A Scalable RC Architecture for Mean-Shift Clustering

Stefan Craciun[*], *Student Member, IEEE*, Gongyu Wang[*], *Student Member, IEEE*, Alan D. George[*], *Fellow, IEEE*, Herman Lam[*], *Member, IEEE*, Jose C. Principe[†], *Fellow, IEEE*

[*]NSF Center for High-Performance Reconfigurable Computing (CHREC)
[†]Computational Neuro-Engineering Laboratory (CNEL)
Department of Electrical and Computer Engineering, University of Florida Gainesville, Florida, 32611-6200
(email: craciun@chrec.org, wangg@chrec.org, george@chrec.org, hlam@chrec.org, principe@cnel.ufl.edu)

Abstract— **The mean-shift algorithm provides a unique non-parametric and unsupervised clustering solution to image segmentation and has a proven record of very good performance for a wide variety of input images. It is essential to image processing because it provides the initial and vital steps to numerous object recognition and tracking applications. However, image segmentation using mean-shift clustering is widely recognized as one of the most compute-intensive tasks in image processing, and suffers from poor scalability with respect to the image size (*N pixels*) and number of iterations (*k*): $O(kN^2)$. Our novel approach focuses on creating a scalable hardware architecture fine-tuned to the computational requirements of the mean-shift clustering algorithm. By efficiently parallelizing and mapping the algorithm to reconfigurable hardware, we can effectively cluster hundreds of pixels independently. Each pixel can benefit from its own dedicated pipeline and can move independently of all other pixels towards its respective cluster. By using our mean-shift FPGA architecture, we achieve a speedup of three orders of magnitude with respect to our software baseline.**

Keywords—mean-shift; image segmentation; hardware acceleration; FPGA; reconfigurable computing

I. Introduction

Nearly 30 years ago in 1975, Fukunaga and Hosteler proposed a clustering technique [1] that would later make a considerable impact in the clustering domain due to its unique unsupervised and non-parametric nature. In 1995 Cheng [2] established a rigorous mathematical background for the algorithm, proving its convergence in a finite number of iterations and giving it the familiar "mean-shift" name. The contributions of Comaniciu & Meer in the early 2000s demonstrated the performance advantages of the mean-shift algorithm [3], by efficiently applying it to image-processing applications, mainly image segmentation, tracking and edge detection. For image segmentation, the mean-shift algorithm has become an increasingly popular solution, yielding good results and providing a solid stepping stone for high-level vision tasks [4]. Image segmentation maps the input image to smaller pixel regions that share a common feature. This method helps in analyzing and interpreting the image, transforming it from a random collection of pixels to a unique arrangement of recognizable objects. Image segmentation is used to locate and separate objects from the background and from each other, to find boundaries, contours or any general region of interest. It is also instrumental to a multitude of domains with applications in computer/machine vision, medical imaging/diagnosis [5], satellite imaging and face recognition [6] to name just a few. However, Fukunaga and Hosteler stated in their first publication that their algorithm "*may be costly in terms of computer time and storage.*" Even for today's much improved computational platforms, their assessment remains true. The task is particularly challenging because the algorithm scales poorly with both the number of pixels (*N*) and number of iterations (*k*): $O(kN^2)$.

Speeding up multidimensional clustering in statistical data analysis has been the focal point of numerous papers that have employed different techniques to accelerate the clustering algorithms. These techniques primarily focus on incorporating better distance functions, better techniques for stopping early, and data-dimensionality reductions [7]. Other approaches have transitioned from conventional CPU platforms to GPUs [8] and many-node CPU platforms [9], in an effort to accelerate the convergence speed. Recently, efforts to accelerate the mean-shift algorithm have focused on FPGA technology [10] and have achieved the most promising results to date. In this paper, we tackle the problem of prohibitive execution time by first breaking the algorithm down to its smallest grain, which is the single-pixel movement. We start by first showing that the fine granularity of this algorithm allows all pixels to be shifted in parallel towards their respective clusters due to their computational independence. By first designing a pipeline dedicated to the individual movement of one pixel, and then scaling this approach to incorporate hundreds more, we demonstrate that the algorithm can be accelerated without incurring significant overhead. The computational platform we propose is ideal for this scenario, consisting of gate-array fabric on which these pipelines can be replicated, so that each one can process individual pixel movement over multiple iterations. This embarrassingly parallel approach is first replicated to effectively utilize all FPGA resources and then further scaled up to a board-level architecture (4 coupled FPGAs). The speedup we achieve, by unrolling the outer loop and clustering pixels in parallel (wide parallelism) while pipelining the inner loop and accumulating pairwise pixel interaction every clock cycle (deep parallelism), is more than one-thousand fold.

The organization of the paper will proceed by first establishing the mathematical background and presenting a quick overview of the baseline algorithm. In Section III, the

This work was supported in part by the I/UCRC Program of the National Science Foundation under Grant Nos. EEC-0642422 and IIP-1161022.

proposed parallel architecture model is presented in detail. In Section IV, we present numerical results while section V outlines future work and our continued efforts to optimize and further scale our current architecture. Finally, in Section VI, we draw conclusions and key insight from this work.

II. Mathematical Background

A. Mean-shift description

The mean-shift algorithm has been intuitively called a "hill climbing" technique, because the data points move in the direction of the estimated gradient. It is also commonly named a "mode seeking" algorithm, because the data points cluster around the nearest density maxima (the modes of the data). The underlying fundamental approach of the mean-shift is to transform the sampled feature space into a probability density space and locate the peaks of the probability density function (PDF). The regions of highest density are the cluster centers. This method is considered an unsupervised clustering method because the number of clusters and their location is determined only by the distribution of the data set.

B. Mathematical framework

The Parzen window technique is a very popular way of evaluating the PDF of a random variable. The kernel density estimator for a 3-dimensional space given *n* samples can be expressed as:

$$\hat{f}(x) = \frac{1}{nh^3}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right), \tag{1}$$

where *h* is the kernel size or bandwidth parameter. A wide range of kernel functions can be used in estimating the PDF. The most frequently used kernel and the one we have chosen to evaluate the PDF with is the Gaussian kernel:

$$K(x) = exp^{-\frac{x^2}{2\sigma^2}} \tag{2}$$

The next step is to calculate the gradient of the PDF $\nabla\hat{f}(x)$ and set it equal to zero for finding the peaks of the density function. The resulting mean-shift equation using the Gaussian kernel is:

$$m(x) = \frac{\sum_{i=1}^{n} x_i exp^{-\frac{(x-x_i)^2}{2\sigma^2}}}{\sum_{i=1}^{n} exp^{-\frac{(x-x_i)^2}{2\sigma^2}}} \tag{3}$$

The term *m(x)* is the new location of point *x* while the distance traveled by the point from location *x* to *m(x)* is coined as the "mean-shift" distance.

III. Hardware Architecture

A. Approach

We have chosen to focus on the Gaussian mean-shift (GMS) algorithm for several key reasons. GMS is primarily a non-parametric and unsupervised clustering technique. This advantage makes it a perfect candidate for an autonomous setting where human guidance is restricted. GMS does not require any prior information with respect to the number of clusters, their shape, or the distribution of the data. The mean-shift algorithm is also embarrassingly parallel due to its high granularity, and can be efficiently mapped to hardware for faster execution without sacrificing performance. Pixels move in the direction of the estimated PDF gradient at every iteration, but the gradient vector estimation (Eq. 3) can be evaluated at every pixel location in parallel. This key advantage is leveraged through our hardware design.

Each pixel in the dataset maps to a unique point in the feature space based on its grayscale value, along with an x and y coordinate location (*pixel(x, y, grayscale)*). Even though this is an iterative algorithm, as pixels in the feature space converge toward their respective clusters, their new locations are never updated in the original image (as in the blurring GMS). The gradient vector is estimated based on the pixel distribution of the original image, allowing us to achieve hazard-free parallelization across multiple devices as long as each device has a copy of the entire image in memory. Each FPGA could thus segment a different region of the image without encountering boundary problems.

By successfully exploiting the inherent mean-shift parallelism, we have designed a scalable hardware architecture capable of clustering hundreds of pixels in parallel. Each pixel benefits from its own dedicated pipeline and moves independent of all other pixels towards its cluster. This approach can be easily scaled to incorporate multiple FPGAs without incurring significant overhead as can be observed in the experimental results section. By parallelizing the mean-shift algorithm and clustering regions of the image consisting of hundreds of pixels in parallel, we have leveraged the advantages of reconfigurable computing and efficiently decreased computational complexity (loop unrolling).

B. Pipeline Architecture

Image segmentation requires the pairwise distance computations of all pixels for evaluating the PDF. Since we utilize the Gaussian kernel to estimate the PDF we designed a hardware block (Fig. 1) that can quickly compute squared Euclidean distances.

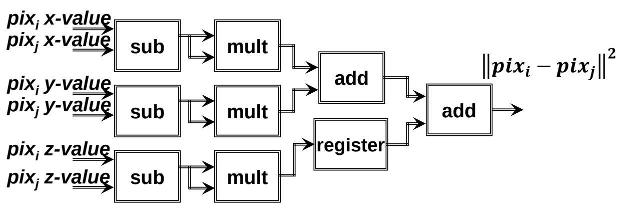


Fig. 1. Pipelined Euclidean distance hardware diagram.

The pipelined design allows a new pixel to be input every clock cycle, and enables us to efficiently stream pixel values from memory without stalling. The Euclidean distance block is further integrated in the pipeline design for evaluating the Gaussian kernel. Since the architecture features a 32-bit fixed-point implementation the exponential function used in Eq. 2 is quantized to a finite look-up table. Fig. 2 shows a graph of our quantized (10 values) exponential approximation overlapping the true analog function.

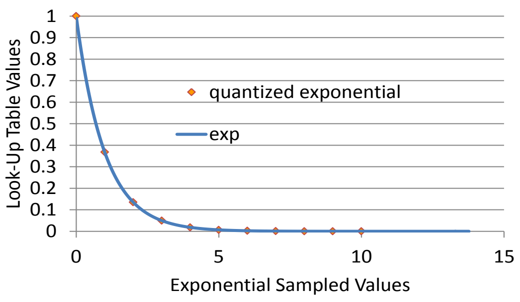


Fig. 2. Look-up table estimation of exponential function.

Fig. 3 shows a complete diagram of the 32-bit fixed-point pipeline. The input *pixel_i* (*x*, *y* and *z* value) is the pixel being moved by the pipeline to a new location, while *pixel_j* (*x*, *y* and *z* values) represents streaming pixel values from memory. All computational blocks are fully pipelined and can handle a steady stream of input pixels from memory. The four outputs in Fig. 3 represent the numerator and denominator values of Eq. 3. By dividing the first three outputs with the last output, we obtain the new *x*, *y* and *z* locations of the "*mean-shifted"* pixel (Eqs. 4, 5, 6).

$$x_j^{new} = \frac{\sum_{i=1}^{N} x_i * exp^{-\left(\frac{\|pix_i - pix_j\|^2}{2\sigma^2}\right)}}{\sum_{i=1}^{N} exp^{-\left(\frac{\|pix_i - pix_j\|^2}{2\sigma^2}\right)}} \tag{4}$$

$$y_j^{new} = \frac{\sum_{i=1}^{N} y_i * exp^{-\left(\frac{\|pix_i - pix_j\|^2}{2\sigma^2}\right)}}{\sum_{i=1}^{N} exp^{-\left(\frac{\|pix_i - pix_j\|^2}{2\sigma^2}\right)}} \tag{5}$$

$$z_j^{new} = \frac{\sum_{i=1}^{N} z_i * exp^{-\left(\frac{\|pix_i - pix_j\|^2}{2\sigma^2}\right)}}{\sum_{i=1}^{N} exp^{-\left(\frac{\|pix_i - pix_j\|^2}{2\sigma^2}\right)}} \tag{6}$$

Each pipeline could have its dedicated fixed-point divider to compute the results shown in Eqs. 4, 5 and 6. However, the hardware resources required by a fixed-point divider would pose considerable restrictions on the number of pipelines that

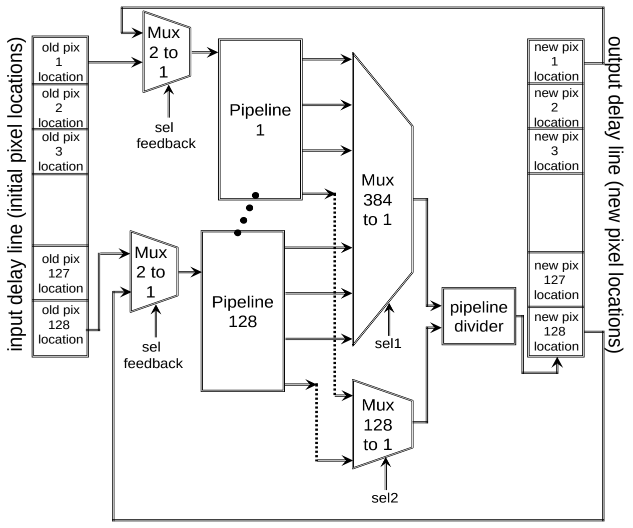


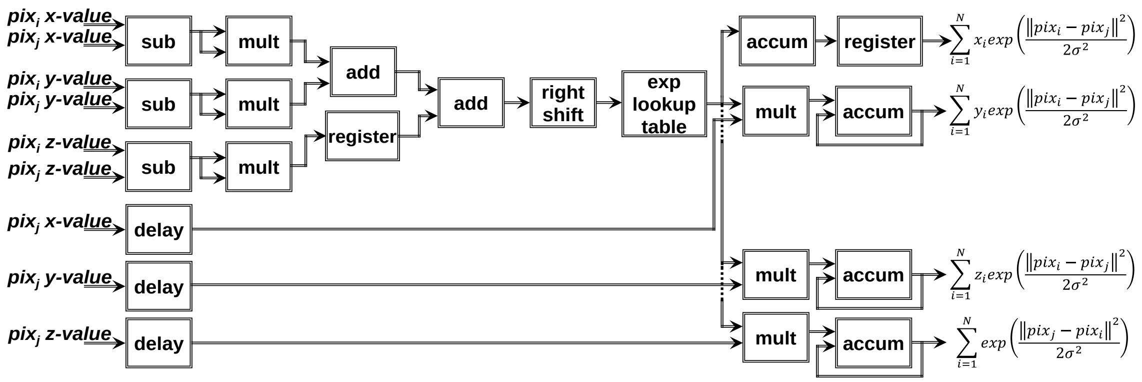
Fig. 4. Feedback loop closed for running multiple iterations.

would fit on the FPGA. Fig. 4 shows a more conservative approach for the hardware resource utilization. The pipelined 32-bit divider is shared to service all pipelines with the addition of two extra multiplexers. The multiplexers serve the purpose of delaying each pipeline output by one clock cycle. This approach helps us save considerable hardware resources while only adding a negligible time penalty (less than 0.1% of the overall execution time).

The application is scaled by replicating the pipelines to efficiently allocate all the available FPGA hardware resources. Each pipeline is dedicated to servicing the movement of one pixel. The last remaining step is to configure the pipelines for multiple iterations by closing the feedback loop from the output registers back to the pipeline input registers (Fig. 4).

C. Dataflow

The algorithm follows a repetitive pattern. The pixels being moved towards their modes are read from memory and stored in the onboard input delay registers (Fig. 4). The entire image is then streamed from memory to every pipeline (low memory bandwidth needed) and the output results are stored in the output delay registers. This process is repeated for multiple iterations until every pixel converges to a mode. A new batch of pixels is then read from memory until the entire image is serviced.



pix_i x-value
pix_j x-value
pix_i y-value
pix_j y-value
pix_i z-value
pix_j z-value
pix_j x-value
pix_j y-value
pix_j z-value
sub
mult
add
register
add
right shift
exp lookup table
delay
accum
register
mult
accum
$\sum_{i=1}^{N} x_i exp\left(\frac{\|pix_i - pix_j\|^2}{2\sigma^2}\right)$
$\sum_{i=1}^{N} y_i exp\left(\frac{\|pix_i - pix_j\|^2}{2\sigma^2}\right)$
$\sum_{i=1}^{N} z_i exp\left(\frac{\|pix_i - pix_j\|^2}{2\sigma^2}\right)$
$\sum_{i=1}^{N} exp\left(\frac{\|pix_j - pix_i\|^2}{2\sigma^2}\right)$

Fig. 3. Pipeline block diagram.

D. *Fast and accurate exploration of design space*

In previous subsections, we described the hardware architecture for image segmentation that is scalable to multiple FPGAs. Some readers may notice that our design is not fully optimized (e.g., pipeline stalls while loading a new batch of dedicated pixels). We made this informed decision to focus on more productive design choices from the entire design space.

Table 1. Design paths based on 1-FPGA 1-iteration design

Steps	0	1	2	3	4
Path 1	1 Iter.; 1 FPGA	5 iter.; 1 FPGA;	5 iter.; 2 FPGAs	5 iter.; 4 FPGAs	5 iter.; Apply path 2; 4 FPGAs;
Path 2		Double-buffer pipeline outputs	Double-buffer pipeline dedicated pixel input	Stall-less pipelines	N/A
Path 3		2 FPGA; 1 iter.	Broadcast image to 2 FPGAs	N/A	N/A
Path 4		4 FPGA; 1 iter.	Broadcast image to 4 FPGAs	N/A	N/A

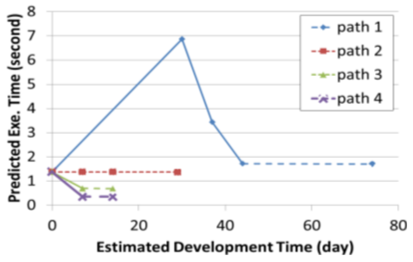


Fig. 5. Predicted execution time of design paths.

We created abstract models that contain timing parameters and high-level behavioral code. Using these models we estimated the execution time. We identified four design paths that were considered as shown in Table 1. Every path consists of several steps, each of which represents a distinct feature added to the design, including new functionality such as multi-iteration (e.g., path 1), and optimizations such as double-buffering (e.g., path 2). We then predicted execution times for every step along each path. Combining this information, we created the productivity curve for each path as shown in Fig. 5. We concluded that path two (optimization of the pipelines), does not provide much speedup to either 1-iteration or 5-iteration designs and that scaling to multiple FPGAs clearly provides the most productive way of achieving higher performance.

IV. Experimental Results

A. *Platform Description*

The RC platform used to test the performance of our proposed architecture is Novo-G, a reconfigurable supercomputer housed at the NSF Center for High-Performance Reconfigurable Computing (CHREC) [11]. The FPGAs targeted for implementing the scalable mean-shift architecture are Altera Stratix-III E260s as part of the GiDEL PROCStar III quad-FPGA boards. This family of FPGAs features 768 18×18 multipliers and 256K logic elements, with 4.25GB of dedicated memory in three parallel banks. The software baseline is coded in C, compiled using GCC with optimization –O4, and executed on an AMD 2.26 GHz Xeon E5520 with 4GB of DDR400 RAM.

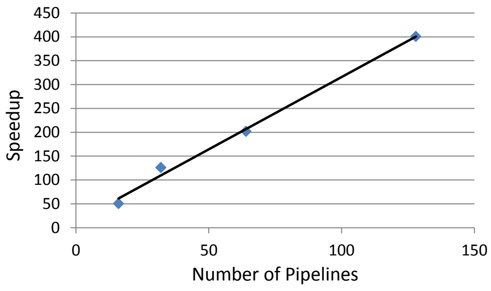


Fig. 6. Speedup achieved as number of pipelines are increased.

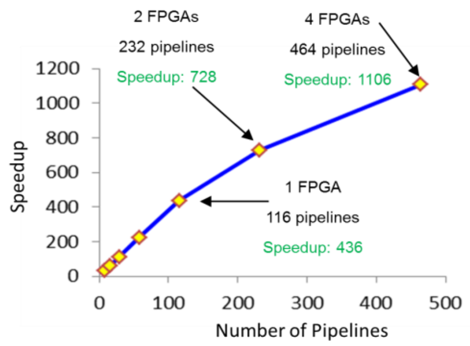


Fig. 7. Speedup achieved for multi-FPGA image segmentation.

B. *Performance Results*

Our architecture first incorporates eight pipelines and is gradually scaled by increasing this number up to a maximum of 128 pipelines on one FPGA. From our compilation results we conclude that our limiting resource is the DSP block 18-bit elements. A maximum of 128 pipelines could fit on one Stratix III E260 FPGA allowing us to effectively compute the PDF gradient at 128 different locations in parallel and cluster 128 pixels at the same time. The two contributing factors to the achieved performance are the pipelined architecture (deep parallelism), along with the parallelized pipeline approach (wide parallelism). Not only can we cluster multiple pixels in parallel but we can also clock in a new pixel from memory to each pipeline without stalling. Fig. 6 shows the correlation between the number of pipelines and the speedup achieved on one FPGA. As expected the speedup is linearly dependent on the amount of loop unrolling.

As we continue scaling our architecture beyond one FPGA, the overhead associated with the host-to-FPGA data transfers prevent the linear trend in speedup observed for one FPGA. Fig. 7 shows the speedup increase as multiple FPGAs are used to tackle the segmentation task. We can conclude that our proposed hardware architecture accelerates the mean-shift algorithm over 1000 times using four FPGAs and can continue to be further scaled to incorporate more FPGAs due to its low overhead characteristics. Speedup is not only increased through scaling the architecture to cluster more pixels in parallel, but also by running multiple iterations inside a pipelined architecture.

C. *Segmentation Visual Results*

For testing the performance of our fixed-point architecture we segmented two grayscale images of various complexities and resolutions, and compared our hardware

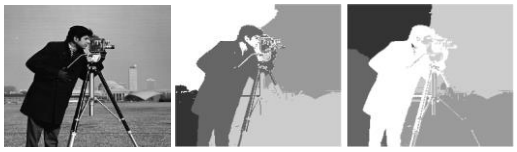


Fig. 8. Left to right: Cameraman: 256 by 256 8-bit grayscale image, floating-point segmentation and fixed-point segmentation result.

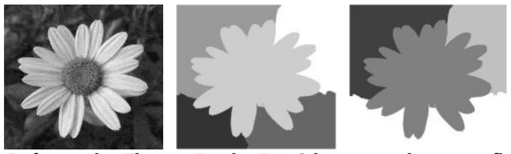


Fig. 9. Left to right: Flower: 512 by 512 8-bit grayscale image, floating-point segmentation and fixed-point segmentation result.

fixed-point results to a floating-point software implementation. We then calculated the mean squared error (MSE) between the final location of the pixels in the floating-point implementation and our hardware fixed-point architecture. The MSE values of 187 for the cameraman 8-bit 256 by 256 grayscale image and 100 for the flower 8-bit 512 by 512 grayscale image are then compared to the radius of the smallest cluster. For the cameraman image the smallest cluster radius is 213 while for the flower image the smallest radius is 202. In both cases the MSE values are smaller than the radii of the smallest clusters, meaning that our fixed-point quantization errors do not significantly affect the final segmentation results. Another measure that quantifies our clustering accuracy is the misclassification error. The misclassification errors for the two images prove that less than 10% of pixels converge to a different cluster in hardware and software. For each segmentation experiment, the kernel size is fixed to 128, while each pixel is clustered over 32 iterations. Figs. 8 and 9 show each grayscale image next to the segmented results for both our fixed point architecture and a software floating point implementation. In both the software and hardware segmentation figures, each cluster is colored in using a different grayscale value. Since we employed a large kernel size (128) we expect the PDF to be relatively smooth with few maxima resulting in relatively low number of segments. Also note that the main differences in Figs. 8 and 9 are in the way the background is segmented and not the foreground objects.

V. Future Work

The current architecture has been scaled to run on a maximum of four Stratix III E260 FPGAs on the GiDEL PROCStar III board. However, our speedup results show that the increase in overhead associated with a multi-FPGA implementation is marginal, and that the architecture can be further scaled for increased speedup on multiple boards. Our future work will focus on scaling the mean-shift architecture beyond the current board-level design. Also, since speedup is dependent on the number of pipelines we can fit on every FPGA, we will target larger Stratix IV and V FPGAs.

VI. Conclusions

The mean-shift algorithm provides a non-parametric and unsupervised clustering technique that is used in a multitude of applications ranging from object recognition, to tracking and quality control to name just a few. Image segmentation is a low-level image processing task that can be solved using mean-shift clustering. Although the mean-shift approach yields good results for image segmentation, its computational challenges have prohibited its impact in the image processing domain, with runtimes that have precluded real-time applications. This paper proposes a scalable architecture that accelerates the mean-shift by allocating dedicated hardware to clustering hundreds of pixels in parallel. Due to the high granularity of the mean-shift algorithm, we can evaluate the PDF gradient vector at different locations of the feature space in parallel, which allows us to move the pixels in parallel toward their respective clusters. Our scalable architecture consists of fixed-point 32-bit pipelines that are replicated to effectively utilize all hardware resources of the FPGA fabric. Our proposed architecture is tested on a PROCStar III board using all four Stratix III E260 FPGAs. The maximum achieved speedup for a 464 by 332 8-bit grayscale image is 1164, while the small overhead penalty associated with scaling the architecture to incorporate multiple FPGAs shows that our design can be further scaled to span multiple boards for better speedup results.

VII. References

[1] K. Fukunga, L. D. Hosteler, "The Estimation of the Gradient of a Density Function, with Application in Pattern Recognition," *IEEE Trans. Information Theory (IT)*, Vol. 21, Issue. 1, pp. 32-40, Jan. 1975

[2] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 17, No. 8, pp. 790-799, Aug. 1995.

[3] D. Comaniciu, P. Meer, "Mean Shift Analysis and Applications," *Proc. Seventh International Conference on Computer Vision*, pp 1197-1203, Sept. 1999

[4] M. J. Deilamani, R. N. Asli, "Moving Object Tracking Based on Mean Shift Algorithm and Features Fusion," *International Symposium on Artificial Intelligence and Signal Processing*, pp. 48-53, June 2011.

[5] P. Bai, C. Fu, M Cao, Y. Han, "Improved Mean Shift Segmentation Scheme for Medical Ultrasound Images," *Fourth International Conference on Bioinformatics and Biomedical Engineering (iCBBE)*, pp. 1-4, June 2010

[6] A. Yamashita, Y. Ito, T. Kaneko, H. Asama, "Human Tracking with Multiple Cameras Based on Face Detection and Mean Shift," *IEEE International Conference on Robotics and Biometrics*, pp. 1664-1671, Dec. 2011.

[7] W. Al-Nuaimy, Y. Huang, A. Eriksen, V. T. Nguyen, "Automatic Feature Selection for Unsupervised Image Segmentation," *Applied Physics Letters*, Vol. 77, Issue: 8, pp. 1230-1232, Aug. 2000.

[8] J. Zhang, S. Luo, X. Liu, "Weighted Mean Shift Object Tracking Implemented on GPU for Embedded Systems," *International Conference on Control Engineering and Communications Technology (ICCECT)*, pp. 982-985, Dec. 2012.

[9] H. Wang, J. Zhao, H. Li, J Wang, "Parallel Clustering Algorithms for Image Processing on Multi-Core CPUs," *International Conference on Computer Science and Software Engineering*, Vol. 3, pp. 450-453, 2008.

[10] U. Ali, M. B. Malik, K. Munawar, "FPGA/Soft-Processor Based Real-Time Object Tracking System," *Fifth Southern Conference on Programmable Logic (SPL)*, pp 33-37, April 2009.

[11] A. George, H. Lam, and G. Stitt, "Novo-G: At the Forefront of Scalable Reconfigurable Computing*," IEEE Computing in Science & Engineering (CiSE)*, Vol. 13, No. 1, Jan/Feb. 2011, pp. 82-86.