J Real-Time Image Proc (2018) 14:379–394
https://doi.org/10.1007/s11554-014-0459-1

ORIGINAL RESEARCH PAPER

# A real-time, power-efficient architecture for mean-shift image segmentation

**Stefan Craciun · Robert Kirchgessner · Alan D. George · Herman Lam · Jose C. Principe**

Received: 30 April 2014 / Accepted: 29 September 2014 / Published online: 16 October 2014


**Abstract** Image segmentation is essential to image processing because it provides a solution to the task of separating the objects in an image from the background and from each other, which is an important step in object recognition, tracking, and other high-level image-processing applications. By partitioning the input image into smaller regions, segmentation performs the balancing act of extracting the main areas of interest (objects and important features) that further help to interpret the image, while remaining immune to irrelevant noise and less important background scenes. Image-segmentation applications branch off into a plethora of domains, from decision-making applications in computer vision to medical imaging and quality control to name just a few. The mean-shift algorithm provides a unique unsupervised clustering solution to image segmentation, and it has an established record of good performance for a wide variety of input images. However, mean-shift segmentation exhibits an unfavorable computational complexity of $O(kN^2)$, where $N$ represents the number of pixels and $k$ the number of iterations. As a result of this complexity, unsupervised image segmentation has had limited impact in autonomous applications, where a low-power, real-time solution is required. We propose a novel hardware architecture that exploits the customizable computing power of FPGAs and reduces the execution time by clustering pixels in parallel while meeting the low-power demands of embedded applications. The architecture performance is compared with existing CPU and GPU implementations to demonstrate its advantages in terms of both execution time and energy.

**Keywords** FPGA · Reconfigurable computing · Hardware acceleration · Mean-shift · Unsupervised clustering · Image segmentation · Gradient density estimation

## 1 Introduction

The mean-shift algorithm was proposed by Fukunaga and Hostetler [1] as a non-parametric clustering technique based on the concepts of kernel density estimation previously developed by Parzen [2]. The motivating factor behind this algorithm was the lack of information available about the true probability-density function. Fukunaga used the existing Parzen approach to obtain a differentiable density estimate and then computed the gradient of the density. The mean-shift method uses the gradient estimate of the probability-density function (PDF) as a clustering force that moves the data points in a given dataset towards the closest density peak (mode). Every point moves by an amount equal to the density gradient estimated at that location over multiple iterations until all data points converge to their closest mode. The mean-shift algorithm has been intuitively called a hill-climbing process, where the hills are formed by the surface of the PDF, the peaks are the modes of the dataset, and all points move in the direction of the estimated gradient (uphill).

S. Craciun (✉) · R. Kirchgessner · A. D. George · H. Lam
Department of Electrical and Computer Engineering, NSF Center for High-Performance Reconfigurable Computing (CHREC), University of Florida, Gainesville, FL 32611-6200, USA
e-mail: craciun@hcs.ufl.edu

J. C. Principe
Computational Neuro-Engineering Laboratory (CNEL), Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611-6200, USA

### 1.1 Background

The mean-shift algorithm has made a considerable impact in the clustering domain because it provides a non-parametric solution and does not require any prior knowledge about the distribution of the input data. The mean-shift algorithm has been labeled unsupervised and as a result can be used for autonomous applications, where the user is not required to provide any input parameters. In 1995, Cheng [3] established a rigorous mathematical background and gave the gradient estimation method its recognized *mean-shift* name. Fukunaga and Hosteler originally applied their algorithm to kernel smoothing and signal de-noising. It was not until the early 2000s that the mean-shift algorithm was applied to image processing. The work of Comaniciu and Meer [4–6] demonstrated the performance advantages of the mean-shift algorithm by efficiently applying it to segment images, track objects, and find contours/edges. The mean-shift algorithm has since been primarily applied to the image-processing domain because it provides an unsupervised solution to image segmentation. Mean-shift segmentation has been successfully used in medical imaging for volume calculation, malign tissue localization [7], and detection of functional connectivity in the brain [8]. In computer vision, image segmentation has been used as the initial step for object recognition such as face recognition [9] and tracking [10, 11]. In quality control, image segmentation using the mean-shift algorithm has been proven to be efficient at locating wafer defects [12].

The goal of image segmentation is to cluster all pixels to discrete locations, and thus quantize the input image into pixel subsets that share common visual characteristics such as a distinct structure, color, or texture. Using the mean-shift algorithm, the forefront objects are separated by contrast from a uniform background. This method helps in analyzing and interpreting the image, transforming it from a random collection of pixels to a unique arrangement of recognizable objects. However, the main challenge that impedes the impact of the mean-shift algorithm for embedded applications is its computational complexity. The creators of the algorithm, Fukunaga and Hostetler, recognized from the very beginning that their algorithm "*may be costly in terms of computer time and storage*." Their assessment holds true even for current computational platforms as a result of the ever-increasing resolution of images, a well-known and documented trend for the past decade [13].

In this paper, we design a scalable, power-efficient, and pipelined hardware architecture that addresses the prohibitive execution time challenge of the mean-shift algorithm. Our architecture exploits the intrinsic deep and wide parallelism of the algorithm. We leverage the inherent coarse granularity of the mean-shift algorithm by dedicating independent hardware pipelines to cluster multiple pixels in parallel (wide parallelism). The hardware architecture transforms the mean-shift algorithm into a streaming application, allowing pairwise interactions between pixels to accumulate every clock cycle (deep parallelism) and, as a result, accelerate the estimation of the PDF gradient. The platform used is ideal for the algorithm's fine granularity, consisting of a gate-array fabric on which the pipelines can be replicated, so that each one can process individual pixel movement over multiple iterations. Our approach is to first replicate the pipelines to effectively utilize all the resources on one FPGA, and then further scale-up to a board-level architecture (four coupled FPGAs). We compare the execution time and power consumption of our fixed-point architecture with the most up-to-date GPU implementations of image segmentation [14–19] and also a CPU software baseline. The ideal combination of superior execution times coupled with considerably lower power makes our design well suited for embedded autonomous applications.

The organization of the paper proceeds by first exploring the related research (Sect. 1.1), and then providing the mathematical background and a brief overview of the baseline algorithm (Sect. 2). In Sect. 3, the proposed parallel architecture is presented in detail. Sect. 4 features numerical and visual results, and in Sect. 5 we draw conclusions and key insight from this work.

### 1.2 Related research

From an information-theoretic perspective, the mean-shift algorithm has been proven to minimize Renyi's cross entropy [20]. The cross information forces exerted by the original dataset on each sample help cluster the points around the modes. Renyi's cross entropy [21] reaches its minimum value when all points have converged to their respective modes. Furthermore, the mean-shift algorithm has an important advantage over other algorithms based on gradient-ascent or decent cost functions. The step size, which is a free parameter associated with gradient-based algorithms, does not exist. The step-size parameter is embedded in the mean-shift method, and thus ensures that points move faster during the initial iterations and slow down as they approach their modes. The well-known problem of step-size selection is thus avoided. Multidimensional clustering has been a major topic of study in statistical data analysis [22, 23] with numerous research papers proposing theoretical methods of accelerating the clustering process. These methods include techniques to reduce data-dimensionality, improvements to distance functions, and techniques for stopping before the final convergence. Other authors have transitioned from mainstream computing platforms to many-node CPUs [24] and GPUs [14–19] in an effort to map the clustering

algorithms to hardware more efficiently and leverage the intrinsic parallelism. Recently, efforts to accelerate the mean-shift algorithm have focused on FPGA technology and have achieved the most promising results to date for embedded applications [25]. However, FPGA implementations of the mean-shift algorithm have exclusively targeted tracking applications rather than image segmentation. In contrast to segmentation applications, which cluster the entire image pixel set, tracking applications require only a small subset of pixels in the input image to be clustered. The subset corresponds to the pixels inside a rectangular frame centered over the object that is being tracked. FPGA architectures have been successfully used to track objects in real time as part of embedded system applications [26–28]. However, despite these efforts, the trend of increasing image resolutions makes the mean-shift algorithm impractical for embedded real-time image-segmentation applications when image sizes exceed VGA resolution (640 by 480). The mean-shift algorithm scales poorly with both the number of pixels $N^2$ and number of iterations ($k$) as $O(kN^2)$. Although performance is reliable for a wide range of input images yielding good segmentation results, the execution time continues to prohibit this algorithm from approaching the real-time barrier for images above VGA resolution. Furthermore, the power required to segment images on conventional CPU or GPU platforms has impeded the algorithms impact in the embedded arena where it is most needed [29]. Our proposed solution leverages the fine granularity of the algorithm by evaluating the PDF gradient at multiple data points in parallel, and thus clusters multiple pixels in parallel. The FPGA fabric provides the ideal low-power computational platform on which pixels can be processed in parallel, resulting in a decrease in computational complexity and real-time segmentation of images above VGA resolution.

## 2 Mathematical background

In 1995, Cheng et al. [3] coined the gradient estimation of a density function proposed by Fukunaga and Hostetler [1] as the "*mean-shift*" algorithm. This term is now exclusively used when referring to clustering methods based on the gradient estimation. The term "*mean-shift*" has endured the test of time because it describes the algorithm in a very simple and intuitive form. The data points, which in our case refer to pixel values, move in an iterative process or "*shift*" towards the closest local density peak or local "*mean*." In the research literature, some authors have described the mean-shift algorithm as a mode-seeking algorithm [3] because the data points cluster around the nearest density maxima. Other authors have intuitively called the mean-shift a hill-climbing technique [30]. The hills refer to the density surface of the data that exhibits peaks at locations where points have a high density, and is flat in regions where data points are sparse. On this probability surface, each data point travels in the direction of the estimated gradient, up the probability hill, until it reaches the closest peak and becomes stable.

### 2.1 Mathematical framework

We start by describing the Parzen window technique, which is the most popular method to evaluate the density function of a random variable. The multivariate kernel density estimator for a 3D space consisting of $n$ samples and kernel size $h$ is shown in Eq. (1):

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

For radially symmetric kernels, $K(x) = c_{k,3}k(\|x\|^2)$, where $c_{k,3}$ is the 3D normalization constant ensuring that $K(x)$ integrates to 1. The kernel density can be rewritten as shown in Eq. (2):

$$\hat{f}_{c,k}(\vec{x}) = \frac{c_{k,3}}{nh^3}\sum_{i=1}^{n} k\left(\left\|\left(\frac{\vec{x} - \vec{x_i}}{h}\right)\right\|^2\right) \tag{2}$$

The next step is to find the peaks (modes) of the density function by equating the gradient to zero:

$$\nabla\hat{f}(\vec{x}) = 0 \tag{3}$$

$$\nabla\hat{f}(\vec{x}) = \frac{2c_{k,3}}{nh^3}\sum_{i=1}^{n}(\vec{x} - \vec{x_i})k'\left(\left\|\left(\frac{\vec{x} - \vec{x_i}}{h}\right)\right\|^2\right) = 0 \tag{4}$$

By substituting in the above equation $g(s) = -k'(s)$ the gradient can be written as shown in Eqs. (5) and (6):

$$\nabla\hat{f}(\vec{x}) = \frac{2c_{k,d}}{nh^3}\sum_{i=1}^{n}(\vec{x_i} - \vec{x})g\left(\left\|\left(\frac{\vec{x} - \vec{x_i}}{h}\right)\right\|^2\right) = \tag{5}$$

$$= \frac{2c_{k,d}}{nh^3}\left[\sum_{i=1}^{n} g\left(\left\|\frac{\vec{x} - \vec{x_i}}{h}\right\|^2\right)\right]\left[\frac{\sum_{i=1}^{n}\vec{x_i}\,g\left(\left\|\frac{\vec{x} - \vec{x_i}}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\vec{x} - \vec{x_i}}{h}\right\|^2\right)} - x\right] \tag{6}$$

In Eq. (6), the first term represents the density estimation at point x, while the second term is the mean shift. The most widely used kernel when estimating the PDF is the Gaussian kernel:

$$K(\overrightarrow{x}) = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{\overrightarrow{x}^2}{2\sigma^2}\right)} \tag{7}$$

The kernel size previously labeled $h$ is now substituted by the symbol $\sigma$. The resulting mean-shift formula using a Gaussian kernel is shown in Equation (8):

$$m(\overrightarrow{x}) = \frac{\sum_{i=1}^{n} \overrightarrow{x_i} e^{-\frac{(\overrightarrow{x}-\overrightarrow{x_i})^2}{2\sigma^2}}}{\sum_{i=1}^{n} e^{-\frac{(\overrightarrow{x}-\overrightarrow{x_i})^2}{2\sigma^2}}} \tag{8}$$

For image-segmentation applications, the input data set is represented by grayscale images. The pixels of the input image are clustered in a three-dimensional space, where each pixel has a row and column location ($x$, $y$ coordinate) and a grayscale value ($z$ coordinate).

$$\overrightarrow{\text{pixel}} = [x, y, z] \tag{9}$$

Each data point shifts by an amount equal to the gradient of the PDF estimated at its current location. The amount of pixel movement in each of the three dimensions is expressed in Eqs. (10–12). It is important to note that the pixel movement in each of the three dimensions is independent.

$$x_j^{\text{new}} = \frac{\sum_{i=1}^{n} x_i \exp^{-\frac{\left|\left|\overrightarrow{\text{pixel}_i}-\overrightarrow{\text{pixel}_j}\right|\right|^2}{2\sigma^2}}}{\sum_{i=1}^{n} \exp^{-\frac{\left|\left|\overrightarrow{\text{pixel}_i}-\overrightarrow{\text{pixel}_j}\right|\right|^2}{2\sigma^2}}} \tag{10}$$

$$y_j^{\text{new}} = \frac{\sum_{i=1}^{n} y_i \exp^{-\frac{\left|\left|\overrightarrow{\text{pixel}_i}-\overrightarrow{\text{pixel}_j}\right|\right|^2}{2\sigma^2}}}{\sum_{i=1}^{n} \exp^{-\frac{\left|\left|\overrightarrow{\text{pixel}_i}-\overrightarrow{\text{pixel}_j}\right|\right|^2}{2\sigma^2}}} \tag{11}$$

$$z_j^{\text{new}} = \frac{\sum_{i=1}^{n} z_i \exp^{-\frac{\left|\left|\overrightarrow{\text{pixel}_i}-\overrightarrow{\text{pixel}_j}\right|\right|^2}{2\sigma^2}}}{\sum_{i=1}^{n} \exp^{-\frac{\left|\left|\overrightarrow{\text{pixel}_i}-\overrightarrow{\text{pixel}_j}\right|\right|^2}{2\sigma^2}}} \tag{12}$$

## 3 Hardware architecture

The deep parallelism of the mean-shift algorithm along with its intrinsic fine granularity makes it well suited for hardware acceleration. The algorithm evaluates the PDF gradient at every data point and moves that point in the direction of the gradient. The PDF gradient estimation is computationally independent at each data point, providing a parallelization advantage that can be leveraged through our hardware architecture. Using dedicated hardware pipelines to compute the PDF gradient at multiple data points in parallel allows us to shift pixels simultaneously towards their closest mode. Theoretically, all data points could be clustered in parallel given unlimited hardware resources; however, for existing reconfigurable platforms, this task becomes impossible for higher-definition images where the pixel count surpasses two million. An efficient approach is to divide the input image into smaller sections where all pixels can be clustered in parallel. The size of each section depends on the amount of hardware resources available on the FPGA.

### 3.1 Approach

The concept diagram in Fig. 1 explains the hardware mapping approach. The input image stored in external memory is partitioned into sections that are transferred to Block RAM (BRAM) in consecutive order. The size of the sections depends on the amount of BRAM available on the FPGA. A batch of pixels from BRAM are serviced by an equal number of pipelines. These are the pixels clustered in parallel. Each pipeline shifts its dedicated pixel over multiple iterations from the original position in the input image, represented by its $x$, $y$ coordinates and grayscale

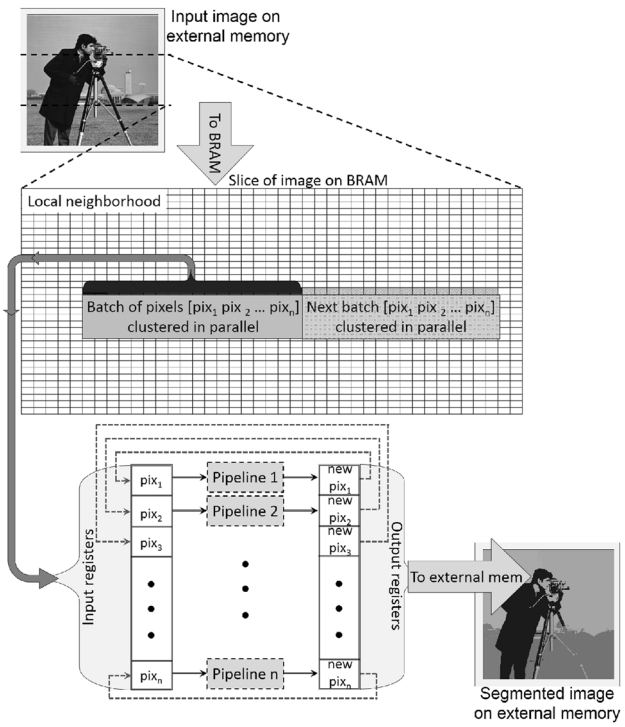



**Fig. 1** Hardware mapping of the mean-shift algorithm using dedicated pipelines to cluster pixels in parallel

value, to the closest mode. When all pixels in the batch have converged to their respective modes, the final locations of the clustered pixels are stored back to an external memory, and the next batch is serviced. In our experimental results section, the mean-shift algorithm is individually mapped to two different FPGAs, first a Stratix-III and then a Stratix-IV from Altera. We show that the larger FPGA can parallelize the clustering task over a larger area of the image. The architecture is then further scaled to include multiple FPGAs using a divide and conquer approach. Our results prove that there is a linear relationship between the number of pipelines used to cluster the pixels in the image and the overall acceleration of the algorithm's execution time. The only requirement for scaling our architecture over multiple FPGAs is that each FPGA must have a copy of the original image.

Another important step of the mean-shift algorithm that tailors the algorithm to hardware acceleration is the Gaussian kernel accumulation defined in Eqs. (10–12). Given a local neighborhood of pixels, the PDF gradient is an accumulation of pairwise kernel computations between the point at which the PDF gradient is calculated ($\text{pixel}_\text{j}$) and all of the pixels in the local neighborhood ($\text{pixel}_\text{i}$) as shown in Equations (10–12). In these equations, the local neighborhood is composed of pixels 1 to $n$. Since the kernel function can be mapped to a pipelined architecture, as is the case for our design, the PDF gradient computation is transformed into a streaming application. All data points that belong to the local neighborhood are streamed from memory, allowing the accumulation of a new pairwise kernel computation at every clock cycle. Furthermore, since the batch of consecutive pixels that are simultaneously clustered share the local neighborhood, the same input data is streamed from memory to each pipeline. All pipelines require the same input stream of data, and, as a result, the memory-bandwidth requirements remain constant as the architecture is scaled to parallelize the movement of more pixels. In summary, the proposed hardware architecture not only clusters pixels in parallel by computing the PDF gradient at different locations simultaneously (wide parallelism), but also transforms the PDF gradient estimation into a streaming application (deep parallelism) that requires a constant memory bandwidth independent of the scaling factor. The fixed-point approach gives us control over the way that we tailor the precision of the pipeline architecture. Without incurring errors, we can use the least number of bits necessary at every stage of the pipeline and maximize the number of pipelines replicated on the FPGA fabric. We propose a novel pipelined architecture, with feedback, capable of running multiple iterations of the Gaussian mean-shift algorithm to accelerate image-segmentation applications.

### 3.2 Pipeline architecture

In this section, we explain the design choices made at every stage of the architecture and describe the main building blocks that are used to construct the pipeline. We start by describing the core of the kernel density function and progress towards the final accumulators and the outputs of the pipeline, which are eventually fed back to the input for iterative clustering.

The first pipeline block is the Euclidean distance calculation. For a three-dimensional space, as is the case with grayscale images, each pixel is described by a 3D vector composed of $x$, $y$ Cartesian coordinates, and an 8-bit $z$ (grayscale) value. The Euclidean distance is chosen as our distance metric because we use the Gaussian kernel for density estimation. The Gaussian kernel, which computes the PDF based on pairwise Euclidean distances, is the most widely employed kernel for density estimation. Other kernel functions that are more easily mapped to hardware, such as the Epanechnikov [1] or the Quartic kernel [1], can also be used. The Euclidean distance block allows us to efficiently compute the pairwise distances between pixels without stalling the pipeline. A diagram of the Euclidean distance is shown in Fig. 2.

The stream of Euclidean distances scaled by the kernel size or bandwidth parameter $\sigma$ is then fed into a fixed-point exponential look-up table (LUT). The exponential function is quantized to discrete values as shown in Fig. 3, allowing us to efficiently estimate the Gaussian kernel without having to integrate a resource-intensive, floating-point function into the pipeline. The kernel size parameter $\sigma$ is hard-coded into the LUT values. We will later analyze the errors incurred using a fixed-point precision LUT rather than the exponential floating-point hardware core.

The Euclidean distance along with the Gaussian kernel LUT are integrated into the overall pipeline architecture shown in Fig. 4. The output of the exponential LUT is the Gaussian kernel estimation as shown in Eq. (7). The exponential LUT output is further multiplied with the $x$, $y$, and $z$ coordinates of all the neighboring pixels before accumulating the products in each of the three dimensions. Equations (14–16) show the products accumulated in each

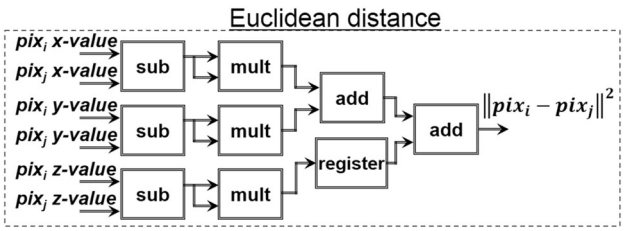



**Fig. 2** Pipelined architecture for Euclidean distance calculation

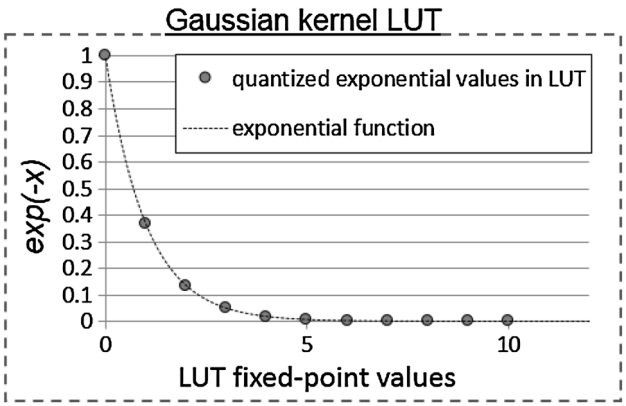



**Fig. 3** Quantized exponential function values for Gaussian kernel LUT

dimension. Each dimension has a dedicated accumulator inside the pipeline. Our architecture takes advantage of this parallelization opportunity, allowing the gradient component in each dimension to be accumulated in parallel. In addition to the three gradient accumulators, the pipeline features a fourth accumulator as shown in Eq. (13), and used for the Gaussian kernel summation. The accumulated value of all Gaussian kernels will be used as a normalization factor for all three dimensions. Equations (13–16) represent the outputs of the pipeline at the end of each iteration. The new $x$, $y$ and $z$ coordinates of the shifted pixel are obtained by dividing the accumulated outputs of Eqs. (14–16) by the accumulated value of Eq. (13).

$$A = \sum_{i=1}^{n} \exp^{-\frac{\left|\left|\text{pixel}_i - \text{pixel}_j\right|\right|^2}{2\sigma^2}} \tag{13}$$

$$B = \sum_{i=1}^{n} x_i \exp^{-\frac{\left|\left|\text{pixel}_i - \text{pixel}_j\right|\right|^2}{2\sigma^2}} \tag{14}$$

$$C = \sum_{i=1}^{n} y_i \exp^{-\frac{\left|\left|\text{pixel}_i - \text{pixel}_j\right|\right|^2}{2\sigma^2}} \tag{15}$$

$$D = \sum_{i=1}^{n} z_i \exp^{-\frac{\left|\left|\text{pixel}_i - \text{pixel}_j\right|\right|^2}{2\sigma^2}} \tag{16}$$

An embarrassingly parallel approach is employed to scale the architecture. The pipeline design in Fig. 4 is replicated on the FPGA fabric to efficiently use all available resources. The complexity of the algorithm is reduced by assigning independent pipelines to cluster multiple pixels simultaneously.

The final normalization stage requires a division. However, the hardware resources required by a fixed-point divider core are relatively large. Dedicating a division hardware core to the output of each pipeline would impose harsh restrictions on the number of pipelines that can fit on the FPGA fabric. The extensive resources required by a division in hardware lead to an important design choice. To maximize the number of pipelines, and thus maximize the number of pixels clustered in parallel, a preferred design choice is to stagger the outputs of the pipelines by employing two multiplexers, and share only one final divider for all the pipelines. The use of the two multiplexers adds a relatively small latency equal to the number of pipelines, but helps to save important resources that are allocated to reducing the algorithms computational complexity. A high-level diagram for the entire pipeline architecture is shown in Fig. 5.

For each pipeline, the two inputs are the streaming pixel values from the local neighborhood (BRAM) and the pixel being clustered. The outputs are the new coordinates of that pixel shifted in the direction of the PDF gradient. The pixels move toward their closest density maxima over multiple iterations, and therefore, the output values at iteration $n$ (new pixel 3D locations) constitute the input values at iteration $n + 1$. A 2-to-1 multiplexer is used at the input of each pipeline to select between the initial starting location of a pixel on the first iteration, and the new output location which is fed back from the pipeline output (Fig. 6).

The mean-shift clustering task is executed over multiple iterations, requiring the input dataset to be streamed through the pipelines multiple times until all pixels have converged to their respective mode. Therefore, it is more efficient to store the section of the image that is reused (local neighborhood used to evaluate the PDF gradient) to BRAM. Reading from BRAM as opposed to external memory banks decreases access latency and increases throughput. The pixels that are moving toward their respective modes need to be accessed every clock cycle; therefore, these values are stored on internal FPGA registers. The following steps summarize the data flow required by the algorithm:

1. *Transfer image from host to FPGA external memory bank A.*
2. *Store portion of image used for PDF gradient estimation to internal memory (BRAM).*
3. *Store pixels being clustered to internal FPGA registers.*
4. *Stream BRAM data over multiple iterations and feedback output results to pipeline inputs.*
5. *Write final pixel location to external memory bank B.*

When the entire image has been processed, the final clustering results are read by the host from memory bank B and used to segment the input image.

## 4 Results and analysis

All experimental results were conducted on Novo-G, a reconfigurable supercomputer developed and hosted by the

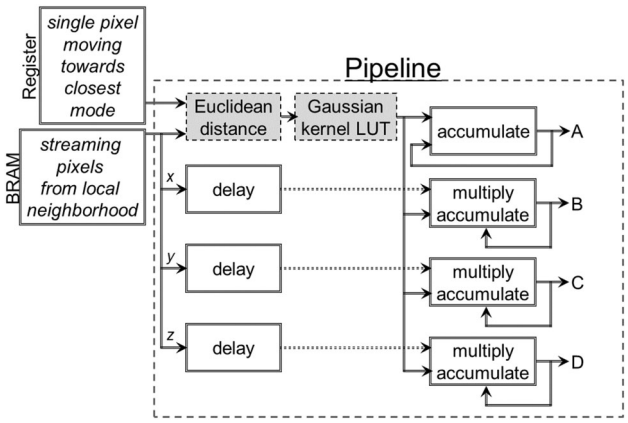



**Fig. 4** Overview of entire pipeline architecture

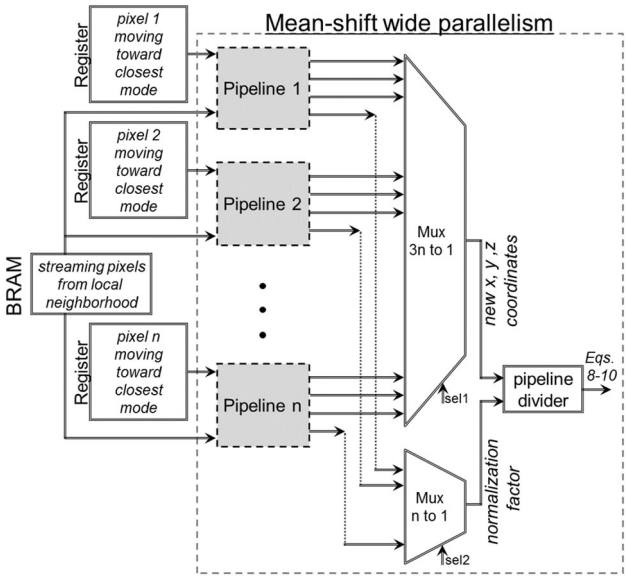



**Fig. 5** Pipelines replicated to cluster individual pixels

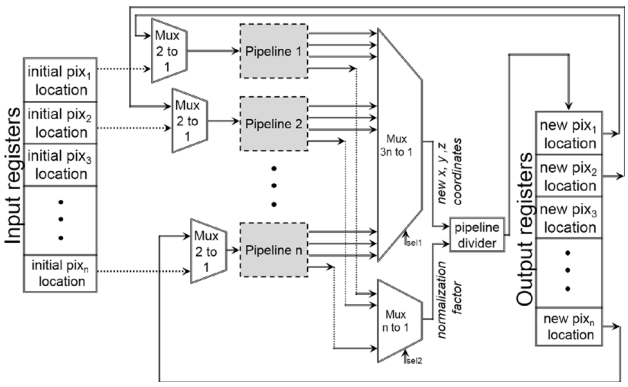



**Fig. 6** Closing the feedback loop for running multiple iterations

NSF Center for High-Performance Reconfigurable Computing (CHREC) at the University of Florida. The FPGAs targeted for implementing the scalable mean-shift architecture were Altera's Stratix-III E260s and Stratix-IV E530s integrated in the GiDEL PROCStar III and PROCStar IV quad-FPGA boards, shown in Fig. 7.

Each Stratix-III E260 (65 nm) features 768 18×18 multipliers and 256 k logic elements, with 4.25 GB of dedicated memory in three parallel banks on the GiDEL PROCStar III board. Each Stratix-IV E530 (40 nm) features 1024 18×18 multipliers and 813 k logic elements, with 8.5 GB of dedicated memory in three parallel banks on the GiDEL PROCStar IV board. Both the overall runtimes and clustering results obtained from our hardware architecture were compared to the most recently published mean-shift clustering results executed on GPU platforms, and also a C software baseline running on one core of an Intel Xeon E5520 processor.

### 4.1 Middleware

To enable portability across heterogeneous FPGAs (Stratix-III and Stratix-IV), and to improve productivity, an RC Middleware (RCMW) [31] was used to help develop our application. Using RCMW, we were able to select the ideal resource interfaces for our mean-shift algorithm pipelines, and describe our application in the RCMW XML metadata format. The RCMW toolset enabled us to generate the top-level application stub without worrying about low-level platform specifics. As a result, we were able to execute our application seamlessly across the GiDEL PROCStar III and PROCStar IV boards.

The RC Middleware provides a set of standardized hardware and software resources to enable productivity and to simplify FPGA application development. On the hardware side, RCMW provides a set of standardized memory–resource interfaces which automatically handle data packing and unpacking, alignment, and flow control. On the software side, RCMW handles the platform initialization, thread management, and provides an object-oriented application representation specific to our application. These hardware and software abstractions reduced the overhead in developing our hardware accelerator and platform driver, improving our productivity.

### 4.2 Experimental results

We performed multiple clustering experiments over a set of images with different resolutions to test the performance of our hardware architecture. The execution times of our hardware architecture were compared to a software C-code baseline, compiled using GCC with O3 optimization and executed on an Intel Xeon E5520 (2.26 GHz) processor with 4 GB of DDR400 RAM. For this comparison, we used images of lower resolution (up to 150 k pixels) and estimated the PDF gradient using the entire data set (every

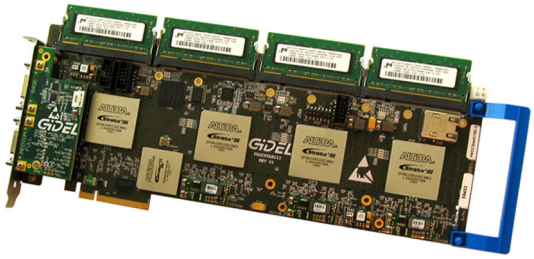


**Fig. 7** GiDEL PROCStar III quad-FPGA board

pixel in the image). We also compared the performance of our hardware architecture to the best execution times obtained from the most recently published GPU image-segmentation implementations [14–19]. Higher-resolution images were used for this comparison (up to 300 k pixels), and the PDF gradient was estimated by only using the pixels within a local neighborhood. Clustering over a local neighborhood is predominantly used for larger images, where pixels that are farther than a given threshold will have a negligible or zero influence on the gradient estimation. The size of the local neighborhood was always chosen as a function of the kernel size, and guaranteed that the estimated gradient would always point toward the closest mode. As expected, the amount of parallelism extracted from the mean-shift algorithm is directly related to the number of hardware resources available on the FPGA. Section 4.2.1 proves that a Stratix-IV FPGA, which has approximately double the amount of hardware resources available on a Stratix-III, will segment images twice as fast. Our experiments proved that a linear relationship exists between the number of hardware resources used to map the mean-shift algorithm on the FPGA fabric and the amount of acceleration achieved. As a result of the scalability of our architecture, the mean-shift algorithm can be accelerated over multiple FPGAs as demonstrated in Sect. 4.2.1. Our experiments predict that a HD image ($1920 \times 1080$) can be segmented in real time using eight Stratix-IV FPGAs, or the equivalent of two GiDEL PROCStar IV boards.

#### *4.2.1 Speedup*

The intrinsic wide parallelism of the mean-shift algorithm allowed us to parallelize the clustering task by moving multiple pixels in parallel. We observed that a linear relationship exists between the number of pixels clustered in parallel using dedicated hardware pipelines and the resulting speedup performance. The graph in Fig. 8 shows the linear relationship between the number of dedicated pipelines replicated on the FPGA fabric to cluster independent pixels, and the speedup obtained with respect to the CPU software baseline.

We continued the embarrassingly parallel approach by scaling our architecture beyond one FPGA. The linear increase in acceleration characterizing the single FPGA implementation exhibited a slight decrease in slope when the architecture was scaled to multiple FPGAs. The sub-linear slope was a consequence of the overhead associated with the multi-FPGA case. For the multi-FPGA scenario, the communication-to-computation ratio increased and prevented the same linear acceleration observed for the single FPGA case. Data transfers from the host (CPU) to the FPGAs interfered with the optimal FPGA execution starting times. Prior to the processing stage, a copy of the input image had to be transferred to each FPGA's external memory bank. As a result, not all FPGAs could start the clustering task at the same time and entered a queuing process until the data transfers were completed. Figure 9 shows the increase in speedup for a multi-FPGA implementation on the GiDEL PROCStar III platform, for up to four Stratix-III E260 FPGAs.

#### *4.2.2 Resource utilization*

Figures 10 and 11 show the percentage resource utilization for a single FPGA implementation as the architecture was scaled to incorporate an increasing number of pipelines on a Stratix-III and -IV, respectively. We can conclude from the resource utilization results that the limiting resources when the architecture was scaled were the DSP block 18-bit elements. The DSP resources were fully utilized after replicating the pipelines 64 times on the Stratix-III FPGA and 128 times on the Stratix-IV FPGA. Using these results, we concluded that an FPGA containing more DSP resources would be a better fit for our scalable architecture. The Stratix-IV FPGA has approximately double the hardware resources of the Stratix-III, allowing further acceleration of the mean-shift algorithm by a factor of 2.

In Fig. 12, our speedup measurements show that one Stratix-IV FPGA achieved approximately the same speedup as two Stratix-IIIs.

#### *4.2.3 Power consumption*

For embedded applications that focus on image-processing algorithms, real-time processing is not the only constraint for a viable solution. Power consumption is another strict requirement that must be met. Our FPGA architecture not only accelerated the mean-shift clustering for real-time segmentation of images surpassing VGA resolution, but also benefited from a minimal increase in power consumption as the architecture was scaled to feature more pipelines. Our experiments are performed on a quad-FPGA board, therefore, we subtract the idle power consumed by the three FPGAs that are not used from the total power

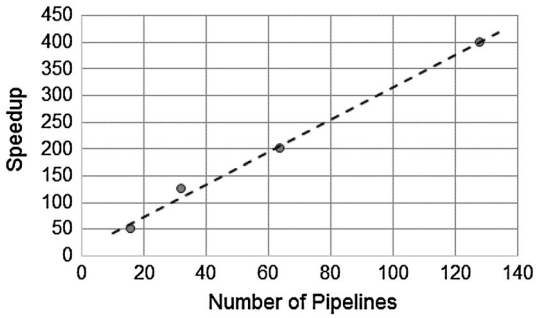



**Fig. 8** Linear increase in speedup (Stratix-III E260 vs. single-core Xeon E5520) vs. number of replicated pipelines

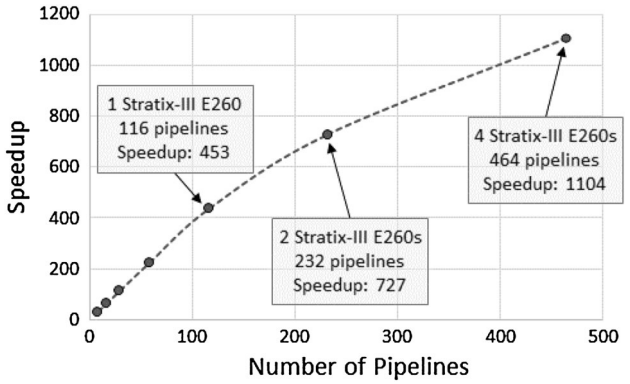



**Fig. 9** Increase in speedup (vs. single-core Xeon E5520) for a multi-FPGA (Stratix-III E260) implementation

consumed by the GiDEL Procstar IV board. We measured the increase in power consumption while scaling our architecture. Figure 13 shows the increase in power as more pipelines were replicated on the Stratix-IV E530 FPGA fabric. The increase in power consumption was minimal ($\approx 4$ Watts) when compared to the amount of pipelines replicated on the FPGA ($\approx 250$ pipelines). The slope in Fig. 14 provides key insight into the efficiency of our hardware architecture. For every Watt consumed by scaling our architecture to incorporate more pipelines, we accelerated the mean-shift algorithm by 288 times. As a result of the achieved speedup-per-Watt ratio, we conclude that our novel architecture is efficiently scaled (from a power perspective) to parallelize the mean-shift algorithm.

#### *4.2.4 Comparison to GPU platforms*

We performed the mean-shift clustering task by evaluating the PDF gradient over a local neighborhood so that we could segment images of higher resolution, and compare our execution time against recently published GPU implementations. GPU platforms have had considerable success in parallelizing fine-grained algorithms [32] that exhibit inherent parallelism. It is not surprising that image

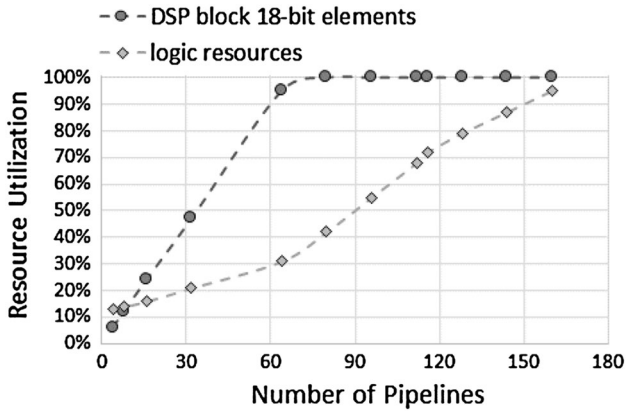



**Fig. 10** Resource utilization as more pipelines are replicated on Stratix-III E260

segmentation has generated much interest in the GPU community, with many authors efficiently mapping the algorithm to GPU platforms [14–19]. When compared to existing CPU software baselines, the mean-shift algorithm has benefitted from a substantial amount of acceleration on GPU platforms [33, 17]. GPUs have successfully driven execution times for images up to VGA resolution ($640 \times 480$) under the real-time threshold (30 fps). However, higher-resolution images have not been segmented in real time. GPUs also consume a relatively high amount of power in comparison to FPGAs and, therefore, FPGAs offer a unique and attractive solution to embedded applications that require low-power consumption. Our architecture can leverage the parallelization opportunities offered by the fine-grained, mean-shift clustering algorithm, while also lowering power consumption.

We compared the execution speed of four recently published image-segmentation GPU implementations including a GeForce 9800 GT (65 nm) [16], GTX 295 (55 nm) [15] and GTX 580 (40 nm) [18] from Nvidia, as well a Radeon HD 5850 (40 nm) [19] from AMD, with the performance of our hardware architecture for different image sizes (Fig. 15).

The execution times for the mean-shift algorithm are relatively similar to both our proposed FPGA architecture, using a single Stratix-IV FPGA (40 nm), and the latest GPU implementations. However, the average power consumption is the distinguishing factor between the two platforms. Our proposed architecture has a relatively low average power consumption even when all resources are completely utilized (34.6 W), while the reported GPU power is on the order of 300 Watts. Given that execution times are similar, normalizing the amount of power consumed to complete the clustering task by the number of pixels will result in the graph shown in Fig. 16. The FPGA architecture outperforms the GPU solutions when power is taken into account. The power advantage of our novel

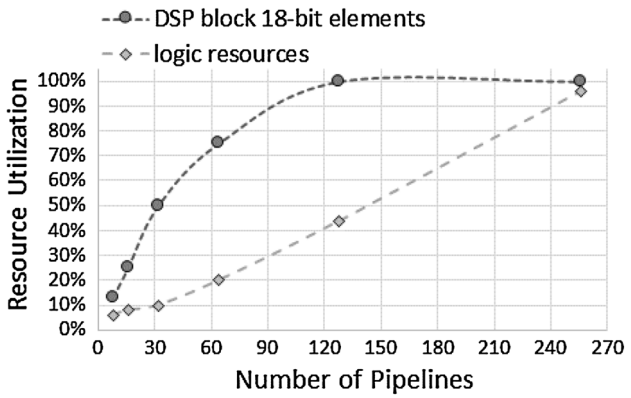



**Fig. 11** Resource utilization as more pipelines are replicated on Stratix-IV E530

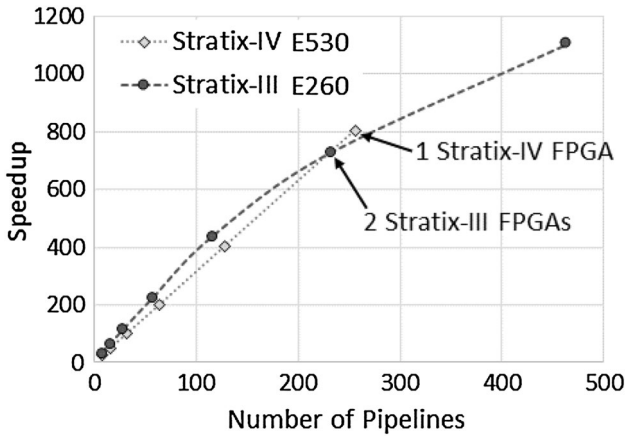



**Fig. 12** Speedup across two FPGA platforms (vs. single-core Xeon E5520)

scalable architecture over GPU implementations demonstrates that our design is a better candidate for embedded applications where power is an essential limiting factor. Furthermore, as shown in Fig. 9, our architecture is scalable beyond a single FPGA, with a close-to-linear speedup increase. According to the trend exhibited in Fig. 9, a HD image (1920 by 1080) can be segmented in real time (30 fps) using eight Stratix-IV FPGAs on two GiDEL PROC-Star IV boards.

### *4.2.5 Future device trends*

The apparent trend in newly emerging GPU and FPGA devices is geared towards combining processing technologies of higher density, higher performance and lower power. For example, the newest Stratix 10 FPGAs (2013) from Altera are built on the Intel 14 nm Tri-Gate process, and offer significantly increased performance with respect to previous generation devices, while lowering power consumption [34]. By contrast, the state of the art GPU

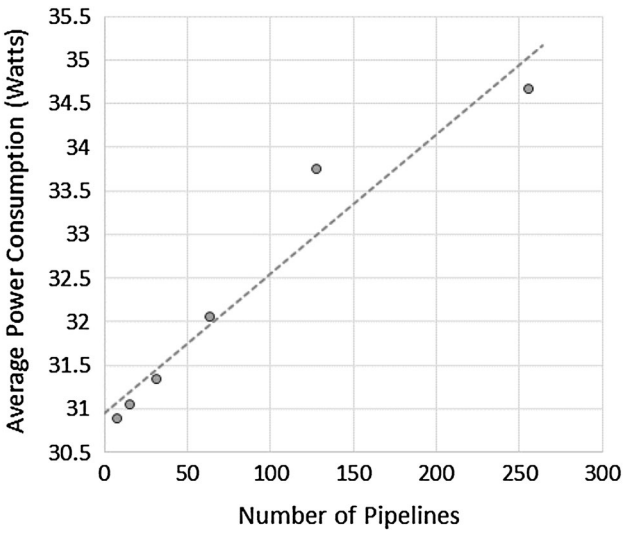



**Fig. 13** Increase in power consumption on Stratix-IV E530 as more pixels are clustered in parallel

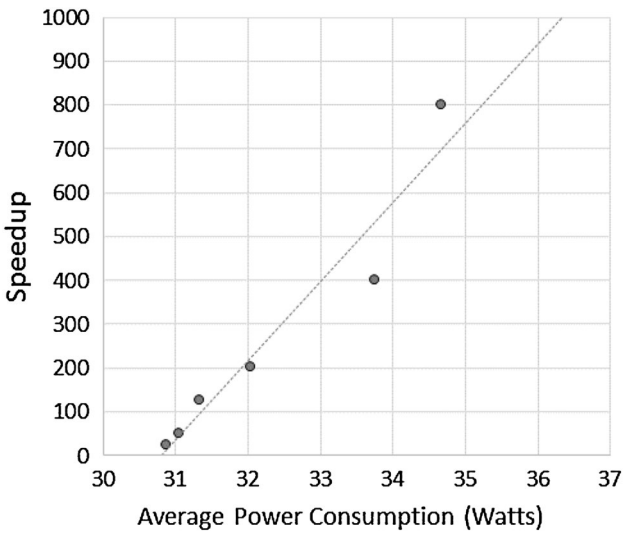



**Fig. 14** Resulting speedup as average power consumption is increased on Stratix-IV E530

technology, such as the Tegra K1 from NVIDIA [35], is currently targeting mobile platforms by maintaining high performance while lowering power consumption. We predict that technological advances in FPGA fabrication could have a considerable impact on our mean-shift architecture. The additional DSP multiplier blocks coupled with the increase in maximum operating frequency could facilitate the extraction of a higher degree of parallelism from the fine-grained, mean-shift algorithm. Our scalable wide parallelism approach, where pipelines are replicated on the FPGA fabric to accelerate execution time, would further benefit from the increase in available resources. However, GPU implementations of the mean-shift algorithm, which

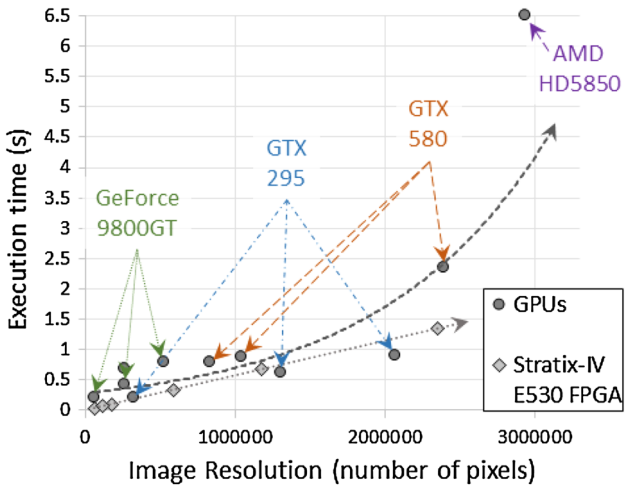



**Fig. 15** Image-segmentation execution times as a function of image resolution

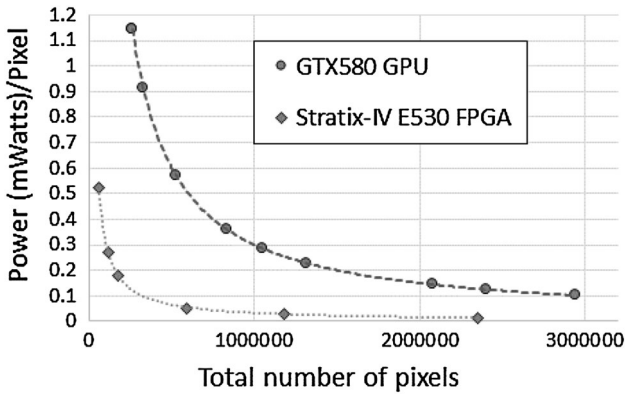



**Fig. 16** Average power consumed for clustering the entire image normalized by the number of pixels in the image

have had limited impact on embedded applications, could now provide an attractive alternate solution to FPGA mean-shift architectures. New lower-power, hybrid GPUs, such as the Tegra K1 from NVIDIA, could be a potential competitor for embedded systems that target real-time segmentation of HD images.

### 4.3 Visual results

Four different grayscale images labeled cameraman, coins, rice, and flowers were used to quantify the clustering accuracy of our hardware architecture. The four images are very different, ensuring that a variety of clustering tasks can be solved using the mean-shift algorithm mapped to our hardware architecture. The clustering tasks range from segmenting small scattered objects over a uniform background to objects that occupy a large surface of the input image in front of a non-uniform background. Each figure below will show the input grayscale image along with the

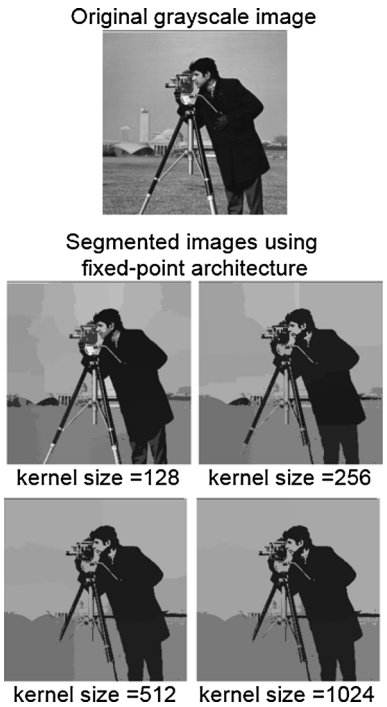



**Fig. 17** Cameraman (large-size object over a uniform background); FPGA segmentation results for different kernel sizes

segmentation results for four different kernel sizes (Figs. 17, 19, 21, 23). 3D snapshots of pixel movements over multiple iterations are shown in Figs. 18, 20, 22 and 24. Each 3D representation tracks the pixels' movement as they progressively converge to the modes of the dataset. The clusters that form in the 3D space represent distinct segments in the original image. The final segmented image is constructed by coloring the pixels in each cluster with their average grayscale color. The kernel size (free parameter) controls the number of modes that are formed. A smaller kernel size will produce a relatively large number of modes, resulting in the image being segmented into many smaller sections, while a larger kernel size ensures that the number of segments remains relatively low.

We also conducted an error analysis to quantify the accuracy of the clustering results. A single-precision, floating-point mean-shift software baseline was used to detect the errors incurred by our fixed-point architecture. Our hardware architecture uses fixed-point cores for all mathematical operations along with a LUT that approximates the exponential function for the Gaussian kernel computation. The mean-squared error (MSE) is used to quantify the extent of the errors resulting from our fixed-point approximations. The MSE measures the difference between the final positions of the clustered pixels resulting from our hardware architecture, and the floating-point mean-shift software baseline. Table 1 records the MSE for

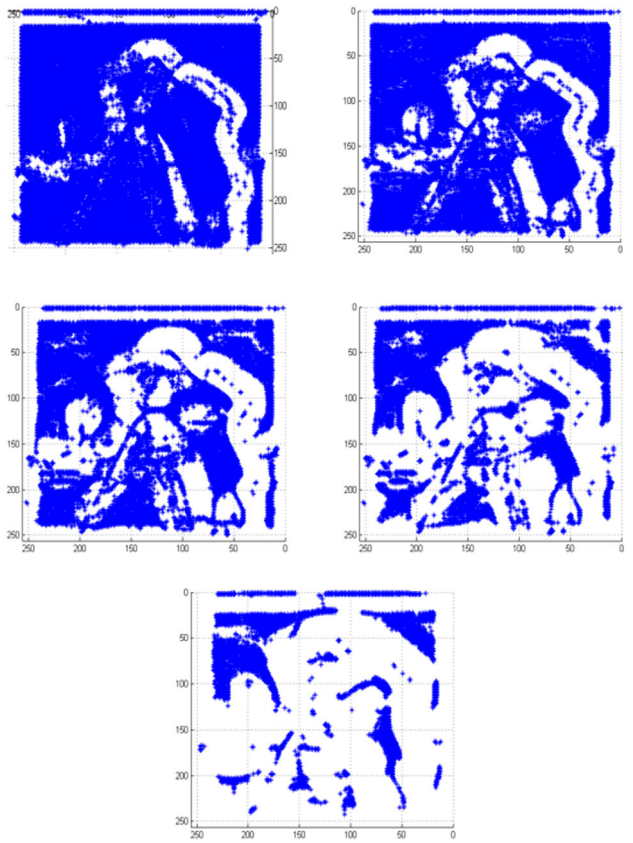


**Fig. 18** Cameraman; pixels clustering in 3D over multiple iterations

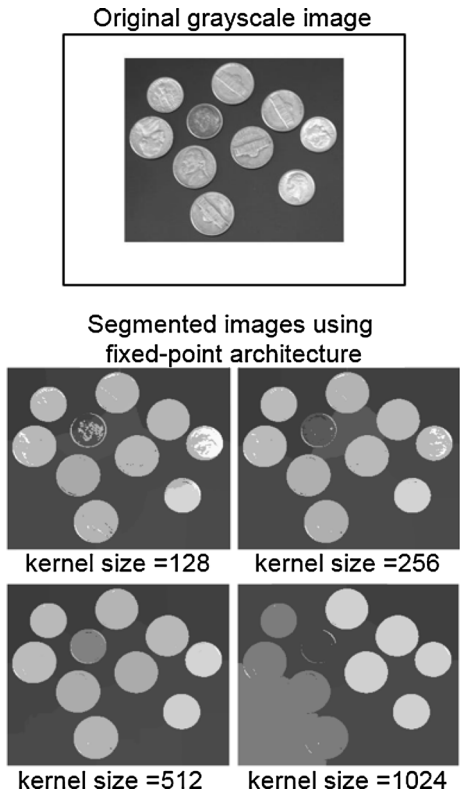



**Fig. 19** Coins (medium-size objects over a uniform background); FPGA segmentation results for different kernel sizes

**Table 1** MSE measured between fixed-point architecture and floating-point software baseline

| Image | MSE | Smallest cluster radius |
|---|---|---|
| Cameraman | 4.2 | 213 |
| Coins | 3.8 | 156 |
| Rice | 3.3 | 57 |
| Flower | 4.6 | 202 |

each of the four grayscale images. The radius of the smallest cluster for each test image is also measured and reported in Table 1. Since the MSE of each segmented image is always less than the radius of the smallest cluster, we conclude that the majority of pixels converge toward the same clusters in hardware and in software. Also, the number of clusters formed and their locations are identical in hardware and software. The differences in final pixel locations that occur between software and hardware result from the quantized distance that pixels travel over each iteration in our fixed-point architecture. Pixels move in the correct direction towards their closest mode, however, the distance traveled is different in software than in our fixed-point architecture. Therefore, even though pixels converge to the correct mode, the final location inside the cluster will

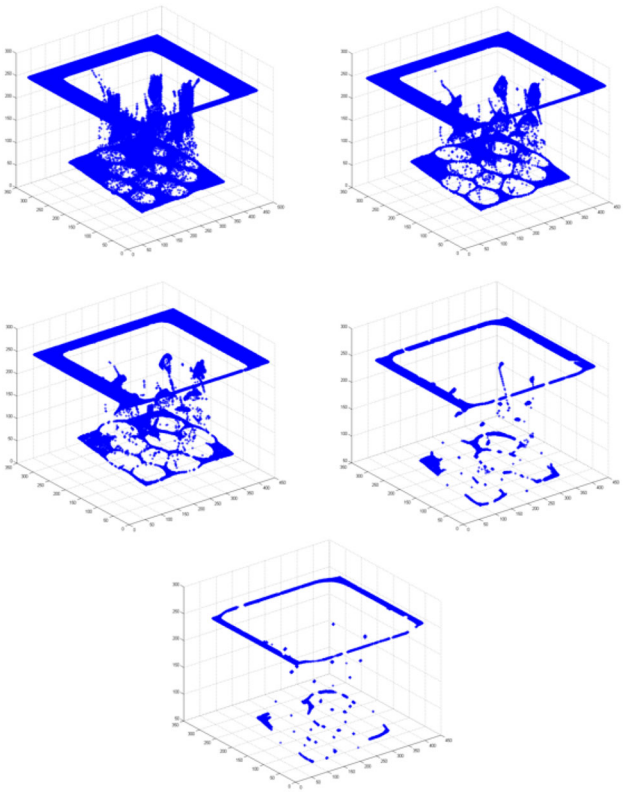


**Fig. 20** Coins; pixels clustering in 3D over multiple iterations

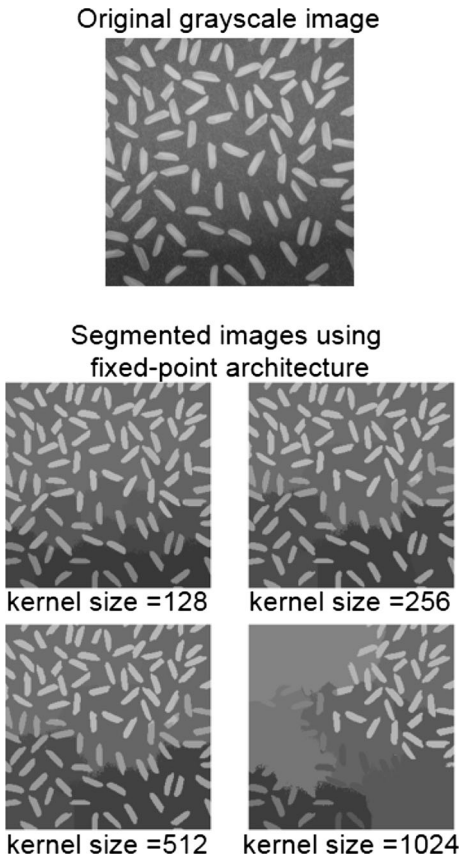



**Fig. 21** Rice (small-size objects scattered over a uniform background); FPGA segmentation results for different kernel sizes

be different in hardware and software. For each test image, if the calculated MSE would be greater than the radius of the smallest cluster, we would conclude that some pixels are converging to the wrong cluster, and that our precision has a negative impact on the performance of the mean-shift algorithm. However, for all our test images, the MSE is consistently far less than the radius of the smallest cluster as shown in Table 1, and therefore, the floating-point approximations do not affect the segmentation results.

Another advantage of our design is the possibility of tailoring the dynamic range of each dimension, which is comparable to assigning different clustering weights in the $x$, $y$, or $z$ dimensions. By discriminating between the three dimensions, we can force pixels to shift more in one of the three dimensions and better control the shape of each segment. If objects resemble a spherical shape, moving pixels by the same distance in each direction (3D) is suggested. However, if objects deviate from a spherical shape, the user should control the discriminative clustering, and allow more movement along the $x$, $y$ or $z$ directions. For example, with the cameraman grayscale image, we show that by assigning a larger dynamic range in the x dimension we can better segment the cameraman from the background (Fig. 25).

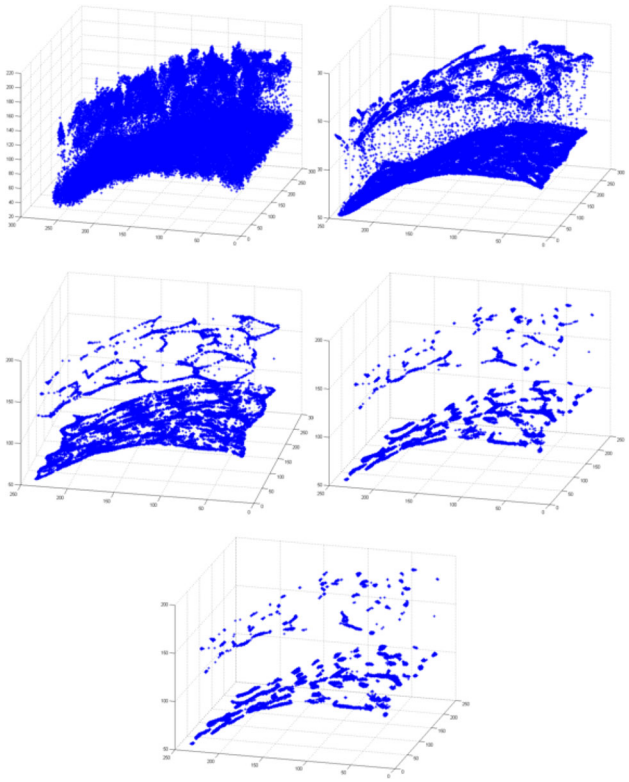


**Fig. 22** Rice; pixels clustering in 3D over multiple iterations

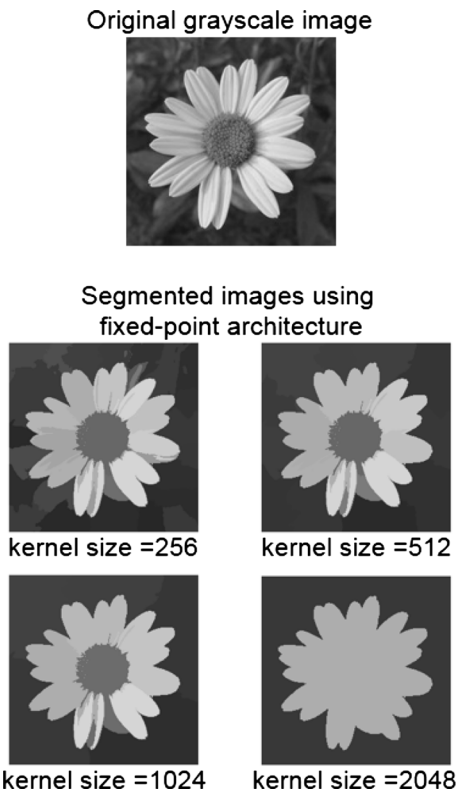



**Fig. 23** Flower (large-size object in front of a non-uniform background); FPGA segmentation results for different kernel sizes

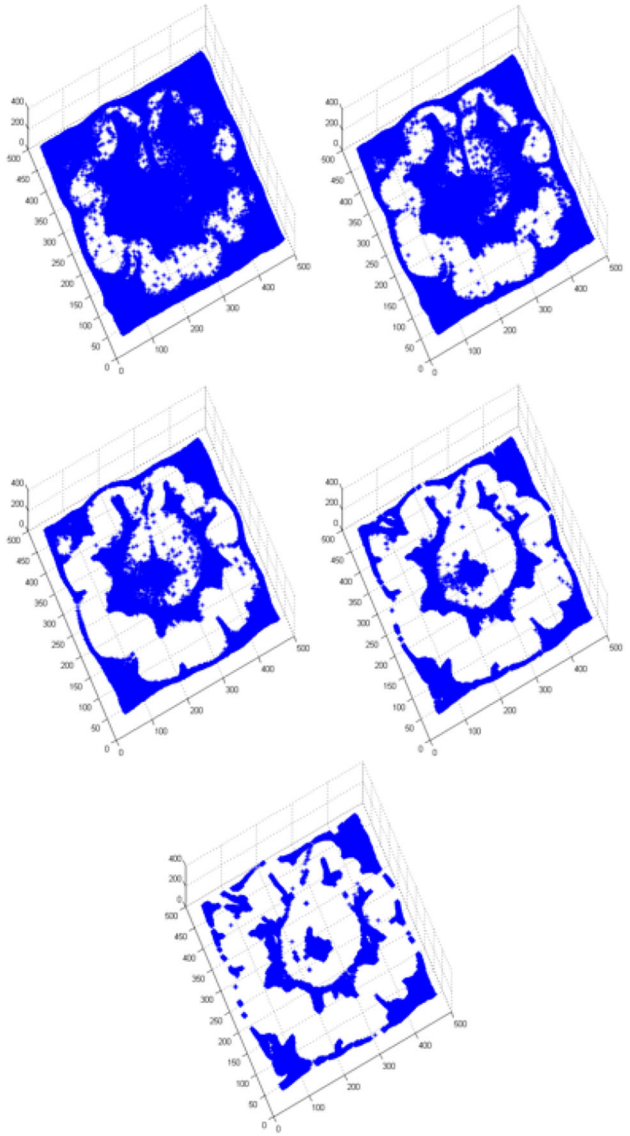


**Fig. 24** Flower; pixels clustering in 3D over multiple iterations

## 5 Conclusions

The mean-shift algorithm provides a non-parametric and unsupervised clustering solution to image segmentation that is used in a multitude of applications such as object recognition, tracking, and quality control to name just a few. Furthermore, the mean shift is a gradient-ascent algorithm that exhibits an adaptive step size, helping users avoid the step-size selection problem. 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 for images beyond VGA resolution (640 $\times$ 480).

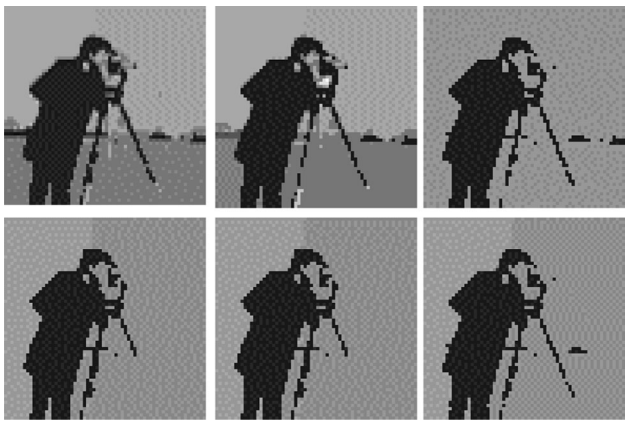


**Fig. 25** Discriminative clustering along the $x$ direction

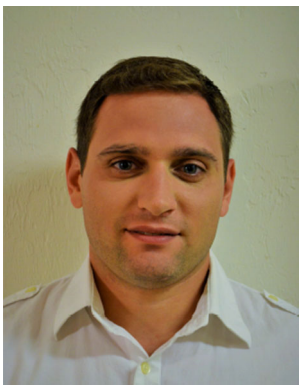
This paper proposes a scalable architecture that accelerates the Gaussian mean-shift algorithm by allocating dedicated hardware pipelines to cluster multiple pixels in parallel. Due to the fine granularity of the mean-shift algorithm, we can evaluate the PDF gradient at different locations in parallel and shift pixels toward their closest mean simultaneously. Our scalable architecture consists of fixed-point pipelines that are replicated to effectively use all hardware resources of the FPGA fabric. The architecture is tested on both PROCStar III and PROCStar IV boards using Stratix-III E260 and Stratix-IV E530 FPGAs, respectively. The small overhead penalty associated with scaling the architecture to incorporate multiple FPGAs demonstrates that our design can be further scaled to span multiple boards and can potentially segment high-definition images (1920 by 1080) on two GiDEL PROCStar IV boards. Another important achievement of our design is low-power consumption. Our architecture running on one Statix-IV FPGA, as part of the GiDEL ProcStar IV board, consumes less than 35 Watts of average power. The achieved speedup together with the low-power consumption demonstrates that our architecture is an excellent platform for embedded applications. When compared to the most current GPU implementations, our design achieved comparable speedup while lowering power consumption. The segmentation results were compared to a floating-point C code baseline to ensure no errors resulted from our fixed-point approximations.

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

## References

1. Fukunaga, K.L.D.H.: The estimation of the gradient of a density function, with application in pattern recognition. In: IEEE Trans. Information Theory (IT), vol. 21, 32–40 (1987)

2. Parzen, E.: On estimation of a probability density function and mode. Ann. Math Stat. **21**(1), 1065–1076 (1962)
3. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Trans. Pattern Anal. Mach. Intell. **17**(8), 790–799 (1995)
4. Comaniciu, D.P.M.: Mean shift analysis and applications. Proc Seventh International Conference on Computer Vision. **1**(1), 1197–1203 (1999)
5. Comaniciu, D.P.M.: Mean shift: A robust approach toward feature space analysis, ieee trans. pattern analysis and machine intelligence. IEEE Trans. Pattern Anal. Mach .Intell. **24**(5):603–619 (2002)
6. Comaniciu, D.P.M., Ramesh, V.: Real-time tracking of non-rigid objects using mean shift. Proc 2000 IEEE Conference on Computer Vision and Pattern Recognition. **2**(1):142–149 (2000)
7. Bai, P., Fu, M.C.Y.H.C.: Improved mean shift segmentation scheme for medical ultrasound images. Fourth Intern. Conf. Bioinf. Biomed. Eng. (iCBBE). **1**(1), 1–4 (2010)
8. Bottger, J., Schafer, G.L.A.V.D.S.M.A.: Three-dimensional mean-shift edge bundling for the visualization of functional connectivity in the brain. IEEE Trans Vis. Comp. Gr. **20**(3), 471–480 (2014)
9. Yamashita, A., Ito, T.K.H.A.Y.: Human tracking with multiple cameras based on face detection and mean shift. IEEE Intern. Conf. Robot Biom. **1**(1), 1664–1671 (2011)
10. Shotton, J., Blake, R.C.A.: Multiscale categorical object recognition using contour fragments. IEEE Trans. Pattern Anal. Mach. Intell. **30**(7), 1270–1281 (2008)
11. Deilamani, M.J.R.N.A.: Moving object tracking based on mean shift algorithm and features fusion. Artificial Intelligence and Signal Processing (AISP), 2011 International Symposium on. (1):48–53 (2011)
12. Du-Ming, Tsai, J.Y.L., Yuan-Ze.: Mean shift-based defect detection in multicrystalline solar wafer surfaces. IEEE Trans. Indus. Inf. **7**(1), 125–135 (2011)
13. Ranchin, T.M.M., Wald, L.: The arsis method: a general solution for improving spatial resolution of images by the mmean of sensor fusion. Fusion of Earth Data: Merging Point Measurements, Raster Maps and Remotely Sensed Images (EARSeL) 1(1):53–58 (1996)
14. Faro, A.S.P., Giordano, D.: Integrating unsipervised and supervised clustering methods on a gpu platform for fast image segmentation. 3rd International Conference on Image Processing Theory, Tools and Applications (IPTA) 85–90 (2012)
15. Alexey, A., FWBD Tomas Kulvicius: Facing the Multicore-Challenge, Real-Time Image Segmentation on a GPU. Lecture Notes in Computer Science, vol. 6310, 1st edn. Springer, Berlin Heidelberg (2010)
16. Fulkerson, B., SS: Really quick shift: Image segmentation on a gpu. In: Kutulakos, K.N. (ed.) Trends and Topics in Computer Vision. Lecture Notes in Computer Science, vol 6554, 350–358. Springer, Berlin Heidelberg (2012)
17. Jun Zhang, X.L., Luo, S.: Weighted mean shift object tracking implemented on gpu for embedded sustems. Intern. Conf. Control Eng. Commun. Technol. **1**(1), 982–985 (2012)
18. F Galluzzo, H.H.N.S., Barbosa D.: Segmentation framework for 3d echocardiography. IEEE International Ultrasonics Symposium (IUS) 2639–2642 (2012)
19. Feng W.Y.Z, Xiang, H.: An improved graph-based image segmentation algorithm and its gpu acceleration. 2011 Workshop on Digital Media and Digital Content Management (DMDCM) 237–241 (2011)
20. Rao, S., de Martins, A.M., Principe, J.C.: Mean shift: An information theoretic perspective. Trans. Pattern Anal. Mach. Intell. **30**(3), 222–230 (2009)
21. Renyi, A.: On measure of entropy and information. Proc Fourth Berkeley Symp Math Stat and Prob. **1**(1), 547–561 (1961)
22. Carreira-Perpinan, M.A.: Acceleration strategies for gaussian mean-shift image segmentation. IEEE Comp. Soc. Conf. Comp. Vision Pattern Recognit. **1**(1), 1160–1167 (2006)
23. Saegusa, T.T.M.: An fpga implementation of k-means clustering for color images based on kd-tree. Intern. Conf. Field Program Logic Appl (FPL) **1**(1), 1–6 (2006)
24. Wang, H., HLJW JZhao.: Parallel clustering algorithms for image processing on multi-core cpus. Intern. Conf. Comp. Sci. Softw Eng. **2**(1), 450–453 (2008)
25. Ali, U., Malik, K.M.M.B.: Fpga/soft-processor based real-time object tracking system. Fifth South. Conf. Program. Logic (SPL) **1**(1), 33–37 (2009)
26. Pandey, M.S.J.U.K.S.R., Borgohain, D.: Real-time histogram computation in kernel-based tracking system. International Conference on Advanced Electronic Systems (ICAES) 171–174 (2013)
27. Trieu, D.B.K.T.M.: An implementation of the mean-shift filter on fpga. Intern. Conf. Field Program. Logic Appl. (FLP) 219–224 (2011)
28. Lu, X.S.Y., Ren, D.: Fpga-based real-time object tracking for mobile robot. Intern. Conf. Audio Lang. Imag. Process. (ICALIP) 1657–1662 (2010)
29. Stolkin, R., Florescu, M.B.C.H.I.: Efficient visual servoing with the abcshift tracking algorithm. IEEE Intern. Conf. Robot. Automation (ICRA). **1**(1), 3219–3224 (2008)
30. Carreira-Perpinan, M.A.: Gaussian mean-shift is an em algorithm. IEEE Trans. Pattern Anal. Mach. Intell. **29**(5), 767–776 (2007)
31. Kirchgessner, R.A.G., Lam, H.: Reconfigurable computing middleware for application portability and productivity. Intern. Conf. Appl-Spec. Syst., Arch Process. (ASAP). 1(1):211–218 (2013)
32. Kalgin, K.: Implementation of fine-grained algorithms on graphical processing units. 10th International Conference on Prallel Computing Technologies 207–215 (2009)
33. Sirotkivic, J., Dujmic, V.P.H.: Accelerating mean-shift image segmsegmentation ifgt on massively parallel gpu. 36th International Conference on Information & Communications Technology Electronics and Microelectronics (MIPRO) 279–285 (2013)
34. Altera, C.: Altera announces breakthrough advantages with generation 10; http://newsroom.altera.com/press-releases/nr-altera-generation-10.htm. Tech. rep., Altera (2013)
35. Coombes, D.: Tegra k1 whitepaper. Tech. rep, NVIDIA (2014)

**Stefan Craciun** is a Ph.D. candidate and research assistant at NSF Center for High-Performance Reconfigurable Computing (CHREC) at the University of Florida, ECE department. He completed his M.S. degree in Electrical and Computer Engineering at the University of Florida in 2009. Currently, he is pursuing his Ph.D. degree under the supervision of Prof. Alan D. George, the founder of the NSF CHREC center at the University of Florida, and the co-supervision of Dr. Jose C. Principe, the founder of the CNEL center at the University of Florida. His research topic specializes in FPGA-based architectures that enable efficient mapping of image-processing algorithms for real-time execution. His applications focus on low-power embedded systems for onboard information extraction such as

edge detection and feature extraction as well as high-level decision making such as object recognition and tracking applications.

**Robert Kirchgessner** is a Ph.D. candidate and research assistant at NSF Center for High-Performance Reconfigurable Computing (CHREC) at the University of Florida, ECE department. He received his B.S. degrees in Electrical and Computer Engineering, and the M.S. degrees in Electrical Engineering from the University of Florida. He is currently pursuing his Ph.D. degree under the supervision of Prof. Alan D. George. His research interests include high-performance reconfigurable architectures, FPGA-based application portability and productivity tools, and FPGA-based graph-processing architectures and applications.

**Alan D. George** is Professor of ECE at the University of Florida, where he founded and directs the NSF Center for High-Performance Reconfigurable Computing (CHREC). He received the B.S. degree in CS and the M.S. in ECE from the University of Central Florida, and the Ph.D. in CS from the Florida State University. His research interests focus upon high-performance architectures, networks, systems, services, and applications for reconfigurable, parallel, distributed, and fault-tolerant computing. Dr. George is a Fellow of the IEEE.

**Herman Lam** is an Associate Professor of Electrical and Computer Engineering at the University of Florida and the Associate Director of CHREC, the NSF Center for High-Performance Reconfigurable Computing. He has over 25 years of research and development experience in the areas of distributed computing, service-oriented computing, database management, and most recently high-performance and reconfigurable computing. He is the co-developer of the Novo-G reconfigurable supercomputer, the most powerful reconfigurable computer in the academic world. Novo-G, containing over 400 top-of-the-line FPGAs, serves as a testbed for the study of methods and tools for the acceleration and deployment of scientifically impactful big-data applications on a scalable heterogeneous system.

**Jose C. Principe** is Distinguished Professor of Electrical and Biomedical Engineering at the University of Florida since 2002. He is BellSouth Professor and Founding Director of the University of Florida Computational Neuro-Engineering Laboratory (CNEL). He joined the University of Florida in 1987, after an eight-year appointment as Professor at the University of Aveiro, in Portugal. Dr. Principe holds degrees in electrical engineering from the University of Porto, Portugal, University of Florida, USA (Master and Ph.D.), and Honoris Causa degrees from the Universita Mediterranea in Reggio Calabria, Italy, Universidade do Maranhao, Brazil and Aalto University, Finland.