Int. J. Advance Soft Compu. Appl, Vol. 10, No. 1, March 2018
ISSN 2074-2827;

Automatic 2D Image Segmentation Using Tissue-Like P System

Rafaa I. Yahya[1, 2]**, Siti Mariyam Shamsuddin**[2]**, Salah I. Yahya**[3, 4]**,**
Shafaatunnur Hasan[2]**,and Bisan Alsalibi**[2, 5]

[1]Department of Computer, Collage of Science, University of Al-Mustansiriyah,
Baghdad, Iraq
e-mail: rafaa_ismaeel79@uomustansiriyah.edu.iq

[2]UTM Big Data Center, Ibnu Sina Institute for Scientific and Industrial Research,
Universiti Teknologi Malaysia, UTM Skudai, 81310 Johor, Malaysia
e-mail: mariyam@utm.my; shafaatunnur@utm.my

[3]Department of Software Engineering, Faculty of Engineering, Koya University,
Kurdistan Region, Iraq
e-mail: salah.ismaeel@koyauniversity.org

[4]Department of Computer Science and Engineering, School of Science and
Engineering, University of Kurdistan Hewler, Kurdistan Region, Iraq

[5]School of Computer Sciences, Universiti Sains Malaysia, USM, Malaysia
e-mail: besansalipi@gmail.com

Abstract

This paper uses P-Lingua, a standard programming language that is designed specifically for P systems, to automatically simulate the region-based segmentation of 2D images. P-Lingua, which is based on membrane computing, links to Java Netbeans using the PLinguaCore4 Java library to automatically codify the pixels of the input image as long as automatically draw the output segmented image. Many methods have been suggested previously and used for artificial image segmentation, but to the best of our knowledge, none of those techniques were automatic, where the image was codified manually and the visualization of the output image was done manually in the tissue simulator which takes time and effort, especially when dealing with large images in the system. Two types of pixel adjacency have been utilized in this paper, namely; 4-adjacency and 8-adjacency. The jacquard index method has been used to measure the accuracy of the segmentation. The results of the proposed method demonstrated that beside its ability to automatically segmenting 2D images with arbitrary sizes, it is more efficient and faster than the tissue simulator tool, since the latter needs the input image to be codified manually pixel by pixel which makes it impractical for real-world applications.

Keywords: *Membrane computing, P-Lingua, Region-based image Segmentation, Tissue-like P systems.*

1 Introduction

Membrane computing (MC) has emerged as a recent branch of natural computing, which is mainly based on the assumption that the flow of metabolites within the compartmental architecture and functioning of biological cells can be interpreted as a flow of information for computations [1]. The computational devices in MC are known as P systems in honour of their initiator G. Paun [2]. More importantly, P systems are massively parallel and distributed computing models processing multisets of objects based on predefined evolution rules in a non-deterministic manner. Owing to the inherent parallelism, MC has the ability of solving non-deterministic polynomial time (NP)-complete problems in polynomial or often in linear time and this feature has been shown extensively in the literature. The basic ingredients of a typical P system model consist of; membrane structure, a set of evolution rules and multisets of objects.

The main novelty of MC relies on the fact that membrane systems look at the whole cell as a computing device. “A membrane system consists of a hierarchical arrangement of membranes delimiting regions, which represent various compartments of a cell, and with each region containing bio-chemical elements of various types and having associated rules, which represent bio-chemical processes taking place inside the cell,” [3].

Basically, MC devices or P systems constitute three different models of computing devises depending on the particular features of the cell which are:

a) Cell-like P systems (inspired from the structure of the cell).
b) Tissue-like P systems (inspired from organization of cells in tissue)
c) Spiking neural-like membrane systems (inspired from the way neurons are linked in neural nets) [4].

In this paper, tissue-like P system is used for segmentation of the 2D images.The fundamental objective of digital image processing is to extract meaningful information from images without human assistance. Segmentation is an important task of image processing for satellite and medical images [5]. Technically speaking, in computer vision [6], segmentation is the process of partitioning a digital image into multiple segments (sets of pixels), aiming to simplify or modify the representation of an image to be more expressive and easier to analyse and understand. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image, such that pixels with the same label share certain visual characteristics.

Recently, MC techniques have been used in solving problems arising from digital images. Image segmentation has been addressed extensively in literature. For example, Christinal, Díaz-Pernil and Jurado [7] presented a tissue-like P systems to improve the standard edge-based segmentation method using MC rules. Christinal, et al. [8] presented a MC rules framework to solve the threshold problem using cell-like P system wherein the solution has been reached in linear time depending on the number of pixels of the input image. Carnero, et al. [9] proposed a new tool (hardware system) to obtain a segmentation of digital images, the system uses MC method along with a hardware programming (VHDL) language to develop an ad–hoc processor to solve the problem where the implementation of edge based detection for noise removal is used. Similarly, Reina-Molina, et al. [10] presented a new segmentation approach based on a tissue-like P system with the use of MC rules with multiple auxiliary cells for solving segmentation problem. Díaz-Pernil, et al. [11], proposed a new software tool for performing a segmentation of 2D digital images based on MC platform. However, they did not explain in detail the technical aspects of implementing the tool and they did not measure the accuracy of the segmentation, as well. Christinal, Díaz-Pernil and Real [12] developed a tissue-like P system. using MC rules, to design a region based segmentation algorithm in a constant number of steps. In their approach, 4-adjacency relation between pixel's neighborhoods has been adopted for 2D digital images where 6-adjacency relation between voxel neighborhoods has been used for 3D digital images. Peña-Cantillana, et al. [13] proposed a bio-inspired algorithm to solve the threshold problem by using a membrane computing and implemented a novel device architecture called Compute Unified Device Architecture (CUDA^{TM}). Carnero, Dıaz-Pernil and Gutiérrez-Naranjo [14], proposed a hardware programming tool called field programmable gate arrays (FPGAs) to implement tissue-like P system for solving segmentation problems by using rules of MC. Sheeba, et al. [15] proposed a tissue-like P system (MC rules) to enhance segmentation methods in medical image but they did not measure the accuracy of the segmentation. Christinal, et al. [16] proposed a bio-inspired MC via tissue-like P system, MC rules, to perform a parallel color segmentation of images using a threshold method. Yang, et al. [17] proposed a novel membrane algorithm, tissue-like P system, to develop a region-based segmentation. Along this line, Peng, et al., [18] proposed a novel threshold segmentation approach using cell-like P system (MC algorithm) to improve threshold segmentation. Díaz-Pernil, et al. [19] presented CUDA^{TM} to implement tissue-like P system (MC rules) for segmentation of images with gradient-based edge detection. Furthermore, Peng, et. al [20] proposed a novel segmentation by adaptive traditional region based colour segmentation method using tissue-like P system. In [21], Isawasan, et al. developed a tissue-like P system using MC rules to segment hexagonal images wherein the segmentation has been done in 7 steps. Peng, et al. [22] proposed a novel method using cell-like P system with membrane algorithm to solve the optimal multi-level thresholding problem. Yahya, et al. [23] presented a region based segmentation with tissue like P system rules that implement a simple artificial image with a more detailed illustration of how P system works furthermore, where different colour relations have been explored to show the effect of colour on the segmentation results.

Interestingly, a new research line has been recently launched in which MC has been adopted to solve several problems related to digital imagery. For example, region based segmentation of 2D and 3D images has been investigated in [12]. However, the main drawback of that approach is that the image has been manually codified in the tissue simulator. Thus, in this paper, we present a dynamic software using P-Lingua for region-based segmentation of 2D digital images based on the segmentation rules presented in [12]. The same face image (14×12 pixel image) that used in [12] has been adopted in this paper, but resized to 14×14 to validate our presented system and its ability to deal with large size. Basically, region-based segmentation is performed based on MC and implemented in P-Lingua with artificial images (apple.png, house.png and face.png) which are loaded automatically in the system. P-Lingua is an official programming language for MC that offer a general syntactic framework that could define a unified standard for MC, covering a broad variety of models [24].

Most recently, Yahya et al. [25] proposed a tissue-like P system with region-based and edge-based segmentations to segment two dimensional hexagonal images, where-in P-Lingua programming language has been used to implement and validate the pro-posed system. The achieved experimental results clearly demonstrated the effective-ness of using hexagonal connectivity to segment two dimensional images in a less number of rules and computational steps. The experimental results have shown that using the hexagonal connectivity is more efficient than the four connectivities, where the number of rules and computational steps have been reduced from 9 to 7 steps.

A comprehensive and an up to date survey pertaining to image processing techniques using MC has been presented in [26].

This paper is organized as follows: Section 2 provides a basic definition of tissue-like P system and describes the use of tissue-like P system for automatic region-based segmentation of 2D images. Section 3 presents the methodology of segmentation with membrane computing using P-Lingua. Section 4 evaluates the proposed approach. Section 5 concludes the paper and suggests some directions for the future work.

2 Tissue-like P system for Image Segmentation

2.1 Tissue-Like P System

Tissue-like P system was firstly introduced by Martí [27]. The structure of tissue-like P system is organized as a graph representing all the communication channels existing between interacting cells. The set of interconnections is dynamic and, in the course of the evolution of the system, it can possibly change whosesoever a new communication way is established or another one is closed [28]. From the computational perspective, the essential characteristic of tissue-like P system is that membranes do not have electrical charges as in the cell-like P systems. Typically, the form of tissue-like P systems model with input of degree $q \geq 1$ is a tuple

$\Pi = (\Gamma, \Sigma, \mathcal{E}, w_1, \ldots, w_q, R, i_\Pi, o_\Pi)$

Where

a) Γ is a finite alphabet, whose symbols are called objects,

b) $\Sigma(\subset \Gamma)$ is the input alphabet,

c) $\mathcal{E} \subseteq \Gamma$ is the list of objects in the environment, each one of arbitrarily infinite copies,

d) $w_1, \ldots, w_q$ are strings over Γ representing the multi-sets of objects associated with the cells at the initial configuration,

e) R is a finite set of rules of the form;
   1. Communication rule: $(i, u/v, j)$, for $i, j \in \{0,1,2,\ldots,q\}, i \neq j, u, v \in \Gamma$;
   2. Division rules: $[a]_i \rightarrow [b]_i [c]_i$, where $i \in \{1,2,\ldots,q\}$ and $a, b, c \in \Gamma$;

f) $i_\Pi \in \{1,2,\ldots,q\}$ Refers to the input cell,

g) $o_\Pi \in \{0,1,2,\ldots,q\}$ Refers to the output cell.

In the typical framework of MC, each cell is viewed as a computing unit working in a maximally parallel and non-deterministic way. The configuration is an instantaneous description of the P system at a particular moment, where a sequence of computation steps can be applied in a parallel manner to obtain a new configuration. A computation is said to be successful if it halts, reaching a specific configuration where no more rules can be applied to the current objects. With a halting computation, the associated output can be codified by the content of the output membrane.

2.2 Tissue- like P system for Automatic Region Based Segmentation of 2D images

Region-based techniques rely on common patterns in intensity values within a cluster of neighbouring pixels. The cluster is referred to as the region, and the goal of segmentation algorithm is to group regions according to their anatomical or functional roles [29].The main aim of region-based segmentation is to use image characteristics to map individual pixels in an input image to sets of pixels called regions that might correspond to an object [30].

Segmentation in digital imagery has several interesting features which makes it suitable for mechanisms inspired by nature. One of them is that it can be solved parallely and locally. Regardless of how large is the picture, the segmentation process can be performed in parallel in different local areas of the picture. Another interesting feature is that the basic required information can be easily encoded by different bio-inspired representations [14].

The previous work of Yahya, et al. [23], which was based on the rules adapted by Christinal, Díaz-Pernil and Real Hepzipah [12], implemented a simple apple image to provide a more detailed illustration of how P systems works by using tissue simulator, and to prove that in MC the algorithm requires only 9 steps to obtain region-based segmentation regardless of the size of the image. Furthermore, different colour relations have been explored to show the effect of colour on the segmentation results.

In this paper, an automatic region-based segmentation will be applied to artificial images (.png) rather than codifying the image manually using tissue simulator. In order to automatically load the images in the system, the segmentation rules will be written using P-Lingua programming language. Furthermore, to deal with 2D images automatically, P-Lingua will be linked to Java Netbeans using the PLinguaCore4 Java library. PLinguaCore4 is a tool integrated in java library that includes a parser to handle P-Lingua input files and check possible programming errors (both lexical/syntax and semantics) [31]. This library is used to implement MC with region-based segmentation of 2D images. The artificial images that have been used are apple.png with size of 13×13 and three colours, face.png with seven colours and size of 14×14 and house.png with three colours and size of 30×30.

3 Methodology of Segmentation with Membrane Computing Using P-Lingua

As mentioned previously, PLinguaCore4 is a JAVA library which was developed as a software framework for tissue-like P system [32], cell-like P system [33] and neural-like p system [34]. In this paper, the tissue-like P system rules will be implemented using P-Lingua to segment the artificial images. Fig. 1 shows the steps of performing the segmentation using P-Lingua, where the algorithm of region–based segmentation has been achieved.

3.1 Adding P-Lingua library (PLinguaCore4) to the Java platform

In the first step of this work, PLinguaCore4 library has been added to Java. This library implements all functions supported by P-Lingua (model definition, simulator and formats). In addition to that, it reports the rules and membrane structure that is read from a file where a P system is defined, detects errors in the file and reports them and, if the P system is defined in the P-Lingua language, it locates the error in the file. There are two types of format in P-Lingua where the first is the input format, whose file can be read via PLinguaCore4, and the second is the output format, whose files can be generated by PLinguaCore4.

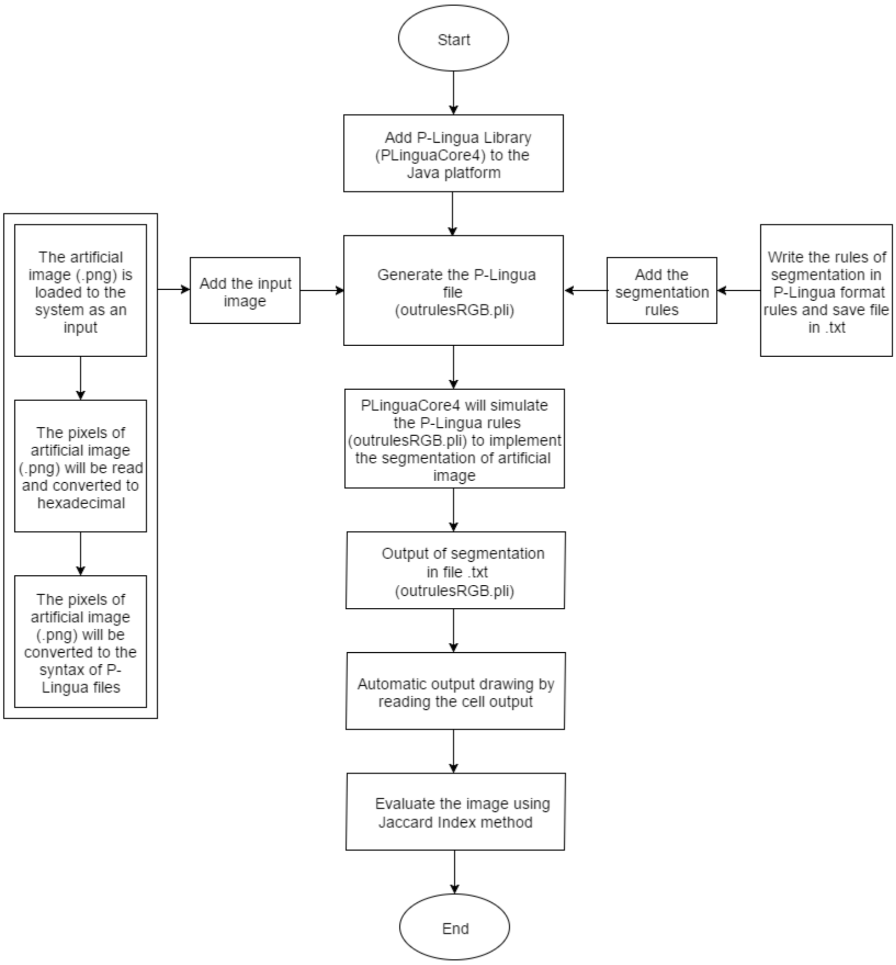

Start
Add P-Lingua Library (PLinguaCore4) to the Java platform
The artificial image (.png) is loaded to the system as an input
The pixels of artificial image (.png) will be read and converted to hexadecimal
The pixels of artificial image (.png) will be converted to the syntax of P-Lingua files
Add the input image
Generate the P-Lingua file (outrulesRGB.pli)
Add the segmentation rules
Write the rules of segmentation in P-Lingua format rules and save file in .txt
PLinguaCore4 will simulate the P-Lingua rules (outrulesRGB.pli) to implement the segmentation of artificial image
Output of segmentation in file .txt (outrulesRGB.pli)
Automatic output drawing by reading the cell output
Evaluate the image using Jaccard Index method
End

Fig. 1: Flowchart of segmentation of artificial image (.png) in P-Lingua.

3.2 Generating P-Lingua File (outrulesRGB.pli)

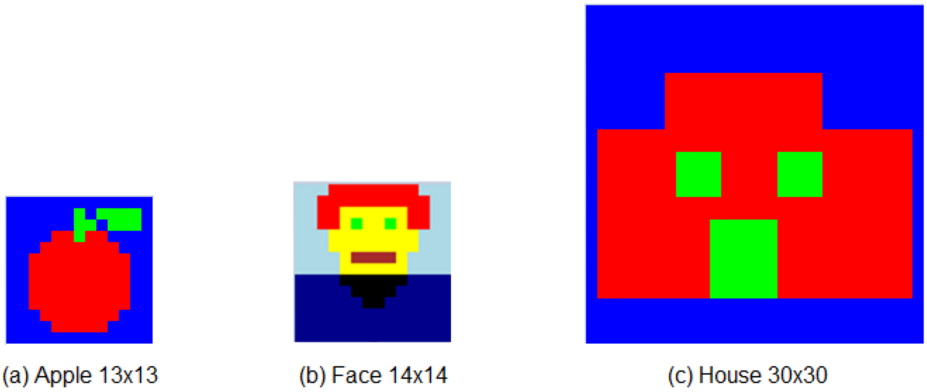
When the PLinguaCore4 library is linked to Java Netbeans, the next step is to generate the P-Lingua file (outrulesRGB.pli) by concatenating the input artificial image with the segmentation rules as discussed in the following two steps.

3.2.1 Input image type (png)

a) Loading the image: The image is loaded into the system as an input image. In tissue-like P system, the image is considered as a cell and the pixels of the image are represented as objects in the cell. The system will read the image and convert it to the standard P-Lingua syntax. Specifically, every pixel (object) has the colour and coordinates (*x*, *y*). The artificial images that used are drawn with a paint program in the format (.png). In this paper, the 2D images as shown in Fig. 2 are; an artificial image named apple.png with size of 13×13 and 3 colours, face.png image with size of 14×14 and 7 colours, and house.png image with size of 30×30 and 3 colours.

b) The pixels of artificial image will be read and converted to hexadecimal: The pixels should have colour and coordinates as mentioned earlier. In getting the colour the pixel will be read and converted from the RGB format to hexadecimal format (hexadecimal values give information about colours including colour models). The pixels of artificial image will be read and converted using Java Netbeans. Technically speaking, this step is used to link each pixel of the image to its corresponding P-Lingua syntax (e.g., ffff0000 is converted to R{*x*, *y*}, where *x* and *y* are the coordinates of this pixel in the image) as shown in Fig. 3.

c) The pixels of the artificial image (. png) will be converted to the syntax of P-Lingua: All the pixels of the image (cell) after converting have the standard format which can be recognized by the P-Lingua compiler. The syntax of the pixels in the P system has the name of the colour and the coordinates so the name of the colour will match with the value of the pixel and coordinates followed by {} as shown in Fig. 4. All the pixels are converted to the syntax of P-Lingua.


(a) Apple 13x13

(b) Face 14x14

(c) House 30x30

Fig. 2: Input artificial images (.png).

3.2.2 Writing the rules of segmentation

In this paper, the segmentation rules adopts the rules of MC for 2D image segmentation method used in [12]. The rules are written in the P-Lingua format and saved in a text file. After loading the image, the image will be automatically codified as an input and concatenated to the rules file to the final form.pli file named outrulesRGB.pli.

Red (255, 0, 0)	🟥	→ ffff0000	Dark Blue (0, 0,139)	🟦	→ ff00008b
Green (0, 255, 0)	🟩	→ ff00ff00	Light Blue (173, 216, 230)	🟦	→ ffadd8e6
Yellow (255, 255, 0)	🟨	→ ffffff00	Black (0, 0, 0)	⬛	→ ff000000
Brown (165, 42, 42)	🟫	→ ffa52a2a			

Fig. 3: Illustration of converting the pixels of the artificial image to hexadecimal.

Red	🟥	→ ffff0000 → R{i,j}	Dark Blue	🟦	→ ff00008b →db{i,j}
Green	🟩	→ ff00ff00 → G{i,j}	Light Blue	🟦	→ ffadd8e6 → lb{i,j}
Yellow	🟨	→ ffffff00 → Y{i,j}	Black	⬛	→ ff000000 → Bl{i,j}
Brown	🟫	→ ffa52a2a → Br{i,j}			

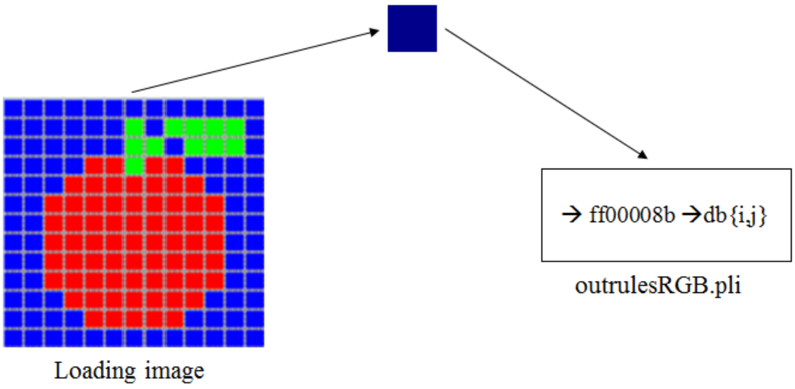


Fig. 4: Steps of converting pixel of the artificial image (.png) to object (pixels in syntax of P-Lingua).

Technically, a 2D digital image can be represented by a matrix where each pixel in the image is an element of the matrix. Basically, there are two types of neighbourhoods, surrounding a pixel which is considered in this paper, namely 4-adjacency and 8-adjacency.In 4-adjacency, a 4 neighbourhood

$$\{(x-1,y),(x,y+1),(x+1,y),(x,y-1)\}$$

contains only the pixels above, below, to the left and to the right of the central pixel (x,y) as shown in Fig. 5. On the contrary, in 8-adjacency, an 8-neighbourhood adds to the four neighbourhoods four diagonal neighbours which are;

$\{(x-1,y-1),(x-1,y),(x-1,y+1),(x,y+1),(x+1,y+1),(x+1,y),(x+1,y-1),\ (x,\ y-1)\}$

as illustrated in Fig. 4.

If the image contains two colours which are green and blue, then if the 4-adjacency is considered the segmentation rules will be as the following:

1. $[G\{i,j\},B\{i,j+1\}]'1 <--> [Gx\{i,j\},B\{i,j+1\}]'0;$
2. $[G\{i,j\},B\{i,j-1\}]'1 <--> [Gx\{i,j\},B\{i,j-1\}]'0;$
3. $[G\{i,j\},B\{i+1,j\}]'1 <--> [Gx\{i,j\},B\{i+1,j\}]'0;$
4. $[G\{i,j\},B\{i-1,j\}]'1 <--> [Gx\{i,j\},B\{i-1,j\}]'0;$

Similarly, if the 8-adjacency is considered, then four more rules will be added to cover the diagonal pixels as described in the following:

1. $[G\{i,j\},B\{i,j+1\}]'1 <--> [Gx\{i,j\},B\{i,j+1\}]'0;$
2. $[G\{i,j\},B\{i,j-1\}]'1 <--> [Gx\{i,j\},B\{i,j-1\}]'0;$
3. $[G\{i,j\},B\{i+1,j\}]'1 <--> [Gx\{i,j\},B\{i+1,j\}]'0;$
4. $[G\{i,j\},B\{i-1,j\}]'1 <--> [Gx\{i,j\},B\{i-1,j\}]'0;$
5. $[G\{i,j\},B\{i-1,j+1\}]'1 <--> [Gx\{i,j\},B\{i-1,j+1\}]'0;$
6. $[G\{i,j\},B\ \{i-1,j-1\}]'1 <--> [Gx\{i,j\},B\{i-1,j-1\}]'0;$
7. $[G\{i,j\},B\{i+1,j-1\}]'1 <--> [Gx\{i,j\},B\{i+1,j-1\}]'0;$
8. $[G\{i,j\},B\{i+1,j+1\}]'1 <--> [Gx\{i,j\},B\{i+1,j+1\}]'0;$

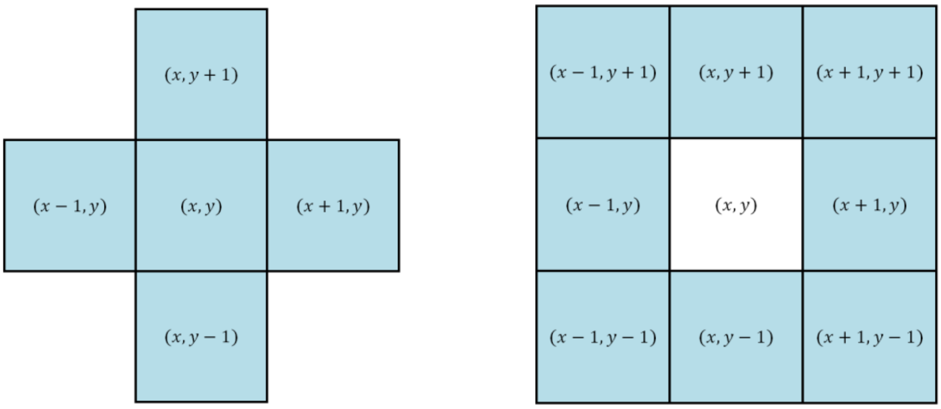


Fig. 5: Illustration of the different neighbourhoods of 2D image; (a) 4-neighborhood (4-adjacency) and (b) 8-neighborhood (8-adjacency).

3.3 The PLinguaCore4 will simulate the P-Lingua rules (outrulesRGB.pli)

The PLinguaCore4 will simulate the P-Lingua rules (outrulesRGB.pli) to implement the segmentation of the artificial image (.png). All the steps above are to prepare the P-Lingua file (image and rules of segmentation) for simulation by the PLinguaCore4 Java library.

3.4 Output of Segmentation in file.txt (outrulesRGB.txt)

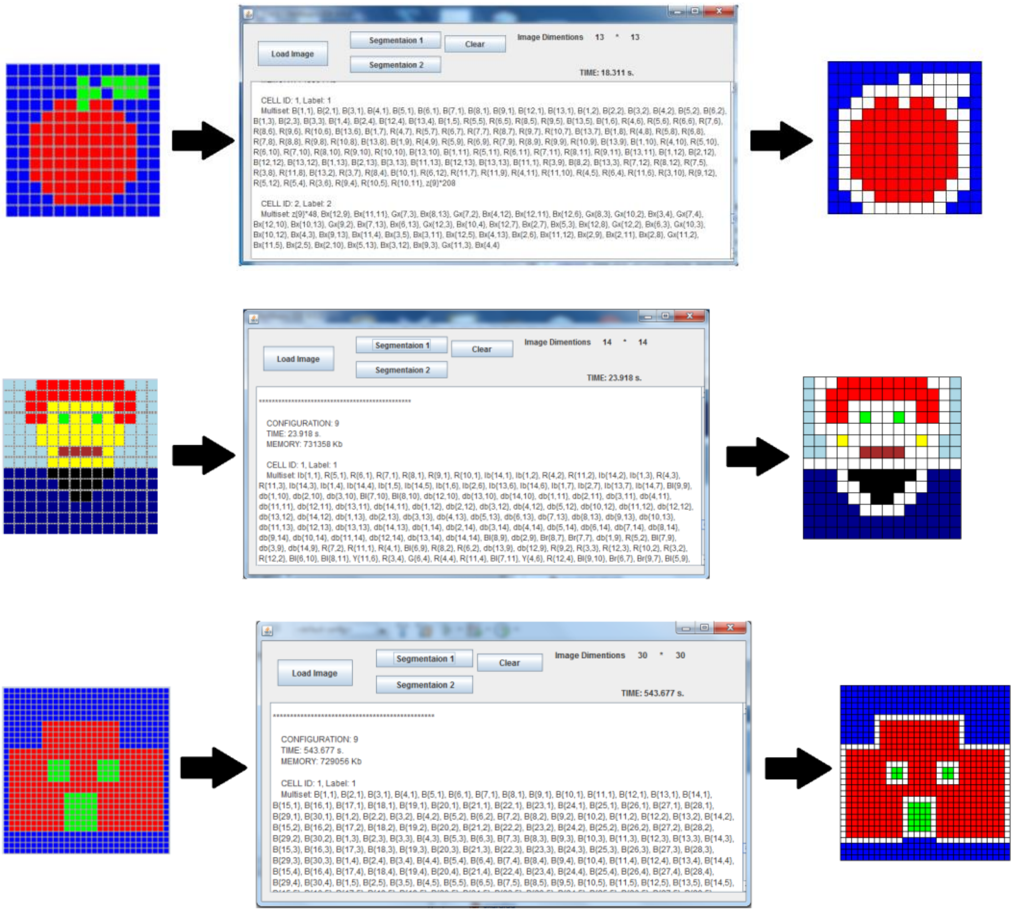
The result of segmentation is in the file that is generated after segmentation, called (outrulesRGB.txt). This file contains the details of every step of segmentation, it contains information such as the configuration, input of the cell, the output of cell, the environment, the time of every step and final time of execution. The final step which shows the segmented image is shown in Fig. 6 pertaining to the 4-adjacency and in Fig. 7 for the 8-adjacency. To make our system more convenient and easy to use, we have replaced the manual drawing of the output to automatic drawing by using C#, a multi-paradigm programming language.

3.5 Reading Output and Visualization

Once the image is segmented using MC, bearing in mind that our model contains two different cells, the output can be read from any of these cells as needed. As we consider region-based segmentation in this paper, the output will be read from cell two only. It is worth mentioning that cell two contains the border and edge pixels. In order to draw the image automatically according to the region-based criteria, the original image will be loaded to the system. The pixels of the original image will be used to draw the segmented image. Then, in order to mark the border pixels, cell two will be read and the pixels in cell two will be located in the segmented image and converted to white colour to indicate the borders. By doing so, the algorithm of the automatic drawing of segmented artificial image is achieved, where a snapshot of the program implementing this is shown in Fig. 8, and the obtained region-based segmented images are shown in Fig. 9.

4 Evaluation of the Proposed Approach

In order to illustrate the effectiveness of the proposed segmentation approach, the segmentation results obtained by 4-adjacency are compared with the results obtained using 8-adjacency. To this end, the algorithm will be evaluated in terms of accuracy and efficiency. Accuracy corresponds to the degree to which the delineation of the object corresponds to the truth, whereas the efficiency corresponds to the amount of time required to perform the segmentation. Two types of evaluation are used. The first type was performed by visualizing the output of the segmented images, including the face used in [12] as shown in Fig. 10.

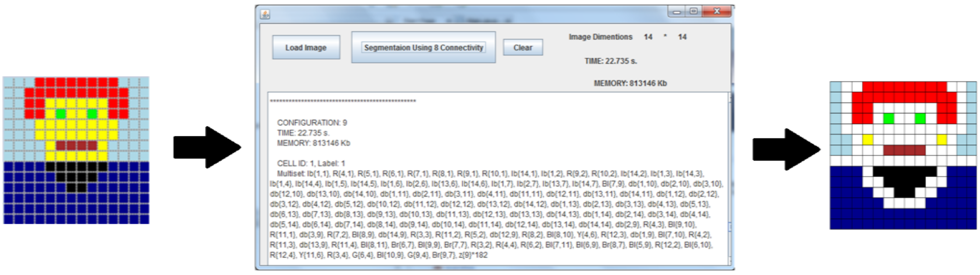


```
  CELL ID: 1, Label: 1
  Multiset: B{1,1}, B{2,1}, B{3,1}, B{4,1}, B{5,1}, B{6,1}, B{7,1}, B{8,1}, B{9,1}, B{12,1}, B{13,1}, B{1,2}, B{2,2}, B{3,2}, B{4,2}, B{5,2}, B{6,2},
B{1,3}, B{2,3}, B{3,3}, B{1,4}, B{2,4}, B{12,4}, B{13,4}, B{1,5}, R{5,5}, R{6,5}, R{8,5}, R{9,5}, B{13,5}, B{1,6}, R{4,6}, R{5,6}, R{6,6}, R{7,6},
R{8,6}, R{9,6}, R{10,6}, B{13,6}, B{1,7}, R{4,7}, R{5,7}, R{6,7}, R{7,7}, R{8,7}, R{9,7}, R{10,7}, B{13,7}, B{1,8}, R{4,8}, R{5,8}, R{6,8},
R{7,8}, R{8,8}, R{9,8}, R{10,8}, B{13,8}, B{1,9}, R{4,9}, R{5,9}, R{6,9}, R{7,9}, R{8,9}, R{9,9}, R{10,9}, B{13,9}, B{1,10}, R{4,10}, R{5,10},
R{6,10}, R{7,10}, R{8,10}, R{9,10}, R{10,10}, B{13,10}, B{1,11}, R{5,11}, R{6,11}, R{7,11}, R{8,11}, R{9,11}, B{13,11}, B{1,12}, B{2,12},
B{12,12}, B{13,12}, B{1,13}, B{2,13}, B{3,13}, B{11,13}, B{12,13}, B{13,13}, B{11,1}, R{3,9}, B{8,2}, B{13,3}, R{7,12}, R{8,12}, R{7,5},
R{3,8}, R{11,8}, B{13,2}, R{3,7}, R{8,4}, B{10,1}, R{6,12}, R{11,7}, R{11,9}, R{4,11}, R{11,10}, R{4,5}, R{6,4}, R{11,6}, R{3,10}, R{9,12},
R{5,12}, R{5,4}, R{3,6}, R{9,4}, R{10,5}, R{10,11}, z{9}*208

  CELL ID: 2, Label: 2
  Multiset: z{9}*48, Bx{12,9}, Bx{11,11}, Gx{7,3}, Bx{8,13}, Gx{7,2}, Bx{4,12}, Bx{12,11}, Bx{12,6}, Gx{8,3}, Gx{10,2}, Bx{3,4}, Gx{7,4},
Bx{12,10}, Bx{10,13}, Gx{9,2}, Bx{7,13}, Bx{6,13}, Gx{12,3}, Bx{10,4}, Bx{12,7}, Bx{2,7}, Bx{5,3}, Bx{12,8}, Gx{12,2}, Bx{6,3}, Gx{10,3},
Bx{10,12}, Bx{4,3}, Bx{9,13}, Bx{11,4}, Bx{3,5}, Bx{3,11}, Bx{12,5}, Bx{4,13}, Bx{2,6}, Bx{11,12}, Bx{2,9}, Bx{2,11}, Bx{2,8}, Gx{11,2},
Bx{11,5}, Bx{2,5}, Bx{2,10}, Bx{5,13}, Bx{3,12}, Bx{9,3}, Gx{11,3}, Bx{4,4}
```

```
*********************************************


  CONFIGURATION: 9
  TIME: 23.918 s.
  MEMORY: 731358 Kb

  CELL ID: 1, Label: 1
  Multiset: lb{1,1}, R{5,1}, R{6,1}, R{7,1}, R{8,1}, R{9,1}, R{10,1}, lb{14,1}, lb{1,2}, R{4,2}, R{11,2}, lb{14,2}, lb{1,3}, R{4,3},
R{11,3}, lb{14,3}, lb{1,4}, lb{14,4}, lb{1,5}, lb{14,5}, lb{1,6}, lb{2,6}, lb{13,6}, lb{14,6}, lb{1,7}, lb{2,7}, lb{13,7}, lb{14,7}, Bl{9,9},
db{1,10}, db{2,10}, db{3,10}, Bl{7,10}, Bl{8,10}, db{12,10}, db{13,10}, db{14,10}, db{1,11}, db{2,11}, db{3,11}, db{4,11},
db{11,11}, db{12,11}, db{13,11}, db{14,11}, db{1,12}, db{2,12}, db{3,12}, db{4,12}, db{5,12}, db{10,12}, db{11,12}, db{12,12},
db{13,12}, db{14,12}, db{1,13}, db{2,13}, db{3,13}, db{4,13}, db{5,13}, db{6,13}, db{7,13}, db{8,13}, db{9,13}, db{10,13},
db{11,13}, db{12,13}, db{13,13}, db{14,13}, db{1,14}, db{2,14}, db{3,14}, db{4,14}, db{5,14}, db{6,14}, db{7,14}, db{8,14},
db{9,14}, db{10,14}, db{11,14}, db{12,14}, db{13,14}, db{14,14}, Bl{8,9}, db{2,9}, Br{8,7}, Br{7,7}, db{1,9}, R{5,2}, Bl{7,9},
db{3,9}, db{14,9}, R{7,2}, R{11,1}, R{4,1}, Bl{6,9}, R{8,2}, R{6,2}, db{13,9}, db{12,9}, R{9,2}, R{3,3}, R{12,3}, R{10,2}, R{3,2},
R{12,2}, Bl{6,10}, Bl{8,11}, Y{11,6}, R{3,4}, G{6,4}, R{4,4}, R{11,4}, Bl{7,11}, Y{4,6}, R{12,4}, Bl{9,10}, Br{6,7}, Br{9,7}, Bl{5,9},
```

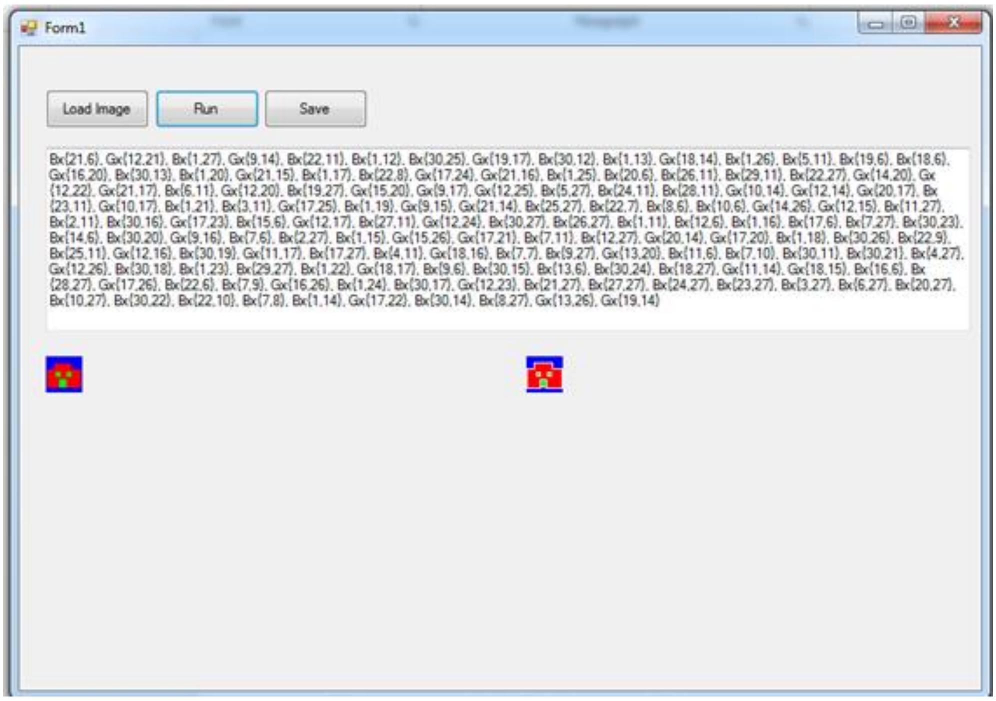
```
*********************************************


  CONFIGURATION: 9
  TIME: 543.677 s.
  MEMORY: 729056 Kb

  CELL ID: 1, Label: 1
  Multiset: B{1,1}, B{2,1}, B{3,1}, B{4,1}, B{5,1}, B{6,1}, B{7,1}, B{8,1}, B{9,1}, B{10,1}, B{11,1}, B{12,1}, B{13,1}, B{14,1},
B{15,1}, B{16,1}, B{17,1}, B{18,1}, B{19,1}, B{20,1}, B{21,1}, B{22,1}, B{23,1}, B{24,1}, B{25,1}, B{26,1}, B{27,1}, B{28,1},
B{29,1}, B{30,1}, B{1,2}, B{2,2}, B{3,2}, B{4,2}, B{5,2}, B{6,2}, B{7,2}, B{8,2}, B{9,2}, B{10,2}, B{11,2}, B{12,2}, B{13,2}, B{14,2},
B{15,2}, B{16,2}, B{17,2}, B{18,2}, B{19,2}, B{20,2}, B{21,2}, B{22,2}, B{23,2}, B{24,2}, B{25,2}, B{26,2}, B{27,2}, B{28,2},
B{29,2}, B{30,2}, B{1,3}, B{2,3}, B{3,3}, B{4,3}, B{5,3}, B{6,3}, B{7,3}, B{8,3}, B{9,3}, B{10,3}, B{11,3}, B{12,3}, B{13,3}, B{14,3},
B{15,3}, B{16,3}, B{17,3}, B{18,3}, B{19,3}, B{20,3}, B{21,3}, B{22,3}, B{23,3}, B{24,3}, B{25,3}, B{26,3}, B{27,3}, B{28,3},
B{29,3}, B{30,3}, B{1,4}, B{2,4}, B{3,4}, B{4,4}, B{5,4}, B{6,4}, B{7,4}, B{8,4}, B{9,4}, B{10,4}, B{11,4}, B{12,4}, B{13,4}, B{14,4},
B{15,4}, B{16,4}, B{17,4}, B{18,4}, B{19,4}, B{20,4}, B{21,4}, B{22,4}, B{23,4}, B{24,4}, B{25,4}, B{26,4}, B{27,4}, B{28,4},
B{29,4}, B{30,4}, B{1,5}, B{2,5}, B{3,5}, B{4,5}, B{5,5}, B{6,5}, B{7,5}, B{8,5}, B{9,5}, B{10,5}, B{11,5}, B{12,5}, B{13,5}, B{14,5},
```

Fig. 6: Final result of segmented images using 4-adjacency; (a) Apple.png 13×13, (b) Face.png 14×14 and (c) House.png 30×30.



```
*********************************************


  CONFIGURATION: 9
  TIME: 22.735 s.
  MEMORY: 813146 Kb

  CELL ID: 1, Label: 1
  Multiset: lb{1,1}, R{4,1}, R{5,1}, R{6,1}, R{7,1}, R{8,1}, R{9,1}, R{10,1}, lb{14,1}, lb{1,2}, R{9,2}, R{10,2}, lb{14,2}, lb{1,3}, lb{14,3},
lb{1,4}, lb{14,4}, lb{1,5}, lb{14,5}, lb{1,6}, lb{2,6}, lb{13,6}, lb{14,6}, lb{1,7}, lb{2,7}, lb{13,7}, lb{14,7}, Bl{7,9}, db{1,10}, db{2,10}, db{3,10},
db{12,10}, db{13,10}, db{14,10}, db{1,11}, db{2,11}, db{3,11}, db{4,11}, db{11,11}, db{12,11}, db{13,11}, db{14,11}, db{1,12}, db{2,12},
db{3,12}, db{4,12}, db{5,12}, db{10,12}, db{11,12}, db{12,12}, db{13,12}, db{14,12}, db{1,13}, db{2,13}, db{3,13}, db{4,13}, db{5,13},
db{6,13}, db{7,13}, db{8,13}, db{9,13}, db{10,13}, db{11,13}, db{12,13}, db{13,13}, db{14,13}, db{1,14}, db{2,14}, db{3,14}, db{4,14},
db{5,14}, db{6,14}, db{7,14}, db{8,14}, db{9,14}, db{10,14}, db{11,14}, db{12,14}, db{13,14}, db{14,14}, db{2,9}, R{4,3}, Bl{9,10},
R{11,1}, db{3,9}, R{7,2}, Bl{8,9}, db{14,9}, R{3,3}, R{11,2}, R{5,2}, db{12,9}, R{8,2}, Bl{8,10}, Y{4,6}, R{12,3}, db{1,9}, Bl{7,10}, R{4,2},
R{11,3}, db{13,9}, R{11,4}, Bl{8,11}, Br{6,7}, Bl{9,9}, Br{7,7}, R{3,2}, R{4,4}, R{6,2}, Bl{7,11}, Bl{6,9}, Br{8,7}, Bl{5,9}, R{12,2}, Bl{6,10},
R{12,4}, Y{11,6}, R{3,4}, G{6,4}, Bl{10,9}, G{9,4}, Br{9,7}, z{9}*182
```

Fig. 7: Final result of segmented image (Face.png; 14×14) using 8-adjacency.



```
Bx{21,6}, Gx{12,21}, Bx{1,27}, Gx{9,14}, Bx{22,11}, Bx{1,12}, Bx{30,25}, Gx{19,17}, Bx{30,12}, Bx{1,13}, Gx{18,14}, Bx{1,26}, Bx{5,11}, Bx{19,6}, Bx{18,6},
Gx{16,20}, Bx{30,13}, Bx{1,20}, Gx{21,15}, Bx{1,17}, Bx{22,8}, Gx{17,24}, Gx{21,16}, Bx{1,25}, Bx{20,6}, Bx{26,11}, Bx{29,11}, Bx{22,27}, Gx{14,20}, Gx
{12,22}, Gx{21,17}, Bx{6,11}, Gx{12,20}, Bx{19,27}, Gx{15,20}, Gx{9,17}, Gx{12,25}, Bx{5,27}, Bx{24,11}, Bx{28,11}, Gx{10,14}, Gx{12,14}, Gx{20,17}, Bx
{23,11}, Gx{10,17}, Bx{1,21}, Bx{3,11}, Gx{17,25}, Bx{1,19}, Gx{9,15}, Gx{21,14}, Bx{25,27}, Bx{22,7}, Bx{8,6}, Bx{10,6}, Gx{14,26}, Gx{12,15}, Bx{11,27},
Bx{2,11}, Bx{30,16}, Gx{17,23}, Bx{15,6}, Gx{12,17}, Bx{27,11}, Gx{12,24}, Bx{30,27}, Bx{26,27}, Bx{1,11}, Bx{12,6}, Bx{1,16}, Bx{17,6}, Bx{7,27}, Bx{30,23},
Bx{14,6}, Bx{30,20}, Gx{9,16}, Bx{7,6}, Bx{2,27}, Bx{1,15}, Gx{15,26}, Gx{17,21}, Bx{7,11}, Bx{12,27}, Gx{20,14}, Gx{17,20}, Bx{1,18}, Bx{30,26}, Bx{22,9},
Bx{25,11}, Gx{12,16}, Bx{30,19}, Gx{11,17}, Bx{17,27}, Bx{4,11}, Gx{18,16}, Bx{7,7}, Bx{9,27}, Gx{13,20}, Bx{11,6}, Bx{7,10}, Bx{30,11}, Bx{30,21}, Bx{4,27},
Gx{12,26}, Bx{30,18}, Bx{1,23}, Bx{29,27}, Bx{1,22}, Gx{18,17}, Bx{9,6}, Bx{30,15}, Bx{13,6}, Bx{30,24}, Bx{18,27}, Gx{11,14}, Gx{18,15}, Bx{16,6}, Bx
{28,27}, Gx{17,26}, Bx{22,6}, Bx{7,9}, Gx{16,26}, Bx{1,24}, Bx{30,17}, Gx{12,23}, Bx{21,27}, Bx{27,27}, Bx{24,27}, Bx{23,27}, Bx{3,27}, Bx{6,27}, Bx{20,27},
Bx{10,27}, Bx{30,22}, Bx{22,10}, Bx{7,8}, Bx{1,14}, Gx{17,22}, Bx{30,14}, Bx{8,27}, Gx{13,26}, Gx{19,14}
```

Fig. 8: Snapshot of the program, based on algorithm that implemented to segmenting image automatically.

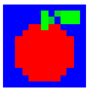
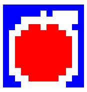
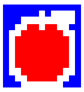
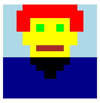
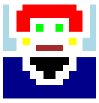
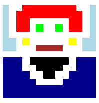
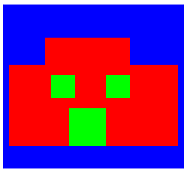
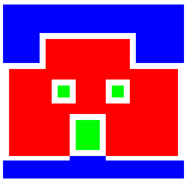
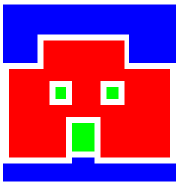
Orginal image	Segmented image with 4-adjacency	Segmented image with 8-adjacency
		
		
		

Fig. 9: Three images; apple.png, face.png and house.png, before and after region-based segmentation.

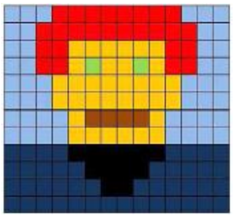
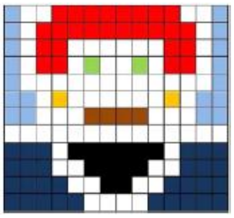
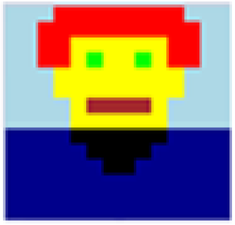
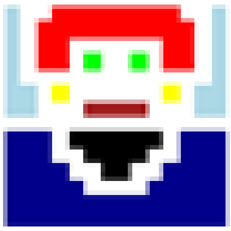
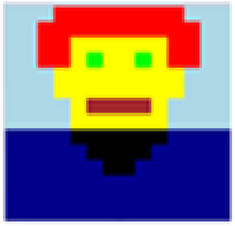
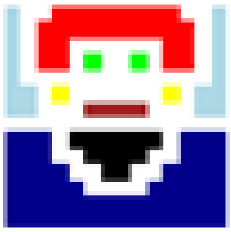
Work of Christinal (Christinal, Díaz-Pernil *et al.* 2011) [12]	 Original image (14×12)	 Segmented image
Proposed method (4-adjacency)	 Original image (14×14)	 Segmented image
Proposed method (8-adjacency)	 Original image (14×14)	 Segmented image

Fig. 10: Visualization evaluation between Christinal's study [12] and this study.

The second type of evaluation is by Jaccard Index method [35]. Jaccard Index method is a typical region-based evaluation metrics which corresponds to the ratio of the number of matching pixels to the total number of both matching pixels and mismatching pixels. Jaccard method aims at measuring the similarity between the ground truth of the original image and the segmentation result. The segmentation result and its corresponding ground truth are typically represented as S and G respectively. Jaccard Index is defined as follows:

$$\text{Jaccard Index} = \frac{A\,(G \cap S)}{A\,(G \cup S)} \tag{1}$$

Where A (.) is the operation of counting amount. In (1), the numerator corresponds to the number of matching pixels (true positives), whereas the denominator counts the total number of matching and mismatching pixels.

Basically, it is an absolute “0” or “1” assignment and counting problem in the Jaccard Index. Note that the much closer the Jaccard accuracy value to one, the better the segmentation. It can be noted from Table 1 that the 4-adjacency is slightly better than the 8- adjacency.

Table 1: Jaccard Index accuracy comparisons between 4 and 8-adjacencies.

Results	Apple	Face	House
4-adjacency	0.6377	0.8413	0.7627
8-adjacency	0.6377	0.8413	0.7601

The time and memory space of image segmentation using 4-adjacency is shown with details in Table 2. The time and memory space of image segmentation using 8-adjacency is shown with details in Table 3. From the results of Table 2, it can be noted that as the image increase in size and colours, the number of rules will be increased and consequently the time of execution will be increased. Furthermore, it is notable that 8-adjacency is much faster than the 4-adjacency. The hardware platform used in this paper is a PC with processor Intel**Error! Reference source not found.** ®Core^{TM} i5-M430, 2.27 GHZ CPU, 4 GB of memory. The main advantage of P-Lingua language is that it enabled the system to be automatic rather than codifying the artificial images manually. Furthermore, it is more flexible, faster and convenient to use than the tissue simulator.

Table 2: Time and memory space of image segmentation (4-adjacency).

Results	Apple	Face	House
Time (in seconds)	18.311 s	23.918 s	543.677 s
Memory space	688332 kb	731358 Kb	729056 kb

Table 3: Time and memory space of image segmentation (8-adjacency).

Results	Apple	Face	House
Time (in seconds)	17.408 s	22.735 s	438.017 s
Memory space	714111 kb	813146 kb	536321 kb

Finally, we can conclude that analysis of the experimental results demonstrates the effectiveness of the segmentation approach and provides valuable insights into the performance and characteristics of the proposed segmentation algorithm. It is worth mentioning that we cannot estimate the difference between the 4 and 8-adjacency accurately because the artificial images used in this work are relatively small. In the future work, we plan to compare the 4 and 8-adjacency using real medical images to accurately estimate the difference.

5 Conclusion

Segmentation has special characteristics which enable it to be suitable for methods of natural computing. However, for those techniques to be more effective and practical in real applications, a suitable software tool is of great importance. In this paper, we presented a MC software tool for automatically segmenting 2D images using P-Lingua programming language. Two types of pixel adjacency have been used in this paper, namely 4-adjacency and 8-adjacency. In contrast to the tissue simulator, the proposed software is able to deal with images of different sizes and has the potential of becoming a helping tool for dealing with large medical images. According to the literature of related work pertaining to image segmentation using MC, the face image used by Christinal, et al., has been adopted in this paper to validate the presented system. Christinal, et al., codified the input image manually, pixel by pixel, and the visualization of the output image was achieved manually in the tissue simulator, which takes time and effort especially when dealing with large images. The strategy that proposed by Christinal, et al., is impractical when dealing with large real images. Another limitation found in the literature is that only 4-adjacency relationship has been considered. Hence, we can summarize the contributions of the paper as follows: firstly, the input image has been codified automatically by linking P-Lingua with the Java platform. Secondly, two ways of adjacency relationships have been investigated and compared in terms of performance. Finally, the output of the image has been read and the segmented image has been rendered automatically instead of reading the output pixels and drawing the segmented image manually as in the previous works which used the tissue simulator tool.

However, there are still some drawbacks of the proposed approach which need to be addressed thoroughly. The first limitation of the proposed approach is that it can only deal with a limited number of colours in the sense that it cannot be applied directly to real images with multiple colours. Accordingly, to deal with real medical images, the binarization of the image prior to segmentation is of considerable importance. Secondly, the proposed approach has been only simulated using a sequential architecture which in turn does not exploit the massive parallelism inherited in P systems. In our future work, to fully make use of the MC parallelism, a parallel architecture such as CUDA^{TM} will be used to gain higher performance speedups over the serial implantation.

References

[1] Păun, G. (2002). Introduction: Membrane Computing—What It Is and What It Is Not, In *Membrane Computing* (pp. 1-6). Springer, Singapore.

[2] Păun, G. & Rozenberg, G. (2002). A guide to membrane computing. *Theoretical Computer Science, Elsevier*, *287*(1), 73-100.

[3] Bernardini, F. & Gheorghe, M. (2205). Membrane systems for molecular computing and biological modelling, University of Sheffield.

[4] Păun, G. & Pérez-Jiménez, M.J. (2006). Membrane computing: brief introduction, recent results and applications. *Biosystems, Elsevier*, *85*(1), 11-22.

[5] Somasundaram, S. & Alli, P. (2011). A Review on Recent Research and Implementation Methodologies on Medical Image Segmentation. *Journal of Computer Science*. *8*(1), 170-174.

[6] Shapiro, L. & Stockman, G.C. (2001). Computer Vision. *Prentice Hall*.

[7] Christinal, H.A., Díaz-Pernil, D. & and Jurado, P.R. (2009). Segmentation in 2D and 3D image using tissue-like P system, In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (pp. 169-176). Springer International Publishing.

[8] Christinal, H.A., et al. (2010). Thresholding of 2D images with cell-like P systems. *Romanian Journal of Information Science and Technology (ROMJIST), 13*(2), 131-140.

[9] Carnero, J., et al. (2010). Image segmentation inspired by cellular models using hardware programming. In *3rd International Workshop on Computational Topology in Image Context.*

[10] Reina-Molina, Carnero, R.J. & Diaz-Pernil, D. (2010). Image segmentation using tissue-like P systems with multiple auxiliary cells. *Image-A*, *1*(3), 143-150.

[11] Díaz-Pernil, D., et al. (2010). A bio-inspired software for segmenting digital images. I*n Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference on.* IEEE.

[12] Christinal, H.A., Díaz-Pernil, D. & Real, P. (2011). Region-based segmentation of 2D and 3D images with tissue-like P systems. *Pattern Recognition Letters, Elsevier*, *32*(16), 2206-2212.

[13] Peña-Cantillana, F., et al. (2011). A parallel implementation of the thresholding problem by using tissue-like P systems. In *Computer Analysis of Images and Patterns.* Springer, Singapore.

[14] Carnero, J., Dıaz-Pernil, D. & Gutiérrez-Naranjo, M.A. (2011). Designing tissue-like P systems for image segmentation on parallel architectures. In *Ninth Brainstorming Week on Membrane Computing* (pp. 43-62).

[15] Sheeba, F., et al. (2011). Segmentation of peripheral blood smear images using tissue-like P systems. In *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2011 Sixth International Conference on*. IEEE.

[16] Christinal, H.A., et al. (2012). Colour Segmentation of 2D Images with Thresholding. In *Eco-friendly Computing and Communication Systems* (pp. 162-169). Springer, Singapore.

[17] Yang, Y., et al., (2013). A Region-based Image Segmentation Method Under P Systems. *J. Inf. Comput. Sci*, *10*(10), 2943-2950.

[18] Peng, H., et al., (2012). Image thresholding with cell-like P systems. *Membrane Computing. Proceedings. Tenth Brainstorming Week on* (p. 03).

[19] Díaz-Pernil, D., et al., (2013). Segmenting images with gradient-based edge detection using Membrane Computing. *Pattern Recognition Letters, Elsevier*. *34*(8), 846-855.

[20] Peng, H., et al. (2014). A Region-based Colour Image Segmentation Method Based on P Systems. *Romanian Journal of Information Science and technology*. *17*(1), 63-75.

[21] Isawasan, P., et al. (2014). Region-based segmentation of Hexagonal digital images using membrane computing. In *Membrane Computing (ACMC), 2014 Asian Conference on* IEEE.

[22] Peng, H., Wang, J. & Pérez-Jiménez, M.J. (2014). Optimal multi-level thresholding with membrane computing. *Digital Signal Processing, Elsevier, 37*, 2014. 53-64.

[23] Yahya, R.I., et al. (2015). Membrane Computing for 2D Image Segmentation. *Int. J. Advance Soft Compu. Appl*, 7(1).

[24] Díaz-Pernil, D., et al. (2009). A P-lingua programming environment for membrane computing, In *Membrane Computing*, (pp. 187-203). Springer.

[25] Yahya, R.I., Shamsuddin, S.M., Hasan, S. & Yahya, S.I. (2016). Tissue-like P system for Segmen-tation of 2D Hexagonal Images. ARO-The Scientific Journal of Koya University. 4(1), 35-42. doi: 10.14500/aro.10135.

[26] Yahya, R.I., Shamsuddin, S.M., Yahya, S.I., Hasan, S., Al-Salibi, B. & Al-Khafaji, G. (2016). Image Segmentation Using Membrane Computing: A Literature Survey. In: *Gong M., Pan L., Song T., Zhang G. (eds) Bio-inspired Computing – Theories and Applications. BIC-TA 2016. Communications in Computer and Information Science* (pp. 314-335). Springer, Singapore.

[27] Martí, C., Păun, G. Pazos, J. (2003). Tissue P systems. *Theoretical Computer Science. 296*(2), 295-326.

[28] Bianco, L. (2007). Membrane models of biological systems. Citeseer.

[29] Siddiqui, F.K. & Richhariya, V. (2013). An Efficient Image Segmentation Approach through Enhanced Watershed Algorithm. *Computer Engineering and Intelligent Systems*. *4*(6), 1-7.

[30] Zhao, Y., et al. (2008). Improved watershed algorithm for dowels image segmentation. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on* IEEE.

[31] Frisco, P., Gheorghe, M. & Pérez-Jiménez, M.J. (2014). Applications of Membrane Computing in Systems and Synthetic Biology. Springer, Singapore.

[32] Perez-Hurtado, I., et al. (2014). AP–Lingua based Simulator for Tissue P Systems with Cell Separation. *Scinece and Technology*. *17*(1), 89-102.

[33] García-Quismondo, M., et al. (2009). P-Lingua 2.0: A software framework for cell-like P systems. *Int J Comput Commun Control*. *4*(3), 234-43.

[34] Macías–Ramos, L.F., et al. (2012). A P–Lingua Based Simulator for Spiking Neural P Systems, *in Membrane Computing* (pp. 257-281). Springer, Singapore.

[35] Shi, R., Ngan, K.N. & Li, S. (2014). *Jaccard index compensation for object segmentation evaluation.* In *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE.