molecules

MDPI

Article

A Resolution-Free Parallel Algorithm for Image Edge Detection within the Framework of Enzymatic Numerical P Systems

Jianying Yuan [1,2,3], Dequan Guo [1,2,3], Gexiang Zhang [4,5,*], Prithwineel Paul [5], Ming Zhu [2] and Qiang Yang [4]

1 The Postdoctoral Station at Xihua University Based on Collaboration Innovation Center of Sichuan Automotive Key Parts, Xihua University, Chengdu 610039, China; yuanjy@cuit.edu.cn (J.Y.); guodq@cuit.edu.cn (D.G.)
2 School of Control Engineering, Chengdu University of Information Technology, Chengdu 610225, China; zhuming@126.com
3 School of Aeronautics and Astronautics, University of Electronic Science and Technology, Chengdu 610054, China
4 Robotics Research Center, Xihua University, Chengdu 610039, China; qiangychd@126.com
5 School of Electrical Engineering, Southwest Jiaotong University, Chengdu 610031, China; prithwineelpaul@gmail.com
* Correspondence: zhgxdylan@126.com; Tel.: +86-028-8772-9583

Received: 3 February 2019; Accepted: 23 March 2019; Published: 29 March 2019

Abstract: Image edge detection is a fundamental problem in image processing and computer vision, particularly in the area of feature extraction. However, the time complexity increases squarely with the increase of image resolution in conventional serial computing mode. This results in being unbearably time consuming when dealing with a large amount of image data. In this paper, a novel resolution free parallel implementation algorithm for gradient based edge detection, namely EDENP, is proposed. The key point of our method is the introduction of an enzymatic numerical P system (ENPS) to design the parallel computing algorithm for image processing for the first time. The proposed algorithm is based on a cell-like P system with a nested membrane structure containing four membranes. The start and stop of the system is controlled by the variables in the skin membrane. The calculation of edge detection is performed in the inner three membranes in a parallel way. The performance and efficiency of this algorithm are evaluated on the CUDA platform. The main advantage of EDENP is that the time complexity of $O(1)$ can be achieved regardless of image resolution theoretically.

Keywords: membrane computing; edge detection; enzymatic numerical P system; resolution free

1. Introduction

In recent decades, image processing technology has experienced dramatic growth and widespread applications. Nearly no area escapes impact in some way by digital image processing. Normally, digital image processing includes three main levels, i.e., low-level, mid-level and high-level processing [1]. As one of the most basic operators in low-level image processing, edge detection can preserve the important structural properties of an image while significantly reducing the amount of data. This excellent property makes it a basic tool for many high-level image processing algorithms and is extensively applied in target tracking [2], image compression [3], and object recognition [4]. An edge can be defined as points in a digital image at which the image brightness changes sharply or has discontinuities. This phenomenon may be caused by depth discontinuous, illumination changes, or intrinsic texture properties of objects. In various edge detection algorithms, the gradient based

method is a type of classic edge detection approach with the merit of simple theory and good performance. However, as convolution calculation (i.e., a classic neighborhood computing in image processing) [5] is involved in this kind of algorithm, the time complexity increases squarely with the increase of image resolution. So it is difficult to deal with images with large resolution, such as remote sensing images, medical images, etc., in real time processing.

In order to achieve real-time calculation of high resolution images, many researchers have put much effort into this problem and several methods have been proposed. Generally, there are two main categories of resolutions. The first type of resolution concerns computational algorithms. In this kind of method, an elaboratively computational algorithm is usually designed to reduce the computational complexity. For the template matching problem, integral image [6] and dual-bound algorithm [7] are two classical approaches to speed up the computation. In [8], Fast LDA feature extraction is present, where steepest descent and conjugate direction methods are combined to optimize the step size in each iteration. In [9], common orthogonal basis extraction is proposed to extract a common basis of collection of matrices. The second category is based on hardware with parallel architecture, such as Graphics Processing Unit (GPU) [10–12] and Field Programmable Gate Array (FPGA) [13,14]. GPU uses hundreds of parallel processor cores executing tens of thousands of parallel threads to rapidly solve large problems having substantial inherent parallelism. However, with the shrinking volume of chips, semiconductor technology begins to reach its physical limits, which means the performance of conventional computing technique based on silicon chip integrated circuit microprocessors will be difficult to improve further [15]. Under this background, some scholars have turned their attention to non-traditional computing, such as quantum computing [16], DNA computing [17] and membrane computing (MC) [18]. MC is a new active branch of natural computing that simulates the function and structure of living cells and tissues, abstracting their biochemical reactions and material exchanges [19]. One of the most prominent features of MC is its capability of generating exponential growth space over a polynomial time, which makes it a promising method to resolve the conflict between the ever-increasing amount of data in the image processing field and the backward computing power of conventional computer [20]. In recent years, image edge detection and image segmentation [21–24], image smoothing [25], obtaining homology groups of 2D images [26,27], counting cells [28], Enzymatic numerical P systems image thinning [29] and corner detection [30] in MC framework have been vividly studied. In the previous literature about MC and image processing, much work is based on tissue-like P systems. However, when designing a parallel implementation program of an existing image processing algorithm, it is difficult to realize the mathematical formula in "tissue-like P systems language". The reasons for this are as follows. First, the data type of an image is an integer between 0 and 255. When design image processing algorithm uses tissue-like P systems, the image data should be coded to symbolic variables and those symbolic variables need to be decoded to integer for display as the algorithm finished. Second, most image processing algorithms are composed of several steps in determined logical order, which means variables in the membrane system need to be calculated in a deterministic way, rather than in a random manner. Since the rules in tissue-like P systems are implemented randomly, it is difficult to control the execution orders of different rules.

In order to overcome the above shortcomings when tissue-like P systems are combined to image processing, we make the first attempt to introduce enzymatic numerical P system (ENPS) to image processing. Concretely, a parallel algorithm for gradient based edge detection algorithm is designed and tested in the framework of ENPS. Besides the features described in [25], ENPS has another two good properties which make it particularly appropriate for image processing. One is that numerical variables and numerical expressions can be used directly in ENPS. Thus, image data can be directly operated without the additional encoding and decoding process. Another important characteristic is that enzymatic variables can control the execution orders of multiple rules in ENPS, i.e., the algorithms with complex logical steps can be designed easily.

The main contribution of this paper is that a parallel algorithm for image edge detection in the framework of ENPS, namely EDENP, is designed. The significant advantage of EDENP is that it

can achieve the time complexity of $O(1)$ theoretically, no matter how large the image resolution is. Moreover, the performance is equivalent to the performance run on the serial computing platform. This is very important for real projects, because most of the classical image processing algorithms have been widely proven to be effective in practical engineering, so the designed parallel implementation algorithm can be directly applied to the real image processing project without the need to perform large-scale testing. To the best of our knowledge, it is the first time to bridge problems from image processing with ENPS.

The rest of this paper is structured as follows. Section 2 introduces the definition, characteristics and applications of MC and ENPS. The problem statement is elaborated in Section 3. Section 4 discusses the EDENP algorithm in detail. The experiments and results are presented in Section 5. Conclusions are drawn in Section 6.

2. MC and ENPS

MC is a young biocomputing model proposed by Gh.Păun in 2000 [19]. The computational devices in MC are called P systems. Generally, a P system includes three ingredients: (i) the membrane structure; (ii) multisets of objects; (iii) rules of a bio-chemical inspiration. The multisets of objects are placed in the membrane, and evolved according to given rules which are usually applied in a synchronous non-deterministic maximally parallel manner. Since being proposed, MC has received great attention from scientists in many fields [31–33]. In the past 20 years, both the theory [32,34–37] and application [31,38–41] of MC have been greatly developed, and many different classes of P systems have been investigated. According to the way in which membranes are structured, there are three major types of P systems, i.e., cell-like [19], tissue-like [42] and spiking neural P systems [43,44]. Enzymatic numerical P system comes from numerical P system (NPS). NPS is a new special research branch of cell-like P systems, proposed by the founder of MC, Gh.Păun in 2006 [45]. In NPS, multisets of objects associated to membranes are sets of numerical variables, and the evolutionary rules are composed of a production function and a repartition protocol [46–48]. The most common widely application area of NPS is robot controller design [49–52]. Although NPS can deal with numerical variables, it can only execute one production function per membrane at a time. When there are multiple production functions per membrane, one is selected randomly. This limits its application in some situations where the rules should be executed deterministically. In order to solve this problem and expand the application of NPS, ENPS is put forward [24]. It is extended from NPS by introducing enzyme-like variables which can make rules run deterministically [53]. The standard form of ENPS is defined as follows:

$$\Pi = \Big(m, H, \mu, \big(Var_1, E_1, \mathrm{Pr}_1, Var_1(0)\big), \ldots, \big(Var_m, E_m, \mathrm{Pr}_m, Var_m(0)\big)\Big).$$

where:

1. m is the number of membranes used.
2. H is an alphabet that contains m symbols, and $H = \{1, 2, \ldots, m\}$.
3. μ is the membrane structure.
4. Var_i is the set of variables from membrane i and $Var_i(0)$ are the initial values for these variables.
5. Pr_i is the set of rules in membrane i, composed of a production function and a repartition protocol. A typical rule is as follows.

$$\mathrm{F}_{l,i}(y_{1,i}, \ldots, y_{k_l,i})\big|_{e_{j,i}} \rightarrow c_{l,1}|v_1 + c_{l,2}|v_2 + \ldots + c_{l,n_i}|v_{n_i},$$

where $e_{j,i}$ is a variable from Var_i different from $y_{1,i}, \ldots, y_{k_l,i}$ and $v_1, v_2, \ldots, v_{n_i}$. The rule can be executed at a time t only if $e_{j,i} > \min\{y_{1,i}(t), y_{2,i}(t), \ldots y_{k_l,i}(t)\}$. From the definition of ENPS, it is clear that with enzymes-like variables, the system can control multiple production functions to run in parallel in the same membrane deterministically [54]. Hence, it can overcome the disadvantages of traditional NPS that only run one rule nondeterministically at a time in a membrane. The ENPS with deterministic, parallel execution model has already been proved to be Turing universal [55,56]. In [57], it is shown

that any ENPS working in all-parallel mode or one parallel model can be simulated by an equivalent one-membrane ENPS working in the same mode. Since the proposal of ENPS, this model has been successfully applied in a wide range of domains, such as robot control [58], big data field [59], and sequential minimal optimization [60] fields. In this paper, ENPS is used to solve the problem of gradient based image edge detection.

3. Problem Statement

Edges generally occur in areas where the brightness of the image changes dramatically. These changes can be described by image gradients. Usually, a pair of convolution masks are used to estimate the gradients in the x and y directions, respectively, as shown in Equations (1)–(3), where ($Sobel_x$, $Sobel_y$), ($Prew_x$, $Prew_y$), (Rob_x, Rob_y) are three classic pairs of convolution masks. In this paper, we take Sobel operator as an example of gradient based edge detection (GBED). When the masks are sliding over the image, a square of pixels are operated. Then both directional gradients and absolute gradient magnitudes of image are computed, as shown in Equations (4) and (5), where I is the image, (g_x, g_y) are gradients in x and y direction respectively, $g_{i,j}$ is the absolute gradient magnitude of a pixel with coordinate (i, j), $2 \leq i, j \leq n-1$ for image with resolution of $n \times n$.

$$Sobel_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}; \quad Sobel_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{1}$$

$$Prew_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}; \quad Prew_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{2}$$

$$Rob_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}; \quad Rob_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{3}$$

$$g_x = Sobel_x * I; \quad g_y = Sobel_y * I; \tag{4}$$

$$g_{i,j} = \sqrt{{g_{x_{i,j}}}^2 + {g_{y_{i,j}}}^2} \tag{5}$$

When the gradient magnitude $g_{i,j}$ is computed, the difference between it and a predefined threshold θ is used to judge whether this pixel is an edge pixel or not, as presented in Equation (6), where $d_{i,j}$ is the difference. More concretely, if $d_{i,j}$ is greater than or equal to 0, then the pixel is assumed as an edge point, otherwise, it is a background point, as shown in Equation (7). It is worth noting that in real application, before thresholding, the gradient image should be filtered by "non-maximum suppression" for getting more real edges. In this paper, in order to simplify the algorithm, this step is ignored.

$$d_{i,j} = g_{i,j} - \theta \tag{6}$$

$$edg_{i,j} = \begin{cases} 1 & \text{if } (d_{i,j} \geq 0) \\ 0 & \text{if } (d_{i,j} < 0) \end{cases} \tag{7}$$

The program pseudo code of GBED run on conventional serial computer is illustrated in Algorithm 1, where the initial value of $edg_{i,j}$ is set to 0. From Algorithm 1, it can be deduced that the computational complexity is $O(n^2)$ because two loops are involved. When n becomes large, the calculations are very time-consuming under the serial computing platform.

Algorithm 1 The pseudo code of GBED

Input: $I(n*n)$
Output: $edg(n*n)$
1: **for** $i = 2 : n-2$ **do**
2: **for** $j = 2 : n-1$ **do**
3: Computing $g_{x_{i,j}}$
4: Computing $g_{y_{i,j}}$
5: Computing $g_{i,j}$
6: Computing $d_{i,j}$
7: Computing $edg_{i,j}$
8: **end for**
9: **end for**

In order to reduce the calculation time complexity, we attempt to introduce an enzymatic numerical P system to design a high parallel computing algorithm for edge detection. The details of how to design the algorithm will be given in the next section.

4. The EDENP Algorithm

This section starts with the mathematical model of EDENP followed by the detailed description of EDENP. The execution process and resources needed are discussed lastly.

4.1. Mathematical Model of EDENP

From Section 3, we know that the GBED algorithm contains four steps for a certain pixel in an image. In EDENP, the four steps will be executed in a cell-like P system under the control of enzyme variables, as illustrated in Figure 1. The initialization of variables, start and stop of the system will be controlled in the skin membrane. The directional gradients estimation will be completed in membrane 1. The absolute gradient magnitude estimation will take place in membrane 2. Membrane 3 is responsible for computing the image edge. The corresponding membrane structure is illustrated in Figure 2.

The mathematical expression of EDENP is as follows, and

$$\Pi = \Big(m, H, \mu, \big(Var_1, E_1, \mathrm{Pr}_1, Var_1(0)\big), ..., \big(Var_4, E_4, \mathrm{Pr}_4, Var_4(0)\big)\Big),$$

where

1. $m = 4$.
2. $H = \{1, 2, 3, 4\}$.
3. $u = [[[[\ ]_1]_2]_3]_4$.
4. $Var_1 = \{g_{x_{i,j}}, g_{y_{i,j}}\}$, $Var_2 = g_{i,j}$, $Var_3 = \{ed_1, ed_2, ed_3, E_{i,j}, E_{D_{i,j}}\}$, $Var_4 = \{x_{i,j}, edg_{i,j}, \theta, e_{1,1}, E_D\}$.

$x_{i,j}(1 \le i, j \le n)$, are the gray value of pixel with coordinate of (i, j) on the source image plane.

$edg_{i,j}(1 \le i, j \le n)$, are the corresponding edge points of the source image with initial value 0.

$\theta[threshold]$, is a numerical variable which is used as the threshold value for edge detection, and the value of threshold should be predefined.

$g_{x_{i,j}}(1 \le i, j \le n)$, are the horizontal derivative approximations at each pixel.

$g_{y_{i,j}}(1 \le i, j \le n)$, are the vertical derivative approximations at each pixel.

$g_{i,j}(1 \le i, j \le n)$, are the gradient magnitude approximations at each pixel.

$ed_1[0]$, is a numerical variable with initial value 0, which is used as the background value of the edge image.

$ed_2[1]$, is a numerical variable with initial value 1, which is used as the edge point value of the edge image.

$ed_3[-256]$, is a numerical variable with initial value -256, which is used as a intermediate variable.

5. E_k is a set of enzyme variables from membrane k, i.e., $E_1 = [\ ]$,$E_2 = [\ ]$,$E_3 = \{E_{i,j}, E_{D_{i,j}}\}$,$E_4 = \{e_{1,1}, E_D\}$.
6. Pr_k is the set of programs (rules) in membrane k, composed of a production function and a repartition protocol.

$\mathrm{Pr}_{1,CE_{i,j}}$:
$(|x_{i,j+2} + 2x_{i+1,j+2} + x_{i+2,j+2} - x_{i,j} - 2x_{i+1,j} - x_{i+2,j}|)|_{e_{1,1}} \to 1|g_{x_{i,j}}, (2 \le i, j \le n-2)$,

$\mathrm{Pr}_{2,CE_{i,j}}$:
$(|x_{i,j} + 2x_{i,j+1} + x_{i,j+2} - x_{i+2,j} - 2x_{i+2,j+1} - x_{i+2,j+2}|)|_{e_{1,1}} \to 1|g_{y_{i,j}}, (2 \le i, j \le n-2)$,

$\mathrm{Pr}_{3,CE_{1,i}} : 0|_{e_{1,1}} \to |g_{x_{1,i}}; (1 \le i \le n)$,

$\mathrm{Pr}_{4,CE_{n,i}} : 0|_{e_{1,1}} \to |g_{x_{n,i}}; (1 \le i \le n)$,

$\mathrm{Pr}_{5,CE_{i,1}} : 0|_{e_{1,1}} \to |g_{x_{i,1}}; (2 \le i \le n-1)$,

$\mathrm{Pr}_{6,CE_{i,n}} : 0|_{e_{1,1}} \to |g_{x_{i,n}}; (2 \le i \le n-1)$,

$\mathrm{Pr}_{7,CE_{1,i}} : 0|_{e_{1,1}} \to |g_{y_{1,i}}; (1 \le i \le n)$,

$\mathrm{Pr}_{8,CE_{n,i}} : 0|_{e_{1,1}} \to |g_{y_{n,i}}; (1 \le i \le n)$,

$\mathrm{Pr}_{9,CE_{i,1}} : 0|_{e_{1,1}} \to |g_{y_{i,1}}; (2 \le i \le n-1)$,

$\mathrm{Pr}_{10,CE_{i,n}} : 0|_{e_{1,1}} \to |g_{y_{i,n}}; (2 \le i \le n-1)$.

Those rules are used to execute Formula (1). The enzyme in $\mathrm{Pr}_{1,CE_{i,j}} \sim \mathrm{Pr}_{10,CE_{i,j}}$ must exist in enough amount so that the rules can be activated. Specifically, if the value of the enzyme $e_{1,1}$ is greater than variable $x_{i,j}(1 \le i, j \le n)$, then rules $\mathrm{Pr}_{1,CE_{i,j}} \sim \mathrm{Pr}_{10,CE_{i,j}}$ are effective. Since variable $x_{i,j}$ is the gray value of image, the maximum value is 255. So, the initial value of $e_{1,1}$ is set to 256, such that the condition modeled by rule $\mathrm{Pr}_{1,CE_{i,j}} \sim \mathrm{Pr}_{10,CE_{i,j}}$ are satisfied. It is important to note that the number of rules are $n \times n$, and all the rules are executed in parallel.

7. $\mathrm{Pr}_{21,CE_{i,j}} : (\sqrt{g_{x_{i,j}}^2 + g_{y_{i,j}}^2})|e_{1,1} \to 1|g_{i,j}; 1 \le i, j \le n$
$\mathrm{Pr}_{21,CE_{i,j}}$ are the rules which are executed by Formula (5). Hence, after executing $\mathrm{Pr}_{1,CE_{i,j}} \sim \mathrm{Pr}_{10,CE_{i,j}}$, the value of the variables $g_{x_{i,j}}$, $g_{y_{i,j}}$ are obtained. The maximum value of $g_{x_{i,j}}$ and $g_{y_{i,j}}$ is 255, and the enzyme $e_{1,1}$ is 256. So the condition of execution for rules $\mathrm{Pr}_{21,CE_{i,j}}$ is satisfied. Hence, all $n \times n$ rules are executed concurrently.
8. $\mathrm{Pr}_{31,CE_{i,j}} : (2*(g_{i,j} - \theta))| \to 1|\mathrm{g}_{i,j}+1|\mathrm{E}_{i,j}; 1 \le i, j \le n$
$\mathrm{Pr}_{31,CE_{i,j}}$ are the rules which compute $d_{i,j}$ in Formula (6). After executing $\mathrm{Pr}_{31,CE_{i,j}}$, the value of $d_{i,j}$ are obtained, which is equal to variables $g_{i,j}$ and $E_{i,j}$ in rule $\mathrm{Pr}_{31,CE_{i,j}}$.
9. $\mathrm{Pr}_{32,CE_{i,j}} : (ed_1 + 2 * ed_2)|_{\mathrm{E}_{i,j}} \to 1|\mathrm{edg}_{i,j} + 1|E_{D_{i,j}}$; $\mathrm{Pr}_{33,CE_{i,j}} : (0 * ed_1 + 0 * ed_3)|_{\mathrm{E}_{i,j}} \to 1|\mathrm{edg}_{i,j} + 1|E_{D_{i,j}}$; $1 \le i, j \le n$.
$\mathrm{Pr}_{32,CE_{i,j}}$ and $\mathrm{Pr}_{33,CE_{i,j}}$ are rules for computing edge value as Formula (7). If $\mathrm{E}_{i,j}$ is greater than or equal to 0, then $\mathrm{Pr}_{32,CE_{i,j}}$ and $\mathrm{Pr}_{33,CE_{i,j}}$ are executed. Because ed_1 is 0, and ed_3 is -256, so $\mathrm{E}_{i,j} \ge \min(ed_1, ed_2)$ and $\mathrm{E}_{i,j} \ge \min(ed_1, ed_3)$. The execution condition of $\mathrm{Pr}_{32,CE_{i,j}}$ and $\mathrm{Pr}_{33,CE_{i,j}}$ is satisfied. If $\mathrm{d}_{i,j} < 0$, only $\mathrm{Pr}_{33,CE_{i,j}}$ will be executed. Because $\mathrm{E}_{i,j} \ge \min(ed_1, ed_3)$ and $\mathrm{E}_{i,j} < \min(ed_1, ed_2)$, only the execution condition of $\mathrm{Pr}_{33,CE_{i,j}}$ can be satisfied. After executing $\mathrm{Pr}_{32,CE_{i,j}}$ and $\mathrm{Pr}_{33,CE_{i,j}}$, variables $edg_{i,j}$ will be set to 1 if $\mathrm{d}_{i,j} \ge 0$ and every variable $E_{D_{i,j}}$ will be assigned.

10. $\mathrm{Pr}_{main} : (0 * E_{D_{1,1}} + 0 * E_{D_{1,2}} + \ldots + 0 * E_{D_{m,n}} + 1)| \rightarrow 1|E_D$
    Pr_{main} is a rule contained in membrane 4, which controls the stop condition of the P system. For pixel (i, j), if all the enzyme variables $E_{D_{i,j}}$ are assigned, the condition for Pr_{main} is meet. Enzyme variable E_D is set to 1 by rule Pr_{main}, and the system stops running.

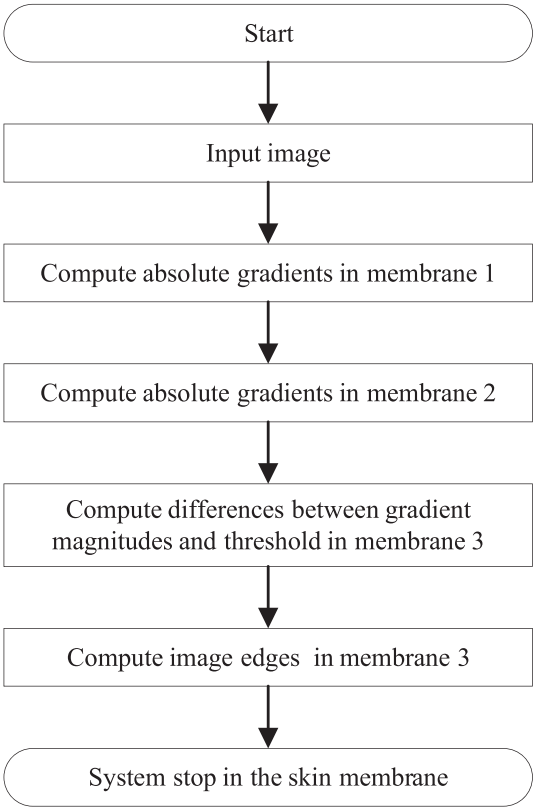


Figure 1. The flowchart of EDENP.

4.2. The Structure and Execution Processes of EDENP

As shown in Figure 2, the structure of EDENP includes four membranes. The system begins to start when the input variables $x_{i,j}$ representing the gray value of source image at location (i, j) appear in the skin membrane. The whole process includes five steps.

Step 1: Horizontal and vertical derivative approximations of every pixel are computed in membrane 1 by using rules of $\mathrm{Pr}_{1,CE_{i,j}}$~$\mathrm{Pr}_{10,CE_{i,j}}$ in a parallel manner. When the directional gradients are computed, membrane 2 will be activated.

Step 2: The gradient magnitude of all the pixels are obtained at the same time with rules of $\mathrm{Pr}_{21,CE_{i,j}}$ in membrane 2.

Step 3: The comparisons between the gradient magnitudes of all pixels and the predefined threshold are executed by rules of $\mathrm{Pr}_{31,CE_{i,j}}$ in membrane 3.

Step 4: The edge pixels are detected and marked with 1, while the background pixels are marked with 0 by rules of $\mathrm{Pr}_{32,CE_{i,j}}$ and $\mathrm{Pr}_{33,CE_{i,j}}$ in membrane 3.

Step 5: The system stop condition is satisfied and the system stops working by rules of Pr_{main} in membrane 4.

So as described above, only five steps are needed in the proposed algorithm for images with arbitrary resolution. Since we do not change the mathematical model of Sobel based edge detection, the detection result by our proposed method is the same as if run on a serial computing platform.

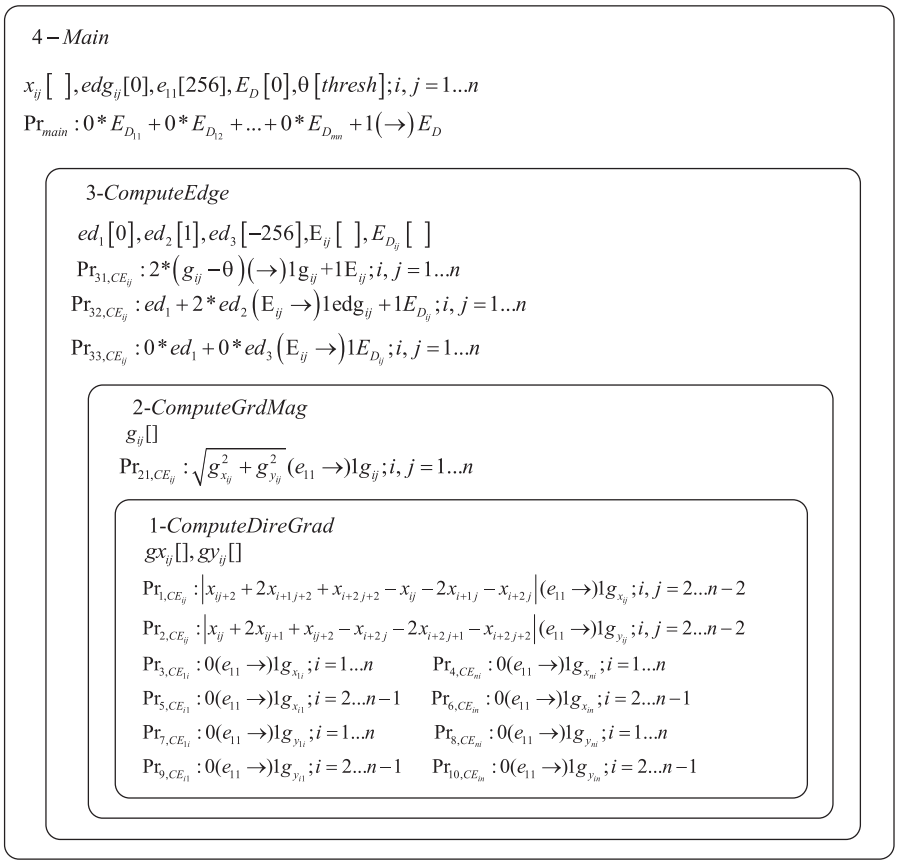


Figure 2. The structure of EDENP.

4.3. Complexity and Resources Analysis

Taking into account that the size of the input data is $n \times n$, and the image is a gray image. The amount of resources needed is illustrated in Table 1. From Table 1, we can see that there are $(7n^2 + 6)$ variables, including $(2n^2 + 2)$ enzymatic variables and $(5n^2 + 4)$ numerical variables. $(6n^2 + 4)$ rules are involved in this system. The total storage space is 1 cell with $(13n^2 + 10)$ molecules. So the space complexity is $O(n^2)$ theoretically. The time complexity is $O(1)$ because the number of execution steps is 5, which implies the computational efficiency is constant for images under arbitrary resolutions.

From the above analysis, we can see that the core of the proposed algorithm is to use space to replace time to obtain high-performance parallel computing, which is exactly the prominent characteristic of MC. Since molecules are used as storage units in a real biological computer, huge storage space can be utilised when this algorithm is implemented on it. So we think the proposed parallel algorithm is effective for images with high resolutions, at least at a theoretical level.

Table 1. Complexity and resources needed for EDENP.

Term	Necessary Resources
Initial number of cells	1
Number of enzymatic variables	$2n^2 + 2$
Number of numerical variables	$5n^2 + 4$
Number of rules	$6n^2 + 4$
Execution steps	5

5. Experiments and Results

In this section, both the performance and efficiency of our proposed EDENP algorithm are evaluated. Since there is no hardware implementation of MC systems at present, the only way to

test the behaviors of the designed P systems is to simulate them in conventional computers. In this paper, a parallel computing architecture, Compute Unified Device Architecture (CUDA), is used as the simulating platform, as it has been reported in literature [24,61]. The parameters of the platform on which our experiments are carried out are illustrated in Table 2. The threshold θ for all the experiments is set to 0.2.

Table 2. Parameters the computer used.

Term	Parameters
CPU model	Intel(R) Core(TM) i7-7700HQ
cache memory	8 MB, 16-Way, 64 byte lines
main memory	16 GB (2* DDR4 2400MHz)
hard disc	SSD, SK hynix SC308 SATA 128GB, 600 Mbps; MQ01ABD100, 1TB
GPU model	Nvidia GeForce GTX 1050 Ti (4 GB)
execution steps	5

5.1. Performance Evaluation

Two case studies are considered to evaluate the performance of the proposed method for different types of images. Since the proposed algorithm is in the framework of MC, edge detection methods based on tissue-like *P* systems [21,24] are chosen as comparison methods. Algorithms in the literature [21,24] are sketched and implemented on a CPU platform using the MATLAB program.

5.1.1. Qualitative Evaluation

Case study 1 is considered to evaluate the performance of the three algorithms for images with rich textures. Four images named *rice*, *cameraman*, *mri*, and *AT3_lm4_01* randomly collected from the MATLAB Image Tool Box are used as testing samples in this experiment, as shown in Figure 3a,e,i,m. Figure 3b–d,f–h,j–l,n–p show the detailed qualitative edge detection results of the three algorithms for the four images. It can be clearly observed from Figure 3b,f,j,n, that the contours of the objects can be detected, but meanwhile the noise in the background is also detected, which will make the following image processing, such as object recognition, more difficult to deal with. The results by reference [21] are shown in Figure 3c,g,k,o. It can be seen that there are too many small edges, and the main outlines of the targets can hardly be found even by human eyes. The results of EDENP are illustrated in Figure 3d,h,l,p, from which we can see that not only the main contours of objects can be detected successfully, but also the noise is well suppressed.

Case study 2 is used to test the performance of the three methods for images with less texture, in which images named *toyobjects*, *circbw*, *text*, *testpart1* randomly selected from MATLAB Image Tool Box are used as testing image samples. In image *toyobjects*, each object has a constant gray value, while the other three images are binary images. Like in Case 1, the detected edge results by the three approaches are shown in Figure 4. Figure 4b,f,j,n clearly show that there are many discontinuous edges when using algorithm in reference [24], while the other two methods can detect the edges completely. When comparing the thickness of the edges, it is obvious to see that the method in reference [21] can achieve the thinnest edges, then the EDENP method, and the edges detected by [24] is the thickest. Although the method in [21] can obtain the finest edges, those edges often have burrs, as shown in Figure 5. Figure 5a,e are the whole edge image of *toyobjects* and *circbw*. Figure 5b–d,f–h are the local enlargement of areas in pink rectangles in Figure 5a,e. Areas marked in green in Figure 5b,f are some examples of discontinuous edges by [24]. When comparing Figure 5c,g with Figure 5d,h, it is clear that edges by EDENP are much smoother than by algorithm [21].

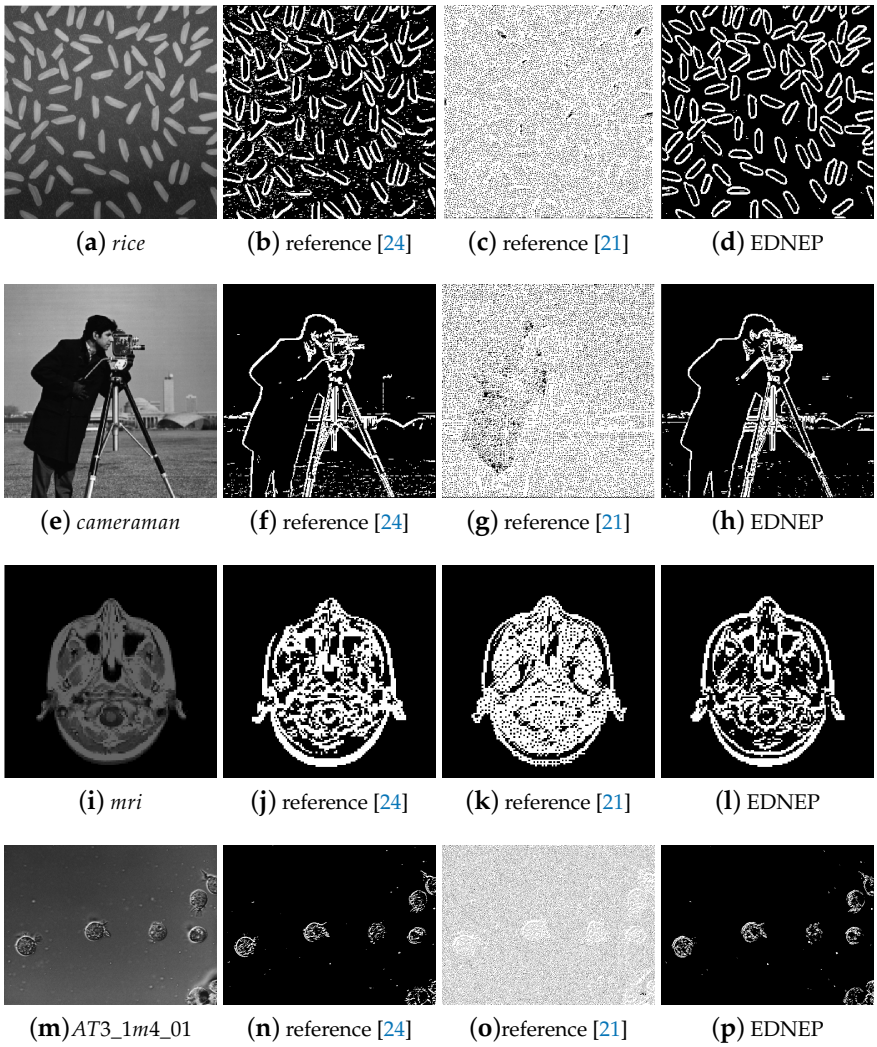


Figure 3. Edge detection results of images with rich texture (the first column: the source gray images; the second to the last column: results by using methods in [21,24] and EDENP respectively).

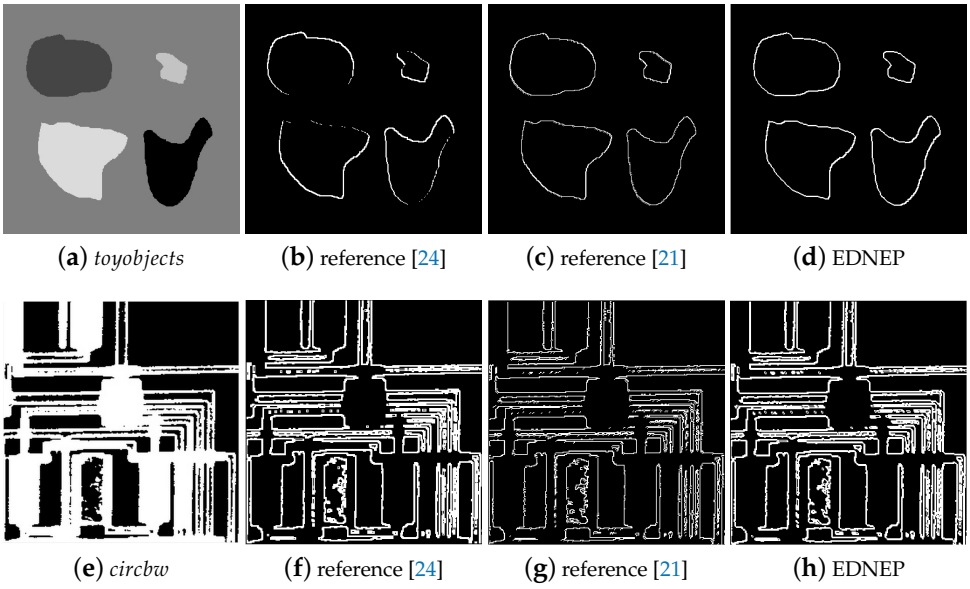
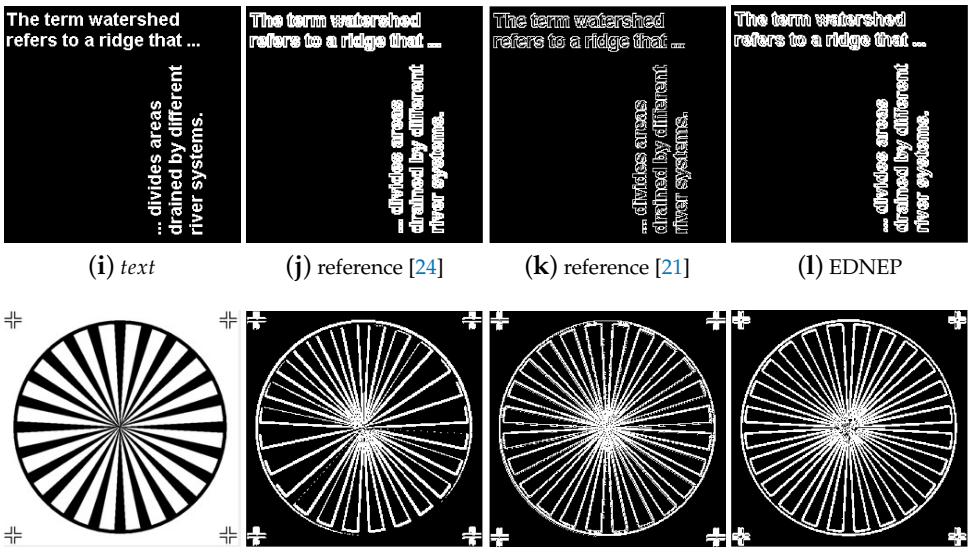


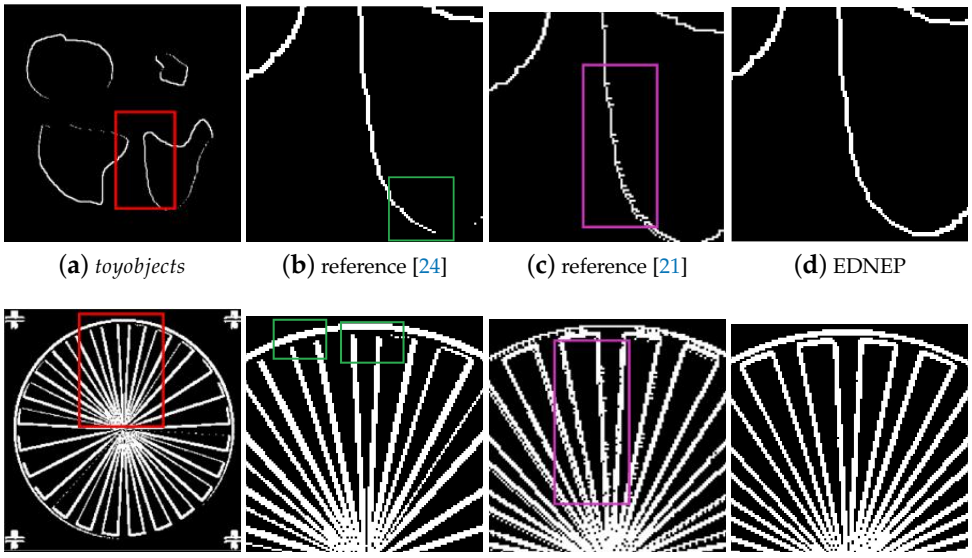
Figure 4. *Cont.*



(**i**) *text* (**j**) reference [24] (**k**) reference [21] (**l**) EDNEP

(**m**) *testpart*1 (**n**) reference [24] (**o**) reference [21] (**p**) EDNEP

Figure 4. Edge detection results of images with less texture (the first column: the source gray images; the second to the last column: results by using methods in [21,24] and EDENP, respectively).



(**a**) *toyobjects* (**b**) reference [24] (**c**) reference [21] (**d**) EDNEP

(**e**) *testpart*1 (**f**) reference [24] (**g**) reference [21] (**h**) EDNEP

Figure 5. Edge detection results of *toyobjects* and *testpart*1 (the first column: the edge image; the second to the last columns: the local edge image enlarged by using methods in [21,24] and EDENP respectively).

5.1.2. Quantitative Evaluation

The confidence degree of the edge image is one of the most used indexes for evaluating the authenticity of the edge pixels. In general, the greater the edge confidence degree is, the more reliable the edges are. In this paper, we use this index to evaluate the performance of the edge detection algorithm quantitatively, whose mathematical definition is presented in reference [62].

Table 3 provides the comparison results of the three methods in terms of edge confidence degree. It can be seen from Table 3 that the EDENP method has the highest edge confidence degree for images with both high and low texture, which means edges detected by EDENP have less false edges.

Through the above quantitative and qualitative results, it can be deduced that the method in reference [21] is nearly invalid for grayscale images with rich texture. For images with less textures,

this method can get the fine edges of the objects. However, the edges are not smooth in some cases because of the false burr edge points. The approach in [24] cannot get the whole contours of the objects due to the discontinous edges detected for images with both rich and less rich textures. The EDENP algorithm has the highest performance and can obtain clear, continuous, and authentic edges of images with both rich and less rich textures.

Table 3. The edge confidence degree.

	Reference [24]	Reference [21]	EDENP
rice	0.75	0.56	**0.84**
cameraman	0.66	0.32	**0.74**
mri	0.63	0.56	**0.68**
AT3_lm4_01	0.44	0.12	**0.5**
toyobjects	0.85	0.76	**0.86**
circbw	0.94	0.93	**0.95**
text	0.93	0.90	**0.94**
testpart1	0.81	0.79	**0.86**

In this paper, only edge detection methods in the framework of MC are chosen for a comparison. From the above experimental results, we can see that the proposed algorithm has better performance compared with the existing tissue-like based edge detection methods. The fundamental reason for this is that with the help of "enzyme variables" in ENPS, the rules can be controlled flexibly, thus the existing Sobel edge detection algorithm can be programmed in "membrane computing language" easily.

5.2. Efficiency Evaluation

To better describe the computation efficiency of EDENP, a speedup ratio is defined as the elapsed time of algorithm on CPU platform divided by running time on GPU platform. The running times of images with different resolutions under GPU and CPU platform and corresponding speedup ratios for one image (*camera*) are illustrated in Table 4. From Table 4, we can see, although the computation times of EDENP are independent of resolutions theoretically, it takes different times to execute the EDENP algorithm for the same image at different resolutions. The reason for this is that the programs do not run on real bio-computers. Table 5 gives the speedup ratios results of the other seven images. It can be found that the lowest speedup is 53, and the maximum speedup can reach up to 262. It is obvious that the computing power of the proposed algorithm is much superior compared with the traditional algorithm implemented on CPU platform.

Table 4. Elapsed time of images with different resolution (*cameraman*).

Image Resolution	256^2	384^2	512^2	768^2	1024^2	2048^2	Platform
Elapsed time(ms)	0.014	0.03	0.05	0.12	0.23	0.86	GPU
Elapsed time(ms)	3.5	9.1	4.4	9.6	41.9	72.8	CPU
Speedup ratio	250	303	88	80	182	130	

Table 5. The speedup ratio of seven images.

Image Resolution	256^2	384^2	512^2	768^2	1024^2	2048^2
rice	79	121	79	101	136	82
mri	60	80	77	62	73	66
AT3_lm4_01	80	90	102	172	71	75
toyobjects	187	162	163	81	182	62
circbw	193	213	262	210	176	66
text	167	180	194	118	57	65
testpart1	53	76	100	161	87	64

6. Conclusions

Membrane computing is a new branch of natural computing, and its amazing storage space and high parallel computing characteristics are very suitable for big data processing. Among various membrane systems, the ENPS can directly deal with numeric variables, and the enzyme variables can flexibly control the execution orders of different rules. In this paper, we attempt to apply ENPS to image processing, and take Sobel edge detection as an example. Compared with the previous works which are based on tissue-like P systems, the advantage of the proposed method is that it does not need to encode and decode the image data, and it is easy to write the program for algorithms with complex execution orders in "membrane computing language". The limitation of the proposed algorithm mainly has two aspects. One is that the execution of the algorithm is based on real biological computers. However, there are no universal biological computers at present, so it is difficult to evaluate the real computing efficiency of the proposed algorithm. The other shortage is that the space complexity is $O(n^2)$, which means large storage space is needed for the proposed algorithm. In future research, we will simulate the algorithm on FPGA hardware and try to combine the ENPS with other, more complex image processing algorithms.

Supplementary Materials: The following are available online.

Author Contributions: The research structure was conceived and designed by J.Y. and G.Z.; D.G., M.Z. and Q.Y. wrote the program and performed the experiments; J.Y. and P.P. wrote the paper and analyzed the data; G.Z. made revisions to the final manuscript. The final manuscript was read and corrected by all authors.

Funding: This work was partially supported by the National Natural Science Foundation of China, under Grant #162300410079, #61672437, #61702428; the Sichuan Science and Technology Program China, under Grant #2018GZ0385, #2017GZ0431, #2018GZ0245, #2018GZ0185, #2018GZ0086, #2018GZ0095, #2019YFG0188; Sichuan education department Program China, under Grant #17ZB0095, #17ZB0090; the Talent Import Fund of CUIT China, under Grant #KYTZ201633; the Chengdu Science and Technology Program China, under Grant #2017-GH02-00049-HZ, #2018-YF05-00981-GX and the New Generation Artificial Intelligence Science and Technology Major Project of Sichuan Province China, under Grant #2018GZDZX0043.

Acknowledgments: The authors would like to thank the anonymous reviewers for their valuable suggestions on improving this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zheng, S.; Yuille, A.; Tu, Z. Detecting object boundaries using low-, mid-, and high-level information. *Comput. Vis. Image Underst.* **2010**, *114*, 1055–1067. [CrossRef]
2. Zhao, L.; Zhao, Q.; Liu, H.; Lv, P.; Gu, D. Structural sparse representation-based semi-supervised learning and edge detection proposal for visual tracking. *Vis. Comput.* **2017**, *33*, 1169–1184. [CrossRef]
3. Hua, S.; Wen, H. Moment-preserving edge detection and its application to image data compression. *Opt. Eng.* **2013**, *32*, 1596–1608. [CrossRef]
4. Satpathy, A.; Jiang, X.; Eng, H.L. LBP-based edge-texture features for object recognition. *IEEE Trans. Image Process.* **2014**, *23*, 1953–1964. [CrossRef]
5. Saif, J.; Hammad, M.; Alqubati, I. Gradient based image edge detection. *Int. J. Eng. Technol.* **2016**, *8*, 153–156. [CrossRef]
6. Jung, J.; Lee, H.; Lee, J.; Park, D. A novel template matching scheme for fast full-Search boosted by an integral image. *IEEE Signal Proc. Lett.* **2010**, *17*, 107–110. [CrossRef]
7. Schweitzer, H.; Deng, R.; Anderson, R.F. A dual-bound algorithm for very fast and exact template matching. *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *33*, 459–470. [CrossRef] [PubMed]
8. Ghassabeh, Y.A.; Rudzicz, F.; Moghaddam, H.A. Fast incremental LDA feature extraction. *Pattern Recognit.* **2015**, *48*, 1999–2012. [CrossRef]
9. Zhou, G.; Cichocki, A.; Zhang, Y.; Mandic, D.P. Group component analysis for multiblock data: common and individual feature extraction. *IEEE Trans. Neural Netw.* **2016**, *27*, 2426–2439. [CrossRef]
10. Herout, A.; Josth, R.; Juranek, R.; Havel, J.; Hradis, M.; Zemcik, P. Real-time object detection on CUDA. *J. Real-Time Image Process.* **2011**, *6*, 159–170. [CrossRef]

11. Jiang B. Real-time multi-resolution edge detection with pattern analysis on graphics processing unit. *J. Real-Time Image Process.* **2018**, *14*, 293–321. [CrossRef]
12. Zuo H.; Zhang Q.; Yong, X.; Zhao, R. Fast sobel edge detection algorithm based on GPU. *Opto-Electron. Eng.* **2009**, *36*, 8–12. [CrossRef]
13. Jiang, J.; Liu, C.; Ling, S.R. An FPGA implementation for real-time edge detection. *J. Real-Time Image Process.* **2018**, *15*, 787–797. [CrossRef]
14. Nausheen, N.; Seal, A.; Khanna, P.; Halder, S. A FPGA based implementation of Sobel edge detection. *Microprocess Microsy* **2018**, *56*, 84–91. [CrossRef]
15. Paolo,B.; Sipp, D. Regulation: Sell help not hope. *Nature* **2014**, *510*, 336–337. [CrossRef]
16. Jiang, N.; Dang, Y.; Wang, J. Quantum image matching. *Quantum Inf. Process.* **2016**, *15*, 3543–3572. [CrossRef]
17. Tsaftaris, S.A.; Katsaggelos, A.K.; Pappas, T.N.; Papoutsakis E.T. How can DNA computing be applied to digital signal processing?. *IEEE Signal Process. Mag.* **2004**, *21*, 57–61. [CrossRef]
18. Díaz-Pernil, D.; Gutierrez-Naranjo, M; Peng, H. Membrane computing and image processing: A short survey. *J. Membr. Comput.* **2019**. [CrossRef]
19. Păun, Gh. Computing with membranes. *J. Comput. Syst. Sci.* **2000**, *61*, 108–143. [CrossRef]
20. Alsalibi, B.; Venkat, I.; Subramanian, K.; Lebai, L.; De, W. The impact of bio-inspired approaches toward the advancement of face recognition. *ACM Comput. Surv.* **2015**, *48*, 1–33. [CrossRef]
21. Christinalh, A.; Díaz-Pernil, D.; Real, P. Region-based segmentation of 2D and 3D images with tissue-like P systems. *Pattern Recognit. Lett.* **2011** *32*, 2206–2212. [CrossRef]
22. Díaz-Pernil, D.; Gutiérrez-Naranjo, M.; Molina-Abril, H.; Real, P. Designing a new software tool for digital imagery based on P systems. *Nat. Comput.* **2012**, *11*, 381–386. [CrossRef]
23. Carnero, J.; Díaz-Pernil, D.; Molina-Abril, H.; Real, P. Image segmentation inspired by cellular models using hardware programming. In Prceedings of the 3rd International Workshop onComputational Topology in Image Context, Chipiona, Spain, 10–12 November 2010; Volume 1, pp. 143–150.
24. Díaz-Pernil, D.; Berciano, A.; Peña-Cantillana, F.; Gutiérrez-Naranjo, M. Segmenting images with gradient-based edge detection using membrane computing. *Pattern Recognit. Lett.* **2013**, *34*. 846–855. [CrossRef]
25. Peña-Cantillana, F.; Díaz-Pernil, D.; Christinal, H.; Gutiérrez-Naranjo, M. Implementation on CUDA of the smoothing problem with tissue-like P systems. *Int. J. Nat. Comput. Res.* **2011**, *2*, 25–34. [CrossRef]
26. Alsalibi, B.; Venkat, I.; Subramanian, K.; Christinal, H. A bio-inspired software for homology groups of 2D digital images. In Proceedings of the Asian Conference on Membrane Computing (ACMC), Coimbatore, India, 18–19 September 2014; IEEE Computer Society: Washington, DC, USA, 2014. [CrossRef]
27. Díaz-Pernil, D.; Christinal, H.; Gutiérrez-Naranjo, M.; Real, P. Using membrane computing for effective homology. *Appl. Algebr. Eng. Commun.* **2012** *23*, 233–249. [CrossRef]
28. Ardelean, I.; Díaz-Pernil, D.; Gutiérrez-Naranjo, M.; Peňa-Cantillana, F.; Reina-Molina, R.; Sarchizian, I. Counting cells with tissue-like P systems. In Proceedings of the Tenth Brainstorming Week on Membrane Computing, Seville, Spain, 30 January–3 February 2012; Fénix Editora: Seville, Spain, 2012.
29. Reina-Molina, R.; Díaz-Pernil, D.; Gutiérrez-Naranjo, M. Cell complexes and membrane computing for thinning 2D and 3D images. In Proceedings of the Tenth Brainstorming Week on Membrane Computing, Seville, Spain, 30 January–3 February 2012; Fénix Editora: Seville, Spain, 2012.
30. Berciano, A.; Díaz-Pernil, D.; Christinal, H.; Venkat, I. First steps for a corner detection using membrane computing. In Proceedings of the Asian Conference on Membrane Computing, Coimbatore, India, 18–19 September 2014; IEEE Computer Society: Washington, DC, USA, 2014.
31. Enguix, G. Preliminaries about some possible applications of P systems in linguistics. *Lect. Notes Comput. Sci.* **2002**, *2597*, 74–89. [CrossRef]
32. Cabarle, F.; de la Cruz, R.; Zhang, X.; Jiang, M.; Liu, X.; Zeng, X. On string languages generated by spiking neural P systems with structural plasticity. *IEEE Trans. Nanobiosci.* **2018**, *17*, 560–566. [CrossRef]
33. Song, T.; Zeng, X.; Zheng, P.; Jiang, M.; Rodríguez-Patón, A. A parallel workflow pattern modelling using spiking neural P systems with colored spikes. *IEEE Trans. Nanobiosci.* **2018**, *17*, 474–484. [CrossRef] [PubMed]
34. Mayne, R.; Phillips, N.; Adamatzky, A. Towards experimental P-systems using multivesicular liposomes. *J. Membr. Comput.* **2019**. [CrossRef]
35. Mitrana, V. Polarization: a new communication protocol in networks of bio-inspired processors. *J. Membr. Comput.* **2019**, published online. [CrossRef]

36. Pan, L.; Păun, Gh.; Zhang, G.; Neri, F. Spiking Neural P Systems with Communication on Request. *Int. J. Neural Syst.* **2017**, *27*, 1750042. [CrossRef]
37. Orellana-Martín, D.; Valencia-Cabrera, L.; Riscos-Núñez, A.; Pérez-Jiménez, M.J. P systems with proteins: A new frontier when membrane division disappears. *J. Membr. Comput.* **2019**. [CrossRef]
38. Sánchez-Karhunen, E.; Valencia-Cabrera, L. Modelling complex market interactions using PDP systems. *J. Membr. Comput.* **2019**. [CrossRef]
39. Zhang, G.; Rong, H.; Neri, F.; Pérez-Jiménez, M.J. An optimization spiking neural P system for approximately solving combinatorial optimization problems. *Int. J. Neural Syst.* **2014**, *24*, 1440006. [CrossRef] [PubMed]
40. Zhang, G.; Cheng, J.; Gheorghe, M.; Meng, Q. A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems. *Appl. Soft Comput.* **2013**, *13*, 1528–1542. [CrossRef]
41. Wang, T.; Zhang, G.; Zhao, J.; He, Z.; Wang, J.; Pérez-Jiménez, M.J. Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems. *IEEE Trans. Power Syst.* **2015**, *30*, 1182–1194. [CrossRef]
42. Martín-Vide, C.; Păun, Gh.; Pazos, J.; Rodríguez-Patón, A. Tissue P systems. *Theor. Comput. Sci.* **2003**, *296*, 295–326. [CrossRef]
43. Ionescu, M.; Păun, Gh.; Yokomori, T. Spiking neural P systems. *Fund. Inform.* **2006**, *71*, 279–308. [CrossRef]
44. Zeng, X.; Pan, L.Q; Pérez-Jiménez, M.J. Small universal simple spiking neural P systems with weights. *Sci. China Inf. Sci.* **2014**, *57*, 1–11. [CrossRef]
45. Păun, G.; Păun, R. Membrane computing and economics: numerical P systems. *Fund. Inform.* **2006**, *73*, 213–227.
46. Zhang, Z.; Wu, T.; Paun, A.; Pan, L. Numerical P systems with migrating variables. *Theor. Comput. Sci.* **2016**, *641*, 85–108. [CrossRef]
47. Pan, L.; Zhang, Z.; Wu, T.; Xu, J. Numerical P systems with production thresholds. *Theor. Comput. Sci.* **2017**, *673*, 30–41. [CrossRef]
48. Zhang, Z.; Pan, L. Numerical P systems with thresholds. *Int. J. Comput. Commun.* **2017**, *11*, 292–304. [CrossRef]
49. Buiu, C.; Vasile, C.; Arsene, O. Development of membrane controllers for mobile robots. *Inform. Sci.* **2012**, *187*, 33–51. [CrossRef]
50. Wang, X.; Zhang, G.; Neri, F.; Jiang T.; Zhao, J.; Gheorghe, M.; Ipate, F.; Lefticaru, R. Design and implementation of membrane controllers for trajectory tracking of nonholonomic wheeled mobile robots. *Integr. Comput.-Aid Eng.* **2016**, *23*, 15–30. [CrossRef]
51. Zhang, G.; Gheorghe, M.; Pérez-Jiminez, M.J. *Real-Life Applications with Membrane Computing*; Springer International Publishing AG: Cham, Switzerland, 2017; pp. 130–141.
52. Mahalingam, K.; Rama, R.; Sureshkumar, W. Robot motion planning inside a grid using membrane computing. *Int. J. Imaging Robot.* **2017**, *17*, 33–51.
53. Pavel, A.; Arsene, O.; Buiu, C. Enzymatic numerical P systems—A new class of membrane computing systems. In Proceedings of the Fifth International Conference on Bio-Inspired Computing: Theories and Applications, Changsha, China, 23–26 September 2010; IEEE Computer Society: Washington, DC, USA, 2010. [CrossRef]
54. Zhang, Z.; Wu, T.; Paun, A.; Pan, L. Universal enzymatic numerical P systems with small number of enzymatic variables, *Sci. China Inf. Sci.* **2018**, *61*, 38–49. [CrossRef]
55. Vasile, C.; Pavel, A.; Dumitrache, I.; Păun, Gh. On the power of enzymatic numerical P system. *ACTA Inform.* **2012**, *49*, 395–412. [CrossRef]
56. Vasile, C.; Pavel, A.; Dumitrache, I. Universality of enzymatic numerical P systems. *Int. J. Comput. Math.* **2013**, *90*, 869–879. [CrossRef]
57. Leporati, A.; Porreca, A.; Zandron, C.; Mauri, G. Improved universality results for parallel enzymatic numerical P systems. *Int. J. Unconv. Comput.* **2013**, *9*, 385–404.
58. Pavel, A.; Buiu, C. Using enzymatic numerical P systems for modeling mobile robot controllers. *Nat. Comput.* **2012**, *11*, 387–393. [CrossRef]
59. Li, W.; Yang, J.; Zhang, J. Handling big data field with enzymatic numerical P System. *J. Sichuan Univ. Nat. Sci. Ed.* **2013**, *45*, 96–104.

60. Pang S.C.; Ding, T.; Rodriguez-Paton, A.; Song, T.; Zheng, P. A parallel bioinspired framework for numerical calculations using enzymatic P system with an enzymatic environment. *IEEE Access* **2018**, *6*, 65548–65556. [CrossRef]
61. Cecilia, J.; García, J.; Guerrero, G.; Martínet-del-Amor, M.; Pérez-Hurtado, I.; Pérez-Jiménez, M. Simulation of P systems with active membranes on CUDA. *Brief Bioinform.* **2009**, *11*, 313–322. [CrossRef] [PubMed]
62. Wang, J.; Bi, J.; Wang, L.; Wang, X. A non-reference evaluation method for edge detection of wear particles in ferrograph images. *Mech. Syst. Signal Process.* **2018**, *100*, 863–876. [CrossRef]