Real-Time Image Segmentation using a Spiking Neuromorphic Processor

Chetan Singh Thakur, Jamal Molin, Ralph Etienne-Cummings
Department of Electrical and Computer Engineering
Johns Hopkins University
Baltimore, MD, USA
cthakur2@jhu.edu

***Abstract*—Segmentation of specific objects in an image is a key task in computer vision, for which various algorithms have been proposed. However, most of these algorithms are software-oriented and have high computational complexity that makes them difficult to implement in hardware for real-time applications. The semi-supervised graph-based random walker (RW) algorithm, which seeks the solution for a large sparse matrix, shows optimal image segmentation performance in software. In this paper, we configured our existing spiking neuromorphic processor to implement an adaptation of the RW algorithm that performs segmentation using an iterative method while preserving accuracy. Our hardware platform can be adapted to achieve an extremely fast segmentation speed of up to 1000 images per second by computing all neurons in parallel. We believe our approach will facilitate the implementation of graph-based computer vision algorithms on neuromorphic hardware for low power, real-time applications. Our approach will be immensely valuable in applications that require high-performance computing to run in real time, such as biomedical image segmentation for image-guided surgery.**

Keywords—FPGA; real-time image segmentation; neuromorphic engineering; VLSI; Laplace equation; graph theory

I. INTRODUCTION

Image segmentation is the process of partitioning an image into clusters of similar pixels corresponding to salient image regions, such as individual surfaces or objects, for easier processing and analysis. Several image segmentation techniques are popular, including those based on threshold, edge, region, cluster, watershed, partial-differential equations (PDEs), and artificial neural networks [1]. These methods can be categorised into manual, semi-automatic, and fully automatic. Semi-automatic or seeded image segmentation requires minimal user guidance or attention input from other processes to define the desired content to be extracted [2]–[4]. The random walker (RW) is one such multi-label interactive algorithm that models the image as a graph where pixels correspond to nodes and are connected to neighbouring pixels via weighted edges. The RW achieves segmentation by solving a discrete Dirichlet problem, but this is computationally expensive for real-time applications [5].

Image segmentation algorithms are primarily software-oriented with minimal concern for hardware implementation, thus restricting their use in several real-time applications such as image-guided surgery and defence surveillance [6], [7]. Several researchers have implemented image segmentation on generic hardware such as the field-programmable gate array (FPGA) [8]–[10]. However, these systems are based on the conventional von Neumann architecture and use fixed-point arithmetic, which requires a large amount of computational resources. The large silicon area requirements and power constraints of these implementations thus restrict their applications to handheld devices. The analogue very large-scale integrated (VLSI) implementation of image segmentation algorithms have also been reported [11], [12]. However, these systems are not easily scalable or configurable.

Neuromorphic engineering offers an attractive alternative for designing systems suited for low power, real-time applications. This is an interdisciplinary approach inspired by the function, structural organisation, and physical foundations of biological nervous systems [13]–[18]. Neuromorphic systems are more robust and orders of magnitude more energy efficient than conventional approaches using user-programmed intelligence.

Here, we utilise an end-to-end stochastic-, event-based neuromorphic system for performing image segmentation. Our image segmentation framework is based on the RW algorithm [5]. In our implementation, images are segmented by solving a discrete PDE using an iterative method as described in [19]. Typically, a scene is partitioned into different objects based on common properties such as depth, motion, or image intensity. Our work uses image intensity as the basis for segmentation, but the framework can be easily modified to incorporate other properties, such as texture or color, for segmentation.

In a neuromorphic approach, neurons communicate via spikes (events) and information is encoded in terms of either spike rate or timing. In this work, we encode the information in terms of the spike rate. The spike-based computation that we use requires simple computational elements and can be scaled to larger systems. Moreover, our graph-based framework can be easily configured to perform other graph-based computer vision tasks. Our system could, thus, be a prototype for building smaller and power-efficient hardware systems to perform image segmentation in real time.

II. ALGORITHM FRAMEWORK

We use a graph-based semi-automatic image segmentation method, where a user or pre-processing algorithm assigns the labels (either foreground or background) for a few image pixels

This work has been supported by the Office of Naval Research under MURI grant N000141010278.

978-1-5090-4780-2/17/$31.00 ©2017 IEEE

(seed points). The segmentation algorithm is formulated on a weighted graph, where each node represents a pixel. These nodes are connected by graph edges and each edge has its own weight. The weight may be considered as a penalty (or cost function) for traveling from one node to another along the edge. Thus, the weights should be such that the desirable region in the image corresponds to larger weights among the nodes in the graph and vice versa. An image is represented as a graph $G = (V, E)$, with vertices or nodes (pixels) $v \in V$ and edges $e \in E \subseteq V \times V$. An edge, e, spanning two nodes v_i and v_j is denoted by e_{ij} and a weight of the edge is denoted as w_{ij}. The graph here is assumed undirected ($w_{ij} = w_{ji}$) with positive weights $(w_{ij} \geq 0)$. The weights in the graph are a function of some combination of image features, where an image 'feature' is any useful information that can be acquired from the image. Usually, the features are computed in a neighbourhood around each graph edge, transformed to give low edge costs to more desirable feature values. Here, we use the Gaussian weighting function, expressed as:

$$w_{ij} = \exp\left(-\beta\left(g_i - g_j\right)^2\right) \quad (1)$$

where, g_i denotes the image intensity at pixel i and β is the free parameter. For a colour image, g_i will represent a vector that handles different colour channels. Equation (1) can be modified to apply other image features such as texture information.

In our semi-automatic segmentation algorithm, there are a set of user-defined nodes (pixels) denoted as V_M, which are few in number, and a set of unmarked pixels V_U, such that $V_M \cup V_U = V$ and $V_M \cap V_U = \phi$. Each predefined node $v_i \subset V_M$ is labelled from the set $C = \{0, 1, 2, \ldots k\}$. In case of binary-labelled segmentation, $C = \{0, 1\}$, where 0 and 1 denote the background and foreground, respectively. Segmentation is complete when all the unlabelled nodes V_U are assigned labels. This segmentation problem can be formulated as a diffusion process defined using the equation below:

$$\frac{\partial I}{\partial t} = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad (2)$$

where, I is an image in 2D space and $I_{x,y}$ denotes pixel intensity at position (x, y). The discrete formulation of (2) for isotropic diffusion (all edges have same weights) can be written as:

$$I_{x,y}^{t+1} = I_{x,y}^{t} + \frac{1}{4}\left(I_{x-1,j} + I_{x+1,j} + I_{x,y-1} + I_{x,y+1} - 4I_{x,y}\right) \quad (3)$$

For a weighted graph (Fig. 1A), (3) will be modified as:

$$I_{x,y}^{t+1} = I_{x,y}^{t} + \frac{1}{d_{x,y}}\left(w_{x-1,y} * I_{x-1,y} + w_{x+1,y} * I_{x+1,y} + w_{x,y-1} * I_{x,y-1} + w_{x,y+1} * I_{x,y+1} - d_{x,y} I_{x,y}\right) \quad (4)$$

As the graph is undirected, $d_{x,y} = \sum_E w_{x,y}$, the summation of the weights of all edges connecting the pixel $I_{x,y}$ to the neighbouring pixels ($\in E$). This algorithm can be summarised as below:

- Generate weights for all edges in the graph using (1).
- Obtain a set of labelled pixels V_M defined interactively or automatically. For binary segmentation, foreground-labelled pixels are assigned the value '1' and background pixels are assigned '0'. This can be generalised for multi-label segmentation by considering each label at a time.
- Iteratively solve (4) in parallel for all pixels, while keeping the value of the seeded pixels constant, and run till the pixel intensity converges.
- After a few initial iterations, if pixels cross a threshold value, they are assigned the maximum value. This trick accelerates the diffusion across the neighbouring pixels.
- Obtain final segmentation by thresholding pixel values. Pixels with intensities higher than the threshold are labelled as foreground, else treated as background. For multi-labelled pixels, individual pixels are assigned a label that maximises their value.

III. Circuit Analogy and Proposed Implementation

The relation between a diffusion process and electrical circuits has been known for a long time [20], [21]. The diffusion equation in a weighted graph can be described as the spread of electrical charge in a resistor–capacitor (RC) circuit. The iterative solution of (4) can be interpreted as a simulation of the circuit shown in Fig. 1B. However, it is difficult to implement this circuit using resistors that have different resistance values for different edges, as the diffusion distance would be difficult to control electrically and the resistors would occupy large silicon area in the VLSI process. A signal injected into a linear resistor network decays exponentially with distance from the source. The diffusion distance is higher if the resistance between the nodes is small and vice versa. Thus, we propose a circuit implementation where transistors replace resistors and the gate voltage of the transistors controls the conductance between two nodes (Fig. 1C). In this weighted-graph circuit, labelled (seeded) node voltages propagate to neighbouring nodes based on their connectivity weights. Each node is built using the circuit shown in Fig. 1C. For a seeded node, $V_{seed} = 1$ (i.e. $\overline{V_{seed}} = 0$), and this switches on transistor T_{in} and allows current I_{in} to be injected into the neighbouring nodes via transistors T_N, T_S, T_W, T_E and into transistor T_m as sink current. Our circuit operates in the weak-inversion region (though it can also operate in the above-threshold region) of the transistor, with an exponential relationship between the gate voltage (V_{gs}) and current (I_{ds}) between the drain and source terminals that is expressed as:

$$I_{ds} = I_{d0} \exp\left(\frac{V_{gs} - V_{ds}}{nU_T}\right) \quad (5)$$

where, I_{d0} is the residual saturation drain current or channel leakage current, n is the slope factor (generally between 1 and 1.5), and U_T is the thermal voltage. The weights in the graph can be set by V_{gs}, which effectively controls the transconductance between neighbouring nodes. In the weighted graph, the weights are calculated based on neighbouring pixel intensities, as defined in (2). For circuit

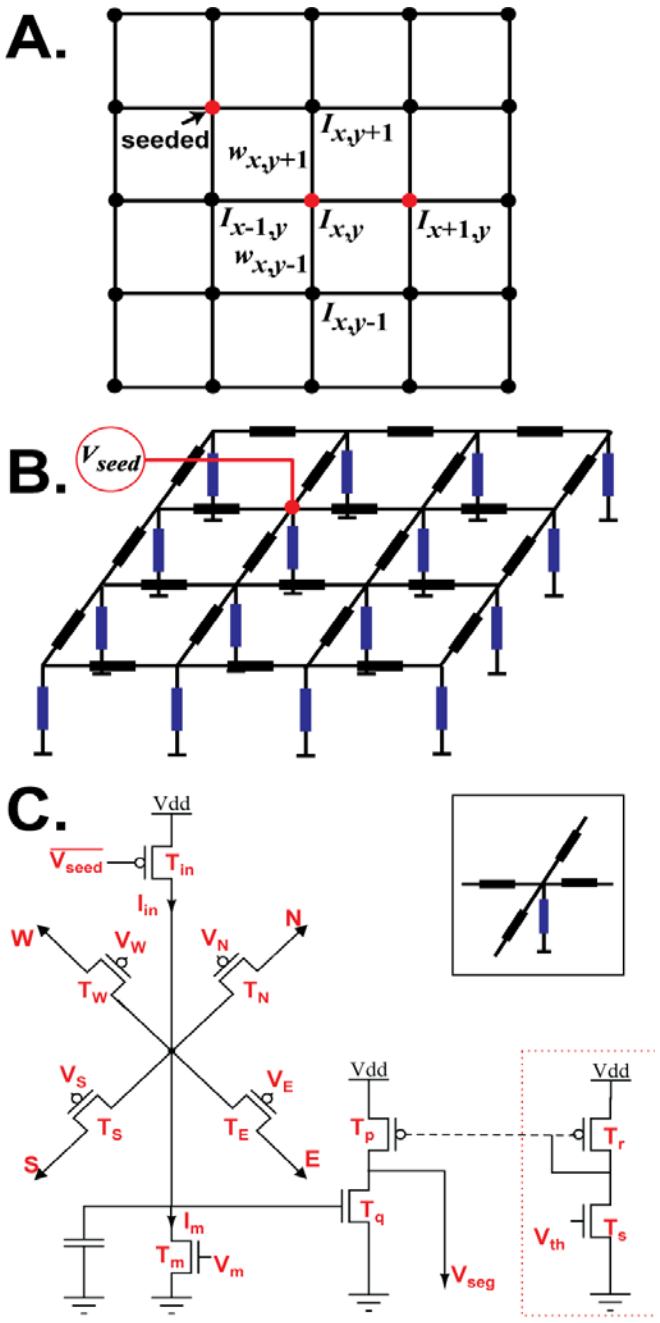

Fig. 1. Circuit analogy of an image represented as a graph. (**A**) The image is represented as a graph, with red points representing the seeded nodes. (**B**) The weighted graph modelled as an equivalent RC (resistor–capacitor) circuit. (**C**) Proposed transistor-level circuit implementation where the weights of a network can be controlled electronically.

implementation, we use (5) to define the weights, where V_{gs} would be high if the neighbouring pixels are of similar intensities, else it would be small. Transistor T_m is the current-limiting device. The remaining current goes to the neighbouring nodes based on the V_{gs} of the connecting transistors. In the unseeded node, current higher than I_m will charge the capacitor to ensure that the node belongs to the image foreground, as labelled by the seeded node. In contrast, if the current in the unseeded node is less than I_m, then V_{seg} will be pulled high and will be a part of the image background. This anisotropic arrangement of the circuit favours voltage propagation in the more conducting direction, set by voltages V_N, V_S, V_W, V_E, and allows segmentation as per the seeded nodes. Transistors T_c and T_d are shared by all the nodes and set the threshold voltage of the detector at V_{th}.

IV. Implementation on FPGA–IFAT

The IFAT, in its original design, is an array of neurons implemented in the VLSI CMOS technology [22]. Here, the capacitors are analogous to neurons and the voltage across the capacitors represents the membrane potential of the neurons. When a neuron receives an event (or spike), the potential across the capacitor increases owing to charge accumulation. When this potential exceeds a globally set threshold, the neuron outputs an event. A dynamically reconfigurable look-up table (LUT) determines the connectivity (synapses) between neurons and the strength of the connections. Here, we implemented the IFAT system using an FPGA framework. Our implementation includes a digital Poisson neuron, where an accumulator represents the neuronal membrane potential, and output spikes are generated based on the membrane potential. Fig. 2 shows the block diagram of the complete FPGA implementation. The stochastic output events generated based on the pixel intensity of the input image serve as input for the FPGA–IFAT. The neurons in the IFAT are configured as a grid (Fig. 1A), where the membrane potential of a neuron represents the node voltage in a graph. The connectivity of the neurons and weights of the graph edges are stored in the LUT. This LUT is implemented as a block RAM on the FPGA. Each value in the LUT block RAM contains 4 destination addresses of the neighbouring pixels and their corresponding weights. The target neuron receives the spike trains from its neighbouring pixels with a mean spike rate proportional to the pixel value multiplied (using AND gate) by the corresponding synaptic weight. These weights are in the range [0,1], so we employ stochastic computation to convert them into a bit stream of a Poisson process. This allows the multiplication of neuron spike trains and weight bit streams using a simple AND gate, thus increasing the area efficiency of the design. After a few iterations, the membrane potential of each neuron converges and does not change with time thereafter. These converged node potentials represent the intensity of the final pixels in the segmented image.

V. Results

Fig. 3 shows the segmentation results for medical and natural images. We considered only grayscale images in our implementation, but the method can be generalised by modifying (1) to incorporate other features such as colour and texture. The images and seeds were chosen to demonstrate the general applicability of the interactive segmentation approach on objects of varying uniformity, size, shape, and contrast. The free parameter β in (1) was chosen as 20. We found that 200 iterations were sufficient to achieve convergence, as the

diffusion rate was much faster initially owing to pixels above a threshold being assigned the maximum value.

images (Fig. 3A3, 3B3). The results are very promising as we achieved the desired segmented objects in real time. Our current existing FPGA–IFAT accesses each neuron

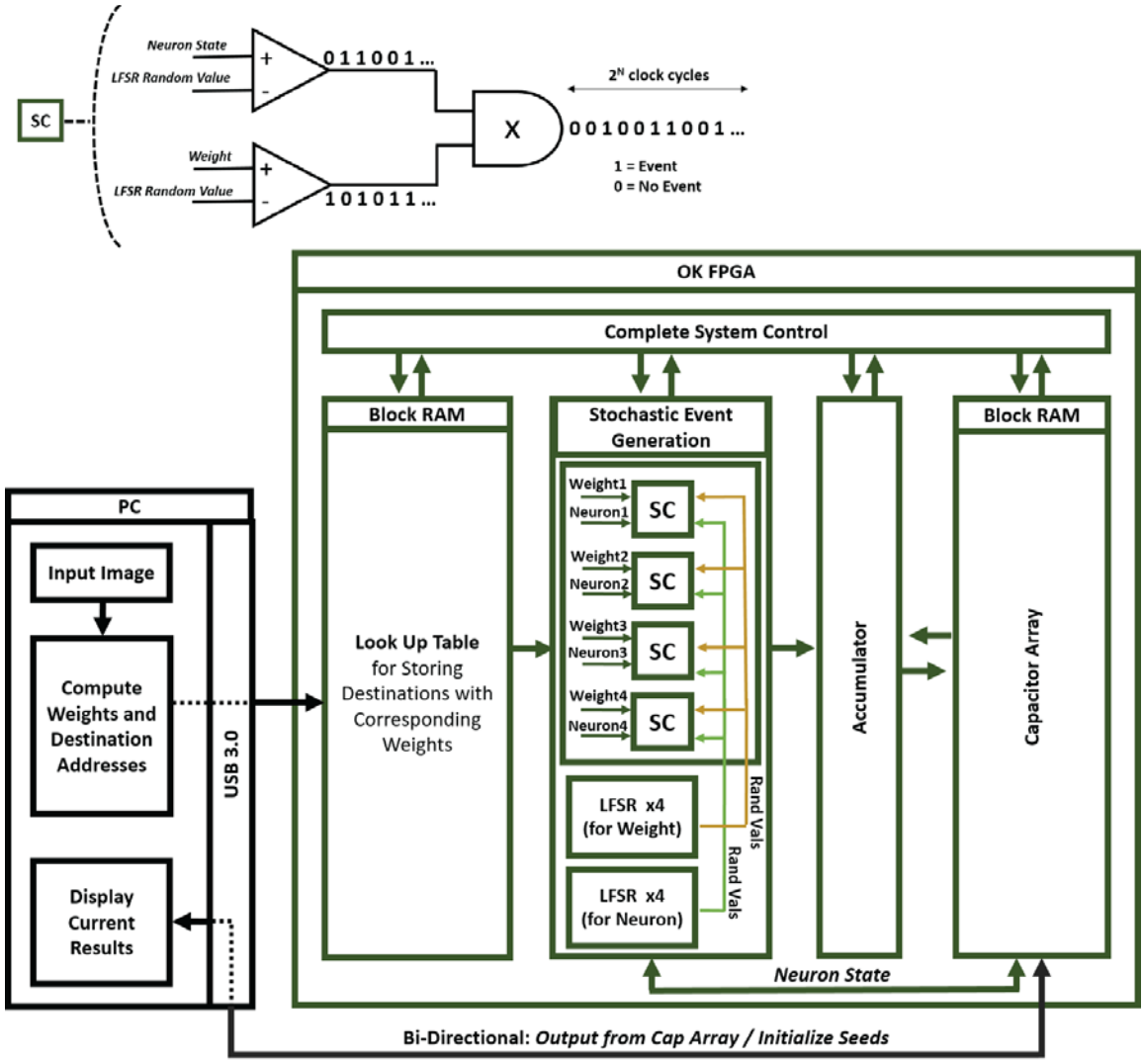

Fig. 2. Block diagram of the complete FPGA–IFAT system operating on stochastic event streams.

Fig. 3A shows an example of multi-label segmentation, where we segmented 2 different objects (girl and sea lion) from the image. Fig. 3A1 and 3B1 show a natural and medical image, respectively, with seeds shown in red. After 200 iterations, we achieved the segmented images (Fig. 3A2, 3B2), which were further thresholded to obtain the binary segmented

sequentially, but our solution can run in parallel and can access all the neurons in parallel for each iteration. This configuration will allow us to achieve segmentation of 1000 images per second using a 100 MHz system clock in the FPGA. Table 1 shows the resources used by our current FPGA–IFAT implementation.

TABLE I. FPGA RESOURCES FOR SEGMENTATION OF AN IMAGE OF SIZE 150 × 150 PIXELS

Resource	Used	Total	Percentage Used
Slice Registers	1,832	202,800	1%
Slice LUTs	2,017	101,400	1%
Occupied Slices	808	25,350	3%
BRAM (36E1)	87	325	26%
BRAM (18E1)	2	650	1%
DSP (48E1)[a]	3	600	1%

[a] DSP blocks only necessary for computing address indices (other methods for indexing could be used for computing addresses without using DSP blocks). No DSP blocks are required for computation using stochastic elements. Multiplications are performed in an event-based manner, over time, as opposed to using traditional DSP blocks.

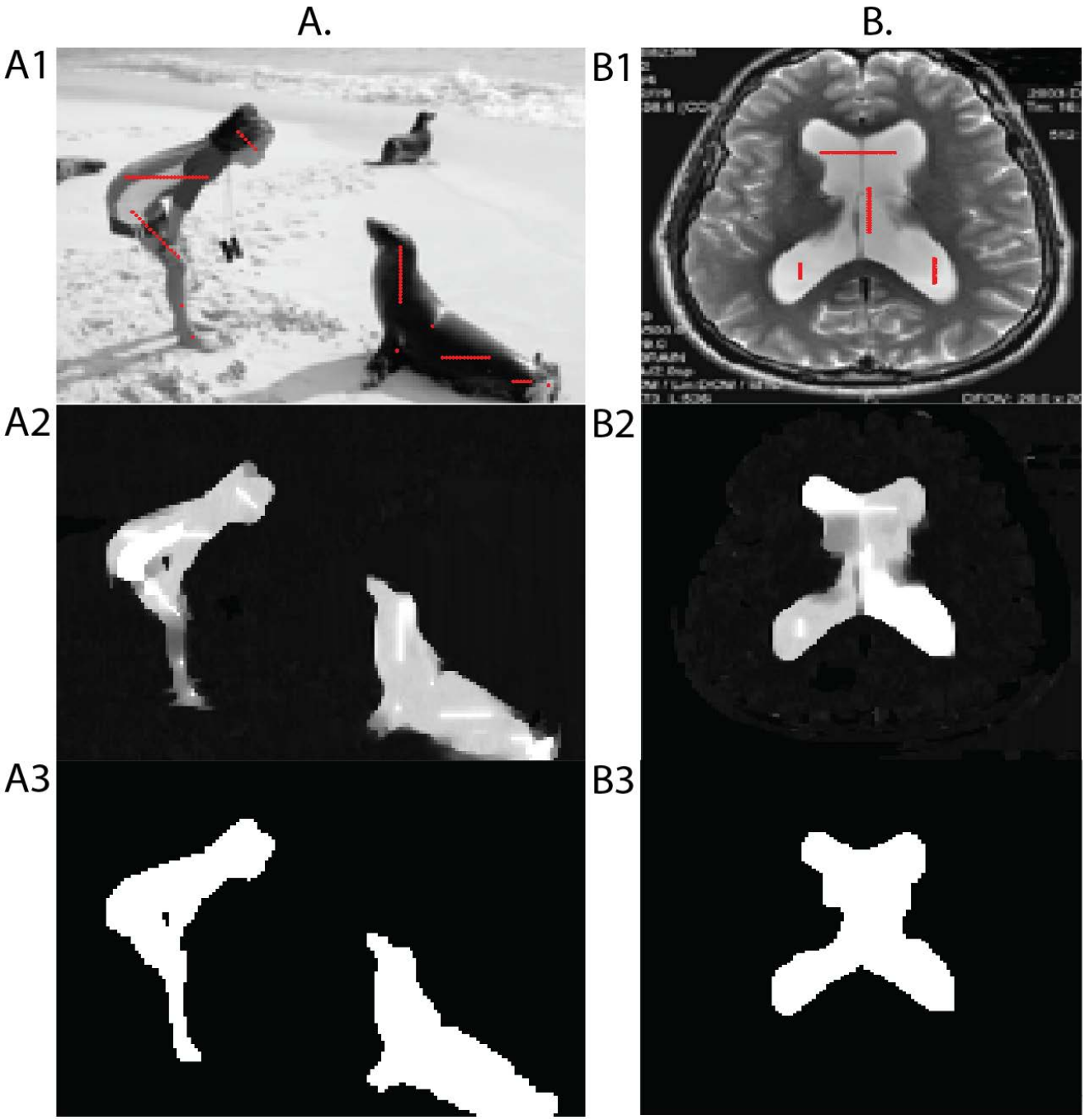

Fig. 3. Segmentation results using our adapted graph-based algorithm. (**A1**) shows a natural image with seeds shown in red, (**A2**) shows the segmentation of the image A1, (**A3**) is achieved after thresholding segmentation output A2. (**B1**) shows a brain magnetic resonance imaging (MRI) scan with seeds shown in red, (**B2**) shows the segmentation of the image B1, (**B3**) is achieved after thresholding segmentation output B2.

VI. Conclusions

Here, we have presented the implementation of a graph-based image segmentation algorithm as an FPGA emulation of our neuromorphic processor IFAT. Our problem formulation was similar to the famous RW algorithm [5], but segmentation was achieved by solving a diffusion equation using an iterative method. We used an existing FPGA–IFAT emulator that accesses each neuron sequentially. However, the current implementation serves as proof-of-concept that the complete system can be parallelised for real-time applications. The main contribution of our work is that it establishes a framework for the hardware implementation of graph-based image segmentation methods in real time. Further, our solution allows parallel distributed processing, i.e. computation can be performed using all neurons in parallel. Our work is significant as it shows that segmentation can be achieved within a network of local-constraint-solving neurons that do not have explicit

access to the global segmentation goal. Additionally, our approach is easily scalable and can be extended for multi-label image segmentation, such as that based on texture and color, simply by changing the weights across the different nodes. Furthermore, the proposed system can be extended for 3D images as well, where each node will be connected to 6 neighbouring nodes, with 2 nodes in each dimension.

REFERENCES

[1] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, no. 9, pp. 1277–1294, Sep. 1993.

[2] A. X. Falcão, J. K. Udupa, and F. K. Miyazawa, "An ultra-fast user-steered image segmentation paradigm: Live wire on the fly," in *IEEE Transactions on Medical Imaging*, 2000, vol. 19, no. 1, pp. 55–62.

[3] A. Falcao, J. K. Udupa, S. Samarasekera, and B. E. Hirsch, "User-steered Image Boundary Segmentation," *Proceedings of SPIE*, vol. 2710, pp. 278–288, 1996.

[4] E. N. Mortensen and W. A. Barrett, "Interactive Segmentation with Intelligent Scissors," *Graphical Models and Image Processing*, vol. 60, no. 5, pp. 349–384, 1998.

[5] L. Grady, "Random walks for image segmentation," *Pami*, vol. 28, no. 11, pp. 1768--1783, 2006.

[6] W. Niessen, "Model-based image segmentation for image-guided interventions," in *Image-Guided Interventions: Technology and Applications*, 2008, pp. 219–239.

[7] S. K. Warfield, F. A. Jolesz, and R. Kikinis, "Real-Time Image Segmentation for Image-Guided Surgery," *Supercomputing, 1998.SC98. IEEE/ACM Conference on*. p. 42, 1998.

[8] P. Dillinger, J. F. Vogelbruch, J. Leinen, S. Suslov, R. Patzak, H. Winkler, and K. Schwan, "FPGA-Based Real-Time Image Segmentation for Medical Systems and Data Processing," *IEEE Transactions on Nuclear Science*, vol. 53, no. 4, pp. 2097–2101, 2006.

[9] D. B. K. Trieu and T. Maruyama, "Real-time image segmentation based on a parallel and pipelined watershed algorithm," *Journal of Real-Time Image Processing*, vol. 2, no. 4, pp. 319–329, 2007.

[10] M. Genovese and E. Napoli, "FPGA-based architecture for real time segmentation and denoising of HD video," *Journal of Real-Time Image Processing*, vol. 8, no. 4, pp. 389–401, 2013.

[11] J. Kramer, R. Sarpeshkar, and C. Koch, "Analog VLSI motion discontinuity detectors for image segmentation," *Circuits and Systems, 1996. ISCAS '96., Connecting the World., 1996 IEEE International Symposium on*, vol. 2, pp. 620–623 vol.2, 1996.

[12] J. G. Harris, "An analog network for continuous-time segmentation," *International Journal of Computer Vision*, vol. 10, no. 1, pp. 43–51, 1993.

[13] C. S. Thakur, R. Wang, T. J. Hamilton, J. Tapson, and A. van Schaik, "A Trainable Neuromorphic Integrated Circuit that Exploits Device Mismatch," Jul. 2015.

[14] C. S. Thakur, T. J. Hamilton, J. Tapson, A. van Schaik, and R. F. Lyon, "FPGA implementation of the CAR Model of the cochlea," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 1853–1856.

[15] E. A. Vittoz, "Analog VLSI signal processing: Why, where, and how?," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 8, no. 1, pp. 27–44, Feb. 1994.

[16] C. S. Thakur, R. M. Wang, S. Afshar, T. J. Hamilton, J. C. Tapson, S. A. Shamma, and A. van Schaik, "Sound stream segregation: a neuromorphic approach to solve the 'cocktail party problem' in real-time," *Frontiers in Neuroscience*, vol. 9, no. September, pp. 1–10, Sep. 2015.

[17] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, "Neuromorphic Silicon Neuron Circuits," *Frontiers in Neuroscience*, vol. 5, no. May, pp. 1–23, 2011.

[18] C. S. Thakur, S. Afshar, R. M. Wang, T. J. Hamilton, J. Tapson, and A. van Schaik, "Bayesian Estimation and Inference using Stochastic Hardware," *Frontiers in Neuroscience*, vol. 10, no. March, pp. 1–28, 2015.

[19] Y. Saad, "Parallel Iterative Methods for Sparse Linear Systems," in *Studies in Computational Mathematics*, vol. 8, no. C, 2001, pp. 423–440.

[20] L. J. Grady and J. R. Polimeni, *Discrete Calculus*. London: Springer London, 2010.

[21] P. G. Doyle and J. L. Snell, "Random Walks and Electric Networks," *American Mathematical Monthly*, vol. 94, no. January, p. 202, 2000.

[22] J. L. Molin, A. Eisape, C. S. Thakur, V. Varghese, C. Brandli, and R. Etienne-Cummings, "Low-Power, Low-Mismatch, Highly-Dense Array of VLSI Mihalas-Niebur Neurons," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017.