An Autonomy Oriented Computing Approach to Image-Component Labeling

Branko Samarzija and Slobodan Ribaric
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, 10000 Zagreb, Croatia
E-mail: {branko.samarzija, slobodan.ribaric}@fer.hr

Abstract

This paper presents an Autonomy Oriented Computing (AOC) approach to gray-level image-component labeling. The basic elements of such AOC systems are autonomous entities placed in an environment. The environment, in our case, is viewed as a two-layer 2D lattice containing a gray-level image in the first layer and a notice board at the second layer. The environment serves as the place where autonomous entities reside, roam and operate. The goal of each autonomous entity is to locate and label image pixels belonging to the homogeneous component according to the specified criteria of the region's homogeneity. During the image exploration and evaluation the entities rely on their reactive and rational behaviors, such as diffusion, breeding and communication. By communicating, the entities are able to determine distinct components by assigning them different labels. Experiments based on a simulation of the proposed AOC system were run over a set of images from "blocks world".

1. Introduction

Approaches based on Autonomy Oriented Computing (AOC) differ from traditional Artificial Intelligence (AI) and Multi-Agent System (MAS) studies in that AOC pays special attention to the role of self-organization, a powerful methodology as demonstrated in Nature, as well as AOC's suitability for problems that involve distributed, locally interacting or rational entities [1]. An AOC system $\mathbf{A}$ is formally described with the 3-tuple: $\mathbf{A} = (\{e_1, e_2, \ldots, e_N\}, \mathbf{E}, \mathbf{\Phi})$, where $\{e_1, e_2, \ldots, e_N\}$ is a group of N autonomous entities, $\mathbf{E}$ is an environment in which entities reside and $\mathbf{\Phi}$ is the system objective function, which is usually a non-linear function of the entity states. The system objective function is incorporated into the performance of the entities through their behavioral rules by implicitly regulating entities toward the desired configuration.

AOC-based approaches are intended to solve hard computational problems. In attempting to do so, AOC system modeling reproduces and relies on lifelike behavior in the computation. With detailed knowledge of the underlying mechanisms, simplified lifelike behavior can be used as a model for a general-purpose problem-solving technique. Such a technique, in our case, refers to low-level image processing. A number of autonomous entities are spread over the image as their environment $\mathbf{E}$ in order to, through their courses of life, replicate some lifelike behaviors on the individual level. This approach leads to a state of the system that gives emerging results at the macroscopic level to the particular image-processing task.

AOC systems as applied to image processing is a relatively new area of research that studies the emergent behaviors in a lattice where entities react to the digital-image environment according to a set of behavioral rules [5]. The basic idea for such research originates from the theory of self-reproducing automata [2], cellular automata [3] and it is also inspired by MAS theory [4] and distributed-behavior-based agent concepts. Several algorithms for image feature extraction based on evolutionary agents have been proposed in [5]: image edge detection, moving-character border searching, multiple-feature extraction and image feature tracking. Furthermore, autonomous agent-based image segmentation has been proposed by [6]. In this approach a digital image is viewed as a two-dimensional cellular environment inhabited by agents. Agents organized into four different classes according to their attribute values had the goal to find and label four homogeneous regions on a given image, relying on reactive behaviors of the directional breeding and diffusion. Face detection and localization using a behavior-based agent approach was proposed in [7][8].

Motivated by Nature-inspired problem solving and the formal framework of AOC system modeling, we propose an approach to gray-level image-component labeling. The proposed AOC-based system is described by the group of autonomous entities with reactive and rational behaviors, as well as the environment containing a given image and the system objective function $\mathbf{\Phi}$.

2. The AOC system for image-component labeling

The image contains a number of pixels that according to their characteristics, based on some statistically defined criteria, can be grouped into homogeneous regions and consequently into components. A gray-level image viewed as a two-dimensional lattice represents an entities' search space,

and that it is the main part of the AOC system's environment. Autonomous entities have a goal to visit, evaluate and mark all the pixels that satisfy the criteria of homogeneity. After that, based on communication among the entities, according to 8-neighborhood connectivity, the marked regions are transformed into components with different labels.

2.1. Environment

The environment $\mathbf{E}$ serves as the domain in which the autonomous entities roam [1]. It contains an internal representation of the problem in the form of a virtual landscape encapsulating all the accessible knowledge and information associated with the problem's structure. For the component-labeling problem environment $\mathbf{E}$ is characterized as a two-layer two-dimensional (2D) lattice of size $W \times H$ containing a gray-level image in the first layer $\mathbf{E_1}$. The layer $\mathbf{E_0}$, called a notice board, consists of a 2D lattice of memory cells (Fig. 1). The notice board holds a socially shared

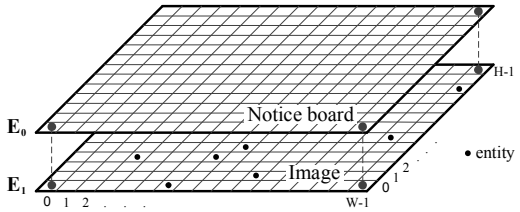


Figure 1. Environment $\mathbf{E}$

memory in the form of a blackboard that is used by all the entities for the purpose of information interchange. In our case the notice board is also used for storing the final result of the component-labeling process.

While roaming from pixel to pixel entities read information from the notice board associated with the pixel at which they are temporarily located and adjust their behavior accordingly (see the algorithm in Fig. 2). Entities also have the ability to leave information at the notice board for the other entities that could possibly visit the same pixel in the future. For the component-labeling problem the entities leave certain patterns called labels that denote different components in the image. After the communication activities of the entities, labels remaining in the notice board represent the final solution of the component-labeling process.

The third main aspect of the environment is that it keeps a central clock that helps to synchronize the behaviors of all the autonomous entities.

2.2. System objective function

The system objective function $\mathbf{\Phi}$ for image-component labeling is defined as a state-oriented function and it maps a multi-set of the internal states of all the entities at the given $cycle$ of the central clock into a set of integer values $\mathbf{Z}$:

$$\mathbf{\Phi}^{(cycle)} : IS^{\infty} \rightarrow \mathbf{Z}, \tag{1}$$

where IS^{∞} denotes a multi-set of $IS = \{active, communicable, sleep, dead\}$ and $\mathbf{Z}$ is a set of integers. Mapping $\mathbf{\Phi}(internal\_state)^{(cycle)}$ returns the number of entities with an internal state equal to $internal\_state$, where $internal\_state \in IS$, at the given central clock $cycle$ ($cycle = 1, 2, 3, \ldots$). The goal values of the system objective function for the image-component labeling are $\mathbf{\Phi}(active)^{(cycle)} = 0$ and $\mathbf{\Phi}(communicable)^{(cycle)} = 0$. These values define the end of any entity activity in the system $\mathbf{A}$.

2.3. Entities

Autonomous entities, based on a simultaneous interaction with their local part of an environment $\mathbf{E}$, have the ability to make changes on it. The local part of the environment $\mathbf{E}$ represents an area in which the entity has the ability to sense and read information contained in the environment necessary to assess its further actions. While doing so, an individual autonomous entity can be viewed as an automaton dynamically governed by its behaviors $\mathbf{B}$, to either reactively or deliberatively operate in its environment [5].

Each entity is formally described by the 3-tuple $(\mathbf{S}, \mathbf{F}, \mathbf{B})$, where $\mathbf{S}$ describes the state of the entity, $\mathbf{F}$ is the entity's evaluation function, and $\mathbf{B}$ is the collection of behaviors performed by the entity.

Entity's state $\mathbf{S}$: The state of the entity is defined as follows:

$$\mathbf{S} = (internal\_state, age, p, m, L_s), \tag{2}$$

where:

$internal\_state$ – defines the entity's internal state, which can be: $active$ is the initial state in which the entity stays while roaming from pixel to pixel and performing pixel evaluation; $communicable$ is the state that the entity enters when it finds an unmarked pixel which is evaluated as logical "*true*" by the evaluation function $\mathbf{F}$ (i.e., the pixel belongs to the homogeneous region). The entity will stay in this state while applying communication behavior with its neighbors in order to determine the unique component label; $sleep$ is the state that the entity enters when the component label is determined; $dead$ is the state that entity enters when its age exceeds its lifespan. Such an entity is not utilizable for further image processing, so it is removed from the environment $\mathbf{E}$.

age - entity's age. All entities initially placed in the system or created as a result of performing self-reproduction behavior from its parent have initial ages set to zero. The entity's age is incremented during the performance of each type of behavior from the $\mathbf{B}$ except the interaction by communication with its neighbors. During communication with its neighbors the entity's ages are frozen (i.e., they are constant over time).

p – entity's position in the environment $\mathbf{E}$. Since all the entities are operating on the pixels this parameter represents the coordinates of the pixel at which the entity is temporarily located.

m - entity's memory. A memory provides the entity's ability to store information obtained by communication

with its neighbors during the component-labeling process. The memory is initially empty.

L_s - entity's lifespan. When the entity's age reaches L_s as a threshold value, an entity changes its state to $dead$ and vanishes from the environment $\mathbf{E}$. This parameter can be viewed as a measure of how much work the entity could perform before it dies.

Entity's evaluation function F: In order to assess if a pixel at which it is currently positioned belongs to the homogeneous region an entity performs the evaluation function $\mathbf{F}$ on its local part of the environment $\mathbf{E}$. The local part of the environment $\mathbf{E}$ for an entity e_k is defined as a square region centered at the entity's current pixel position, denoted by $e_k.p$ with radius R using L_∞ norm.

It is assumed that the homogeneity of the local part of the entity's environment $\mathbf{E}$ (i.e., the image segment) can be specified using the criterion for homogeneity. This criterion is based on the relative contrast, the regional mean and the regional standard deviation of the gray-level intensity (like in [6]). The evaluation function performed by an entity e_k located at the pixel $e_k.p$, where $e_k.p = (i, j)$, $i = 0, 1, 2, \ldots, W-1$ and $j = 0, 1, 2, \ldots, H-1$, is expressed as follows:

$$\begin{aligned}\mathbf{F}(e_k.p, R) = &\, non\_contrast\_criterion(i,j)_{(R)} \\ &\wedge mean\_criterion(i,j)_{(R)} \\ &\wedge std\_criterion(i,j)_{(R)},\end{aligned} \tag{3}$$

where $\wedge$ denotes logical AND, and R is the radius of the entity's local part of the environment $\mathbf{E}$ centered at the pixel $e_k.p$ using L_∞ norm. The result of the evaluation function $\mathbf{F}$ is logical "*true*" or logical "*false*". The non-contrast criterion is expressed as follows:

$$non\_contrast\_criterion(i,j)_{(R)} = \begin{cases} true & \text{if } non\_contrast(i,j)_{(R)} \geq \delta \\ false & \text{otherwise} \end{cases} \tag{4}$$

$$non\_contrast(i,j)_{(R)} = \sum_{m=i-R}^{i+R} \sum_{n=j-R}^{j+R} c(i,j,m,n) \tag{5}$$

$$c(i,j,m,n) = \begin{cases} 1 & \text{if } |\mathbf{E_1}(i,j) - \mathbf{E_1}(m,n)| \leq \eta \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where $\mathbf{E_1}(i, j)$ denotes the intensity value of the pixel (i, j) on the given image. The non-contrast criterion specifies the required number of pixels, which should be equal to, or greater than, a predefined constant δ, within the entity's local part of the environment $\mathbf{E}$. The mean and the standard deviation criteria can be expressed as follows:

$$mean\_criterion(i,j)_{(R)} = \begin{cases} true & \text{if } mean(i,j)_{(R)} \in [\mu_1, \mu_2] \\ false & \text{otherwise} \end{cases} \tag{7}$$

$$mean(i,j)_{(R)} = \frac{1}{(2R+1)^2} \sum_{m=i-R}^{i+R} \sum_{n=j-R}^{j+R} \mathbf{E_1}(m,n) \tag{8}$$

$$std\_criterion(i,j)_{(R)} = \begin{cases} true & \text{if } std(i,j)_{(R)} \in [\lambda_1, \lambda_2] \\ false & \text{otherwise} \end{cases} \tag{9}$$

$$std(i,j)_{(R)} = \frac{1}{(2R+1)} \sqrt{\sum_{m=i-R}^{i+R} \sum_{n=j-R}^{j+R} (\mathbf{E_1}(m,n) - mean(i,j)_{(R)})^2} \tag{10}$$

where μ_1 and μ_2 represent the lower and upper positive predefined thresholds for which the mean criterion is evaluated as logical "*true*" if its value falls between those two limits. Otherwise, the mean criterion will be evaluated as logical "*false*". λ_1 and λ_2 define the lower and upper predefined thresholds for the standard-deviation criterion.

The evaluation function $\mathbf{F}$ = "*true*" at the pixel $e_k.p$ tells the entity that its local part of the environment is homogeneous. In this case the entity changes its state from $active$ to $communicable$ and leaves the mark in $\mathbf{E_0}$ at the position $e_k.p$.

Entity behaviors B: The entity behaviors $\mathbf{B}$ consist of the following behaviors: $Diffuse$, $Self\_reproduce$ and $Communication$.

$Diffuse$ – the entity is performing the reactive behavior $diffuse$ if an evaluation function $\mathbf{F}$ at its current pixel $e_k.p$ returns logical "*false*". The entity will change its current pixel position to another randomly chosen pixel position within its local part of the environment $\mathbf{E}$, which is not occupied by another entity. It should be noted that if the entity is not able to move to another pixel within its local part of the environment $\mathbf{E}$, because all the pixels have already been occupied by other entities, the entity will stay at the current pixel until one of the neighboring pixels becomes free. The entity's age will be increased by 1 for each cycle of the central clock.

$Self\_reproduce$ – if an entity obtains logical "*true*" from the evaluation function $\mathbf{F}$, the entity will leave a mark at the notice board $\mathbf{E_0}$ associated with the entity's current pixel position $e_k.p$. Subsequently, depending on the number of non-occupied pixels within its 8-neighborhood, the entity will reproduce from zero to eight offspring entities. If all the pixels around the entity are occupied by other entities, no offspring entities will be created. After applying the $self\_reproduce$ behavior the entity will change its state to $communicable$, and will be ready to perform $communication$ behavior in the following cycles of the central clock.

$Diffuse$ and $self\_reproduce$ behaviors are regarded as being reactive since they are entirely determined by the local state of the entity's environment.

$Communication$ – this type of behavior is performed only by entities whose state is set to $communicable$. While applying $communication$ behavior an entity is immovable. The entity is located at the pixel that it has previously found and which belongs to the homogeneous region according to the evaluation function $\mathbf{F}$. The entity's goal is to determine the unique component label for

the pixel at its current position $e_k.p$. $Communication$ behavior consists of sending messages ($send\_messages$), processing messages ($process\_messages$) and receiving messages ($receive\_messages$). Communication includes all the neighboring entities located within the entity's 8-neighborhood. The message generated by the entity contains only a number computed from the pixel's position coordinates $e_k.p = (i, j)$ using row ordering (i.e., the content of the message is $i \cdot W + j$, where W is width of the image). This number is unique for each pixel position inside the image lattice since one pixel position can contain only one entity at a time. When the entity receives a message containing a number smaller than its current number computed from its pixel position or stored in its memory $e_k.m$, the entity will store this number in its memory and also pass this number on as a message to all its neighbors. In the other case, messages containing a number greater than, or equal to, the number that is already stored in the entity's memory are not considered and will be discarded.

Through a number of central clock cycles, entities that belong to the same homogeneous region, by performing $communication$ behavior, will get the same number stored in their memories. This number is unique for each component and represents a component label. After a predefined number of central clock cycles being inactive in the state $communicable$ (i.e., not receiving any message), the entity will change its state to $sleep$.

3. The AOC based algorithm for image-component labeling

An algorithm for AOC-based image-component labeling based on the previous description is given in Fig. 3.

The system $\mathbf{A}$ is initialized by clearing the notice board $\mathbf{E_0}$, loading the image to $\mathbf{E_1}$ as well as loading the following parameters: initial number of entities N, entity lifespan L_s, entity local environment radius R, threshold limits δ and η for relative non-contrast criterion, threshold limits μ_1 and μ_2 for region mean criterion, threshold limits λ_1 and λ_2 for standard deviation criterion. Initially, N entities are initialized by setting the following: the entity's internal state to $active$, the entity's age to zero, and the entity's position p to a random pixel within the image lattice sized $W \times H$ that is not previously occupied by another entity. An entity also clears its memory and sets its lifespan L_s, which is constant and is equal for all the entities (Fig. 3; line $1-6$).

An entity performs actions depending on its internal state and the characteristics of the local part of its environment. Every entity in the $active$ state, in order to find a pixel that belongs to the homogeneous region and that has not been detected by any other entity, performs a sequence of actions that includes reading $\mathbf{E_0}$, evaluating $\mathbf{E_1}$ with function $\mathbf{F}$ and writing the results back to the $\mathbf{E_0}$. Depending on the results obtained from $\mathbf{F}$, the entity applies the behaviors $diffuse$ and $self\_reproduce$. After applying the $self\_reproduce$ behavior, the entity changes its internal state to $communicable$ in which it repeatedly performs the behavior $communication$ (Fig. 3; line $9-31$). The activities within a cycle of the central clock are described (Fig. 3; line $8-35$). $Behavior\_communication$ for the entity e_i is described in Fig. 2. In every central clock cycle the entity in the state $communicable$ has the ability to receive messages from its eight neighbors ($receive\_messages$). These messages are temporarily stored in entity's memory $e_i.m$. The entity processes the messages contained in the memory during each central clock cycle. For synchronization purposes the entity processes only those messages that were received in the previous cycle. While processing messages, the entity compares the numbers contained within each received message to the number stored in its memory, which is the smallest number so far. If a new smallest number is found, it is stored in the entity's memory and a message containing the new smallest number is forwarded to all the entity's neighbors. If the entity does not receive any message within $W + H$ central clock cycles, where W and H are the width and height of the analyzed image, the entity will write the smallest number from its memory to the notice board $\mathbf{E_0}$ and change its internal state to $sleep$.

$e_i.m \leftarrow e_i.receive\_messages()$
$e_i.process\_messages()$
$e_i.send\_messages()$
if # of central clock cycles from the last received message $>$ $H + W$ **then**
  $e_i.write\_\mathbf{E_0}$(the smallest number in memory $e_i.m$)
  $e_i.internal\_state \leftarrow sleep$
end if

Figure 2. Detailed description of behavior $communication$

4. Experiments

The proposed AOC-based system has been implemented as a program simulator and it was run on a set of 250 images containing objects from "blocks world" [11]. The goal of the component labeling was to detect all the objects present in a given image and to assign every object a different label. The parameters used in the simulation are obtained from a subset of 30 images, and tested on the remaining 220 images. The simulation parameters are listed in Table 1.

Table 1. Simulation parameters

Parameter	Value
Initial number of entities (N)	100
Entity lifespan (L_s)	10
Local environment radius (R)	3
Relative contrast threshold (δ)	10
Relative contrast threshold (η)	25
Region mean limits $[\mu_1, \mu_2]$	[0, 255]
Standard deviation limits $[\lambda_1, \lambda_2]$	[0,20]

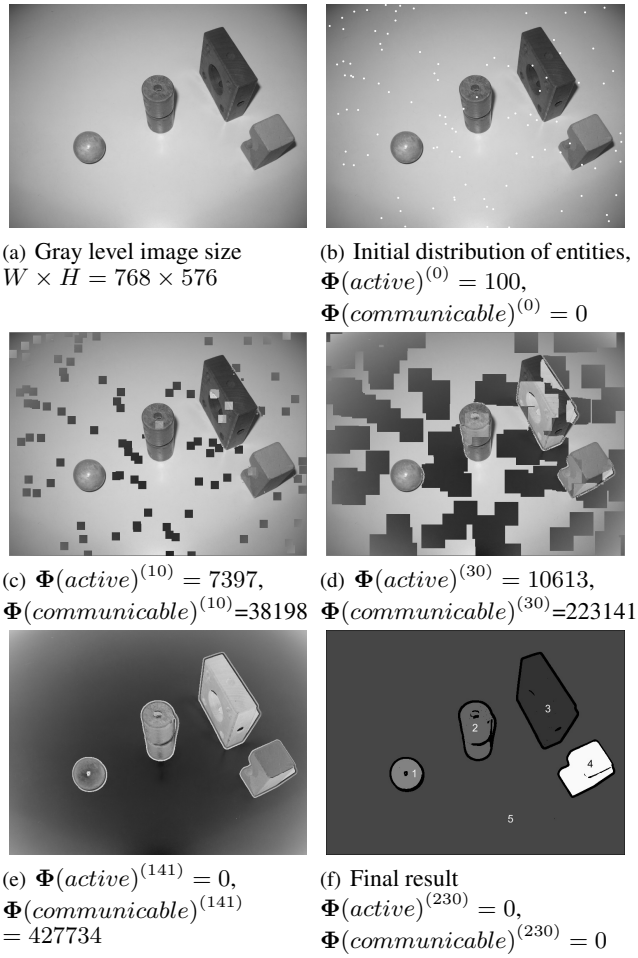
Initially, N entities are randomly distributed over the environment $\mathbf{E}$. Fig. 4 shows a series of intermediate steps during the image segmentation and component labeling. Figure 4.a) represents an input gray-level image of size 768×576. In Fig. 4.b) the initial state of the environment $\mathbf{E}$ is shown: one hundred entities are distributed randomly

1: $\mathbf{E_0} \leftarrow 0, \mathbf{E_1} \leftarrow image$ ▷ clears all cells in the notice board and loads $image$
2: Load parameters: N, L_s, R, δ, η, μ_1, μ_2, λ_1, λ_2
3: Create N entities
4: **for** entity e_i $(i = 1, 2, 3, \ldots,$ N$)$ **do** ▷ entity initialization
5: $e_i.internal\_state \leftarrow active$, $e_i.age \leftarrow 0$, $e_i.p \leftarrow$ random position, $e_i.m \leftarrow 0$, $e_i.L_s \leftarrow L_s$
6: **end for**
7: $\mathbf{\Phi}(active)^{(0)} =$ N and $\mathbf{\Phi}(communicable)^{(0)} = 0$, $cycle = 0$
8: **while** $\mathbf{\Phi}(active)^{(cycle)} > 0$ or $\mathbf{\Phi}(communicable)^{(cycle)} > 0$ **do**
9: **for** entity e_i $(i = 1, 2, 3, \ldots, N_{cycle})$ **do** ▷ N_{cycle} is number of entities in state $active$ in cycle $cycle$
10: **if** $e_i.internal\_state = active$ **then**
11: $e_i.read\_\mathbf{E_0}(e_i.p)$ ▷ read information from the notice board at pixel $e_i.p$
12: **if** e_i is located at pixel not yet visited by any entity **then**
13: $e_i.\mathbf{F}(e_i.p, R)$ ▷ evaluate entity's local part of environment $\mathbf{E}$
14: $e_i.write\_\mathbf{E_0}$(write result of $\mathbf{F}$)
15: **if** e_i is located at pixel that belongs to homogeneous region **then**
16: $e_i.behavior\_self\_reproduce()$
17: $e_i.internal\_state \leftarrow communicable$
18: **else**
19: $e_i.behavior\_diffuse()$
20: **end if**
21: **else**
22: $e_i.behavior\_diffuse()$
23: **end if**
24: $e_i.age = e_i.age + 1$
25: **if** $e_i.age > L_s$ **then**
26: $e_i.internal\_state \leftarrow dead$
27: **end if**
28: **else if** $e_i.internal\_state = communicable$ **then**
29: $e_i.behavior\_communication()$ ▷ See Fig. 2. for detailed description of behavior $communication$
30: **end if**
31: **end for**
32: Remove all entities with $internal\_state = dead$
33: Add new entities generated in this $cycle$ by behavior $self\_reproduce$
34: $cycle = cycle + 1$
35: Evaluate $\mathbf{\Phi}(active)^{(cycle)}$ and $\mathbf{\Phi}(communicable)^{(cycle)}$
36: **end while**
37: Notice board $\mathbf{E_0}$ contains final result (labeled components)

Figure 3. AOC based algorithm for image component labeling

over $\mathbf{E_1}$ (shown as white pixels). $\mathbf{E_0}$ is not shown. Fig. 4.c) shows the state of the system $\mathbf{A}$ at the central clock $cycle = 10$. The shown image is obtained by overlapping the layers $\mathbf{E_0}$ and $\mathbf{E_1}$, where the square-like regions represent parts of the image that have been marked as homogeneous by the entities so far. Fig. 4.d) represents the state of the environment at $cycle = 30$. It can be seen that the number of entities in the state $active$ has increased, but the number of entities in the state $communicable$ has increased a great deal. This is due to the region mean criterion limits μ_1 and μ_2, set as 0 and 255, respectively. In this particular case the region mean criterion is not important, which allows us to obtain a whole image segmented, including all the objects and the background. Only the parts of the image that are very inhomogeneous, like the borders between the objects or the textures with steep gradients, are not segmented. Otherwise, if it is necessary to detect only one type of objects within a known or experimentally determined range of intensity values, the region mean limits could be set accordingly. Fig. 4.e) shows the state of the environment at $cycle = 141$, when there are no entities with the internal state $active$. However, all the remaining entities are in the state $communicable$, which means that at this phase of activity, the entities only perform $communication$ behavior in order to determine the unique component labels for the detected homogeneous regions. At the $cycle = 230$ no entities in the state $active$, nor the state $communicable$ remain in the system $\mathbf{A}$, which means according to the evaluation function $\mathbf{F}$ that this state of the system is considered as the end of the image-component labeling process. During this cycle all the entities, immediately before changing the internal state to $sleep$, wrote the smallest number from their memories to the notice board $\mathbf{E_0}$. Fig. 4.f) depicts the final result of the image-component labeling. $\mathbf{E_0}$ contains five different labels, which mean that four objects and the background have been detected.

During the experiments the initial number of entities was varied between 100 and 50000 (0.022% to 11.3% of the number of total image pixels), while the entity lifespan was constant ($L_s = 10$). We have noticed that, depending on the set parameters for the relative contrast, the regional mean and standard deviation, if many pixels satisfy the entity evaluation function $\mathbf{F}$ then it is better to initially have a larger number of entities, since in that case the



(a) Gray level image size $W \times H = 768 \times 576$

(b) Initial distribution of entities, $\mathbf{\Phi}(active)^{(0)} = 100$, $\mathbf{\Phi}(communicable)^{(0)} = 0$

(c) $\mathbf{\Phi}(active)^{(10)} = 7397$, $\mathbf{\Phi}(communicable)^{(10)}$=38198

(d) $\mathbf{\Phi}(active)^{(30)} = 10613$, $\mathbf{\Phi}(communicable)^{(30)}$=223141

(e) $\mathbf{\Phi}(active)^{(141)} = 0$, $\mathbf{\Phi}(communicable)^{(141)} = 427734$

(f) Final result $\mathbf{\Phi}(active)^{(230)} = 0$, $\mathbf{\Phi}(communicable)^{(230)} = 0$

Figure 4. Simulation Results

component-labeling process tends to complete in a fewer number of central clock cycles. The average number of central clock cycles needed for the image-component labeling was 235 cycles, while the average simulator processing time for the PC-based dual-core processor running at 2.8 GHz was 191ms. Distinct object were successfully labeled on 208 out of 220 tested images. Remaining 12 images (5.5% of 220) produced results in which at least two different objects gained the same label. These images contained two or more overlapped objects with similar gray-level properties and hence the entity evaluation function was unable to differentiate borders between such objects and assigned them the same label.

5. Conclusion

This paper describes an AOC-based approach to image-component labeling. A number of entities with carefully defined properties and behaviors, initially distributed over the image, with the ability to communicate directly or indirectly, store information in the notice board, show an example of how autonomy oriented computing can be utilized for low-level image-processing tasks. The main advantage of the proposed approach is that entities, while diffusing, self-reproducing and communicating, are completely autonomous and they are making decisions directed only by the state of their local part of the environment. This approach also shows a great potential for the further introduction or extension of entities' states and behaviors, as well as modeling the system objective function to achieve other image-processing tasks, such as image feature tracking or image edge detection. In the future, based on our simulations, AOC-based hardware architecture, which enables entities in the system to act in parallel, will provide at least the same quality of component labeling as "classical" approaches, but will also provide us a hardware framework for the implementation of naturally inspired behaviors (ant-colony optimization [9]), with a significant speedup. It can be seen that with such an architecture the number of central clock cycles needed to complete the selected image-processing task can be used as the performance index. For comparison purposes, component labeling based on a connected component-labeling algorithm [10] needs about 20 ms or $5.6 \cdot 10^7$ cycles working on a 2.8 GHz processor for the component labeling. The same task in an AOC-based hardware architecture can be performed in approximately 250 central clock cycles, without counting the time needed for system initialization.

Acknowledgment

The work reported in this paper has been supported by the Croatian Ministry of Science, Education and Sports, as the part of the R&D project Theory, Modeling and Applying of Autonomy Oriented Computing Structures (no. 036-0361935-1954).

References

[1] J. Liu, X. Jin, K. C. Tsui, *Autonomy Oriented Computing*, 1^{st} edition, Springer, New York, 2004.

[2] J. Von Neumann, *Theory of Self-Reproducing Automata*, edited and completed by A. W. Burks, University of Illinois Press, Urbana, 1966.

[3] J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1975.

[4] J. Ferber, *Multi-Agent Systems*, Addison-Wesley, New York, 1999.

[5] J. Liu, Y. Y. Tang, Y. C. Cao, "An evolutionary autonomous agents approach to image feature extraction", *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 2, pp. 141 - 158, July 1997.

[6] J. Liu and Y.Y. Tang, "Adaptive Image Segmentation With Distributed Behavior-Based Agents", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 6, pp. 544 - 551, June 1999.

[7] Y. Wang, B. Yuan, "Fast method for face location and tracking by distributed behaviour-based agents", *IEE Proc. on Vis. Image Signal Processing*, vol. 149, no. 3, pp. 173-178, June 2002.

[8] D. Kipcic, S. Ribaric, "A multi-agent-based approach to face detection and localization", 27^{th} *International Conference on ITI*, pp. 377 - 382, June, 2005.

[9] M. Dorigo, T. Stützle, *Ant Colony Optimization*, MIT Press, Cambridge, Massachusetts, 2004.

[10] R. C. Jain, R. Kasturi, B. G. Schunck, *Machine Vision*, 1^{st} edition, McGraw-Hill, New York, 1995.

[11] P. H. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.