Hindawi Publishing Corporation
ISRN Machine Vision
Volume 2013, Article ID 820216, 6 pages
http://dx.doi.org/10.1155/2013/820216

Research Article

Area Optimized FPGA-Based Implementation of The Sobel Compass Edge Detector

Sanjay Singh,[1] Anil Kumar Saini,[1] Ravi Saini,[1] A. S. Mandal,[1] Chandra Shekhar,[1] and Anil Vohra[2]

[1] *CSIR-Central Electronics Engineering Research Institute (CSIR-CEERI), Pilani, Rajasthan 333031, India*
[2] *Electronic Science Department, Kurukshetra University, Kurukshetra, Haryana 136119, India*

Correspondence should be addressed to Sanjay Singh; sanjay.csirceeri@gmail.com

Received 22 December 2012; Accepted 4 February 2013

Academic Editors: V. Alchanatis and A. Nikolaidis

Copyright © 2013 Sanjay Singh et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents a new FPGA resource optimized hardware architecture for real-time edge detection using the Sobel compass operator. The architecture uses a single processing element to compute the gradient for all directions. This greatly economizes on the FPGA resources' usages (more than 40% reduction) while maintaining real-time video frame rates. The measured performance of the architecture is 50 fps for standard PAL size video and 200 fps for CIF size video. The use of pipelining further improved the performance (185 fps for PAL size video and 740 fps for CIF size video) without significant increase in FPGA resources.

1. Introduction

Edge detection is one of the most important areas in lower level image processing. Quality of detected edges plays a very important role in realization of complex automated computer/machine vision systems [1]. Various edge detection algorithms are available in the literature and give different responses and details to the same input image. The Sobel edge detector is very popular than simple gradient operators due to its property to counteract the noise sensitivity and easier implementation [2]. The accuracy of the Sobel operator for edge detection is relatively low because it uses two masks which detect the edges in horizontal and vertical directions only. The accuracy can be enhanced by using the Sobel compass operator which uses a larger set of masks with narrowly spaced orientations [3, 4]. But use of the Sobel compass edge detector increases the computational complexity significantly for computing edges. It is hard to perform this computationally intensive task in real-time with serial processors. Alternative to this is design of specific hardware (ASICs or FPGAs) for the Sobel compass edge detector. The size and speed of current generation FPGAs are comparable to ASICs, but FPGAs provide the possibility to perform algorithm changes in later stages of the system development and reduce the design cost and time [5]. This makes the FPGAs a suitable choice for such applications.

Some recent FPGA implementations are available in the literature for the Sobel compass edge detector. In [6, 7], the authors discussed the most obvious FPGA implementation of the Sobel compass edge detector, which uses multiple processing elements in parallel to compute gradient along each direction. This increased the FPGA resources. The hardware-software codesign-based approach has been discussed in [8] for Sobel compass operator implementation which uses eight processing elements in parallel. It is observed that, the main focus of most of existing FPGA-based implementations of the Sobel compass edge detector has been on achieving real-time performance by using highly parallel architecture and thus fully ignored the FPGA resources' optimizations. FPGA resource (area) optimization is important for edge detection as it is one of basic modules in the large automated video surveillance system (which uses many complex algorithms requiring large FPGA resources).

In this paper, an FPGA resource optimized hardware architecture for the Sobel compass edge detector for real-time video surveillance applications is investigated. We show that a single processing element is sufficient to compute the gradient along all directions in real-time. As the Sobel compass

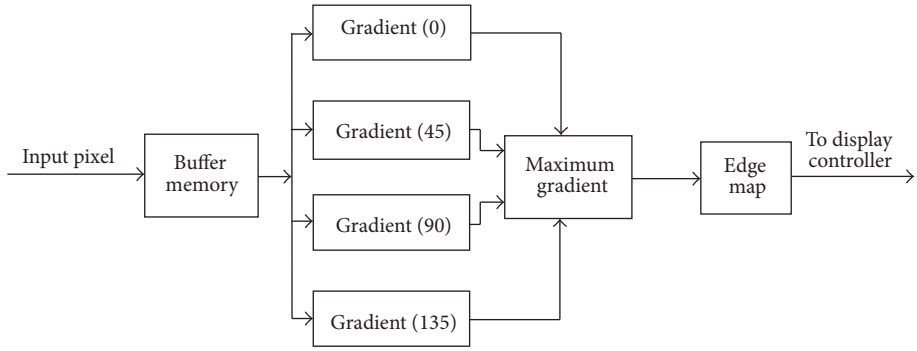


Figure 1: The Sobel compass edge detector block diagram.

edge detector is sliding window operator, smart buffer-based memory architecture is used to move the incoming pixels in computing window. The specific datapaths are designed, and controller is developed to perform the complete task. The design and simulation are done using VHDL and targeted to Xilinx ML510 (Virtex-5 FX130T) FPGA platform. Custom camera interface PCB is designed to interface Sony PTZ camera with ML510 FPGA platform. Complete system is tested for real-world scenario, and it robustly detects the edges in real-time for video sequences captured by camera.

2. The Sobel Compass Edge Detector

In this section, the used algorithm is briefly described, for a more detailed description we refer to [3, 4]. The Sobel operator is widely used for edge detection in images. It has advantage over simple gradient operators because of its property to counteract the noise sensitivity. It is based on computing an approximation of the gradient of the image intensity function. It uses two 3×3 spatial masks (Hx and Hy) which are convolved with the original image to calculate the approximations of the gradient. The Sobel operator uses two filters:

$$Hx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad Hy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{1}$$

These compute the average gradient components across the neighboring lines or columns, respectively. The local edge strength is defined as the gradient magnitude given by

$$GM(x, y) = \sqrt{Hx^2 + Hy^2}. \tag{2}$$

The accuracy of the Sobel operator for edge detection is relatively low because it uses two masks which detect the edges in horizontal and vertical directions only. This problem can be addressed by using the Sobel compass operator which uses a larger set of masks with narrowly spaced orientations. It uses eight masks (H_0, H_{45}, H_{90}, H_{135}, H_{180}, H_{225}, H_{270}, and H_{315}) each providing edge strength along one of the eight possible directions of the compass:

$$\begin{aligned} H_0 &= \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \qquad H_{45} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}, \\ H_{90} &= \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad H_{135} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}, \\ H_{180} &= \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \qquad H_{225} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}, \\ H_{270} &= \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \qquad H_{315} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}. \end{aligned} \tag{3}$$

Only the result of four (H_0, H_{45}, H_{90}, and H_{135}) of the above eight masks must actually be computed since the four others are identical except for the reversed sign. The edge strength E at position (x, y) is defined as the maximum of the eight masks output ($D_0, D_{45}, D_{90}, D_{135}, D_{180}, D_{225}, D_{270}$, and D_{315}), that is,

$$E_{xy} = \max\left(D_0, D_{45}, D_{90}, D_{135}, D_{180}, D_{225}, D_{270}, D_{315}\right). \tag{4}$$

Since $D_{180} = -D_0, D_{225} = -D_{45}, D_{270} = -D_{90}$, and $D_{135} = -D_{315}$, therefore, the above equation can be rewritten as

$$E_{xy} = \max\left(|D_0|, |D_{45}|, |D_{90}|, |D_{135}|\right). \tag{5}$$

Therefore, to find the edges in all possible directions, the four masks (H_0, H_{45}, H_{90}, and H_{315}) must be applied to each pixel of the input image. The Sobel compass edge detector has the advantage of not requiring the computation of squares and square roots (which are considered relatively expensive operations).

3. Proposed Architecture

This section describes the details of the proposed architecture for the Sobel compass edge detector. Figure 1 shows the basic

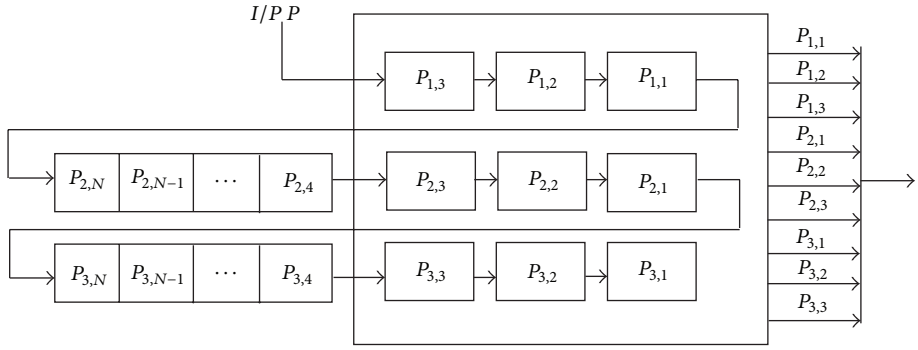


Figure 2: Buffer memory architecture.

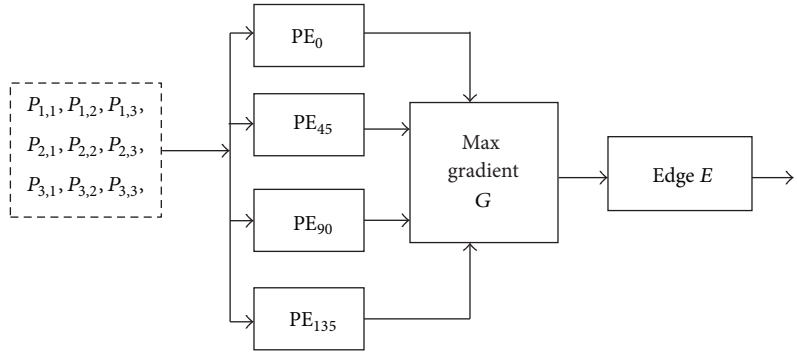


Figure 3: The standard Sobel compass edge detector architecture.

block level data flow diagram for computation of the Sobel compass edge detector. It consists of mainly four stages. In the first stage, the incoming pixel data from camera interface logic is stored in buffer memory. Four gradients along different directions are computed in the second stage. The maximum gradient is selected, and final edge map is computed by comparing the maximum gradient value with a threshold in the third and fourth stages, respectively.

In the Sobel compass operator, the 3×3 masks are used to compute gradient values along different directions over an input image. Therefore, it is necessary to store at least two rows of input image data in FPGA on-chip memory before the processing begins. To achieve this, we have used smart buffer-based memory architecture [9, 10] which utilizes two FIFOs and a set of registers in order to shift the image data into computing window (Figure 2). The length of the shift register depends on input image width.

The standard implementations of the Sobel compass edge detector [6, 7] use four processing elements in parallel for computing gradient along different directions. For comparison of results with our proposed architecture, we coded the standard architecture (Figure 3) in VHDL, synthesized using Xilinx ISE 10.3, and implemented on Xilinx ML510 (Virtex-5) FPGA board. The resulted maximum clock frequency for this architecture is 118.5 MHz.

It is observed that in the above implementation, each processing element (PE) performs the same set of operation

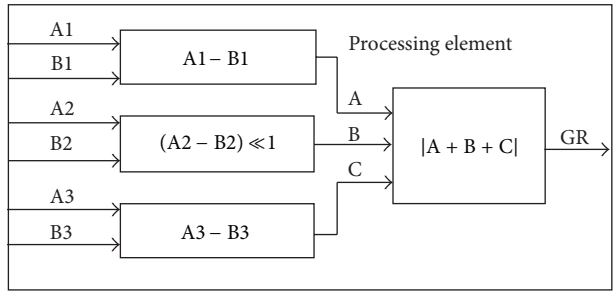


Figure 4: Processing element architecture.

(addition, subtraction, and multiplication by 2) on inputs applied to them. Processing element (PE) architecture for gradient computation is shown in Figure 4. The only difference for four gradient computation units (PE_0, PE_{45}, PE_{90}, and PE_{135}) is in inputs applied to them at a particular time. Therefore, by switching the inputs applied to anyone of processing element in appropriate manner, the same processing element can be used to compute all four gradients along different directions. This forms the basis for the proposed architecture.

For real-time video surveillance applications, the required frame rate is 25 fps (frames per second) for PAL size video (30 fps for NTSC size video). For high speed video surveillance application, for a frame rate of 50 fps for PAL size

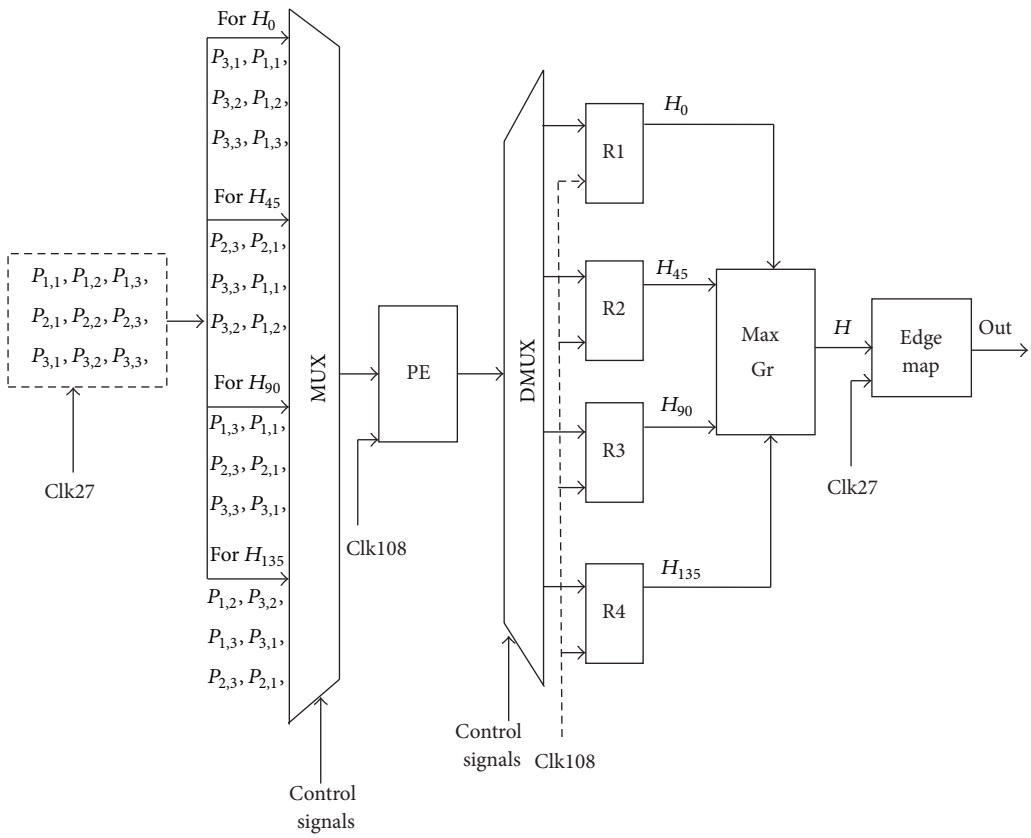


Figure 5: The proposed Sobel compass edge detector architecture.

video (60 fps for NTSC size video), clock frequency at which the image pixel data samples are available is 27 MHz. If the processing element is operated at 108 MHz (4×27 MHz), all the four gradients can be computed for available image pixel data before the arrival of next image pixel data using single processing element. The operating frequency of 108 MHz is well within the limits of maximum operating frequency (118.5 MHz) resulted from synthesis results. The digital clock managers available on FPGA are used to generate 108 MHz clock from 27 MHz pixel data clock.

Figure 5 shows the proposed architecture for real-time computation of edges in an image using the Sobel compass edge detector. Gradient computation for all directions is realized through single processing element (operating at 108 MHz). The PE is used in appropriate sequential order in different time slots for computing gradients along all directions. This architecture greatly economizes on the FPGA resources' usages (area) but needs storage elements to store results for future use and set of multiplexers for switching of inputs and outputs in different time slots. This architecture also requires a controller which insures proper functioning of complete design. Control signals for input selection multiplexer and output selection demultiplexer are generated at 108 MHz so that inputs to processing element can be switched and output can be stored properly. Image pixel data moves through computing window at 27 MHz. Gradient values for directions along 0 degree, 45 degree, 90 degree, and 135 degree are computed by processing element in 4 cycles at 108 MHz clock frequency, and the results are stored in R1, R2, R3, and R4 registers which works at 108 MHz frequency. Finally, the maximum gradient is selected by using the values stored in registers R1 to R4, and final edge map is computed by comparing the maximum gradient value with a threshold. Therefore, before the arrival of next image data samples (at 27 MHz), the final edge map of current pixel data is available.

The performance of the system is further improved by using a pipelined processing element (Figure 6) at the cost of area occupied by pipelined registers. Resulted maximum clock frequency is 405 MHz. Therefore, a frame rate of 185 fps can be achieved for standard PAL size video. The improvement in clock frequency in pipelined architecture is due to reduced maximum combinational path delay in processing element architecture.

4. Results

The proposed architecture has been designed using VHDL, simulated in ModelSim, and synthesized using Xilinx ISE 10.3. It has been implemented on Xilinx ML510 (Virtex-5) FPGA platform. It is capable of computing the edge map of an input image well within the real-time constraints while utilizing much less FPGA resources. The synthesis results (Table 1) reveal that the FPGA resources utilized by proposed edge detection architecture are more than 40% less as compared

Table 1: Comparison of synthesis results.

Synthesis parameters	Standard [6, 7] Figures 3–4	Proposed Figures 4 and 5 (percentage of reduction)	Proposed pipelined Figures 5–6 (percentage of reduction)
FPGA slices	67	40 (40.3%)	42 (37.3%)
Slice LUTs	222	109 (50.9%)	117 (47.3%)
LUT FF pairs	222	118 (46.8%)	133 (40.1%)
Slice register	2	40	95
Route-thrus	16	2 (87.5%)	2 (87.5%)

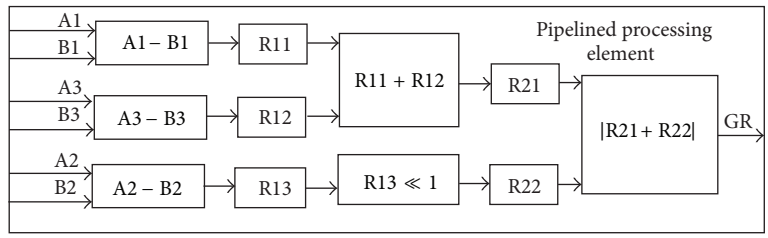


Figure 6: Pipelined processing element architecture.

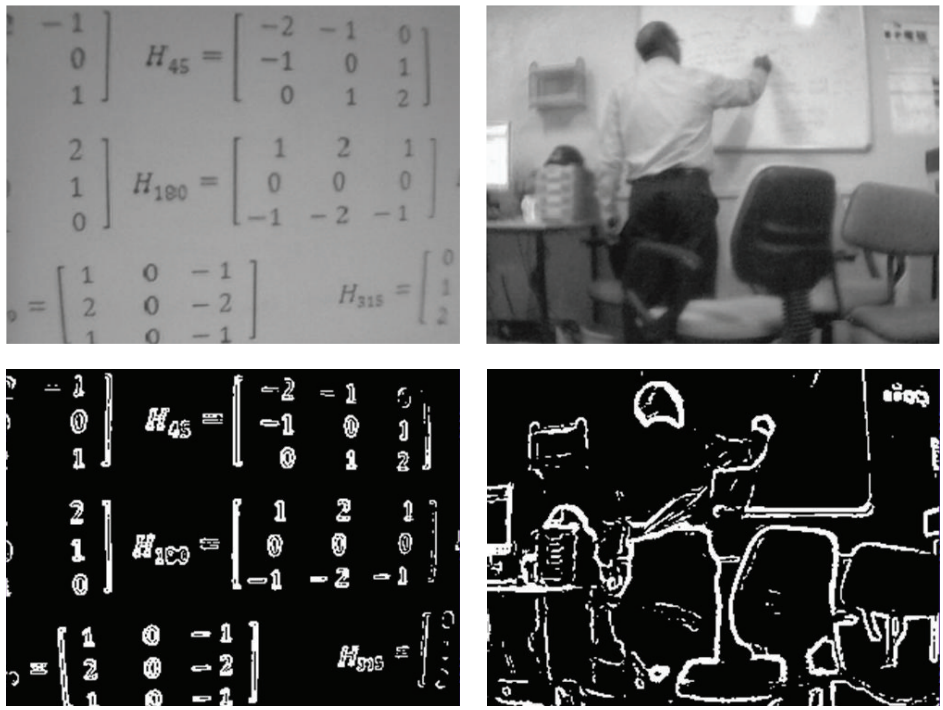


Figure 7: Input test images and output images.

to the standard architecture. The synthesis parameters are number of FPGA slices used by the logic, number of slice LUTs (Look Up Tables), number of LUT Flip Flop Pairs (LUT Flip Flop pair represents one LUT paired with one Flip Flop within a slice), number of slice registers, and number of route-thrus (used for routing only). The resource reduction is not one-fourth of the resources utilized by standard architecture because of resource utilization by additional input and output switching logics. The measured performance of our system at 108 MHz operating frequency for PAL size video is 50 fps (frames per second) and CIF size video is 200 fps. Pipelined implementation resulted in 185 fps frame rate for PAL size video and 740 fps for CIF size video. PAL size and CIF size images are most commonly used video formats for video surveillance cameras. Therefore, implemented system can easily detect edges for high frame rate surveillance applications while utilizing much less FPGA resources. The input test images (PAL size) taken from camera and output images produced by implemented system and displayed on monitor are shown in Figure 7.

5. Conclusion

In this paper, the hardware architecture for the Sobel compass edge detector, implemented on Xilinx ML510 FPGA platform, has been presented. This architecture greatly economizes on the FPGA resources' usages (area). The architecture used more than 40% less FPGA resources as compared to the standard implementations presented in the literature and maintained real-time constraints for video processing. The implemented architecture is integrated with camera and display monitor. The integrated system is tested for real-world scenario. It robustly detects the edge in real-time. It can be efficiently used as part of complex computer vision system without occupying much area.

Acknowledgment

S. Singh would like to thank Mr. Raj Singh, Group Leader, IC Design Group, for his constant guidance, suggestions, support, and encouragement.

References

[1] M. B. Ahmad and T. S. Choi, "Local threshold and boolean function based edge detection," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 674–679, 1999.

[2] T. A. Abbasi and M. U. Abbasi, "A novel FPGA-based architecture for Sobel edge detection operator," *International Journal of Electronics*, vol. 94, no. 9, pp. 889–896, 2007.

[3] W. Burger and M. J. Burge, *Digital Image Processing: An Algorithmic Introduction Using Java*, Springer, New York, NY, USA, 2008.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Pearson Education, New Delhi, India, 2009.

[5] H. Jiang, H. Ardö, and V. Öwall, "A hardware architecture for real-time video segmentation utilizing memory reduction techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 2, pp. 226–236, 2009.

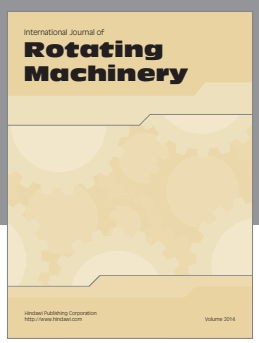
[6] Z. Guo, W. Xu, and Z. Chai, "Image edge detection based on FPGA," in *Proceedings of the 9th International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, pp. 169–171, August 2010.

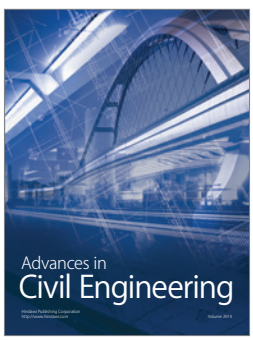
[7] A. Nosrat and Y. S. Kavian, "Hardware description of multi-directional fast sobel edge detection processor by VHDL for implementing on FPGA," *International Journal of Computer Applications*, vol. 47, no. 25, pp. 1–7, 2012.

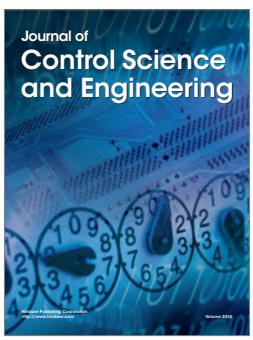
[8] K. C. Sudeep and J. Majumdar, "A novel architecture for real time implementation of edge detectors on FPGA," *International Journal of Computer Science Issues*, vol. 8, no. 1, pp. 193–202, 2011.

[9] Z. Vasicek and L. Sekanina, "Novel hardware implementation of adaptive median filters," in *Proceedings of the 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS '11)*, pp. 1–6, April 2008.

[10] C. Moore, H. Devos, and D. Stroobandt, "Optimizing the FPGA memory design for a sobel edge detector," in *Proceedings of the 20th Annual Workshop on Circuits, Systems and Signal Processing*, 2009.







Hindawi
Submit your manuscripts at
http://www.hindawi.com