Fast Initialization of Active Contours

Towards Practical Visual Interfaces for Human-Robot Interaction

Hossein Mobahi, Majid Nili Ahmadabadi, Babak N. Araabi
Control and Intelligent Processing Centre of Excellence, Robotics and AI Lab,
Department of Electrical and Computer Engineering, University of Tehran
and
School of Cognitive Sciences, IPM
Tehran, Iran
hmobahi@acm.org , mnili@ut.ac.ir , araabi@ut.ac.ir

Abstract— **The field of robotics is currently undergoing a change toward creation of robots that can naturally interact with humans. For achieving this, interactive robots must be endowed with natural interfaces that can sense and respond in real-time. Vision can provide handy information for this purpose by detecting and tracking human limbs to analyze gestures, actions and even emotions. However, real-time processing of visual information is a challenging bottleneck. In this paper, we will introduce a novel method, namely "self-organized contours", that can distinctly accelerate contour initialization, which is the slowest phase in visual tracking. Although the proposed method is general-purpose, it allows immediate initialization of active contours due to its similarity with snake structure. The proposed method is inspired from group behavior in insects and animals, particularly fishes.**

Keywords- real-time vision; active contours; multi-agent systems; human-robot interaction; self-organization; swarm intelligence.

1 INTRODUCTION

The field of robotics is currently undergoing a change. While in the past, robots were dominantly employed in industrial applications for purposes such as manufacturing and transportation, a new generation of interactive robots has recently begun to emerge. For instance, interactive robots are being developed to provide the elderly with assistance in their home or teaching children in elementary schools. Due to the endowment by natural interfaces, interactive robots can be easily instructed or be understood by people unfamiliar to robotics. Vision can provide handy information within natural human-robot interaction framework. For instance, people can naturally communicate with robot through facial gestures such as yes and no. Human's intention and attention can be understood visually by analyzing human's pointing gesture [11] or estimating gaze direction [2]. Moreover, robot's behavior may be reinforced by user's emotional state estimated from facial expressions.

A robot generally sees the environment as a sequence of digital images. There exists a high spatio-temporal correlation between successive images, which can be used to speed up process in images, increase robustness of the system and provide motion information. Therefore, most robot vision systems rely on visual tracking rather than single image analysis. Even when a feature should be interpreted statically without considering its motion, tracking can still be beneficial for fast localization of that feature in the image sequence. This is of special importance in interactive robots due to their need for real-time performance. In summary, for developing a visual interface in an interactive robot two issues have to be addressed, namely, real-time performance and tracking scheme.

To demonstrate interactive behaviors, the robot must be able to work in real-time. Achieving this goal, especially within the visual interface context, is not easy. This is because a large amount of data captured by vision sensors must be processed continually. Particularly, initialization phase is among the slowest tasks in visual tracking due to the need for an exhaustive search over the entire image for finding Region of Interest (ROI). Unfortunately, initialization is not limited to time that a limb first appears. In fact, the tracking process may have to be reinitialized repeatedly, due to the accumulation of tracking error over time and causing a dramatic drop of tracking confidence [7]. Apparently, any acceleration in initialization can be a notable contribution. If achieved, a robot can interactively respond to human's actions even though its tracking module is reinitialized frequently.

The other issue to be addressed in visual interface design for an interactive robot is the tracking scheme. Human limbs are highly deformable and articulated objects that can exhibit different postures and motions. Among a number of available tracking tools, active contour or snake [9] has been very popular and effective for capturing the shape of human's limbs as well as tracking their motion. For instance, snakes have been successfully applied to the analysis of facial features [8, 13, 14, 15] and hand posture estimation [16, 6]. A snake is an energy minimizing spline represented by a finite set of control points. From a given starting point, the snake deforms itself to conform to the nearest salient contour. The initial location of a snake must be provided by other processing.

In this paper, we will introduce a novel method, namely "Self-Organized Contours" (SOC), for fast localization of ROI. Although the proposed method is general-purpose, it allows an immediate initialization of an active contour due

to its similarity with snake structure. Unlike traditional scanning of the whole image for finding and labeling ROI, SOC only explores a small fraction of the image. The proposed method is inspired from group behavior among insects and animals, particularly fishes; simply a swarm of agents that attempt to survive by finding food. While local search by agents leads to partial exploration of the environment, their interaction self-organizes a contour about each ROI.

Applying artificial life and multi-agent ideas to the domain of image processing and specifically feature extraction is not new. In a similar work, Liu and Tang [10] used a different artificial life agent based systems to segment and analyze Chinese documents. Their approach utilizes evolutionary autonomous agents that can self-reproduce, diffuse and cease to exist during the course of interacting with a digital image environment. The evolutionary nature of their proposed agents lies in the way in which the generations of autonomous agents are selected and replicated. The fitness function measures how long it takes the agent to find a feature pixel.

Their approach produces a dense feature map that has two disadvantages for real-time applications. First, it requires a large number of agents and consequently execution time becomes long. Second, it requires a complicated post-processing for shape representation. Determining the shape of feature points is difficult in their approach because a boundary tracking in pixel level is required afterwards and this is time-consuming. Even if such a boundary tracking could be performed, it would face other problems due to the discontinuities. Another problem is thickness of features which makes boundary tracking task take a longer time. In contrast to Liu's approach, our approach represents contours by a few control points, which requires less number of agents and runs faster. Moreover, instead of boundary tracking, a fast and simple algorithm is proposed for forming contours from their potential control points.

This paper is organized as follows. In section 2 we will review self-assembly behavior in fish schools by discussing about local rules that emerge schooling. Section 3 proposes a mathematical model for the discussed rules within a multi-agent framework. These rules are expected to self-organize groups of agents about food regions. Section 4 demonstrates the application of the proposed algorithm on real images for fast localization of faces, hands and lips. It justifies the shorter execution time of the proposed algorithm against scanning the whole image. Finally the paper ends with a conclusion that summarizes the proposed method and its achievements.

2 Biological Inspiration

The SOC takes inspiration from recent studies in swarm intelligence, a novel approach to the design and implementation of intelligent systems inspired by the effectiveness and robustness observed in social insects and in other animal societies [4]. A key role in swarm intelligence is played by the phenomenon of self-organization, whereby global level order emerges in a system from the interactions happening among the system's lower-level components. Moreover, these interactions are based only on local information, without reference to the global pattern [5]. Self-organization can be seen in different animal societies such as colonies of ants and bees, flocks of birds, and schools of fish. A particular form of self-organization is observed in some social insects and animals, which connect one to the other creating complex physical structures. This type of self-organization is referred to as self-assembling [1], and is one of the key elements of our SOC.

A good instance of self-assembling behavior in animal societies can be seen in fish schools. Schooling is where a group of fish acts as a single organism than a collection of individuals. There is a highly coordinated structure, yet no leader or external stimulus that prompts the form of this polarized configuration. Schooling would appear to obey the rules of a distributed model, i.e. each individual applies the same set of simple behavioral rules. 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 [12].

Fish schools do not have a regular geometric form; the structure is loose and it results from each fish's applying a few simple behavior rules. This is similar to the nature of an active contour or snake in image processing where the deformable contour changes as a result of local interactions happening 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 constraint among control points dictates a specific order in snake's movement and significantly limits its exploration ability. Inspired from animal societies, no fixed mates are forced for the agents in our model; mating emerges through agents' local interactions.

Real-time performance is a remarkable motivation for proposing our method. We want the agents to explore the environment (image) with minimal number of steps. Similar cost minimization exists in fish schools too. If a member of the school finds food, the other members can take advantage of the find [3]. Opportunism in fish is triggered when they directly observe the successful fish. However, we will go one step further and enable our agents to communicate with each other from long distances as well. Unlike real world where distance is considered a communication cost, it is has no effect in the simulated world of our agents. By incorporating opportunism behavior and communication ability, agents can cooperatively find food in a reduced time.

Allowing agents to communicate and report food location may result in parasitic behavior that decreases individuals' and consequently swarm's exploration ability. In the extreme case, this may cause the swarm to collapse over the successful agent. To avoid this situation, another intention should be introduced to keep the balance between opportunism and individual-based exploration. In fish schools, fish maximize the searching area by remaining barely in the sight of one another [12]. This can be modeled by repulsive forces emitting from agents for achieving broad exploration.

3 COMPUTATIONAL MODEL

To model the behaviors discussed in pervious section, first a swarm of agents should be created. The agents have simple sensory motor and communicative capabilities. In this paper, we focus on 2D environments, i.e. planar images. However, the proposed model can be applied to 3D images as well, because it entirely relies on vector operations. Initially a number of agents are distributed over the image plane at random locations. This plane acts as the environment where the agents inhabit. Regions that correspond to features of interest, e.g. edges, are treated as food. Formally, food is a scalar function of the environment that responds in locations where features of interest exist. Note that food can be a matter of degree because the feature quality may differ from location to location, e.g. edge strength. The behavior of each agent depends on three state variables, namely position, velocity and energy.

The energy of an agent keeps it alive and it is supplied through nourishing from environmental foods. Once this energy reaches zero the agent dies. We wish, after self-organization of contours, that agents are scattered over feature regions only. In our model, death possibility is helpful when an agent is trapped in a non-nutritious region. If this agent joins the formation process of a contour, it corrupts the whole resulted shape; thus, it is better to be killed. An agent is trapped when other agents surround it such that it cannot move in any direction, due to repulsive forces among agents (this force will be introduced later in this section).

Let us denote the energy of the *i*'th agent by scalar $E_i(t)$. In the beginning, this energy is initialized to a non-zero value. As (1) shows, the energy decreases continually over time and it resets when the agent gets in a nutritious region. Here, d is a decay constant.

$$E_i(t) = E_0 \quad if \quad t = 0 \text{ or } Food(\mathbf{X}_i) > T_F$$
$$\frac{dE_i(t)}{dt} = \begin{cases} -d & ; \quad E_i(t) > 0 \\ 0 & ; \quad otherwise \end{cases} \tag{1}$$

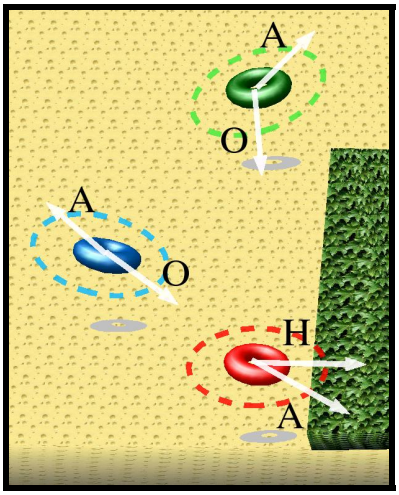

Figure 1. Forces and Agents' FOVs. The agent at the bottom has discovered a nourishing region.

The two other state variables determine the motion of agent in the environment. Two-dimensional position and velocity vectors for the *i*'th agent are denoted by $\mathbf{X}_i$ and $\mathbf{V}_i$, respectively. These state variables are influenced by the superposition of attractive and repulsive force fields emitted from the environment and other agents. Agent's trajectory is evolved over time according to the first order differential equation described in (2).

$$\begin{aligned} \mathbf{F}_i &= K_a\mathbf{A}(\mathbf{X}_i) + K_h\mathbf{H}(\mathbf{X}_i) + K_o\mathbf{O}(\mathbf{X}_i) \\ \dot{\mathbf{V}}_i &= \alpha(\text{sat}(\mathbf{F}_i) - \mathbf{V}_i) \\ \dot{\mathbf{X}}_i &= \mathbf{V}_i \end{aligned} \tag{2}$$

The superposition of partial forces, denoted by $\mathbf{F}_i$, is comprised of three weighted components standing for *avoidance*, *hunting* and *opportunism* force vectors. The computed force is filtered before altering velocity vector to reduce noise in motion and clearly indicate activity level of the agent. The smoothening degree of the filter is equal to *1-*α. For the sake of stability, we bound the velocity of agents is by a *saturation* function. In the rest of this section we describe each of the mentioned forces separately in more details. When reading the following subsections, you may want to refer to Fig. 1. It is a schematic depiction of the mentioned forces that may ease their understanding.

3.1 Hunting

In order for agents to survive, they must sense the environment and look for food. Similar to insects and animals, our agents have a limited field of view (FOV) to sense the environment. Limited FOV decreases computational load of the algorithm. The regions out of FOV are all ignored completely. Nevertheless, the inner regions are not all the same; the farther a point is, the less impact it has on agent's behavior. Given a point within the FOV, we formulate the magnitude of hunting force such that it is 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 (3) by integrating all force vectors that fall inside the FOV.

$$\mathbf{H}(\mathbf{X}_i) = \int_{\mathbf{s}\in FOV} \frac{\mathbf{s}}{\|\mathbf{s}\|} \frac{\text{Food}(\mathbf{X}_i + \mathbf{s})}{\|\mathbf{s}\|} d\mathbf{s} \tag{3}$$

3.2 Avoidance

Avoidance behavior makes agents inhabit almost uniformly over feature points. Uniform distribution of contour points often results in more accurate reconstruction of the underlying shape. As shown in (4), each agent finds the location of its nearest neighbor through communication. The agent then avoids it by the repulsive force emitted from the neighbor.

$$\begin{aligned} k &= \text{Arg} \min_{j \neq i} \left\| \mathbf{X}_i - \mathbf{X}_j \right\| \\ \mathbf{A}(\mathbf{X}_i) &= \frac{\mathbf{X}_i - \mathbf{X}_k}{\left\| \mathbf{X}_i - \mathbf{X}_k \right\|^2} \end{aligned} \tag{4}$$

3.3 *Opportunism*

It may take a long time for an agent to discover food if it relies merely on its own exploration capabilities. To accelerate food localization and contour formation, opportunism behavior is incorporated in agents. An opportunistic agent must first target a rich agent. This is achieved by communicating with other agents to know about their locations and their food qualities. Furthermore, agents in less inhabited regions are favored for approaching a uniform distribution and prevent from collapsing of the swarm (totally or partially) onto the target. This combination is formulated in (5) where k is the target agent of agent i. The contribution of the two criteria to the final decision is determined using a weight factor W.

$$k = \text{Arg} \max_{j \neq i} (\text{Food}(\text{X}_j) + W \min_{p \neq j} \|\text{X}_p - \text{X}_j\|) \tag{5}$$

Once the target agent is found, the measured value that causes its selection can be directly used as the magnitude of opportunism force. However, reminding that food quality may vary in different locations, opportunism can still cause parasitic behavior. This may happen by pushing agents to leave their own food and land about the region discovered with the highest quality. To prevent this trend, opportunism is triggered only when an agent itself does not have access to food. In other words, opportunism occurs when the food quality for an agent is below a threshold like F_0.

$$\mathbf{O}(\mathbf{X}_i) = \begin{cases} \dfrac{\mathbf{X}_k - \mathbf{X}_i}{\|\mathbf{X}_k - \mathbf{X}_i\|}[W \min_{p \neq j} \|\mathbf{X}_p - \mathbf{X}_j\| + \text{Food}(\mathbf{X}_k)] & ; \quad \text{Food}(\mathbf{X}_i) < F_0 \\ 0 & ; \quad \text{otherwise} \end{cases} \tag{6}$$

3.4 *Contour Formation*

The discussed behaviors self-organize the swarm from random initialization toward a specific order, which is settling about feature points, .e.g. image edges. Once the swarm reaches equilibrium, agents are ready to explicitly define the contours that they represent. This is achieved by connecting neighbor agents to each other in the right order. Equilibrium state is perceived by continually measuring the activity of the swarm until it drops below a threshold. At that time contours are formed and simulation is terminated. The activity, denoted by A, is estimated from the average magnitude of motion as shown in (7), where N is the number of survived agents at time t.

$$A(t) = \frac{\sum_{i=1}^{N} \|\dot{\mathbf{X}}_i(t)\|}{N} \tag{7}$$

Figure 2. Self-Organization of Contours about Face and Hand Shown in Three Iteration Steps (Left to Right): 1, 175, and 250.

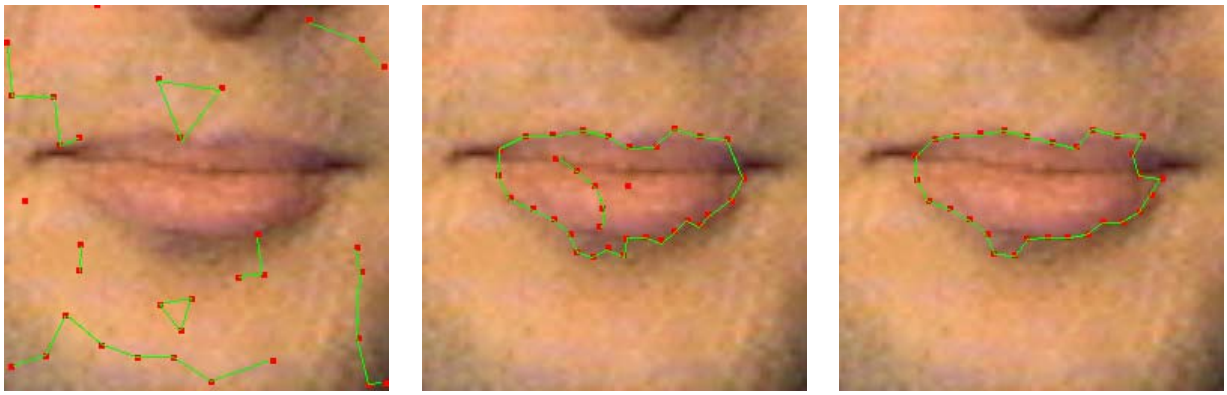

Figure 3. Self-Organization of Contours about Lips Shown in Three Iteration Steps (Left to Right): 1, 175, and 250.

Since we only are only interested in simple and closed contours, each agent must exactly have two neighbors. An agent selects its neighbors based on two factors, namely distance and angle. For distance, closer agents and for angle those that form flatter connections are proffered for becoming neighbor candidates because they form smoother contours. Formally, given an agent with index *i*; it first adds its nearest agent *j* to its neighbor 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 second neighbor. We will 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 neighbor 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 the angle formed between two connections. The neighbor selection procedure is mathematically described in (8).

$$
\begin{aligned}
& j = \operatorname{Arg}\operatorname*{Min}_{p \neq i} \left\| \mathbf{X}_i - \mathbf{X}_p \right\| \\
& S_{i,j} = \{ p \mid p \neq i \neq j \wedge (\left\| \mathbf{X}_p - \mathbf{X}_i \right\| < T_D) \wedge (\frac{(\mathbf{X}_j - \mathbf{X}_i)^T (\mathbf{X}_p - \mathbf{X}_i)}{\left\| \mathbf{X}_j - \mathbf{X}_i \right\| \left\| \mathbf{X}_p - \mathbf{X}_i \right\|} < T_A) \} \\
& S_{i,j} \neq \{\} \rightarrow k = \operatorname{Arg}\operatorname*{Min}_{p \in S_{i,j}} \left\| \mathbf{X}_i - \mathbf{X}_p \right\|
\end{aligned}
\tag{8}
$$

Occasionally it may happen that neighborhood connections become asymmetric. This happens when agent *i* chooses agent *j* as its neighbor while *j* is not chosen as the neighbor of *i*. Asymmetry may result in nodes with more than two neighbors while only two neighbors per agent are allowed for forming a simple closed contour. Therefore, a mechanism is required to detect asymmetries and eliminate them. The answer is formally shown in the equation (9), where N_a denotes the set of neighbors chosen by agent *a*.

$$
i \notin N_j \wedge j \in N_i \rightarrow N_i = (N_i - \{j\}) \cup \operatorname{Arg}\operatorname*{Min}_{p \in S_{i, N_i - \{j\}}} \left\| \mathbf{X}_i - \mathbf{X}_p \right\| \tag{9}
$$

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.

4 Experimental Results

To evaluate the competence of our proposed method against exhaustive scan of image, we carried out some experiments on real images for localizing faces, hands and lips. We used color images taken by BenQ 2300 digital camera with different resolutions. Feature maps were obtained by a simple color-based filter followed by gradient computation. Color classes were modeled using Gaussian distributions in RBG space, e.g. skin or lip color. Feature map computation was treated as a preprocessing phase on the whole image for exhaustive scan. However, in our method the features were computed locally once an agent meets that point of the environment.

TABLE 1. Swarm Parameters

Parameter	Value
α	0.1
D	0.0003
E0	1
Ka	0.8
Kh	0.4
Ko	0.8
TF	10
TD	45
TA	0.7
F0	10
W	0.1
FOV	1

For testing our proposed method, initially 35 agents were distributed over the image at random locations. The time-evolution of agents on two test images are shown in Fig. 2 and Fig. 3. Although it was not necessary to create connections before reaching equilibrium, we did it during intermediate steps merely to visualize gradual formation of the contours. It can be seen that contours could self-organize about ROIs in at most 250 iterations. Obviously, contours could not capture the underlying shapes perfectly. However, this is not a problem because the self-organized contours are just initial contours that are supposed to be refined by other tools. This can be achieved, for instance, through evolution of a snake [9] when minimizing first order derivative of the obtained contour.

The temporal behavior of force magnitudes as well as their superposition is shown in Fig. 4. It is apparent that after 250 iterations the forces exhibit almost a monotone behavior and the activity remains very low. We did not spend much time on tuning agents' parameters. Therefore, even less number of iterations might be possible. Parameter values used in our experiments are listed in Table 1. Some parameters had to be changed for different resolutions. The shown table corresponds only to the parameter set used in 256x256 images. Note that the unit of T_F and F_0 is intensity and of T_D and FOV is pixel.

In traditional pixel by pixel scan, 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 methods indicates that self-organized contours method scans only 13% of image pixels. This is a remarkable achievement in image scanning, comparing with exhaustive search of the image. Since the limited FOV of agents and their communication ability keeps the computation cost of each iteration light-weight, self-organized contours outperform exhaustive scan in with respect to 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 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).

TABLE 2. CLOCK CYCLES TAKEN BY EACH METHOD

Resolution	Clock Cycles	
	Self-Organized Contours	*Exhaustive Scan*
256x256	1428502	93506
640x480	1417473	4811850
800x600	1433933	7593329
1024x1024	1437637	16837443

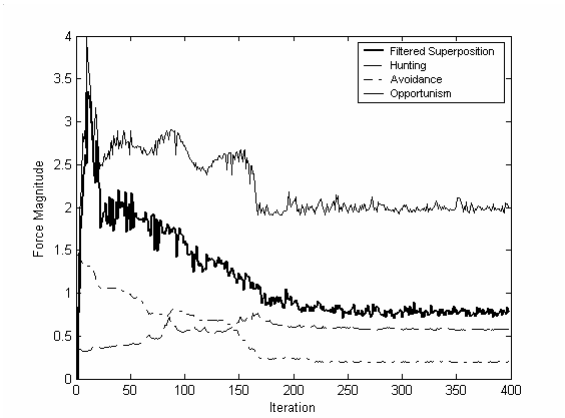

Figure 4. Behavior of Forces over Time

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.

It can be seen that the time taken by self-organized contours is not dependent on image resolution. Although self-organized contours are slower than exhaustive image scan for low resolution images (256x256), they are faster in high resolution. This progress is particularly useful in HRI applications like facial expressions analysis, where high resolution images are required.

5 CONCLUSION AND FUTURE WORKS

We proposed a novel method for contour initialization with real-time performance. As justified by experiments, it outperforms traditional exhaustive image scan. Therefore, it can effectively be applied to visual interfaces of interactive systems where contour initialization must happen as quickly as possible. Our approach was based on a multi-agent framework where agents had simple sensory motor and communication capabilities.

Inspired from behaviors observed in fish schools, we developed a mathematical model for our agents so that they could quickly self-organize themselves about feature points in image. We discussed the how contours are explicitly created from agents at the equilibrium. Our experiments on real images demonstrated that self-organized contours could capture the underlying shapes, guided through food definition, with a good accuracy. Analysis of the execution time also showed that the proposed method works faster than exhaustive image scan in high-resolution images.

The proposed method has several parameters that must be set manually before its execution. Tuning these parameters is possibly time-consuming. Therefore, one important path for future work on this method is automating parameter adjustment. This may be achieved by incorporating learning into the architecture of the agents. 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.

6 REFERENCES

[1] C. Anderson, G. Theraulaz, and J.L. Deneubourg, "Self-assemblages in insect societies", *Insectes Sociaux*, 49:99, 110, 2002.

[2] R. Atienza, A. Zelinsky, "Intuitive Human-Robot Interaction through Active 3D Gaze Tracking" , *11th Int. Symposium of Robotics Research, Siena*, Italy, Oct 2003.

[3] R.G. Bill and W.F. Hermkind, "Drag reduction by formation in spiny lobsters", *Science*, 193, pp. 1146-1148, 1976.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Swarm Intelligence: From Natural to Artificial Systems", *Oxford University Press*, New York, NY, 1999.

[5] S. Camazine, J.L. Deneubourg, N. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, "Self-Organization in Biological Systems", *Princeton University Press*, Princeton, NJ, 2001.

[6] R. Cipolla, P.A. Hadfield, and N.J. Hollinghurst, "Uncalibrated Stereo Vision with Pointing for a Man-Machine Interface, *IAPR Workshop on Machine Vision Applications*, Yokohama, Japan, December 1994.

[7] A. Colmenarez, B.J. Frey, T.S. Huang, "Detection and Tracking of Faces and Facial Features", *International Conference on Image Processing*, 1999: pp. 657-661, Kobe, Japan, 1999.

[8] C. Fan, M. Johnson, C.H. Messom, A. Sarrafzadeh, "Machine Vision for an Intelligent Tutor", *Proceedings of IEEE International Conference on Computational Robotics and Autonomous Systems*, ISSN 0219-6131, 2003.

[9] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes - Active Contour Models", *International Journal of Computer Vision*, 1(4): 321-331, 1987.

[10] J. Liu, Y.Y. Tang, "An evolutionary autonomous agents approach to image feature extraction", *IEEE Transactions on Evolutionary Computation*, Vol.1, No. 2, pp. 141.158, June 1982.

[11] K. Nickel, R. Stiefelhagen, "Recognition of 3D-Pointing Gestures for Human-Robot-Interaction", *Proceedings of Humanoids 2003, Karlsruhe*, Germany, 2003.

[12] B.L. Partridge, "The structure and Function of Fish Schools", *Scientific American*, pp. 114-123, June 1982.

[13] S. Ramadan, W. Abd-Almageed, and C. Smith, “Eye Tracking using Active Deformable Models”, *Proceedings of the III Indian Conference on Computer Vision*, Graphics and Image Processing, India, 2002.

[14] D. Terzopoulus and K. Waters, "Analysis and Synthesis of Facial Image Sequences using Physical and Anatomical models, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(6):569–579, June 1993.

[15] B. Tiddeman, G. Rabey and N. Duffy," Synthesis and Transformation of Three-dimensional Facial Images", *IEEE Engineering in Medicine and Biology*, pp. 64-69, Vol. 18, No. 6, Nov./Dec. 1999.

[16] Z. Wu, P. S. Aleksic, and A. Katsaggelos, "Lip Tracking for MPEG-4 Facial Animation", *International Conference on Multimodal Interfaces (ICMI)*, pp. 293-298, Pittsburgh, October 2002.