Hindawi
Complexity
Volume 2018, Article ID 3618621, 11 pages
https://doi.org/10.1155/2018/3618621

Research Article

Protocomputing Architecture over a Digital Medium Aiming at Real-Time Video Processing

Aoi Tanibata,[1] **Alexandre Schmid**,[2] **Shinya Takamaeda-Yamazaki**,[1] **Masayuki Ikebe**,[1] **Masato Motomura**,[1] **and Tetsuya Asai**[1]

[1]*Graduate School of Information Science and Technology (IST), Hokkaido University, Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido, Japan*
[2]*Microelectronic Systems Laboratory, Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland*

Correspondence should be addressed to Aoi Tanibata; tanibata@lalsie.ist.hokudai.ac.jp

Received 26 October 2017; Accepted 9 January 2018; Published 15 February 2018

Academic Editor: Masashi Aono

A protocomputational architecture is presented that implements the ancient reaction-diffusion model as a microelectronic hardware. A digital medium is selected for the physical mapping of the protoarchitecture as a way to benefit from reliable advanced integrated circuits fabrication technologies. The extraction of dense motion vector fields from textureless objects in a video sequence is selected as a realistic application. Real-time video processing results at 30 fps are achieved using an FPGA physical implementation of the proposed protoarchitecture.

1. Introduction

Modern von Neumann architecture based processors systematically carry over extensive processing tasks governed by the consecutive execution of individual instructions that implement algorithms. The fact is widely accepted that these architectures are suitable to efficiently process a massive amount of simple algebraic operations with limited memory exchange, while their efficiency dramatically decreases at processing tasks that are natural and quick to mammals such as face and object recognition, feature, and saliency extraction. Artificial neural networks [1] and, more recently, machine learning models and algorithms have emerged in response to solving such conceptually complex problems, for example, [2].

Nevertheless, computation has existed in nature prior to the existence of any physical support to engineered and human computing, for example, mathematical structured thinking, geometrical support to calculations, and mechanical and electronic computing systems. The operation and behavior of neurons, the topology of neural networks, and more generally operations of the brain have been a source of inspiration yielding disciplines including computational neuroscience, neuromorphic computing, and engineering [3]. Protocomputing is emerging as a novel research discipline accepting two fundamental research hypotheses; namely, (a) computing is not limited to excitable tissues and mammals and is also processed by simple organisms as well as liquids and materials in the form of biochemical and physical processes and reactions often forming nonlinear dynamical systems with complex behavior and (b) the very fundamental nature of this processing may hold its properties from the origin of life and of the universe.

Still in its infancy, the protocomputing discipline is mostly addressed at the level of theoretical developments, modeling, and algorithmic studies [4]. Decision-solving methodologies and algorithms have been presented to efficiently solve complex optimization problems exploiting specific feature of physical phenomena [5]. Wet-laboratory experiments lend themselves particularly well to an experimental perspective, specifically considering the chemical and biology related approaches. The sophisticated spatiotemporal oscillatory dynamics developed by the primitive single-cell amoeba in solving complex environmental adaptation is used as an inspiration to solving NP-hard problems [6]. The usage of slime mould to constructing various sensing

and computational building blocks is discussed in detail in [7]. One research direction in synthetic biology consists of ancestral sequence reconstruction to gain understanding in mechanisms of life and its origins [8]. Photonic medium is applied to experimentally solving the multiarmed bandit problem exploiting the ultrafast chaotic oscillatory dynamics of lasers [9]. Astronomy and bioastronomy also seek to understand computing of the origin of life.

From an electronic viewpoint, protocomputation may be considered from a classical device-level perspective, that is, as a discipline aiming at performing unconventional computation from benefiting of specific electrical characteristics of novel, post-CMOS devices and thus positioning itself in overlap to advanced switching devices research and material science, for example, single-electron transistors, graphene and quantum devices, and molecular electronics. In general, exploiting such devices requires their circuit-level hybrid usage with CMOS technology. Hence, classical analog MOSFET design is used to amplify excessively low current levels and create functionality that can successfully be interfaced to other circuits and to the outside world.

The majority of the demonstrated protocomputing device-level experiments may be considered as proof-of-concept aiming at understanding the operation and characteristics of the computing media. Prototypes supporting real-world experiments face scaling issues relating to dynamically controlling and adapting the computing media. In this work, we propose exploiting the stable and well-understood digital technology as the computing media. The protocomputing system is implemented as a protoarchitecture that is mapped into the digital media. The proposed protocomputing system consists of a reaction-diffusion process that is observed in fundamental chemical and biological condition developments. A cellular automata is used as the method of implementing the algorithm [10, 11], which is mapped into a digital protoarchitecture. As a merit of the proposed approach, a real application example of video processing is demonstrated in real-time. Abnormal behavior, saliency detection, and attention attraction are part of the reflex cognitive processing of mammals. Motion detection is a prerequisite to these tasks. Motion detection under conditions where the moving object has no texture is used as the application exploiting the protocomputation architecture to achieve real-time.

Section 2.1 details the problem and the selected cellular automata implementation of the reaction-diffusion algorithm. The original RD algorithm is adapted to support its flexible parallel operation as a solution to enable real-time operation over a wide range or size of the input space. Sections 2.2–2.4 present the fundamental protoarchitecture and its implementation into a digital computation medium. The control of the architecture and its real-time scheduling are presented. Section 3.1 presents implementation results and demonstrates the real-time capability of the proposed protoarchitecture over a real-world application. Finally, a methodology that enables scaling the input space (input image size) and that dictates the correct level of parallelism to reach real-time operation is presented in Section 3.2. Section 4 presents the conclusions.

2. Materials and Methods

In the following, a method based on the ancestral reaction-diffusion algorithm is proposed to a real-life video example consisting of creating texture into textureless objects to the aim of enabling the real-time motion vector extraction. A fundamental condition to achieving real-time operation consists in the specific development of a protoarchitecture and its mapping into a digital medium.

Initially developed within the context of video compression techniques [12], motion vector estimation has found new application fields with the advent of modern ubiquitous consumer electronic products and smart vision sensors, with examples including target tracking [13], hand gesture user interface [14], image stabilization [15], surveillance, event analysis and automatic anomaly detection from monitoring cameras, depth map estimation, and 3D vision, which reveal a dynamic research activity over the recent years [16]. Classical motion vector extraction algorithms using block matching [17] are known to deliver accurate results in highly textured regions of images, while they perform poorly in low textured regions. The recent usage of block matching in smart machine-vision applications poses increased constraints on the necessity of a correct motion vector field extraction in real-time. The generated field of motion vectors is used in further algorithmic processing and thus must satisfy severe criteria in terms of spatial homogeneity [18]. For example, postprocessing tasks such as classification may require an accurate density of the motion vector field in order to decrease their error rate. Edges tracking and particle filters have been proposed as a solution to detect and track textureless objects, resulting in computationally complex algorithms, for example, [19], hence with limited practical usage in energy-constrained portable systems.

2.1. Cellular-Automaton Generating Spatial Patterns Aiming at the Motion Vector Estimation of Textureless Objects. Reaction-diffusion (RD) algorithms have initially been introduced to model the concentration dynamics of different chemical species placed in presence of each other within a single container [20]. Diffusion processes represent a fundamental natural phenomena underlying the macroscopic effect observed as a result of numerous irregular microscopic motion of individual particles that spread out as a result of the motion of each. The particles may consist of chemicals, cell or bacteria, or even larger species (animals) [21]. Reaction describes the conversion of one involved particle species into another, as a result of the diffusion of two or more substances.

In its original expression, the RD model is defined over a continuous spatial domain. The dynamics of RD considering the activators and inhibitors over independent spatial fields is adapted following the procedure that is presented in [22]. Diffusion of activators and inhibitors on 1D space is described by the following diffusion equation:

$$\frac{\partial u(x,t)}{\partial t} = D\frac{\partial^2 u(x,t)}{\partial x^2}, \tag{1}$$

where D represents the diffusion coefficient of the activators or inhibitors, x the space, and $u(x,t)$ the spatial concentration at time t. The general solution is given by

$$u(x,t,n) = \frac{1}{2\sqrt{\pi Dt}} \int_{-\infty}^{\infty} u_n(X)\, e^{-(x-X^2)/4Dt} dX, \quad (2)$$

where $u_n(x)$ represents the initial concentration at $t = 0$ and n represents the cycle index, further used in (4) to (6). In [22], two independent diffusion equations for activators and inhibitors were introduced, where (i) diffusion coefficient of inhibitors D_v was set at much larger value than that of activators D_u and (ii) the same initial concentrations were set to the two diffusion equations. Here we introduce a novel method that uses one diffusion equation only to describe the original model equations. First, diffusion of activators is performed during time T_u with initial concentration $u_0(x)$, and the result is obtained as $u(x, T_u, 0)$. Then the diffusion is further continued for additional time T_v, and the result is described by $u(x, T_u + T_v, 0)$. It should be noticed that $u(x, T_u, 0)$ and $u(x, T_u + T_v, 0)$ represent results of diffusions of activators and inhibitors, respectively, because $u(x, T_u + T_v, 0)$ is obtained by diffusion of $u(x, T_u, 0)$ during T_v, which is equivalent to the two-diffusion-equation system under the same initial concentration with $D_v > D_u$. Second, as in [22], differential concentration of activators and inhibitors is amplified by the sigmoid function and is set to the subsequent initial condition $u_1(x)$ as

$$u_1(x) = f\left(u(x, T_u, 0) - u(x, T_u + T_v, 0) - c\right),$$
$$f(x) = \frac{1}{1 + e^{-\beta x}}, \quad (3)$$

where β represents the gain of the sigmoid function and c represents the offset value introduced in [22]. When $T_u \approx 0 \ll T_v$, $u(x, T_u, 0) \approx u_0(x)$, and hence, by assuming $c = 0$ and redefining T_v as T, (3) is simplified as

$$u_1(x) = f\left(u_0(x) - u(x, T, 0)\right). \quad (4)$$

By using $u_1(x)$, subsequent $u_2(x)$ is obtained by

$$u_2(x) = f\left(u_1(x) - u(x, T, 1)\right), \quad (5)$$

and the generalized update equation is

$$u_{n+1}(x) = f\left(u_n(x) - u(x, T, n)\right). \quad (6)$$

Under specific parameters sets, the discrete version of RD algorithms generates repeatable, stable spatial patterns [23, 24] consisting of stripes or spots from an initial image and applying an iterative processing. Henceforth, this property is used to create texture in a video scene and in particular into textureless moving objects. Assuming that the interframe movement is sufficiently small, then the texture follows the movement of the textureless object, enabling the detection of the movement of texture, rather than object edges only, as a limitation of classical algorithms. An algorithm that implements the aforementioned principles has been presented in [25]. The RD process is defined in its fundamental form over a continuous spatial domain. Following the procedure described in [22], the dynamics of the RD process is adapted to support the diffusion of activators and inhibitor in independent discrete fields, which are eventually convoluted into a 2D array of cells.

Two-dimensional RD over images and video sequences is applied from a combination of one-dimensional processing, following the procedure presented in [25] and which delivers patterns that are stable, even in a noisy environment. The theoretical approaches governing the derivation of the reaction-diffusion equation in a continuous time and spatial domains are modified to support the circuit-level approach as expressed in

$$a_i(t+1) = \frac{a_{i-1}(t) + 2a_i(t) + a_{i+1}(t)}{4}, \quad (7)$$

where a is a natural number representing the pixel intensity, i indexes the pixel in the row of an image, and t represents the number of diffusion processes, or diffusion steps. A discrete update consists of a number of diffusion steps iteratively applied from the initial row of pixel intensities. The subsequent reaction consists of subtraction and amplification by a nonlinear logistic function. Several updates are required to generate stable patterns.

This study extends the earlier theoretical study to the implementation of the reaction-diffusion pattern generation algorithms as a protoarchitecture that is mapped into a digital medium. A field-programmable gate array (FPGA) is selected for the physical implementation enabling real-time operation of the protocomputation based application.

2.2. One-Dimensional Reaction-Diffusion Protoarchitecture Aiming at a Digital Media Implementation. A data-flow and a block diagram representing the one-dimensional pattern generation process are shown in Figure 1. The ideal example of a step input is presented in Figure 1(a), which is processed by the system until reaching a stable state after update number 10 as a spatial wave (one-dimensional pattern). The progress of the step input along the first update is shown, evidencing the edge smoothing obtained by the diffusion, the subtraction of the diffused state from the input step, and the result of nonlinear amplification. The state-diagram of the RD system is presented in Figure 1(b), showing the two fundamental states, namely, diffusion and reaction consisting of subtraction and amplification, as well as the iterative data passing process that is required and represented as arrows. The state-diagram of the system including the filter is presented in Figure 1(c), where the lower part of the state-diagram represents the RD process that is followed most of the time, and the upper part of the state-diagram represents the Filtering process that is followed repeatedly. Recognizing that the Filtering process consists of a regular diffusion step, the integrated state-diagram of Figure 1(d) can be derived, which evidences the possible use of identical resources for diffusion and filtering. Controlling and canceling collision of spatial waves is obtained by the action of the spatial Filtering process that is applied after the first update, and then repeatedly after a fixed number of updates, for example, Figure 1(a), in the Filter process (red box). The filter consists

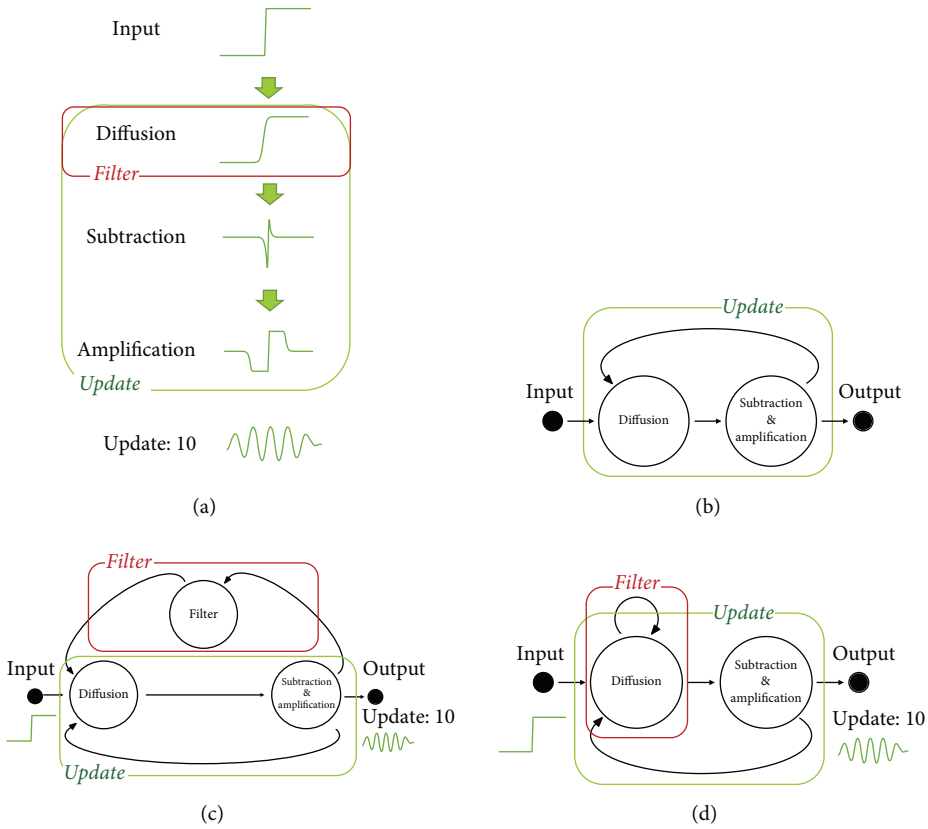

Figure 1: One-dimensional pattern generation concept. (a) Data-flow diagram showing a step input to the Update and Filter processes and the corresponding sequential signal processing, resulting into a stable spacial intensity wave (1D pattern). (b) State-diagram showing the Update process, and (c) state-diagram showing the Update and Filtering processes and evidencing the iterative operation. (d) Integrated state-diagram, evidencing that the filtering is executed as one diffusion process.

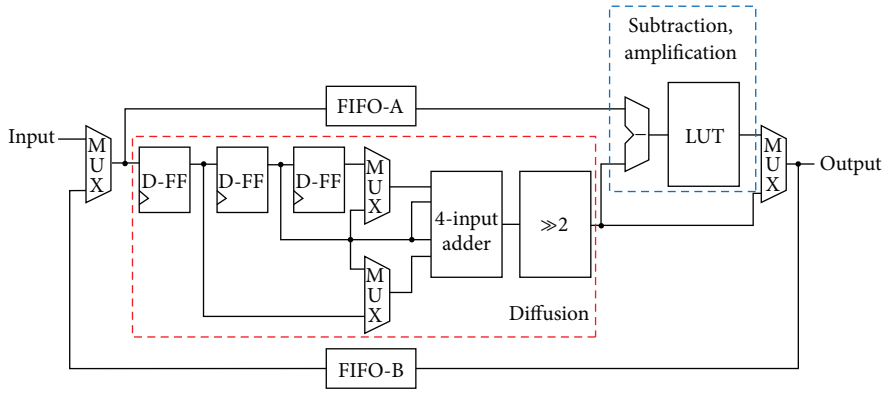

Figure 2: Protoarchitecture of the one-dimensional RD module.

of one step of diffusion (blurring), which is not followed by the reaction process (subtraction and amplification). In addition, the spatial filter controls potential effects of noise which is diffused such that its effect is not amplified, resulting into damping out its propagation. In practical terms, the filtering repetition frequency, the number of diffusion steps, and the maximal gain of the sigmoid function factors are determined empirically; for example, in our application case, one diffusion filtering step is applied every four updates while the gain of the sigmoid function is equal to five.

The protoarchitecture of the one-dimensional processing module is derived from [26] and consists of four major submodules presented in Figure 2. The first diffusion submodule is in charge of the calculation of diffusion according to (7). The second reaction submodule processes the subtraction and amplification required to complete one update. Finally,

two line buffers implemented as first-in-first-out (FIFO) memory segments FIFO-A and FIFO-B temporarily store one row or column, and one processed row or column, respectively. Data is transferred and processed as a word, and thus all operators and transfer lines process or store and, respectively, carry words, which is not drawn to increase visibility of the figure; 12-bit words are used in the specific developed application further detailed. The entire protoarchitecture operates in streaming mode, where data is constantly being processed from consecutive memory locations. Hence memory management is reduced to global read/write operation control, and no address needs to be generated for the internal process.

The input is streamed from a CMOS sensor or a temporary input frame buffer. In both cases, the control of addresses is limited to a row-by-row scanning scheme, which is part of any CMOS imager, while it consists of a simple counter, when the frame buffer memory is used. An input multiplexer serves as a source selection device. The RD process starts with the acquisition of one line (row or column) of a frame in a video sequence that is serially streamed to the input terminal as consecutive words representing the pixel intensity of neighboring, consecutive pixels. The input multiplexer is set to select which input to pass, which is delivered to the diffusion submodule, while it is also stored into FIFO-A in the first step of diffusion. FIFO-A has the capacity to store an entire row or column and is deactivated upon completion of storing.

In parallel to delivering to FIFO-A, consecutive pixels are also delivered to the diffusion block. A delay line formed of three clocked D flip-flop (D-FF) banks accepts incoming pixels, such that they store three neighboring pixels in a row after three clock cycles. Hence, the D-FF bank located in the middle position stores pixel a_i, the D-FF bank located on the rightmost part of the chain stores pixel a_{i-1}, and the D-FF bank located on the leftmost part of the chain stores pixel a_{i+1}. This situation is suitable to execute the diffusion computation of pixel a_i according to (7). The three pixel intensity values are routed through multiplexers to the four inputs of an adder. Eventually, the adder output is shifted to the right two times which implements the operation of division by four. This process is repeated at each clock cycle, thus in a streaming mode, and the diffusion of the new consecutive central pixels a_i is computed. Each new computed diffusion value is routed to FIFO-B by a multiplexer. Hence, at completion of the diffusion, one line (row or column) resulting from one diffusion step is stored in FIFO-B. The two multiplexers located at the input of the adder may be reconfigured to properly handle limit conditions that occur at the boundary of the image; in general, constant boundary conditions are admitted, where the limit value, for example, a pixel in row 0 or column 0, is used two times in replacement of values of a_{i-1} and a_i that do not exist, because they are outside of the physical array. The central pixel a_i always connects to two of the four-input adders, for example, to realize the multiplication coefficient equal to two.

A new step of diffusion can be started immediately after completion of a previous step. The input multiplexer is set to route FIFO-B into the diffusion submodule, and the process described above repeats. During this new step, no imager input is provided, and FIFO-A is deactivated, storing the initial nonprocessed frame. Several steps of diffusion may be processed this way, each implementing one iteration of inhibitor diffusion expressed as $u(x, T, n)$ in (6); their numerical count implements parameter T.

The reaction operation starts while the last step of diffusion is streamed through the diffusion submodule. The reaction submodule (subtraction and amplification) and FIFO-A are activated as soon as diffusion of the first diffused pixel is completed, that is, delivered at the output of the diffusion submodule. The first diffused pixel is subtracted from the first pixel stored in FIFO-A that pertains to the original nondiffused line. The result of subtraction is delivered to the final processing consisting of a sigmoid operation implementing the following function, $\zeta_a(x) = (\tanh(ax/2) + 1)/2$. Function $\zeta_a(x)$ and its parameters are tailored to the application by precomputation of the values. The function is implemented as a lookup table (LUT) storing the precomputed values and enabling fast and accurate result delivery. The rightmost multiplexer routes RD data from the output of the reaction submodule into FIFO-B. At the conclusion of the reaction computation, one update is completed.

Subsequent updates indexed $n + 1$ in (6) are performed until a stable row pattern is generated, yielding $u_{n+1}(x)$. No input is delivered from the image sensor or input frame buffer, and the input data originates from FIFO-B. Hence, a new update starts with delivering the data content of FIFO-B to the diffusion submodule and FIFO-A, in parallel, and proceeds following the procedure described above.

As determined from the algorithmic study, some Filtering process is periodically required and is interleaved between two updates. The Filtering process consists of a diffusion step, that is, without reaction. Consequently, the filtering operation is applied strictly using the diffusion submodule, while the reaction and FIFO-A submodules are deactivated.

At completion of the full RD computation of one line, results are delivered to the output. Subsequently, the entire RD process repeats, using the next line until all rows and columns have been individually processed by the one-dimensional module. At this point, the one-dimensional RD process of one frame is completed. Processed data is delivered for two-dimensional aggregation, and a new frame can be processed by the one-dimensional RD module.

2.3. Two-Dimensional Reaction-Diffusion Protosystem. Two-dimensional RD is applied to acquired images as the application of the one-dimensional RD process to every row and column, individually, as depicted in Figure 3(a). The hardware used to process each row or column is identical and consists of the circuit earlier presented in Figure 2. Nevertheless, a sequential processing of rows and columns would not yield real-time operation. As a benefit of the independent handling of row and columns, the process can be accelerated by parallel processing. Two columns and two rows are concomitantly processed in a one-dimensional RD computation. The subsequent two-dimensional aggregation consists of a multiplication of the row and column intensity

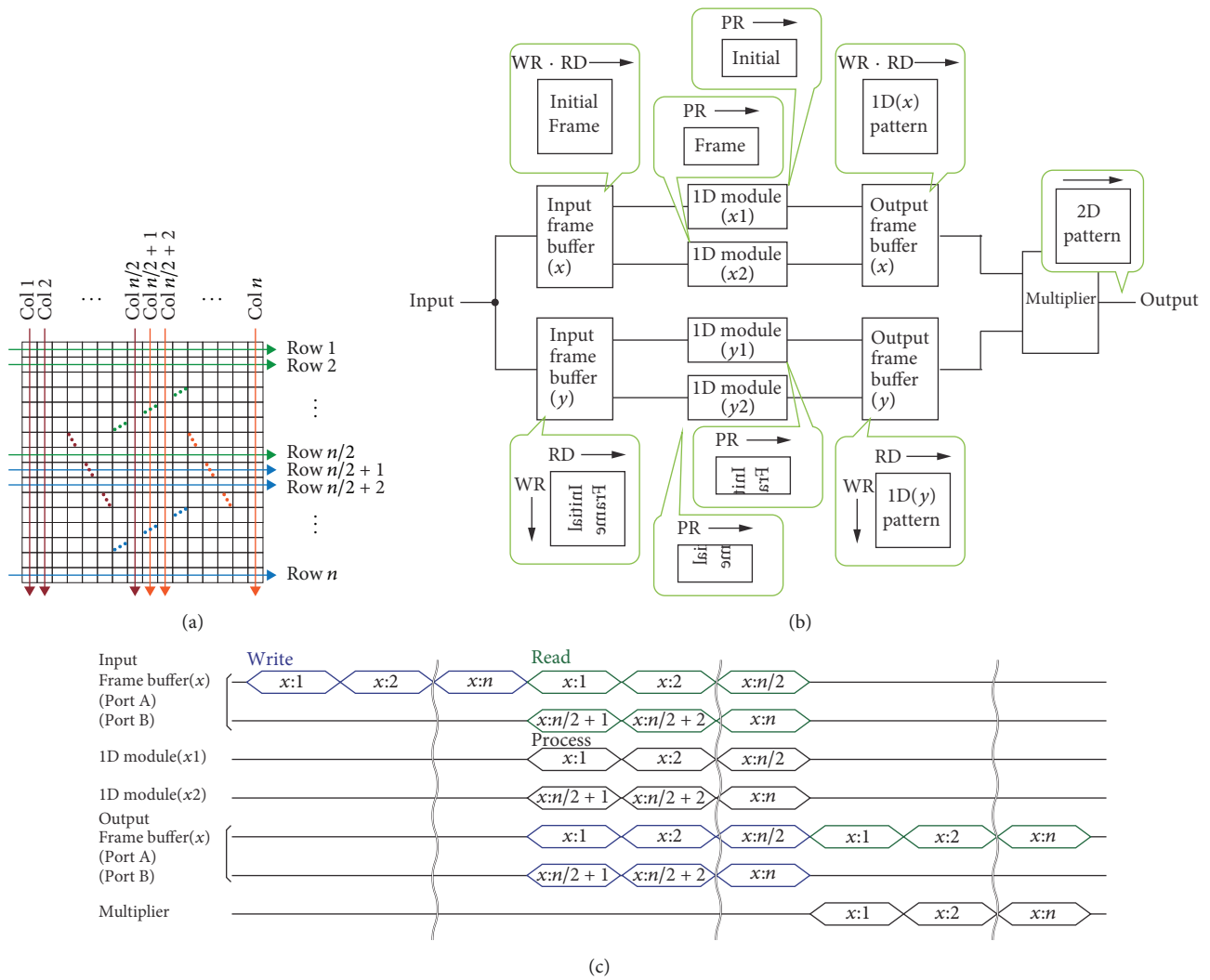

FIGURE 3: Protoarchitecture and operation of the two-dimensional RD system. (a) Independent row and column processing in a one-dimensional algorithm. (b) Block-level system protoarchitecture. (c) Detailed timing chart.

values of a pixel obtained from the RD individual processing of rows and columns.

The two-dimensional real-time RD processing protoarchitecture is presented in Figure 3(b). RD, WR, and PR, respectively, denote read, write, and processing operations. Data streaming to the system input originates from an image sensor, in real-time, while the output is to be delivered to further processing units, for example, extracting the motion vectors by operating block matching in consecutive resulting RD frames and further exploiting this result. The architecture comprises two parallel data paths, each processing two rows and two columns, respectively, in parallel. Two input frame buffers store images of identical horizontal and vertical resolution, for example, 120 × 120 pixels or 250 × 250 pixels. The two images are processed by four one-dimensional RD modules, and the results are stored into two output frame buffers. The signals that control the four subpaths are identical which significantly simplifies the controller. In order to achieve this feature, the image that is stored into input frame buffer (y) is a rotated and mirrored version of the image stored in input frame buffer (x). This storage organization is achieved by appropriate address generation at memory writing. Hence, in read mode and using identical frame buffer addresses, input frame buffer (x) delivers two consecutive rows while input frame buffer (y) delivers two consecutive columns. Finally, a multiplier aggregates the row and column data obtained for every pixel from output frame buffers (x) and (y) and delivers the RD image to further processing.

The detailed timing chart of the system is summarized in Figure 3(c). Here, frame buffer write operations are marked in blue, read operation is marked in green, and RD module processing is marked in black. The change of a read/write/process mode is written over the first involved data, while the vertical wavy line depicts data that is not displayed to improve visibility. Only the timing chart of the upper data path handling two rows is shown, for the sake of clarity. The timing chart of the lower data path handling two columns can be obtained by replicating the timings of the

Table 1: FPGA synthesis results of the core system.

Characteristic	120 × 120	250 × 250	Characteristic	120 × 120	250 × 250
Total logic elements	499	2,221	Total registers	656	3,029
Total memory bits	745,472	3,203,072	→ Usage rate	13%	57%
RAM blocks (·/553)	92	396	DSP blocks[a]	1	15
Total RD modules	4	20[b]	Max. frequency	69.12	66.9
(Horizontal, vertical)	2, 2	10, 10	(F_{max}) MHz		
Processing fps at F_{max}	34.43	38.89	Processing fps at 50 MHz	24.91	29.06

[a]Multiplier used in the RD module. [b]Operation is pipelined in order to restrain FPGA BRAM utilization.

upper data path, while changing x into y when input frame buffer (y) is in read model. Here $\langle x : n\rangle$ denotes row n of the frame, and $\langle y : n\rangle$ denotes column n.

Hence, two-dimensional frame processing for texture generation is performed by providing the one-dimensional RD module with data organized in a two-dimensional array following a time-division scheme, which reduces the complexity of the circuit and system as a whole, while the parallel operation accelerates the global processing to reach to real-time operation.

2.4. Hardware Implementation and Target Platform. The protoarchitecture of the RD system is implemented in a hardware platform with FPGA for real-time processing. A Terasic DE10 Standard board embedding a Cyclone V FPGA and equipped with a TRDB-D5M, 5 M pixel CMOS sensor image acquisition peripheral board are used. Data is acquired from the image sensor in 8-bit RGB format. Results are displayed using an external display screen connected through a VGA link. The board operation frequency is 50 MHz while the core circuits could operate up to F_{max} (Table 1), and the image acquisition rate is programmable up to 70 fps (frames per second).

The operational characteristics and synthesis results pertaining to the core of the system that embeds the 1D module of Figure 2 are presented in Table 1. The entire core is considered, specifically including input frame buffers supporting the 120 × 120 and 250 × 250 window processing. In this latter case, the level of 1D module parallelism is increased from 4 to 20 and pipelining is applied as a way to limit FPGA block-RAM usage at constant input frame buffer bit size. However, service circuits such as the camera interface are deliberately not included into the synthesis results.

3. Results and Discussion

The operation of the entire two-dimensional RD system protoarchitecture is confirmed using numerical simulation and seminatural images. Consecutive video images of size 120 × 120 pixels are created from a natural background that is covered by a textureless moving object. Though the resulting image appears synthetic to human eyes, the recreated scheme in fact conforms an expected real condition, where an object that appears without texture moves into a scene. The vanishing of texture could be resulting from local saturation of imager pixels due to a high-intensity reflection in the scene, for example, white object moving under high illumination, or insufficient gain of the imager, potentially resulting from an image sensor self-adjusting its gain to a high dynamic range environment.

Figure 4 presents numerical simulations of the aforementioned situation. A Verilog RTL model of the hardware is used to process shown data. In order to obtain visually meaningful evidence, ten frames separate the original input shown in Figure 4(a) with respect to Figure 4(b); the movement of the original object is clearly perceptible. The motion vectors extracted using the conventional block-matching technique are compared to the result of the RD-based proposed technique. The results are presented in Figure 4, where (a) and (c) pertain to the same image, and (b) and (d) pertain to another same image. The motion vectors obtained from a conventional block-matching algorithm are shown in red in (a) and (b), clearly evidencing that motion can only be detected at the edges perpendicular to the motion. In contrast, (c) and (d) show the vector field obtained using the proposed technique and hardware that is simulated using an RTL model, where a dense array of red motion vectors is detected inside the object, along edges of the generated patterns that have followed the textureless object movement.

3.1. Real-Time Hardware Protosystem. The hardware platform presented in Section 2.4 is used to generate real-time results of RD pattern generation in natural movies acquired from the camera, in order to further generate motion vector fields. The image field is limited to 250 × 250 pixels by hardware physical resources. Results are presented in Figure 5 confirming the correct operation of the protosystem.

A real indoor environmental situation is used to confirm the capacity of the algorithm and hardware implementation that consists of a single dark object of rectangular shape moving in front of a text background. The processed frame size is equal to 120 × 120 pixels, and the object movement has been made sufficiently significant, for example, to create qualitatively visible results. The background may be acquired and processed slightly blurred due to the limited depth of field of the unsophisticated optical lens system that is used. Images are extracted from the acquired and processed video flow and shown in Figure 5. The dynamic range of images is compressed to 3 bits prior to processing, which can be achieved in hardware by straightforward wire connecting. Hence, images may appear dark to human observers. The motion vector field that is obtained from software block matching of RD

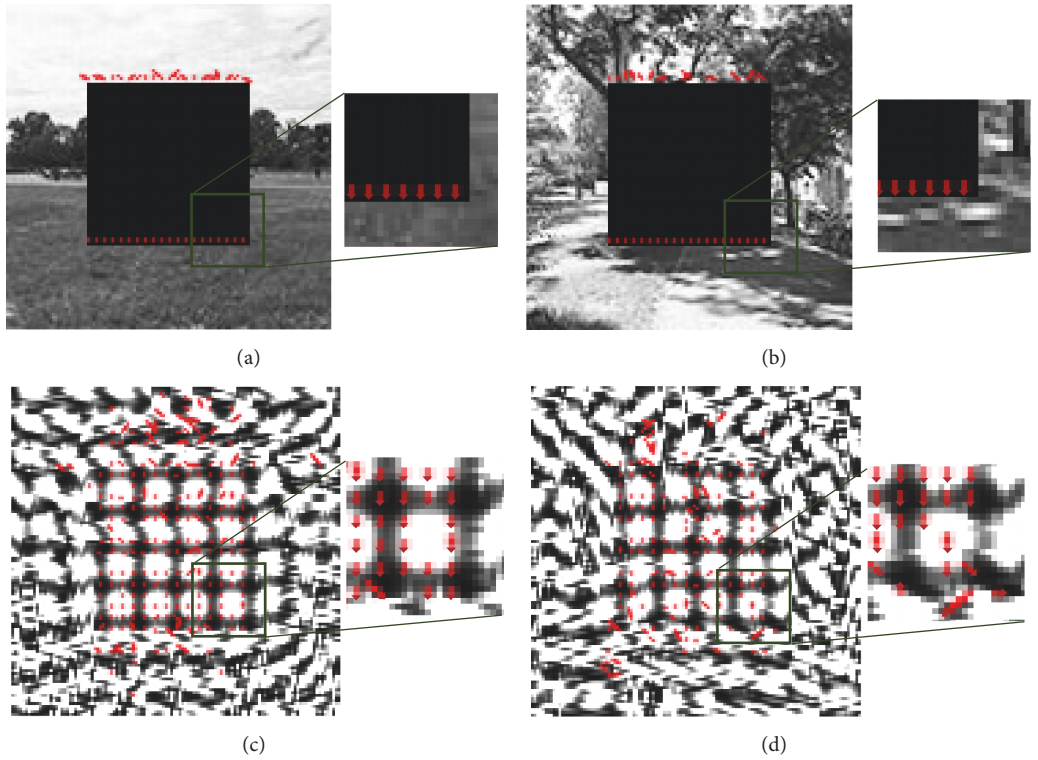

Figure 4: Simulations of an object movement corresponding to one pixel and subsequent detection of motion vectors depicted in red. ((a) and (b)) Motion vector results using a conventional block-matching algorithm; ((c) and (d)) corresponding results using the proposed RD-based method and hardware.

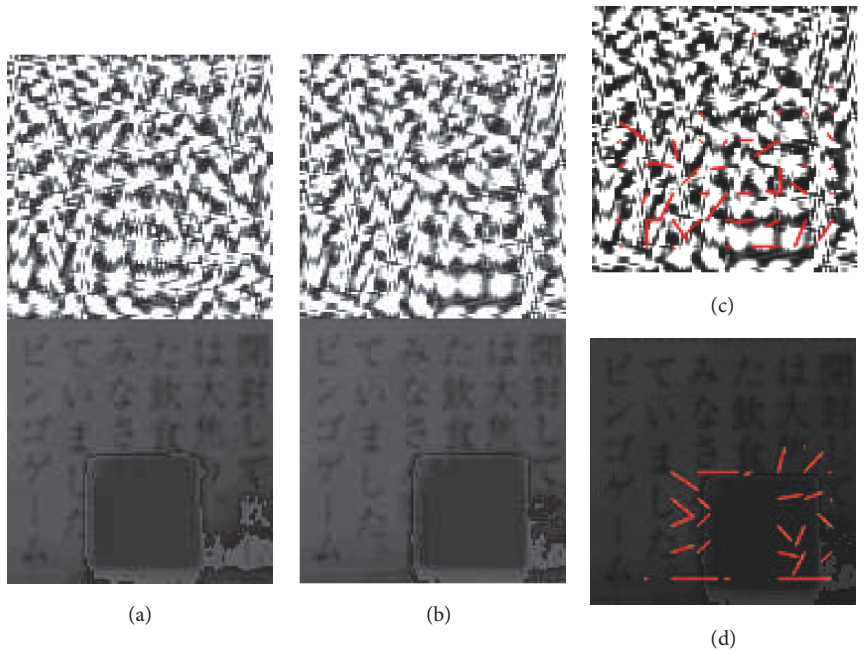

Figure 5: Real-time extraction of RD patterns of a textureless object moving over a text background and subsequent estimation of the corresponding motion vector field. (a) First acquired frame and corresponding RD pattern; (b) second acquired frame and corresponding RD pattern. (c) Close-up of the RD pattern obtained from the second acquired frame with overlaid motion vectors and (d) close-up of the second acquired frame with overlaid motion vectors that are acquired using a conventional block-matching algorithm.

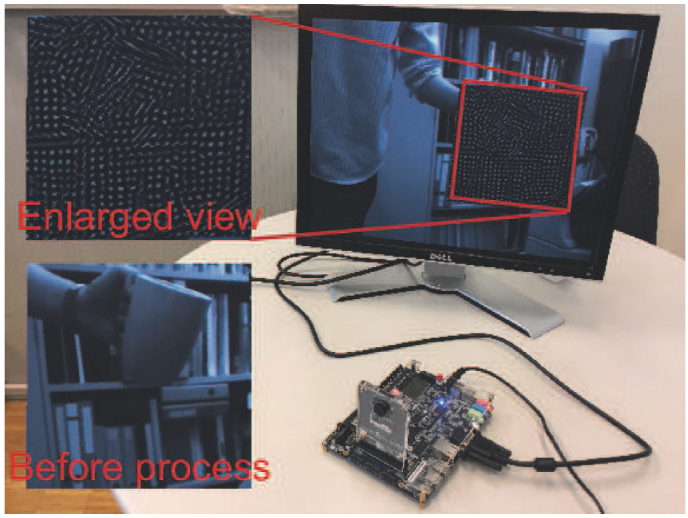

Figure 6: Real-time experimental setup. The processed window is superposed to the full VGA background. The unprocessed and processed window are enlarged in the inlets.

frames is presented in Figure 5(c). The generation of a dense field of motion vectors that appear in red color inside the moving textureless object is confirmed. Some unexpected motion vectors appear outside the area of the moving object. Those which appear in regions previously covered by parts of the moving object are incorrect, though expected; they confirm the necessity to satisfy the hypothesis of a slow moving object or high acquisition speed. The system is very sensitive to any movement, but also to luminance changes within the scene, as a benefit of the iterative RD process that includes a high amplification nonlinear process. Hence, some vectors may be generated within the scene that do not correspond to a movement, but correspond to a parasitic change of luminance. These vectors are incorrect, but very few. A different set of parameters of the algorithms may fix these incorrectly determined movement vectors, however, to the cost of a decrease in the proportion of the correctly detected vectors. Consequently, as a real-time system, a trade-off between the speed of operation (number of iterations) and the performance metrics that characterize the system must be found. As in any sensor and detection system, these performance metrics are defined from true positive and negative, false positive and negative detections, with respect to the total set of expected vectors. These metrics must be extracted from empirical experiments carried over multiple sets of data, that is, various videos acquired with different conditions, for example, image size, motion distance, and illumination. This systematic study is beyond the scope of this paper. In addition, conforming to the theory of reaction-diffusion systems, the proposed system is observed to operate in optimal conditions consisting of a high contrast of intensity between the object and its background, and in presence of a highly textured background where existing features promote local and global fixing of the RD patterns. Finally, the motion vector field obtained using classical block matching only generates vectors at the edges, or underneath zones that were covered by the object in the previous frame. Consequently, the number of vectors that appear in Figure 5(d) is expected to significantly reduce with a smaller movement of the textureless object. In contrast, the proposed technique results should improve with low amplitude of the movement, where the number of incorrect vectors should decrease, while correct vectors should remain stable.

The real-time experimental setup is presented in Figure 6. The optimized parallel implementation of the protosystem enables real-time operation in a window of 250 × 250 pixels. The enlarged view shows the patterns generated inside the cup that is rendered textureless due to its color and a high-intensity illumination of the scene. The window size of 250 × 250 pixels does not represent any limitation of the method or protoarchitecture but relates to the memory capacity of the FPGA. Hence, window size with respect to hardware resource scaling is a major concern that is discussed in the following.

3.2. Real-Time Hardware Scaling of the Protoarchitecture. The computational capacity of protocomputational systems is dictated by the nature of the supporting medium, as well as its physical extent. The concept of real-time operation as such is absent in natural computation. In contrast, the emulation of a natural phenomenon over a substrate of a different type poses issues of accuracy with respect to reflecting natural timings. This issue is well understood in computer science and has been addressed from the concept of real-time that stipulates the time elapsed to compute the simulated model conforming to the time required by the real environment. It is common for emulating systems including microelectronic and computer systems to take benefit of virtual, parallel, or redundant implementations that compute in parallel and support the real-time capability. Within this context, scaling the extent of the protocomputation system has little or no meaning with respect to a natural system. However, considering the selected video application, and the parallelization concept discussed above, scaling appears a relevant hardware issue

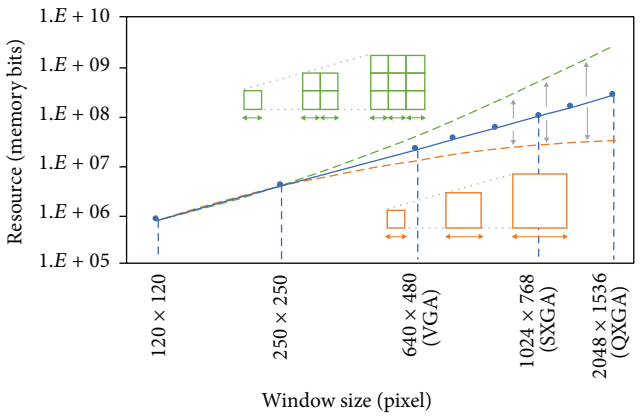

Figure 7: Window size and hardware resource scaling (log-log scale).

that is fully supported by the proposed protoarchitecture, and is discussed in the following.

The size of the image that is acquired is generally dictated by application or image sensors standards. The hardware implementing the RD algorithm should adapt, for example, to cover the entire image. Considering real-time operation a compulsory specification results into the stringent need to allocate additional hardware resources enabling scaling processing to large image sizes.

System scaling may be supported taking three possible protoarchitectural adaptations into account. The BRAM-parallel protoarchitecture (BPA) consists of increasing the number of pixels in each row while also increasing the level of parallelism to keep 30 fps, that is, the number of parallel data paths (1D modules) in Figure 3. Eventually, an image may form one single window. A different level of parallelism should be selected pertaining to the number of units processing the rows and the columns in parallel, in order to accommodate rectangular image formats. The module-parallel protoarchitecture (MPA) dictates parallelizing BPA blocks of small size. Some overlap between neighboring small-size windows may be used to guarantee coherency of the generated motion vectors. The fixed boundary conditions of the RD process dictate this overlap, whose size depends on the RD parameters. Finally, a hybrid of the BPA and MPA may be considered.

The decision criterion regarding the suitable scaled protoarchitecture is based on the analysis of the hardware resources that are theoretically required in a first-order consideration of the window size, as presented in Figure 7. The theoretical analysis (blue trace) dictates a linear relationship between window size and hardware resources. The BPA and MPA architectures are equivalent at the location of this theoretical curve. In practical terms, however, second-order effects must be considered, consisting of the necessity to adapt RD parameters to the size of the window, potentially to the application. For example, the number of RD updates that must be applied to obtain stable patterns may vary with respect to the window size, which eventually reflects into hardware resources allocation. Hence, a decrease of some key parameters may result in a real curve proportionally lower than theoretically predicted and potentially a tendency to saturate (red). The BPA architecture is preferable under these conditions, which takes full benefit of the hardware resource scaling. In contrast, an increase of some key parameters may result in a real curve proportionally higher than theoretically predicted and potentially a tendency to an exponential behavior (green). The MPA architecture is preferable under these conditions, which restrains the hardware scaling to a strict theoretical behavior.

4. Conclusions

This paper demonstrates the correctness of an approach consisting of using a digital physical medium to the purpose of implementing the reaction-diffusion algorithm. A dedicated protoarchitecture is developed, which is subsequently mapped into the implementation media. Real-time operation of a realistic video processing application is demonstrated using the proposed method and protoarchitecture, whereas classical algorithms are known to fail. Specifically, the reaction-diffusion is applied to textureless objects moving into a video frame acquired at 30 fps in order to create texture allowing the generation of dense fields of motion vectors in real-time. Simulations and real operation results using windows of sizes up to 250×250 pixels confirm the suitability of the algorithm, the protocomputing architecture, and its hardware implementation. A scaling method is presented that supports real-time operation.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This study was supported by the JSPS Grants-in-Aid for JSPS Fellows and a Grant-in-Aid for Scientific Research on Innovative Areas [2511001503] from the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan.

References

[1] S. Haykin, *Neural Networks and Learning Machines*, Pearson, 3rd edition, 2008.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning, Adaptive Computation and Machine Learning series*, MIT Press, Mass, USA, 2016.

[3] A. Schmid, "Neuromorphic microelectronics from devices to hardware systems and applications," *Nonlinear Theory and Its Applications, IEICE*, vol. 7, no. 4, pp. 468–498, 2016.

[4] Y. Suzuki, "Harness the Nature for Computation," in *Natural Computing and Beyond*, vol. 6 of *Proceedings in Information and Communications Technology*, pp. 49–70, Springer, Tokyo, Japan, 2013.

[5] S.-J. Kim, "Efficient decision making by harnessing the computational power of physical phenomena," *IEICE Tech. Rep*, vol. 116, no. 514, pp. S2016–45, March 2017.

[6] M. Aono, M. Naruse, S.-J. Kim et al., "Amoeba-inspired nanoarchitectonic computing: solving intractable computational problems using nanoscale photoexcitation transfer dynamics," *Langmuir*, vol. 29, no. 24, pp. 7557–7564, 2013.

[7] A. Adamatzky, *Advances in Physarum Machines, Sensing and Computing with Slime Mould*, vol. 74, Springer, 2016.

[8] B. Kaçar and E. A. Gaucher, "Towards the recapitulation of ancient history in the laboratory: combining synthetic biology with experimental evolution," in *Proceedings of the International Conference on the Simulation and Synthesis of Living Systems*, pp. 11–18.

[9] M. Naruse, Y. Terashima, A. Uchida, and S.-J. Kim, "Ultrafast photonic reinforcement learning based on laser chaos," *Scientific Reports*, vol. 7, no. 1, article no. 8772, 2017.

[10] M. Markus and B. Hess, "Isotropic cellular automaton for modelling excitable media," *Nature*, vol. 347, no. 6288, pp. 56–58, 1990.

[11] J. R. Weimar and J.-P. Boon, "Class of cellular automata for reaction-diffusion systems," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 49, no. 2, pp. 1749–1752, 1994.

[12] A. J. Tabatabai, R. S. Jasinschi, and T. Naveen, "Motion Estimation Methods for Video Compression - A Review," *Journal of The Franklin Institute*, vol. 335, no. 8, pp. 1411–1441, 1998.

[13] T. Yokoyama, T. Iwasaki, and T. Watanabe, "Motion vector based moving object detection and tracking in the MPEG compressed domain," in *Proceedings of the 7th International Workshop on Content-Based Multimedia Indexing, CBMI 2009*, pp. 201–206, Greece, June 2009.

[14] M. Tanaka, Japan patent Kokai, JP2014-52934A, 2014.

[15] Y. Takagi, Japan patent Kokai, JP2012-15959A, 2012.

[16] M. P. Vijaykumar, A. Kumar, and S. Bhatia, "Latest trends, applications and innovations in motion estimation research," *International Journal of Scientific & Engineering Research*, vol. 2, p. 1.

[17] N. K. Parmar and M. H. Sunwoo, "Recent progress on block-based motion estimation techniques," *IETE Technical Review*, vol. 32, no. 5, pp. 356–363, 2015.

[18] M. Mori, T. Itou, M. Ikebe, T. Asai, T. Kuroda, and M. Motomura, "FPGA-based design for motion vector estimation exploiting high-speed imaging and its application to motion classification with neural networks," *Journal of Signal Processing*, vol. 18, no. 4, pp. 165–168, 2014.

[19] C. Choi and H. I. Christensen, "3D textureless object detection and tracking: An edge-based approach," in *Proceedings of the 25th IEEE/RSJ International Conference on Robotics and Intelligent Systems, IROS 2012*, pp. 3877–3884, Portugal, October 2012.

[20] I. R. Epstein and J. A. Pojman, *An Introduction to Nonlinear Chemical Dynamics, Oscillations, Waves, Patterns, and Chaos*, Oxford University Press, 1998.

[21] J. D. Murray, in *Mathematical Biology I: An Introduction*, Springer, New York, NY, USA, 3rd edition, 2002, Chapter 11.

[22] Y. Suzuki, T. Takayama, I. Motoike, and T. Asai, "Striped and spotted pattern generation on reaction diffusion cellular automata: theory and LSI implementation," *International Journal of Unconventional Computing*, vol. 3, p. 13, 2007.

[23] J. D. Murray, in *Mathematical Biology I: An Introduction*, Springer, New York, NY, USA, 3rd edition, 2002, Chapter 7.

[24] M. Gerhardt and H. Schuster, "A cellular automaton describing the formation of spatially ordered structures in chemical systems," *Physica D*, vol. 36, pp. 209–221, 1989.

[25] M. Ushida, A. Schmid, T. Asai, K. Ishimura, and M. Motomura, "Motion vector estimation of textureless objects exploiting reaction-diffusion cellular automata," *International Journal of Unconventional Computing*, vol. 12, no. 2-3, pp. 169–187, 2016.

[26] K. Ishimura, K. Komuro, A. Schmid, T. Asai, and M. Motomura, "FPGA implementation of hardware-oriented reaction-diffusion cellular automata models," *Nonlinear Theory and Its Applications, IEICE*, vol. 6, no. 2, pp. 252–262, 2015.