Swarm Contours: A Fast Self-Organization Approach for Snake Initialization

HOSSEIN MOBAHI,[1,2,3] MAJID NILI AHMADABADI,[1,2] AND BABAK NADJAR ARAABI[1,2]

[1]Control and Intelligent Processing Center of Excellence, Department of Electrical and Computer Engineering, University of Tehran, Tehran 14395, Iran; [2]School of Cognitive Sciences, Institute for Studies in Theoretical Physics and Mathematics, Tehran 19395, Iran; [3]Department of Computer Science, University of Illinois at Urbana Champaign, Urbana, Illinois 61801

This paper was submitted as an invited paper resulting from the "Understanding Complex Systems" conference held at the University of Illinois–Urbana Champaign, May 2005

Received October 17, 2005; revised April 5, 2006; accepted May 2, 2006

A major obstacle in real-time performance of a visual tracking system is its initialization phase. Inspired by social behavior in fish and ant groups, a fast self-organization approach to active-contour initialization is proposed. Contours are emerged during two separate phases of aggregation and self-assembly. Aggregation is achieved by a superposition of simpler behaviors, hunting, avoidance, and opportunism. Self-assembly, which constitutes the explicit contour formation, occurs by mating behavior when the swarm becomes quite stable. Experiments indicate that the proposed method outperforms exhaustive image search for finding contours in high resolutions. © 2006 Wiley Periodicals, Inc. Complexity 12: 41–52, 2006

Key Words: image processing; active contours; self-organization; swarm intelligence

1. INTRODUCTION

Visual tracking constitutes a basic but essential problem in computer vision and image processing. In fact, many popular applications of computer vision such as motion estimation, surveillance, video database management, vision-based robot control, augmented reality environments, and high-resolution satellite depend on visual tracking. Basically, in visual tracking one is interested in estimation of inter-frame correspondence of items within an image sequence.

For the sake of tractability, often a compact parametric model is used to encode the shape of the object being tracked. A popular representation scheme is to overlay a bounding box around the object of interest [1]. Depending on the shape of the object, a circle, oval, and parabola, or a combination are other alternatives [2]. However, for complex-shaped objects, however, a contour model should be used. A popular contour model in image processing is an active contour or snake [3], which is also the focus of this work. A snake is a contour approximated by a finite number of control points and is able to refine its shape iteratively toward the desired shape.

Once we decided about the representation, the second problem is establishing inter-frame correspondence for the object being tracked. A common approach for estimating such correspondence is *Bayesian sequential estimation* [4].

Hossein Mobahi, E-mail: hmobahi2@uiuc.edu

This method recursively estimates a time-evolving distribution that describes the object state conditional on all observations seen so far. Using such estimation from past observations, the tracker may predict the next state of the object, which reduces search space of a feature detector accordingly. The reduction of search space not only improves robustness of a tracker by eliminating false positives, but also speeds up the detection process.

Nevertheless, initially there are not enough frames to launch such a recursive estimator. Consequently, no prediction can be made during initialization and the whole image must be searched exhaustively for detecting objects of interest, which is very time-consuming. Worse is that, for relatively long video streams the tracker may have to be reinitialized repeatedly, as the accumulated tracking error increases such that the tracking confidence drops below a threshold [5]. Therefore, initialization is a significant speed bottleneck in real-time visual tracking. The initial exhaustive search checks all pixels in the image for certain features, e.g., edge or color, to localize the object and fit it to the given representation.

Regardless of how fast the feature extraction operators are, the exhaustive search of an image becomes computationally expensive as the image size grows. This fact has limited the practical resolution of images used in real-time video processing, particularly for color videos [6]. A possible solution to this problem would be performing a partial exploration of the image while using the same simple feature detectors and yet achieving a satisfactory localization. However, we would benefit only when the partial exploration algorithm is not computationally expensive itself.

In this article, we propose an efficient algorithm for localizing objects in a digital image by exploring only a small fraction of the image. Because each object contour is represented by a set of discrete points, this contour can directly be considered as the initial condition of a snake. The proposed method automatically obtains the number of required snakes as well as the appropriate number of the control points. Experiments confirm that this method works faster than exhaustive search, without resorting to special hardware or parallel processing facility.

Our proposed idea is mainly inspired by the social behavior arising in fish schools and ant colonies. In the past two decades, engineering has enjoyed a substantial effort for transforming models of social behavior of insects and animals into useful optimization or control algorithms. These algorithms are usually referred to as *swarm intelligence* [7]. Using distributed and simple multiagent methods, these techniques generally achieve reasonable solutions of hard problems in a practical amount of time. For example, optimization algorithms inspired by models of cooperative food retrieval in ants have been unexpectedly successful in real-world problems such as telecommunication [8] and vehicle routing [9].

This article is organized as follows. In section two, related works in the literature of multiagent and swarm techniques for image processing are reviewed. The essentials of active contours are summarized in section three. Section four focuses on self-organization of patterns in social animals. Specifically this section brings out properties of aggregation and self-assembly which are useful in our model. Mathematical models for the behaviors of interest are illustrated in section five. Section six evaluates the proposed algorithm using real images taken in different resolutions. Finally, section seven contains our conclusion and possible paths for future works on the proposed algorithm.

2. RELATED WORKS

Swarm intelligence is a special case of systems composed of several agents capable of reaching goals that are difficult to achieve by an individual system. Here we will first provide a summary of the literature on recent multiagent works for solving vision problems. Then we will clarify its difference with swarm intelligence and narrow down our survey to swarm approaches. Finally, we will clarify how these works differ from our proposed algorithm.

A part of these works can be criticized for merely replacing traditional algorithms by multiagent systems without any major and explicit improvement. In these researches, the multiagent solution seems just as a sophisticated metaphor without providing better results than traditional techniques [10,11]. However, some studies have applied multiagent ideas in order to achieve practically distinctive results. The goal of these efforts are improving precision and robustness or reducing the processing time. Recent studies in robustness direction have generally focused on providing the agents with knowledge and deliberative capabilities [12,13]. In these works, low-level processing is still carried out by traditional algorithms.

On the other side are applications of multiagent approach for reducing processing time. Because of the distributed and asynchronous nature of multiagent systems, parallel processing is an apparent answer for speeding up image processing. For example, Yanai [14] proposed a multiagent architecture implemented on a highly parallel computer with 16 processing elements, each of which was allocated to a separate agent. Besides this trivial application, there were attempts with the same goal but using uniprocessor platforms. For instance, Zhou et al. [15] introduced an agent-based architecture that uses a utility optimization technique to guarantee that important vision tasks are fulfilled even under resource (e.g., CPU and memory) constraints.

In all of the mentioned works, low-level image processing was based on traditional algorithms. There is another research avenue in applications of multiagent systems in image processing that explores their capacity for accomplishing low-level operations. Before reviewing these works,

© 2006 Wiley Periodicals, Inc.
DOI 10.1002/cplx

we first explain what characteristics agents in a swarm possess. In contrast to the discussed multiagent systems that were based on small-size population of complicated agents, a swarm consists of simple reactive agents in a relatively large population. These agents usually interact through a shared and quite large environment, e.g., the image. Limited percept-motor capabilities of agents are generally ingredients of swarm simplification that makes it suitable for development of local functions such as low-level image processing.

In the realm of swarm methods, particle swarm optimization (PSO) has recently raised a lot of interest in image processing. The PSO idea has been applied to different areas of computer vision including object recognition [16] and image clustering [17]. In a PSO, each individual encodes the whole solution of the problem. Therefore, when the search space is very large (as in image processing) execution of a PSO is lengthy. In contrast, we prefer the solution to be distributed among its members. This might reduce the complexity of the problem that each agent has to solve individually and therefore decrease the processing time. In the following, we only focus on swarm methods that utilize a distributed solution.

Rodin et al. [18] present a system in which the agents move within the image and modify it gradually. There is no direct communication among the agents; rather they influence each other by modifying the shared environment (image). Their system is applied to detection of concentric striate, like the year rings of trees, with darkening and lighting agents. The authors claim that their result is hard to achieve by traditional global feature detection methods.

Germond et al. [19] propose an agent-based segmentation of MRI brain scans in dark and white matters. Agents can resolve conflicts on a first-come, first-served basis without any negotiation. Spinu et al. [20] introduced a system with low-level agents that produce partial data-driven, edge-based segmentations of medical images that are merged into a global result. The agents have only indirect influence on the global result and have no possibility to communicate with the other agents operating on the image.

Liu and Tang [21] used a swarm system to detect homogeneous regions and applied it to brain scan images. Each agent can test pixels around it in a circular neighborhood. Neighboring pixels are evaluated in terms of simple statistics such as the variance or the mean of gray level. If an agent finds that its neighborhood satisfies the conditions to be a region, the central pixel will be marked and new agents will be generated in this neighborhood; otherwise the agent will exhibit a diffusion behavior by moving to a new location within its neighboring region. When the age of an agent exceeds its life span, the agent will vanish. Also, when the activity of agents drops, the algorithm is stopped.

Their method is well adapted to brain scan images, because regions characteristics are regular for tumors, sane parts, etc. The authors state that their approach is different from conventional methods because the proposed agents do not need to perform an exhaustive search on the image. However, there is no comparative report on execution time of Liu's and other swarm approaches with conventional methods.

In summary, up to our knowledge, no swarm-based image processing system has improved execution time of the algorithm. In addition, we are not aware of any swarm-based method for snake initialization. From the works reviewed in this section, Liu and Tang's is the closest to our proposed model. Yet there are fundamental differences in between. For instance they represent regions by dense maps marked by agents. In other words, the number of agents must be approximately equal to the number of pixels in the regions of interest. Computational cost of such a tremendously large population puts any hope for achieving a fast algorithm in doubt. In this article we will take another route to specifically achieve an economic swarm-based solution for contour localization. The main stream of our approach is based on contours approximated by a few control points, as in snakes. We will show that we could achieve a computationally more effective method.

3. ACTIVE CONTOURS

Because our work is concentrated on snake initialization, we briefly review snake definition and its characteristics. An active contour or snake, first introduced by Kass et al. [3], is a popular instance of free-form deformable models. A snake is described parametrically by $\mathbf{v}(s) = (x(s), y(s))$, where $x(s)$ and $y(s)$ are x,y coordinates along the contour and $s \in [0,1)$ is normalized arc length. The snake model defines the energy of a contour $\mathbf{v}(s)$, the snake energy, E_{snake}, as follows:

$$E_{snake}(\mathbf{v}(s)) = \int_{s=0}^{1} E_{\text{internal}}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s))ds, \qquad (1)$$

where E_{internal} is the internal energy of the contour, imposing continuity and curvature constraints, E_{image} is the image energy constructed to attract the snake to desired feature points in the image. In nonautomatic applications, human operator can interactively impose constraints on the contour through another energy term called external energy.

Internal force of an agent characterizes the regularization properties of the snake. A widely used formulation for internal energy is as shown in (2). It is a weighted sum of elasticity and stiffness derived from the first and the second derivatives of the contour, respectively,

$$E_{\text{internal}}(s) = \frac{1}{2}\,(\alpha(s)\|\mathbf{v}_s\|^2 + \beta(s)\|\mathbf{v}_{ss}(s)\|^2). \qquad (2)$$

Once the forces are defined, each force creates its own potential field and then initial contour evolves by minimizing the overall force in (1) using variational calculus techniques. This evolution of the snake deforms it to conform to the nearest salient contour, and therefore it is very sensitive to its initial condition. Consequently, a snake must be initialized about the item of interest before starting its evolution. The snake algorithm says nothing about initial location of a snake, and this must be determined by another process. However, because of the lack of an efficient method for snake initialization, generally an exhaustive search is performed over the image to coarsely localize items of interest. Next, snakes are initialized at those locations to finely extract their shapes.

4. SWARMS IN NATURE

Social insects and animals like ants, bees, and fish are generally conceived as simple species with no or a low level of intelligence. This is because each individual has limited cognitive ability and limited knowledge about its environment [22–24]. However, these species collectively exhibit exciting problem-solving skills. The defining characteristic of a swarm (a crowd of social insects) is self-organization, i.e., there is no central planning component and any complex behavior of the swarm emerges from interaction of the agents' low level behavior.

These interactions are based on local information, without reference to the global pattern [7,22]. This is how the behavior of many biological systems, such as colonies of ants and bees, flocks of birds, and schools of fish is explained. Animal societies present multiple forms of self-organization. In the following section, we study two of these forms that our method takes inspiration from, namely aggregation and self-assembling.

4.1. Aggregation

Aggregation is a crucial social phenomenon because of being a prerequisite for the development of other forms of cooperation (including self-assemblage). The patterns of aggregation are very diverse, ranging from the gathering of all insects in one site to scattering them among several ones.

We model aggregation behavior according to the observed behavior in a fish school. Schooling means that a group of fish acts as a single organism than a collection of individuals. Schooling would appear to follow the rules of a distributed model, i.e., each individual applies the same set of simple behavioral rules. More specifically, each fish takes into account all fish that swim in its neighborhood, paying more attention to the closest ones and trying to match its velocity and direction with that of its neighbors [25]. Such local interactions result in a highly coordinated structure without any leader to determine the form of this configuration.

The arrangement emerged in a fish school does not have a fixed geometric form; the structure is loose and it results from each fish's applying a few simple behavior rules. This is similar to an active contour or snake in image processing where the contour deforms as the result of local interactions among its control points internally and with the image. However, unlike fish schools where mating emerges dynamically, in snakes' neighborhood, connections must be established a priori. This constrains the evolution space of a snake and reduces its exploratory capability and expressivity. Inspired from animal societies, our model does not force fixed mates for the agents; mating emerges through agents' local interactions.

Specifically, there are three basic observations in a fish school that will constitute our computational model. First, each fish searches its nearby surroundings to hunt food. Second, fish maximize the search area by barely remaining in the sight of one another [25]. Third, if a member of the school finds food, the other members can take advantage of its finding [25]. Note that the area that a fish can hunt (small pieces of) food in is generally much smaller than the area that it can see other (large) fish. We will take advantage of this target-dependent acuity of perception in our model as well.

4.2. Self-Assembling

Self-Assembly is another form of self-organization that is observed in some social insects such as ants, bees, and wasps. In self-assembly each member connects to some other members, creating complex physical structures [26]. For instance, ants of the genus *Oecophylla* [27] are characterized by their capacity to hang on to each other to form chains. This allows the bridging of an empty space, for example, between two branches [28]. Self-assemblages may have different patterns like bridges, curtains, ladders, rafts, tunnels, walls, etc. [26]. In this work we are interested in structures that form a contour. Similar to aggregation, self-assembly follows local rules and the overall chain is resulted by self-organization.

5. COMPUTATIONAL MODEL

Inspired by biological observations reviewed in the previous section, we develop our own creatures adapted for fast image exploration. Because the system implements a swarm of agents, it needs the metaphor of an environment in which the agents are situated. This environment is a set of selected properties of the image such as edges or high-intensity regions. In this article, we focus on two-dimensional (2D) environments, i.e., planar images. However, the proposed model is scalable to 3D images as well, because it entirely relies on vector operations.

© 2006 Wiley Periodicals, Inc.
DOI 10.1002/cplx

5.1. Aggregation

Our proposed swarm consists of a set of agents with simple sensory motor capability. Initially the agents are distributed over the image plane at random locations. Throughout swarm evolution, the agents seek for better food resources by following food trails. Food sources are regions that contain the desired image features, e.g., edges. Formally, food is a scalar function of the environment that is large where features of interest exist. This response can be a matter of degree because the feature quality may differ from location to location, e.g., edge strength. We constrain the definition of food to simple local operations, i.e., dependence on a limited neighborhood, to keep the agents light-weight and efficient. Fortunately, such a simple definition can be realized by a large class of linear or nonlinear local filters known in image processing literature.

The behavior of each agent is defined by three state variables, namely position ($\mathbf{X}$), velocity ($\mathbf{V}$), and energy (E). In the following we explain these variables individually and describe how together they form the behavior of agents. Ultimately, the self-organized structure of the swarm must reflect the shape of the underlying objects. Therefore, the density of food should be higher at the boundary of the desired objects to attract agents there. Nevertheless, it may happen that some agents get stuck in their way to reach the boundaries (due to other forces that will be introduced later). These agents should not participate in contour formation; thus they are eliminated. This is achieved by considering an energy resource for each agent, which decreases over time, but increases when the agent meets a food source. Once the energy reaches zero, the agent dies.

Let us denote the energy of the ith agent by scalar $E_i(t)$. In the beginning, this energy is initialized to a nonzero value. As (3) shows, the energy decreases continually over time, and it resets when the agent moves into a nutritious region. Here, d is a constant:

$$E_i(t) = E_0 \quad \text{if } t = 0 \text{ or } Food(\mathbf{X}_i) > T_F$$
$$\frac{dE_i(t)}{dt} = \begin{cases} -d; & E_i(t) > 0 \\ 0; & \text{otherwise.} \end{cases} \tag{3}$$

The two other state variables determine the motion of an agent in the environment. The 2D position and velocity vectors for the ith agent are denoted by $\mathbf{X}_i$ and $\mathbf{V}_i$, respectively. These state variables are influenced by a superposition of attractive and repulsive force fields emitted from the environment and the other agents. The agent's trajectory is evolved according to a first-order differential equation described in (4). Note that these vectors are time-dependent, but the notation of time is eliminated for the sake of simplicity:

$$\begin{aligned} \mathbf{F}_i &= K_a\mathbf{A}(\mathbf{X}_i) + K_h\mathbf{H}(\mathbf{X}_i) + K_o\mathbf{O}(\mathbf{X}_i) \\ \dot{\mathbf{V}}_i &= \alpha(\text{sat}(\mathbf{F}_i) - \mathbf{V}_i) \\ \dot{\mathbf{X}}_i &= \mathbf{V}_i. \end{aligned} \tag{4}$$

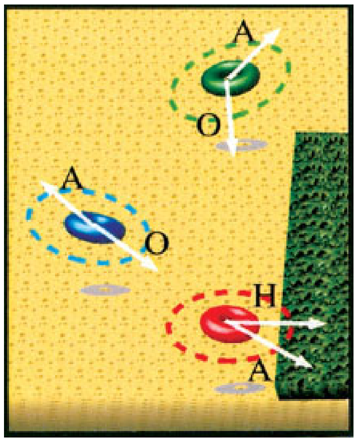
The effective force vector $\mathbf{F}_i$ of each agent is a weighted sum of three partial forces corresponding to avoidance, hunting, and opportunism. This effective force is filtered to reduce the sensitivity of the swarm to agents' local fluctuations. The smoothening degree of the filter is equal to $1 - \alpha$. In addition, for the sake of stability, we limit the maximum velocity of agents using a *saturation* function.

A schematic representation of the agents and their forces are shown in Figure 1. Moreover, Figure 2 shows the effect of each individual force as well as their superposition in a face detection task where the edge of skin-toned regions is considered as food (more details will be presented in section 6). In the rest of this section we describe each of the partial forces separately.

5.1.1. Hunting

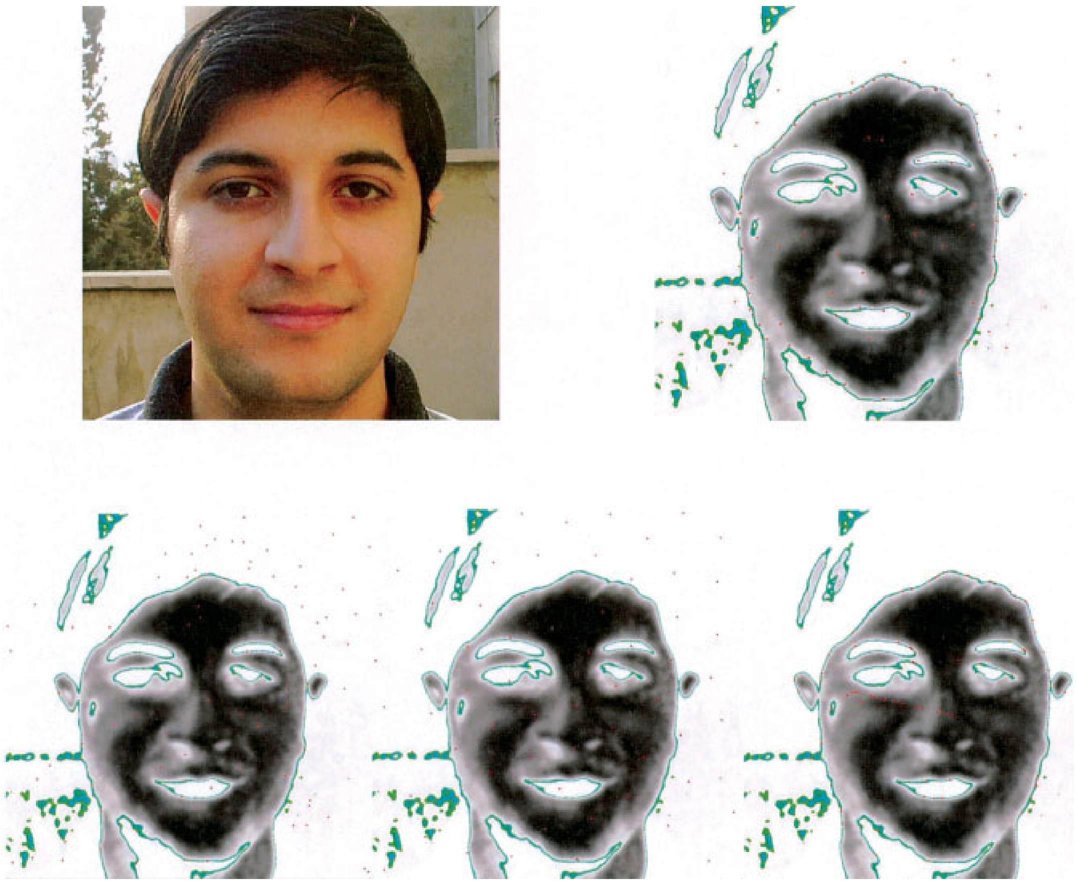
In order for agents to survive, they must sense the environment and seek for food. Similar to insects and animals, our agents have a limited field of view (FOV), which decreases the computational load of the algorithm. Indeed, this is the key point in our method that results in fractional exploration of the image. The regions out of FOV are ignored completely. Nevertheless, not all areas within the FOV are the same; the farther a point is, the less it influences the agent's behavior. Given a point within the FOV, we formulate the magnitude of the hunting force such that it is

FIGURE 1



Forces and hunting. FOV: The agent at the bottom has discovered a nourishing region. The lower agent uses hunting and avoidance, whereas two others head according to avoidance and opportunism. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

FIGURE 2



The swarm result after 250 iterations with different combination of forces. Skin-tone regions are shown by dark and their edges by the following: top left: green, original image; top right: superposition of all forces; bottom left: hunting only; bottom middle: avoidance only; and bottom Right: opportunism only.

proportional to the quality of food at that location and inversely related to its distance. The total hunting force in the entire FOV is computed in (5) by integrating all force vectors that fall inside the FOV. The parameter R defines the resolution of the image by computing the average of width and height of the image. This helps to attain scale invariant constants for the model:

$$\mathbf{H}(\mathbf{X}_i) = R \int_{\mathbf{s}\in FOV} \frac{\mathbf{s}}{\|\mathbf{s}\|} \frac{\text{Food}(\mathbf{X}_i + \mathbf{s})}{\|\mathbf{s}\|} d\mathbf{s}. \tag{5}$$

5.1.2. Avoidance

Avoidance behavior pushes the agents toward a uniform inhabitation around features. Uniform distribution of contour points often results in more accurate reconstruction of the underlying shape. Each agent finds the location of its nearest neighbor according to (6). The agent then avoids it by the repulsive force emitted from the neighbor:

$$k = \operatorname{Arg\,Min}_{j\neq i} \|\mathbf{X}_i - \mathbf{X}_j\|$$

$$\mathbf{A}(\mathbf{X}_i) = \frac{\mathbf{X}_i - \mathbf{X}_k}{\|\mathbf{X}_i - \mathbf{X}_k\|^2}. \tag{6}$$

5.1.3. Opportunism

It could take a long time for an agent to discover food merely on its own exploration. To accelerate food localization and contour formation, opportunism behavior is introduced. An opportunistic agent picks the target agent as which is located in a food-rich location. As mentioned in 4.1, we consider a target-dependent perception for our agents, meaning the FOV for opportunism (which senses other agents) should be much larger than for hunting (which senses food). Here we assume the whole swarm can potentially take advantage of an agent who has found superior food. Considering such a view field as wide as the swarm does not affect speed performance as FOV does for

© 2006 Wiley Periodicals, Inc.
DOI 10.1002/cplx

hunting. This is because the number of agents is relatively insignificant against the number of pixels in a digital image.

Agents in less inhabited regions are favored for being a target so that a uniform distribution of agents is attained. This latter constraint prevents the swarm to collapse over the richest agent. Constrained opportunism is formulated in (7), where agent k is targeted by agent i. The equation combines looking at others' food and the preference of isolated agents through a weight factor W:

$$k = \operatorname*{Arg\,Max}_{j \neq i} \left(\mathrm{Food}(\mathbf{X}_j) + \frac{W}{R} \operatorname*{Min}_{p \neq j} \|\mathbf{X}_P - \mathbf{X}_j\| \right). \qquad (7)$$

Once the target agent has been specified, its selection criterion is directly used as the magnitude of the opportunism force. However, variation of food quality in different locations might alter opportunism to a parasitic behavior. This happens when agents leave their own food and drift toward the agent recognized as possessing the highest food. To avert this trend, opportunism is triggered only when an agent starves. More precisely, opportunism occurs when the food quality for an agent is below a threshold F_0. The final opportunism force is formulated in (8):

$$\mathbf{O}(\mathbf{X}_i) = \begin{cases} R \dfrac{\mathbf{X}_k - \mathbf{X}_i}{\|\mathbf{X}_k - \mathbf{X}_i\|} \times \left[\dfrac{W}{R} \operatorname*{Min}_{p \neq j} \|\mathbf{X}_P - \mathbf{X}_j\| + \mathrm{Food}(\mathbf{X}_k) \right]; & \mathrm{Food}(\mathbf{X}_i) < F_0 \\ 0 \quad ; & \text{otherwise.} \end{cases} \qquad (8)$$

5.2. Self-Assembly

The discussed behaviors self-organize the swarm from a random initial condition to a specific order at equilibrium, which is settling about feature points, e.g., image edges. Reaching the equilibrium is tested by continually measuring the activity of the swarm until it drops off a threshold. The swarm activity, denoted by A, is estimated from the average magnitude of the agents' motion as shown in (9), where N is the number of survived agents at time t:

$$A(t) = \frac{\sum_{i=1}^{N} \|\dot{\mathbf{X}}_i(t)\|}{N}. \qquad (9)$$

Once the swarm reaches equilibrium, the agents become ready for shaping up contours. This is achieved by connecting adjacent agents to each other in a specific order. Because we are only interested in simple and closed contours, each agent must exactly choose two neighbors. An agent selects its neighbors based on two factors, distance and angle. For distance, closer agents and for angle, those that form flatter connections are preferred. These criteria result in smooth and natural contours.

Formally, given an agent with index i; it first adds its nearest agent j to its neighbors set as its first neighbor. Next, any agent whose connection to agent i does not form a sharp angle with the line segment created between i and j is potentially considered as the second neighbor. We denote this set by $S_{i,j}$. From these candidates, the agent with the minimal distance to agent i is selected as the second neighbor and is added to i's neighbors set. Note that the distance from neighbors should not exceed a certain threshold T_D. Similarly, a threshold T_A is used for the cosine of angle between two connections. The neighbor selection procedure is formally described in the following:

$$j = \operatorname*{Arg\,Min}_{p \neq i} \|\mathbf{X}_i - \mathbf{X}_p\|$$

$$S_{i,j} = \left\{ p \mid p \neq i \neq j \wedge (\|\mathbf{X}_p - \mathbf{X}_i\| < RT_D) \wedge \left(\frac{(\mathbf{X}_j - \mathbf{X}_i)^T (\mathbf{X}_p - \mathbf{X}_i)}{\|\mathbf{X}_j - \mathbf{X}_i\| \, \|\mathbf{X}_p - \mathbf{X}_i\|} < T_A \right) \right\}$$

$$S_{i,j} \neq \{\ \} \rightarrow k = \operatorname*{Arg\,Min}_{p \in S_{i,j}} \|\mathbf{X}_i - \mathbf{X}_p\|. \qquad (10)$$

Occasionally it may happen that neighborhood connections become asymmetric. This happens when agent i chooses agent j as its neighbor, whereas i is not chosen as the neighbor of j. Asymmetry may result in nodes with more than two neighbors, whereas only two neighbors per agent are allowed for forming a simple closed contour. Therefore, a mechanism is required to detect asymmetries and eliminate them.

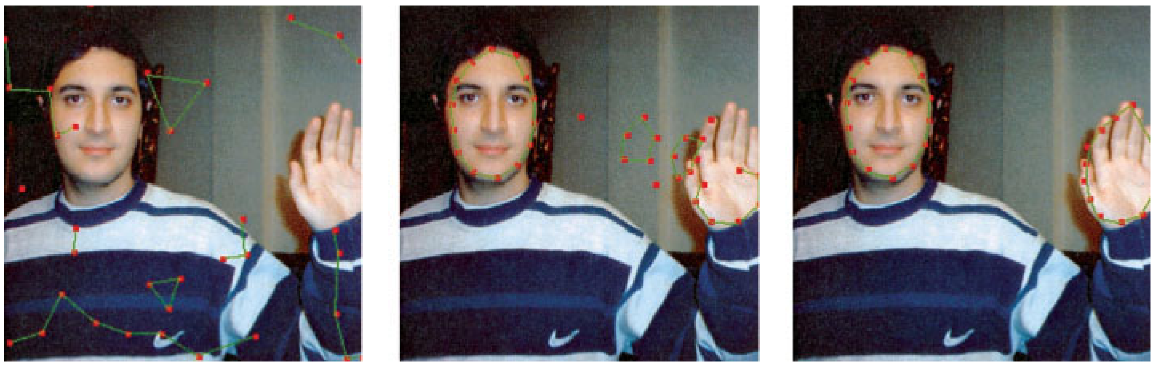
Our suggested mechanisms is formally shown in Equation (11), where N_a denotes the set of neighbors chosen by agent a. Briefly explaining the equation, any connection is checked for asymmetry and once detected, the connection is removed and a new neighbor is selected for that agent. Note that the agent is prohibited to reselect previously removed neighbors. This process is repeated until no more asymmetry remains in the swarm:

$$i \notin N_j \wedge j \in N_i \rightarrow N_i = (N_i - \{j\}) \cup \operatorname*{Arg\,Min}_{p \in S_{i,N_i - \{j\}}} \|\mathbf{X}_i - \mathbf{X}_p\|. \qquad (11)$$

6. EXPERIMENTAL RESULTS

To evaluate the competence of our proposed method against exhaustive search, we carried out experiments on real images for localizing faces, hands, and lips and some objects that could be distinguished by color. We used 20 color images taken by BenQ 2300 camera in very high resolution. These images were down-sampled to different res-

FIGURE 3



Self-organization of contours about face and hand shown in three iteration steps. Left to Right: 1, 175, and 250. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

olutions for the experiment. Feature maps were computed by a simple color-based filter, followed by gradient computation. Color classes were modeled by Gaussian distributions in RBG space as shown in Equation (12):

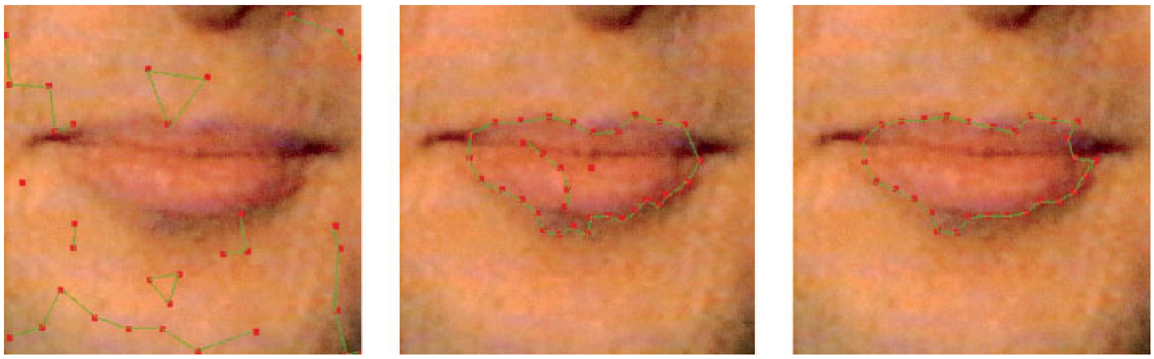
$$\mathrm{Food}(\mathbf{X}) = \mathrm{Food}(x, y) = \|\nabla_{x,y}\exp(-(\mathbf{f}(x, y) - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1} \times (\mathbf{f}(x, y) - \boldsymbol{\mu}))\|$$
$$\mathbf{f}(x, y) \in R \times G \times B. \qquad (12)$$

We applied our algorithm to detect faces, hands, and lips. These parts can be effectively identified by their color [29,30]. The parameters of these Gaussians were estimated by maximum-likelihood technique similar to [30]. This is done as a training phase and prior to execution of the actual swarm algorithm. For exhaustive search, feature map computation was treated as a preprocessing phase on the whole image. However, in our method a feature point is only computed when an agent enters to that point.

For testing our proposed method, initially 35 agents were distributed over each image at random locations. The time-evolution of agents on two test images are shown in Figures 3 and 4. Although it was not necessary to create connections before reaching equilibrium, we did it during the self-organization process merely to visualize gradual formation of the contours. In all experiments, the contours could adequately self-organize about regions of interest in at most 250 iterations. The average activity of the swarm computed from five different images and its threshold value are shown in Figure 5.

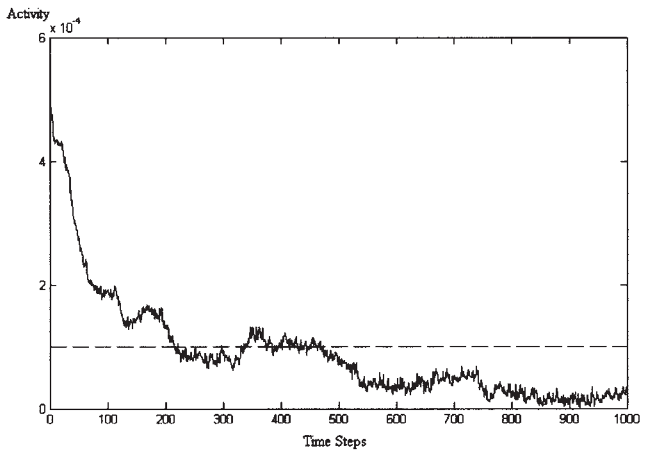
Although the final contours have not perfectly captured the underlying shapes in Figures 3 and 4, they are more than what a snake needs for initialization. In fact, these contours

FIGURE 4



Self-organization of contours about lips shown in three iteration steps. Left to right: 1, 175, and 250. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

FIGURE 5



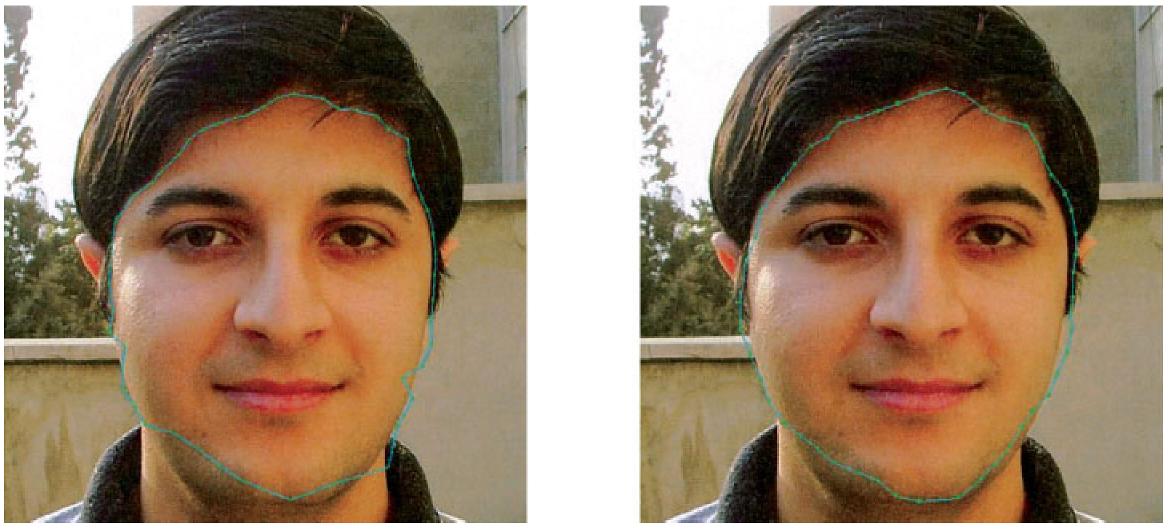
Average activity of the swarm computed from five different test images.

are superior to simple shapes commonly used in computer vision such as a bounding box, circle, or oval, for snake initialization [1,2]. Further refinement is achieved by the snake algorithm itself [3] through minimizing the first-order derivative of the obtained contour. Figure 6 (left) shows another face localization example with agents sensitive to gradient of skin color and the refined contour by snake algorithm in Figure 6 (right).

The temporal behavior of force magnitudes as well as their superposition is shown in Figure 7. It is apparent that after 250 iterations the forces exhibit almost a monotone behavior, and the swarm activity remains very low after that. By spending more time on tuning swarm parameters, even fewer iterations might be possible. Parameter values used in our experiments are listed in Table 1.

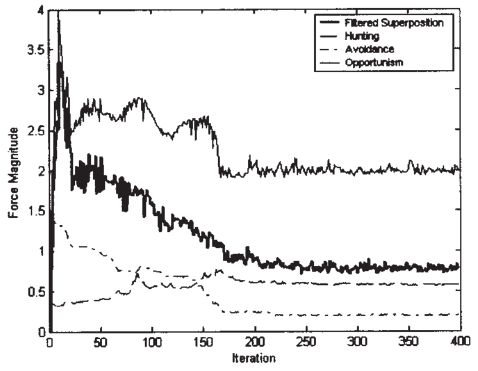
In traditional pixel-by-pixel search, the total number of scanned pixels is equal to the resolution of the image. However, in self-organized contours, multiplying number of agents by iteration steps gives the number of hits that the image is scanned. Computing the ratio of hits in the two

FIGURE 6



Evolution of the skin swarm (left) and the eventual contour (right). Face contour is modified by snake. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

FIGURE 7



Behavior of forces over time.

methods indicates that the self-organized contour method with the tuned parameters can reduce scanned pixels to about 13% of the image pixels. This is a remarkable achievement in image scanning, comparing with exhaustive search of the image. Because the limited FOV of agents keeps the computation cost of each iteration light-weight, self-organized contours outperform exhaustive scan with respect to the execution time too. This was justified by counting clock ticks for complete execution of both algorithms.

We measured the number of clock cycles taken by each method in different image resolutions. Our experimental platform was an Intel Pentium 4 processor with Linux Operating System (Redhat 9.0). Fortunately, Intel has introduced a specific instruction for reading clock cycles in the Pentium, namely Read Time-Stamp Counter (RDTSC). The value of this counter is easily accessible through Machine Specific Register (MSR) macros in Linux. The measured clock cycles for the same algorithm with the same configuration differed a bit on each run. Therefore, we executed the algorithm 10 times for each case and used the average value for comparison. Results are shown in Table 2.

TABLE 1

Swarm Parameters

Parameter	Value
α	0.1
D	0.0003
E0	1
Ka	0.8
Kh	0.0016
Ko	0.0031
TF	10
TD	0.176
TA	0.7
F0	10
W	2.56
FOV	1

TABLE 2

Clock Cycles Taken by Each Method

	Clock Cycles	
Resolution	Self-Organized Contours	Exhaustive Scan
256 × 256	1428502	93506
640 × 480	1417473	4811850
800 × 600	1433933	7593329
1024 × 1024	1437637	16837443

It can be seen that the time taken by self-organized contours does not depend on image size. Although self-organized contours are slower than exhaustive image scan for low-resolution images (256 × 256), they go faster in high resolution. This improvement is particularly notable in applications that require high-resolution images. Note that the choice of color based filters, gradient filter or normal distribution for modeling color are application dependent and must be chosen by user. We utilized these particular features, because they are effective enough for finding a hand, face, or lip contour. Figure 8 shows the emergence of the contour in a completely different test image. Because this is a gray-scale image, we used intensity instead of color and defined brighter areas as food. In general, the choice of suitable features is a big challenge in computer vision. However, given proper features, our method extracts the shape contour of the desired items more efficiently than exhaustive search.

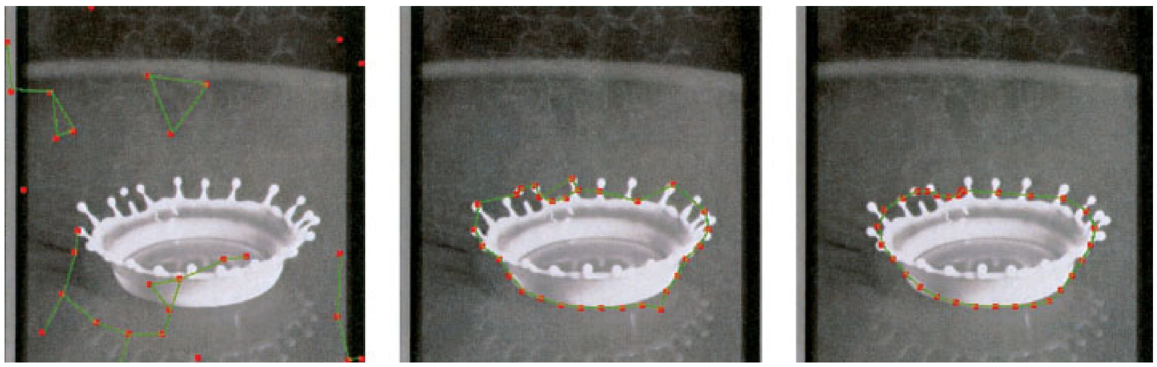
7. CONCLUSION

We proposed a novel method for contour initialization in digital images. Our approach was based on a swarm intelligence framework with agents possessing simple sensory motor capabilities. The swarm algorithm was inspired from animals' behavior, particularly fish schools and ant colonies. Based on some real images, self-organized and emerged contours were illustrated. The results indicated a quite good approximation of the underlying shapes in the image. Experiments revealed that this method outperforms exhaustive image search in high-resolution images. Therefore, it can effectively be applied in real-time applications that deal with large images and where fast processing is demanded. The key to this fast performance is taking advantage of the group behavior of the agents, each of which is a very simple and light-weight local filter.

Similar to other swarm techniques such as PSO [16,17], our proposed method has several parameters to be set manually before its execution. However, we normalized certain

© 2006 Wiley Periodicals, Inc.
DOI 10.1002/cplx

FIGURE 8



The evolution of the contour around the splash in three iteration steps. Left to right: 1, 175, and 250. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

equations so that the parameters become as invariant as possible. Therefore, the same parameter set listed in Table 1, except for T_F and F_0, can lead to satisfactory results for different images. For instance, by introducing variable R in definition of the forces, the overall speed of each agent becomes proportional to the scale of the image. This results in the same behavior in different scales. A similar normalization is used in self-assembly phase to measure the relative distance of the agents. The fact that the proposed values lead to a convergence in a relatively fixed amount of iteration suggests that the temporal parameters, E_0 and D, need no or little change in different images as well.

The parameters T_F and F_0 , which adjust the sensitivity of the feature detector, are the only ones that really need a deeper tuning. In our particular way for defining food, these parameters determine the threshold in the image gradient for extracting edges. Of course, this factor is application dependent in all computer vision tasks and has to be obtained empirically.

Because of the local nature of the food definition, any local filter in image processing domain that is suitable for detection of desired item can be used to represent the food source. This will not require touching the architecture of the agents or significantly changing the parameters of our system (excluding T_F and F_0). Such simple filters have successfully been used in important applications of computer vision tasks such as face detection [29,30]. Nevertheless, there are items that are too complex to be detected by local filters and require global information that the current model cannot support.

Therefore, endowing this work with top-down abilities can be a promising path for the future. This information may bias contour emergence toward specific shapes (e.g., oval, circular, or rectangular) or orientation. Moreover, communication of agents in macro (contour) level may add additional constraints by coordinating their relative position, scale, and orientation. The latter extension, for instance, can be useful in a facial feature tracking scenario, where a set of contours may cooperate to localize them.

Nevertheless, in order to avoid even this slight search over the proposed parameter values and to achieve a fully autonomous system, these parameters could be learned by the agents themselves. Therefore, adding a learning component to the architecture of these agents is a major path for the future of this work. There have been attempts for learning these parameters by artificial life community [31–33] for achieving natural aggregation behavior such as flocking, herding, or schooling. The difficulty in these works is problem is quantifying the quality of the aggregation [33]. Fortunately, in our domain there exists quantitative performance measure that can guide parameter learning. For instance, using a few manually marked contours in training images, agents can learn in supervised manner that what parameters takes them to food in the shortest time.

Finally, uniform distribution of agents is the best guess that currently our method can make. Although this distribution is more likely to happen, it is not always the case. For instance, it is generally more desired to have dense population in areas with high curvature. Other heuristics can be also added to get more accurate distributions when needed. An easy way to increase population density is reproduction of agents. Therefore, such guided reproduction can improve accuracy of contour representation.

REFERENCES

1. Bodor, R.; Jackson, B.; Papanikolopoulos, N. Vision-Based Human Tracking and Activity Recognition; Proceedings of the 11th Mediterranean Conference on Control and Automation, 2003.
2. Yuille, A.L.; Cohen , D.S.; Hallinan, PW. Feature Extraction from Faces Using Deformable Templates; Proceedings of CVPR'89, 1989; pp 104–109.
3. Kass, M.; Witkin, A.; Terzopoulos, D. Snakes: Active contour models. Int J Comput Vision, 1987, 1(4), 321–331.
4. Vermaak, J.; Godsill, S.; Pérez, P. Monte Carlo filtering for multi-target tracking and data association. IEEE Trans Aerospace Electronic Systems 2004, 41(1), 309–332.
5. Colmenarez, A.; Frey, B.J.; Huang, T.S. Detection and Tracking of Faces and Facial Features; International Conference on Image Processing, Kobe, Japan, 1999, pp 657–661.
6. Darrell, T.; Gordon, G.; Woodfill J.; Harville, M, A VirtualMirror Interface using Real-time Robust Face Tracking, Proceedings of the Third International Conference on Face and Gesture Recognition, Nara, Japan, 1998.
7. Bonabeau, E.; Dorigo, M.; Theraulaz, G. Swarm Intelligence: From Natural to Artificial Systems; Oxford University Press: New York, 1999.
8. Di Caro, G.; Dorigo, M. AntNet: Distributed stigmergetic control for communications networks. J Artif Intell Res 1998, 9, 317–365.
9. Gambardella, L.M.; Taillard, E.; Agazzi, G. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In: New Ideas in Optimization; Corne, D.; Dorigo, M.; Hlover, F., Eds.; McGraw-Hill: London, 1999; pp 63–76.
10. Matsuyama, T.; Hwang, V. SIGMA: A Knowledge-Based Aerial Image Understanding System; Plenum Press: New York, 1990.
11. Broggi, A.; Cellario, M.; Lombardi, P.; Porta, M. An evolutionary approach to visual sensing for vehicle navigation, IEEE Trans Indust Electron 2003, 50(1), pp 18–29.
12. Bovenkamp, E.G.P.; Dijkstra, J.; Bosch, J.G.; Reiber, J.H.C. Multi-agent segmentation of IVUS images. Pattern Recognition 2004, 37, 647–663.
13. Hamarneh, G.; McInerney, T.; Terzopoulos, D. Intelligent agents for medical image processing. In: Medical Image Computing and Computer-Assisted Intervention; Niesen, W., Viergever, M., Eds.; Springer: Utrecht, The Netherlands, 2001; pp 66–76.
14. Yanai, K.; Deguchi, K. An Architecture of Object Recognition System for Various Images Based on Multi-Agent; Proceedings of the International Conference on Pattern Recognition, 1998; Vol. I, pp 643–646.
15. Zhou, Q.; Parrott, D.; Gillen, M.; Chelberg, D.M.; Welch, L.R. Agent-based computer vision in a dynamic, real-time environment. Pattern Recognition 2004, 37(4), 691–705.
16. Mirzayans, T.; Parimi, N.; Pilarski, P.; Backhouse, C.; Wyard-Scott, L.; Musilek, P. A swarm-based system for object recognition, Neural Network World 2005, 15(3), 243–255.
17. Omran, M.G.; Engelbrecht, A.P.; Salman A.A. Particle swarm optimization method for image clustering. Int J Pattern Recog Artif Intell 2005, 19(3), 297–321.
18. Rodin, V.; Benzinou, A.; Guillaud, A.; Ballet, P.; Harrouet, F.; Tisseau, J.; Le Bihan, J. An immune oriented multi-agent system for biological image processing, Pattern Recognition 2004, 37(4), 631–645.
19. Germond, L.; Dojat, M.; Taylor, C.; Garbay, C. A cooperative framework for segmentation of MRI brain scans. Artif Intell Med 2000, 20(1), 77–93.
20. Spinu, C.; Garbay, C.; Chassery, J.A Cooperative and adaptive approach to medical image segmentation. In: Lecture Notes in Artificial Intelligence; Barahona, P., Stefanelli, M., Wyatt, J., Eds.; Proceedings of the 5th European Conference on Artificial Intelligence in Medicine; Springer-Verlag, 1995; pp 379–390.
21. Liu, J.; Tang, Y.Y. Adaptive image segmentation with distributed behavior based agents. IEEE Trans Pattern Analysis Machine Intell 1999, 6, 544–551.
22. Camazine, S.; Deneubourg, J.-L.; Franks, N.; Sneyd, J.; Theraulaz, G.; Bonabeau, E. Self-Organization in Biological Systems. PrincetonUniversity Press, Princeton, 2001.
23. Detrain, C. Field study on foraging by the polymorphic ant speciespheidole pallidula. Insectes Sociaux 1990, 37(4), 315–332.
24. Saffre, F.; Furey, R.; Kraft, B.; Deneubourg, J.L. Collective decision-making in social spyders: Dragline-mediated amplification process actsas a recruiting mechanism. J Theor Biol 1999, 198, 507–517.
25. Partridge, B.L. The structure and function of fish schools. Sci Am 1982, 246, 90–99.
26. Anderson, C.; Theraulaz, G.; Deneubourg, J.L. Self-assemblages in insect societies. Insectes Sociaux 2002, 49, 99–110.
27. Lioni, A.; Sauwens, C.; Theraulaz, G.; Deneubourg, J.L. Chain formation in Oecophylla longinoda. J Insect Behav 2001, 15, 679–696.
28. Hölldobler, B.; Wilson, E.O. The Ants; Harvard University Press: Cambridge, MA, 1990.
29. Reveret, L.; Benoit, C. A new 3D lip model for anal-ysis and synthesis of lip motion in speech production. In: Proceedings of the Second ESCA Workshop on Audio-Visual Speech Processing, Terrigal, Australia, 1998.
30. Yang, J.; Lu, W.; Waibel, A. Skin color modeling andadaptation. In: Proceedings of the 3rd Asian Conference on Computer Vision, Vol. 2, 1998; pp 687–694.
31. Reynolds, C.W. An Evolved, Vision-Based Behavioral Model of Coordinated Group Motion. From Animals to Animats 2; Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB92), Meyer, J.A., Roitblat, H.L., Wilson, S.W., Eds.; MIT Press: Cambridge, MA, 1992; pp 384–392.
32. Werner, G.M.; Dyer, M.G. Evolution of Herding Behavior in Artificial Animals. From Animals to Animats 2; Proceedings of the Second International Conference on Simulation of Adaptive Behavior (SAB92), Meyer, J.A., Roitblat, H.L., Wilson, S.W., Eds.; MIT Press, Cambridge, MA, 1992, pp. 393–399.
33. Zaera, N.; Cliff, C.; Bruten, J.; (Not) Evolving Collective Behaviours in Synthetic Fish. From Animals to Animats 4; Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviour, MIT Press: Cambridge, MA, 1996; 635–636.

DOI 10.1002/cplx