is given by

$$S = \frac{1}{1 - f + \frac{f}{s}}$$

Here, theoretical speedup $= \frac{1}{0.2 + \frac{0.8}{38.89}} = 4.537$

The decrease in speedup during practical implementation can be attributed to data-sharing time between the PS and PL.

Speedup provided by individual algorithmic optimisations after hardware synthesis is measured on Vivado HLS. *hls_math.h* provides synthesizable *sqrt()* function for gradient magnitude and inverse square root functions, and *atanf()* function to compute inverse tangent through CORDIC module [24]. The execution times using these functions v/s the optimized functions for 1 computation is shown in Table IV.

TABLE IV: Individual speedup due to different algorithmic optimisations

HOG component	Execution Time (ns)		Speedup
	HLS Library	Optimized	
Gradient Magnitude	236.06	10.56	22.35
Orientation Binning	1867.25	387.03	4.82
Block Normalisation	760.5	141.68	5.37

From Table II, average decode time for a digit and the maximum frame rate for video sampled such that background subtraction yields single new digit is estimated as follows:
Decode time in SW approach = $0.70/20 = 0.035s$
Maximum video capture rate = $28.57fps$
Decode time in HW/SW approach = $0.16/20 = 0.008s$
Maximum video capture rate = $125fps$

B. Accuracy

For testing the accuracy of the SVM classifier, 700 images for every digit are used in training and testing is done using the remaining 300 images. Precision, Recall and F-measure [20] for classification of individual digits are shown in Table V. Digit 8 has a lower recall of 0.86 as an incomplete 8 may be decoded as a 4 or one with a larger loop than the other may be decoded as a 0. The accuracy of prediction has gone down by less than 1% due to hardware approximations as shown in Table VI.

- Gradient Magnitude Approximation:

  The horizontal and vertical gradient components can vary from -255 to 255. The error in approximating the square and square root operations is:
  Average absolute error = 1.90745
  Average relative error = 0.98%

- Normalisation Approximation:
  The error plot for inverse square root approximation used in block normalisation is shown in Figure 8. The error is about 0.16 for 10^{-4} and is less than the order of 10^{-4} for values greater than 1. The cell energies are of the order of 10^6; hence this approximation works very well.

TABLE V: Classification performance for single digits with OpenCV HOG Library

Digit	Precision	Recall	F-measure
0	0.96	0.98	0.97
1	0.97	0.99	0.98
2	0.95	0.95	0.95
3	0.95	0.94	0.94
4	0.95	0.97	0.96
5	0.96	0.96	0.96
6	0.97	0.98	0.97
7	0.95	0.98	0.96
8	0.96	0.86	0.91
9	0.95	0.96	0.96

TABLE VI: Classification performance comparison

Implementation	Average Precision	Average Recall	Average F-measure
OpenCV HOG Library	0.968	0.963	0.965
Hardware optimized HOG	0.957	0.956	0.956

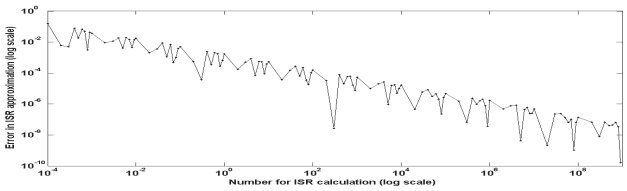

Fig. 8: Error in gradient Inverse Square Root (ISR) computation

C. Resource Utilization

The HOG algorithm has been adapted to hardware implementation to reduce hardware resource requirement. The resource utilization by HOG blocks implemented using *hls_math.h* functions and the optimized functions is shown in Table VII. In addition, Orientation binning via HLS *atanf()* function uses CORDIC module that requires 4 Block RAMs of 16kb size, which can be saved through optimized implementation.

V. CONCLUSION

We have proposed a software/hardware co-design architecture for real-time computer vision applications.

TABLE VII: Individual speedup due to different algorithmic optimisations

HOG component	Function	FF	LUT	DSP 48E
Gradient	HLS Library	1404	1290	2
Magnitude	Optimized	72	112	0
Orientation	HLS Library	9497	10062	13
Binning	Optimized	858	1379	3
Inverse	HLS Library	10039	6765	0
Square Root	Optimized	600	856	5
Overall resources saved		19410 92.7%	15770 87.0%	7 46.7%

We have tested this design on real-time handwritten digit recognition application and achieved a speedup of 38.89 in the execution of the parallelizable portion of the application. An overall speedup of 4.375 has been observed compared to the software implementation of the application. This speedup due to hardware optimisations has only negligibly reduced the accuracy from 96.5% to 95.6%.

Our application spends about 8ms for classifying 1 digit. It can detect the white spaces that are word boundaries and the arrange the digits in the right order. It can also detect if a hand is blocking the view of clear contours and pauses the application and resumes once the hand that is covering the digits is moved away in the process of writing. It is capable of working with a video capture rate of 30 frames per second. However, with the average writing speed being 68 characters per minute, a lower processing rate is used.

This application is easily scalable to include the full character set of any language. Training can be done on any software platform to generate the trained XML file which needs to be included for classification. Existing OCR libraries can also be incorporated. This application will be extremely useful for documenting non-Latin scripts and for language translations. This system can be used for applications like video surveillance, facial and gesture recognition, handwriting recognition and simultaneous conversion of handwritten documents to digital text, augmented reality applications and complete scene registration.

REFERENCES

[1] G. De Micheli and Rajesh K. Gupta, "Hardware/software co-design," in *Proceedings of the IEEE 85.3, 1997*, pp. 349-365.

[2] Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard and V. Vapnik, "Learning algorithms for classification: A comparison on handwritten digit recognition," in *Neural networks: the statistical mechanics perspective, 261, 1995*, pp.276.

[3] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," in *Neural processing letters 9.3, 1999*, pp. 293-300.

[4] Z. Zhang, L. Jin, K. Ding and X. Gao, "Character-SIFT: a novel feature for offline handwritten Chinese character recognition," in *2009 10th International Conference on Document Analysis and Recognition*, pp. 763-767. IEEE.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection,", in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*,vol. 1, pp. 886-893.

[6] T. N. Do and N. K. Pham, "Handwritten Digit Recognition Using GIST Descriptors and Random Oblique Decision Trees," in *Some Current Advanced Researches on Information and Computer Science in Vietnam, 2015*, pp. 1-15. Springer International Publishing.

[7] A. Downton and D. Crookes, "Parallel architectures for image processing," in *Electronics & Communication Engineering Journal 10.3, 1998*, pp. 139-151.

[8] N. Zhang, Y. Chen and J. Wang, "Image parallel processing based on GPU," in *Advanced Computer Control (ICACC), 2010 2nd International Conference on*, vol. 3, pp. 367-370. IEEE.

[9] E. Welch, D. Patru, E. Saber and K. Bengtson, "A study of the use of SIMD instructions for two image processing algorithms," in *Image Processing Workshop (WNYIPW), 2012 Western New York*, pp. 21-24. IEEE.

[10] D. Crookes, K. Benkrid, A. Bouridane, K. Alotaibi and A. Benkrid, "Design and implementation of a high level programming environment for FPGA-based image processing," in *IEE Proceedings-Vision, Image and Signal Processing 147.4, 2000*, pp. 377-384.

[11] C. Kozyrakis and D. Patterson, "Vector vs. superscalar and VLIW architectures for embedded multimedia benchmarks," in *Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture*, pp. 283-293.

[12] B. Cope, P. Y. K. Cheung, W. Luk and S. Witt, "Have GPUs made FPGAs redundant in the field of video processing?," in *Proceedings, 2005 IEEE International Conference on Field-Programmable Technology*, pp. 111-118.

[13] S. Asano, T. Maruyama and Y. Yamaguchi, "Performance comparison of FPGA, GPU and CPU in image processing," in *2009 International Conference on Field Programmable Logic and Applications*, pp. 126-131. IEEE.

[14] L. M. Reyneri,"Implementation issues of neuro-fuzzy hardware: going toward HW/SW codesign," in *IEEE Transactions on Neural Networks 14.1, 2003*, pp. 176-194.

[15] T. V. Huynh, "Design space exploration for a single-FPGA handwritten digit recognition system," in *Communications and Electronics (ICCE), 2014 IEEE Fifth International Conference on*, pp. 291-296. IEEE.

[16] P. F. Felzenszwalb, R. B. Girshick, D. McAllester D. and D. Ramanan, "Object detection with discriminatively trained part-based models," in *IEEE transactions on pattern analysis and machine intelligence, 32.9*, pp.1627-1645.

[17] P. Y. Chen, C. C. Huang, C. Y. Lien and Y. H. Tsai, "An efficient hardware implementation of HOG feature extraction for human detection," in *IEEE Transactions on Intelligent Transportation Systems 15.2, 2014*, pp. 656-662.

[18] L. Deng, "The MNIST database of handwritten digit images for machine learning research," in *IEEE Signal Processing Magazine 29.6, 2012*, pp. 141-142.

[19] J. Hiraiwa and H. Amano, "An FPGA implementation of reconfigurable real-time vision architecture," in *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pp. 150-155. IEEE.

[20] D. M. Powers, David M W, "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation," in *Journal of Machine Learning Technologies 2.1, 2011*, pp. 37–63.

[21] Feature Detection And Description. [Online]. Available: http://docs.opencv.org/2.4/modules/ocl/doc/feature_detection_and_description.html#ocl-hogdescriptor

[22] Getting started with Xillinux for Zynq-7000 EPP. [Online]. Available: http://xillybus.com/doc

[23] FPGA coprocessing for C/C++ programmers. [Online]. Available: http://xillybus.com/tutorials/vivado-hls-c-fpga-howto-4

[24] Vivado Design Suite User Guide: High-Level Synthesis. [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_1/ug902-vivado-high-level-synthesis.pdf

[25] GNU gprof. [Online]. Available: http://sourceware.org/binutils/docs/gprof/