44th IEEE Southeastern Symposium on System Theory
University of North Florida, Jacksonville, FL
March 11-13, 2012

# Efficient FPGA Implementation of Steerable Gaussian Smoothers

Arjun Joginipelly, Alvaro Varela, Dimitrios Charalampidis
Electrical Engineering Department
University of New Orleans
New Orleans, LA, USA

Remy Schott
ECE Department
Florida Institute of Technology
Melbourne, FL, USA

Zachary Fitzsimmons
Computer Science Department
Siena College
Loudonville, New York, USA

*Abstract*— **Smoothing filters have been extensively used in image and video analysis. In particular, directional smoothers have been employed in motion analysis, edge detection, line parameter estimation, and texture analysis. Such applications often necessitate the use of several directional filters oriented at different angles. However, applying a large number of filters commonly requires a significant amount of computing resources. In such cases, real-time performance may be possibly achieved through utilization of hardware devices having parallel processing capabilities. Additionally, techniques can take advantage of the inherent properties of certain smoothing filters. Such a property is steerability, which implies that the outputs of several filtering operations can be linearly combined in order to produce the output of a directional filter at an arbitrary orientation. Although several efficient FPGA implementations of the convolution operation have been presented in the literature for non-separable and separable, research on steerable filter implementations on FPGA is limited. In this paper, steerable Gaussian smoothers are implemented on an FPGA platform. The technique is compared with a software-based implementation. Performance comparisons indicate that the FPGA technique provides significant speed-up factor of at least ~6, utilizing only a small percentage of the FPGA resources.**

***Keywords- Steerability; Directional filter; FPGAs; Gaussian filters; Separable convolution;***

## I. INTRODUCTION

Directional or orientation filters [1] are widely used in computer vision and image processing applications [9], such as motion analysis, edge detection and texture analysis. In general, image components, such as edges and lines, can be characterized by a set of parameters including position, orientation, width or size. One approach for obtaining the response of a filter at any arbitrary position and orientation is to tune the filter to all possible positions and orientations. However, such an approach requires a large number of computations, and is thus not easily implementable in real time. An alternative, more efficient way, is to design a family of basis filters [2] [3], so that a filter tuned at an arbitrary position or orientation can be represented as a linear combination of these basis filters. Therefore, the output of the filter can be expressed as a weighted sum of the basis filter outputs. Filters having this property are called steerable filters.

Steerable smoothers have been used to efficiently smooth images in several directions [1]. In general, smoothing can be performed by convolving the original image with an appropriate mask, such as a Gaussian mask. Convolution [11], requires a significant amount of computational power even if it is implemented on FPGA devices [7][8]. In order to improve the performance of the convolution operation, special filters can be considered. For instance, a reduction in the number of multipliers can be achieved through the use of separable filters [2][10]. A separable filter of size $N \times N$ can be expressed as the convolution between two filters of sizes $N \times 1$ and $1 \times N$, respectively. Isotropic Gaussian filters and Gaussian filters oriented at $0^o$ or $90^o$ are separable. However, Gaussian filters oriented at an arbitrary direction are not separable in general [2]. Therefore, separability alone is not sufficient in designing efficient multidirectional Gaussian filter banks.

In this paper, a Gaussian steerable multidirectional filter bank implementation on FPGA is proposed, based on the work presented in [1]. Although efficient implementations of separable and non-separable 2D convolution techniques have been explored in the past [7][8][9][10], there has been only limited work in the literature related to steerable filter implementations on FPGA [4][5][6]. In [4], a steerable pyramid wavelet construction for image decomposition and feature detection, and its implementation on FPGA was presented. The filter coefficients were obtained in the Fourier domain as the product of a radial frequency function, based on the Erlang function, and an angular frequency function. Some implementation issues, including the quantization of filter coefficients and the error resulted due to the fixed point coefficient representation were discussed. Some similar issues are also discussed in this paper. In [5], steerable filters were used as edge detectors for the purpose of lane detection. The FPGA implementation concentrated only on the first and second order derivatives of 2D Gaussian functions with fixed orientation to reduce computational complexity. In other words, the particular filters used are appropriate for a real time FPGA implementation owing to the fact that they can be steered by a small number of basis filters. On the other hand, if Gaussian smoothers, especially those with strong directional characteristics, were to be steered to an arbitrary direction following an approach similar to the ones presented in [4] and [5], then a large number of basis filters would be

The authors would like to thank the National Science Foundation (NSF) for their support through the REU project 0851618

required. Instead, the implementation employed in this work ensures that, regardless of the desired angular resolution, Gaussian smoothers can be steered via the application of three 1D filtering operations, as described next.

In [1], a Gaussian steerable multidirectional filter bank consisting of $M$ directional 2D Gaussian smoothers is implemented by decomposing each of the $M$ smoothers in three 1D Gaussian filters. The first two 1D filters are employed horizontally and vertically, while the third 1D filter is applied in the direction of interest. The first two 1D filters are essentially equivalent to a 2D separable Gaussian filter, and they are independent of the filter orientation. As a result, they are applied only once, and they do not significantly affect the resource utilization, regardless of the number of filter orientations, $M$, used in the filter bank. The third 1D filter needs to be employed $M$ times. Nevertheless, some initial smoothing has already been applied to the image owing to the first two 1D filters. As a result, a down-sampling factor can be introduced to reduce the number of operations. Therefore, the resource requirements are sufficiently low so that pipelining techniques can be used for a real-time implementation.

The technique in [1] is discussed in more detail in section 2. Since the steerable filter includes a separable filter component, separable filter implementations are discussed in section 3. In section 4 the steerable FPGA implementation is presented. Section 5 presents experimental results. Finally, section 6 closes with some concluding remarks and future work.

## II. BACKGROUND

Steerability implies that the output $O\theta(x,y)$ at location $(x,y)$ of a filtering operation using a filter oriented at an angle $\theta$ can be computed as the linear combination of a finite set of $M$ outputs $\{O\theta_0\ (x,\ y),\ O\theta_1\ (x,\ y),\ \ldots\ldots\ldots,\ O\theta_{M-1}\ (x,\ y)\ \}$ obtained by applying the same filter oriented at directions $\theta_0$, $\theta_1$,……,$\theta_{M-1}$, respectively. Using a more general definition, the $M$ outputs can be obtained using a set of $M$ basis filters, which are not necessarily rotated versions of the same filter. It has been shown [1] that a 2D steerable filter for a directional Gaussian smoother with standard deviations $\sigma_x$ and $\sigma_y$ (where $\sigma_y$ is the standard deviation along the axis of filter orientation) can be expressed as:

$$g_\theta(x,y) = \sum_{r=-R}^{R} g_{iso}\left(x - r\cos(\theta), y - r\sin(\theta)\right) g^{1D}(r) \qquad (1)$$

where it was assumed that the size of $g^{1D}(r)$ is equal to $2R+1$. The filter described in (1) can be applied to image I($x,\ y$) in two steps. In the first step, the filter $g_{iso}(x,\ y)$ is applied to the image:

$$I_{iso}(x,y) = I(x,y) * g_{iso}(x,y) \qquad (2)$$

In the second step, the following operation is applied to the image $I_{iso}(x,\ y)$.

$$I_\theta(x,y) = \sum_{r=-R}^{R} I_{iso}\left(x - r\cos(\theta), y - r\sin(\theta)\right) g^{1D}(r) \qquad (3)$$

The operations described in equations (2) and (3) are equivalent to a 2D convolution between image $I(x,y)$ and a Gaussian directional filter (DSF) oriented at direction $\theta$. The function $g_{iso}(x,y)$ describes a separable filter and can be implemented in an efficient manner. More specifically, $g_{iso}(x,y)$ can be expressed as a product of two 1D Gaussian filters, namely $g_{iso}(x,\ y) = g_x(x)g_y(y)$. Hence $g_{iso}(x,\ y)$ can be applied to $I(x,y)$ by first filtering $I(x,y)$ horizontally using $g_x(x)$ and then vertically using $g_y(y)$. Equation (3) describes a linear combination of shifted versions of the image, $I_{iso}(x - r\cos(\theta),\ y - r\sin(\theta))$, which depend on the filtering direction $\theta$. The coefficients of the linear combination are equal to the values of $g^{1D}(r)$.

This implementation is steerable in the sense that the final output $I_\theta(x,\ y)$ is expressed as a linear combination of output images, $I_{iso}(x - r\cos(\theta), y - r\sin(\theta))$, which are obtained using a set of $2R+1$ filters, $g_{iso}(x - r\cos(\theta), y - r\sin(\theta))$, for $r = -R, \ldots, R$. The isotropic filter $g_{iso}(x,\ y)$ is low pass, while almost 100% of its energy is included within the frequency band $[-3/\sigma_x,\ 3/\sigma_x]$. The output, $I_{iso}(x,\ y)$, obtained by filtering the input image $I(x,\ y)$ with $g_{iso}(x,\ y)$ is band-limited within the frequency range $(-\pi,\ \pi]$ in any direction $\theta$. Thus, equation (3) can be modified without introducing significant aliasing as follows:

$$I_\theta(x,y) = \sum_{k=-[R/D]}^{[R/D]} I_{iso}\left(x - kD\cos(\theta), y - kD\sin(\theta)\right) g^{1D}(kD) \qquad (4)$$

where

$$g^{1D}(r) = g(kD) = \frac{1}{\sqrt{2\pi(\sigma_y^2 - \sigma_x^2)}} \exp\left(-\frac{(kD)^2}{2(\sigma_y^2 - \sigma_x^2)}\right) \qquad (5)$$

and

$$D = \frac{\pi\sigma_x}{3} \geq 1 \text{ is a down sampling factor.} \qquad (6)$$

In equation (4), $[R/D]$ is equal to the integer part of $[R/D]$. Since the range of unique frequencies in discrete signals is $(-\pi,\ \pi]$, $D$ can be as large as the largest integer not greater than $\pi\sigma_x/3$, so that aliasing does not occur. The goal of introducing a down-sampling factor is to further reduce the computational complexity of the filtering operation.

## III. DISCUSSION ON SEPARABILITY

In this section we briefly discuss two different separable techniques, an unpipelined and a pipelined, for the purpose of implementing the filter mask $g_{iso}(x,y)$. As a reminder, $g_{iso}(x,y)$ is a separable filter and is thus equivalent to the convolution of a vertical and a horizontal 1D Gaussian masks. Throughout this section, it is assumed that the input image is of size $N \times N$ and $g_{iso}(x,y)$ of size $P \times P$. Each pixel in the input image and the coefficients of $g_{iso}(x,y)$ are represented using 8 bits. The block diagram representation of unpipelined separable convolution method is shown in Fig. 1. In this method, the image is first convolved with the vertical 1D Gaussian mask and second with the horizontal Gassian mask. At each adder stage rescaling is performed to ensure that the intensity value of the output pixels does not exceed 255 (8 bits). Using this method, 2 clock cycles are required to obtain the required output pixel, independent of image and mask sizes.

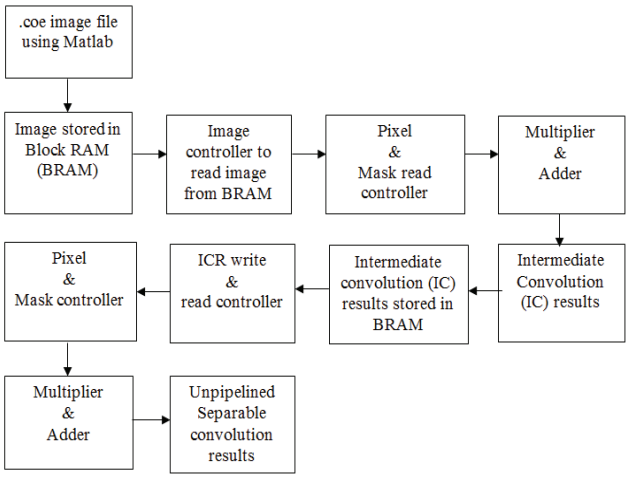


Figure 1. Block diagram representation of unpipelined separable convolution

Alternatively, a pipelining technique can be used to obtain higher throughput. Partial intermediate convolution results i.e. *N* rows and *P* columns of the image are stored in a FIFO or 2D array. This method requires only 1 clock cycle per pixel, regardless of the image and filter sizes. The block diagram representation of this method is shown in Fig. 2:

The above mentioned two separable techniques were implemented on Xilinx Virtex 2 Pro [12] i.e., XU2VP30-ff896 (with a speed grade of -7) for an input image of 158×158 and a Gaussian mask of 7×7. The comparisons are summarized in Table I in terms of resource utilization and processing time. For larger image, number of BRAMS required increases but the LUTs or resources used scale up the same. It should be mentioned at this point that other resources such as SRAM, Flash memory, are not utilized.

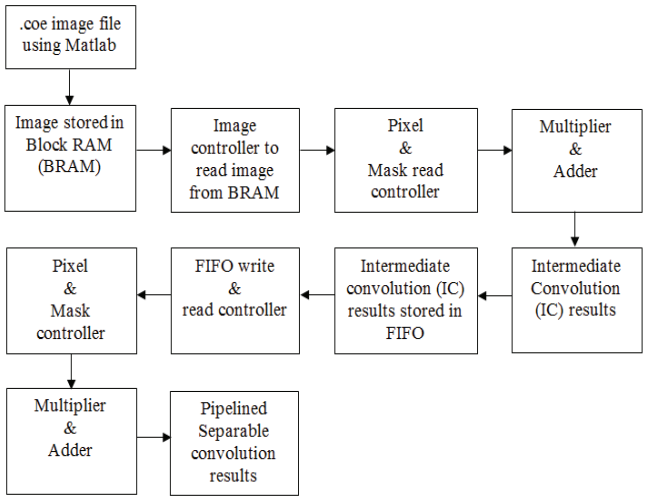


Figure 2. Block diagram representation of pipelined separable convolution

TABLE I. COMPARISON OF SEPARABLE TECHNIQUES

| Method | LUT 27392 | IOB 556 | BRAM 136 | Embedded Multipliers 136 | Clocks per Pixel |
|---|---|---|---|---|---|
| Unpipelined Separable Conv | 4292 (16%) | 10 (2%) | 26 (19%) | 14 (10%) | 2 |
| Pipelined Separable Conv | 14538 (53%) | 10 (2%) | 13 (9.6%) | 14 (10%) | 1 |

Based on Table I, it can be observed that for the unpipelined separable convolution, where intermediate results are stored in BRAM, the performance is reduced due to the access limitations imposed by the BRAM (only 1 pixel per clock cycle - single port access, or 2 pixels per clock cycle - dual port access can be accessed). Using single port access, the throughput in this method is 2 clock cycles per pixel. In the pipelined separable convolution, the throughput of 1 clock cycle per pixel is obtained by saving partial intermediate results in a FIFO or 2D array. However, significantly more hardware resources are required.

## IV. STEERABLE FPGA IMPLEMENTATION

The last stage of the Gaussian steerable approach presented in [1] uses an operation equivalent to 1D filtering at the direction of interest. This operation is applied to the smoothed image obtained by the previous stages, which include filtering of the original image using an isotropic Gaussian filter. As mentioned earlier, the isotropic Gaussian filter can be implemented in a separable manner. In this work, both separable implementations, unpipelined and pipelined, presented in section 3 were used. The Gaussian steerable filter implementation is based on equations (4) and (5). This Gaussian mask can be rotated in different directions to obtain steerable output at different directions. The block diagram representation of the steerable implementation is shown in Fig. 3.

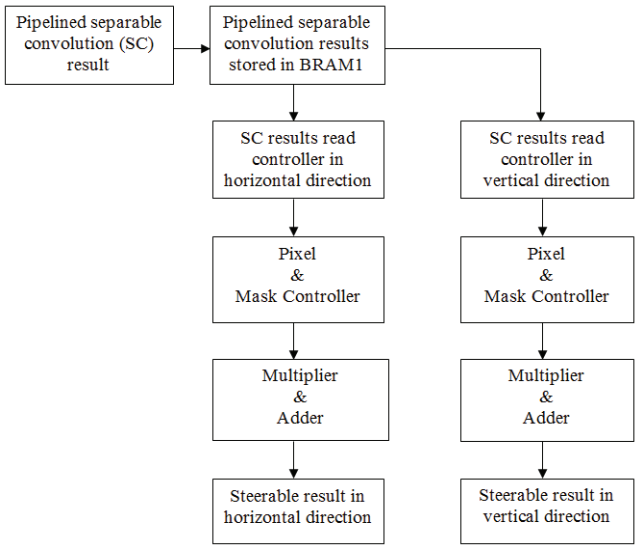


Figure 3. Block diagram representation of steerable implementation in horizontal and vertical directions.

The smoothed images obtained using the unpipelined and pipelined separable methods are stored in BRAMs. A controller is designed to read pixels from the smoothed image in horizontal and vertical directions. It has to be mentioned at this point that a decimation factor using equation (6) is used to reduce the computational complexity. That is, instead of reading consecutive pixels from the image (and filter mask coefficients) for processing, only 1 out of D consecutive image pixels (and filter mask coefficients) are read per output image pixel. A second controller, namely the pixel and mask controller, provides the required pixels to the multiplier and adder blocks. An additional rescaling step is performed at the adder stage to ensure that the intensity value of the output pixels remains between 0 and 255.

The RTL schematic that was obtained using ISE10.1 is shown in Fig. 5 for two steerable filter directions. The leftmost part of the circuit (C1) corresponds to the isotropic Gaussian filter operation, while the middle (C2) and rightmost (C3) parts implement in parallel the overall steerable filter in the horizontal and vertical direction. Additional arbitrary orientations may be obtained in parallel by replicating C2 or C3 a number of times equal to the number of different filter orientations.

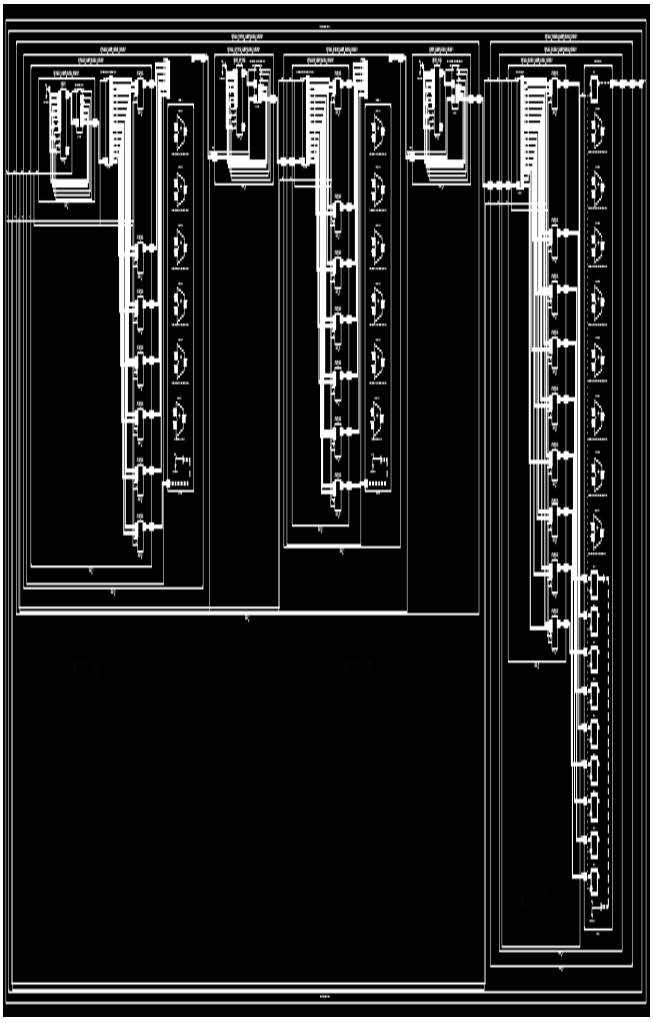

Figure 5. RTL schematic generated by synthesis.

TABLE II. UNPIPELINED AND PIPELINED STEERABLE IMPLEMENTATION

| Method | LUT 27392 | IOB 556 | BRAM 136 | Embedded Multipliers 136 | Clocks per Pixel |
|---|---|---|---|---|---|
| Unpipelined Steerable | 6358 (24%) | 22 (4%) | 52 (38%) | 32 (24%) | 3 |
| Pipelined Steerable | 22464 (82%) | 22 (4%) | 39 (29%) | 32 (24%) | 2 |

## V. EXPERIMENTAL RESULTS

In this section, experimental results are obtained for an image of size 158 × 158, and a separable Gaussian mask of 7 × 7 using the steerable filter implementation. Both separable methods described in section 3 were considered. As proposed in [1] and summarized in section II, isotropic filtering, which is equivalent to two 1D filtering operations (a horizontal and a vertical), followed by a directional 1D filtering operation represents an efficient steerable filtering implementation. In this work, results are presented for directional Gaussian smoothers with standard deviation $\sigma_x$ = 3 and $\sigma_y$ = 5. As a reminder, these two standard deviations do not necessarily correspond to the horizontal and vertical extents of the filter. They rather represent the filter extent with respect to the direction of filter ($\sigma_y$) and the angle vertical to it ($\sigma_x$), which can be arbitrary. Based on these standard deviation values, the directional 1D Gaussian mask is of size 1 × 9, while the 1D masks associated to the isotropic Gaussian mask are of size 1 × 7 and 7 × 1. Depending on which of the two separable implementations is employed, two different steerable implementations are obtained. The design is implemented on Xilinx Virtex 2 Pro [12] i.e., XU2VP30-ff896, with FPGA clock operating at a maximum frequency of 100MHz, in VHDL language using Xilinx ISE 10.1 Software. The comparison between unpipelined and pipelined steerable implementation are summarized in Table 3 in terms of resource utilization and processing time.

From Table II, the following observations can be drawn:

- Steerable implementation in horizontal and vertical directions using the unpipelined separable convolution method requires fewer resources and a throughput of 3 clocks cycles per pixel is achieved.
- Steerable implementation in horizontal and vertical direction using the pipelined separable convolution method requires huge resources and increase with the size of the input image. However an optimum performance of 2 clocks cycles per pixel is achieved.

Through this comparison, it can be observed that there is a trade-off among resource utilization and processing time. Hence the architecture selection can be done depending on the user constraints on whether to go with less resources and more processing time or more resources and less processing time.

It can be seen that the pipelined steerable implementation utilizes almost all the resources available on the board. This can be avoided by storing the original and intermediate processed images in the SDRAM or Flash Memory, which is part of our future work.

## VI. Conclusion And Future Work

In this paper, an efficient steerable filter implementation on FPGA has been presented. Similarly to previous techniques, the proposed approach takes advantage of the separable nature of isotropic Gaussian filters. However, it also employs a steerable implementation in which 2D filtering operations utilizing directional Gaussian smoothing filters, which are not separable in general, are replaced by an equivalent set of three 1D filtering operations. The first two operations are performed in a pipelined manner, in order to achieve high throughput that approaches 2 clock cycles per pixel. The third 1D filtering operation is employed consecutively to the first two operations in order to conserve memory resources.

In addition to using 1D filters, supplementary implementation aspects have been taken into account in order to reduce the on-chip resource utilization. For instance, the rescaling factor, which is effectively removing LSB bits introduced at each adder stage, allows reduction of the number of embedded multipliers.

As part of future work, the technique will be modified to reduce the number of clock cycles per pixel down to 1, by pipelining the pass of the third 1D filter. Of course, it is expected that such pipelining will also decrease the BRAM requirements since only part of the image will need to be stored in the BRAM after the first two 1D passes. However, it will also increase the multiplier usage, since the number of simultaneous multiplications will increase. A solution to the problem is to use block-wise processing by breaking down the image into overlapping zones.

Current results from Table III confirm that the pipelined steerable filter implementation on FPGA (100MHz clock) is significantly faster compared to a C implementation [1] executed on a PC with significantly higher clock speed (Dual Core 2 2.33-GHz).

TABLE III. FPGA Vs PC Comparison Of Execution Times

| Method | Unpipelined Steerable FPGA Implementation | Pipelined Steerable FPGA Implementation | C based Steerable Implementation |
|---|---|---|---|
| Execution Time for a 158x158 image | 0,656 ms | 0.492 ms | 6.5 ms |


## Acknowledgment

The authors would like to thank Mr. Rajesh Chari Chelpuri, MTech (IIT Delhi) and Dr. James Haralambides, Professor at Barry University, Department of Math and Computer Science, for their helpful comments and support.


## References


[1] Dimitrios Charalampidis, " Efficient Directional Gaussian Smoothers", IEEE Geoscience and Remote Sensing Letters, July 2009.

[2] V. Lakshmanan, "A Separable Filter for Directional Smoothing", IEEE Geoscience and Remote Sensing Letters, July 2004.

[3] W.Freeman and E.Adelson, "The Design and Use of Steerable Filters", IEEE Transaction on Pattern Analysis and Machine Intelligence, pp. 891-906, 1991.

[4] C.S Bouganis, P.Y.K Cheung, J.Ng and A. Bharath, " A Steerable Complex Wavelet Construction and its Implementation on FPGA", in Proc. International Conference on Field Programmable Logic and Applications, pp.394-403, 2004

[5] Erke Shang, Jian Li, Xiangjing An and Hangen He, " Lane Detection using Steerable Filters and FPGA Based Implementation", International Conference on Image and Graphics, 2011.

[6] S. Kestur, D. Dantara and V. Narayanan, " A streaming FPGA Implementation of a Steerable Filter for Real Time Applications(abstract only)", Proceeding of 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays, 2011.

[7] Hui Zhang, Mingxin Xia, and Guangshu Hu, "A Multiwindow Partial Buffering Scheme for FPGA Based 2-D Convolvers, IEEE Transaction on Circuits and Systems, Feb 2007.

[8] Franscisco Cardells-Tormo, Pep-Lluis Molinet, " Area Efficient 2-D Shift Variant Convolvers for FPGA Based Digital Image Processing", IEEE Transaction on Circuits and Systems, Feb 2006.

[9] D. Venkateshwar Rao and M. Venkatesan , "Implementation and Evaluation of Image Processing Algorithms on Reconfigurable Architecture using C-based Hardware Descriptive Languages", International Journal of Engineering and Applied Sciences, 2006.

[10] M.S. Andrews, " Architectures for Generalized 2D FIR Filtering using Separable Filter Structures", Proceeding of Acoustics, Speech and Signal Processing, 1999.

[11] R.C. Gonzalez and R.E. Woods. "Digital Image Processing", Prentice Hall 2002.

[12] Xilinx,"Virtex-II Pro Platform FPGA User Guide-UG012",v4.2, Nov 2007