Applied Soft Computing 13 (2013) 3118–3129

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A Swarm Intelligence inspired algorithm for contour detection in images

Ulrich Kirchmaier*, Simon Hawe, Klaus Diepold

Technische Universität München, Arcisstr. 21, 80290 Munich, Germany

ARTICLE INFO

Article history:
Received 31 December 2011
Received in revised form 2 July 2012
Accepted 11 July 2012
Available online 25 July 2012

Keywords:
Contour detection
Boundary detection
Image segmentation
Image processing
Swarm Intelligence

ABSTRACT

Swarm Intelligence uses a set of agents which are able to move and gather local information in a search space and utilize communication, limited memory, and intelligence for problem solving. In this work, we present an agent-based algorithm which is specifically tailored to detect contours in images. Following a novel movement and communication scheme, the agents are able to position themselves distributed over the entire image to cover all important image positions. To generate global contours, the agents examine the local windowed image information, and based on a set of fitness functions and via communicating with each other, they establish connections. Instead of a centralized paradigm, the global solution is discovered by some principal rules each agent is following. The algorithm is independent of object models or training steps. In our evaluation we focus on boundary detection as a major step towards image segmentation. We therefore evaluate our algorithm using the Berkeley Segmentation Dataset (BSDS) and compare its performance to existing methods via the BSDS benchmark and Pratt's Figure of Merit.

1. Introduction

Boundary Detection is commonly regarded as a major step towards Image Segmentation, which is a central problem in computer vision that has been studied for years. Image Segmentation again constitutes an important initial step for high-level tasks in computer vision, like object recognition, image analysis, and scene understanding. It pursues the goal of dividing an image into parts, regions, or objects, which are preferably self-consistent and of meaningful content. Application examples include separating objects considered as foreground from the background, or separating multiple objects from each other.

One of the main difficulties in the field of boundary detection and image segmentation arises from the fact, that not even human beings are able to uniquely and unambiguously decide and agree over a correct segmentation [1]. Different persons are likely to have different opinions on the correct solution to the segmentation problem of any image, especially involving natural images. This lack of an unique ground-truth complicates the comparison of results descending form different segmentation methods. Nevertheless, efforts were taken to establish a framework for objective evaluation of segmentation algorithms. A prominent example is the Berkeley Segmentation Dataset (BSDS) [1], which provides a huge collection of natural images along with human-marked proposals of ground-truth, as well as a precision-recall based measurement for comparison [2]. Fig. 1 shows three example images along with three different human ground-truth suggestions.

The plenty of solutions to the segmentation problem proposed in the literature descend from versatile fields, such as clustering [4], graph theory [5,6], region growing [7], optimization of an evaluation function like energy minimization [8], as well as image feature-based methods [9,10]. The majority of approaches found in the literature can be categorized into one or a subset of these fields, with each adding improved mechanisms to solve specific tasks.

Boundary or contour detection is often performed as an initial step for image segmentation, as boundaries naturally correspond to borders between objects. It describes the process of identifying and locating sharp discontinuities in image regions, which closely relates it to the task of edge detection. Mere edge detection, however, registers on the one hand any type of abrupt changes in image brightness, and misses smooth or subtle transitions of image brightness on the other hand. Regarding the goal of detecting boundaries, this causes false positives like textured regions, or single highlighted spots and true negatives like region boundaries of low contrast.

One recipe to overcome these drawbacks is to combine edge detection techniques with diverse soft computing approaches like Genetic Algorithms, or Fuzzy Logic, etc. [11,12]. Other methods combine oriented local image cues, like brightness, color, and texture descriptors with edge detection to get a global estimation. Such combination methods include multiscale cue combination [3], energy and probability weighting [13], or likelihood-methods [14].

* Corresponding author.
E-mail address: kirchmaier@tum.de (U. Kirchmaier).

1568-4946/$ – see front matter © 2012 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.asoc.2012.07.017

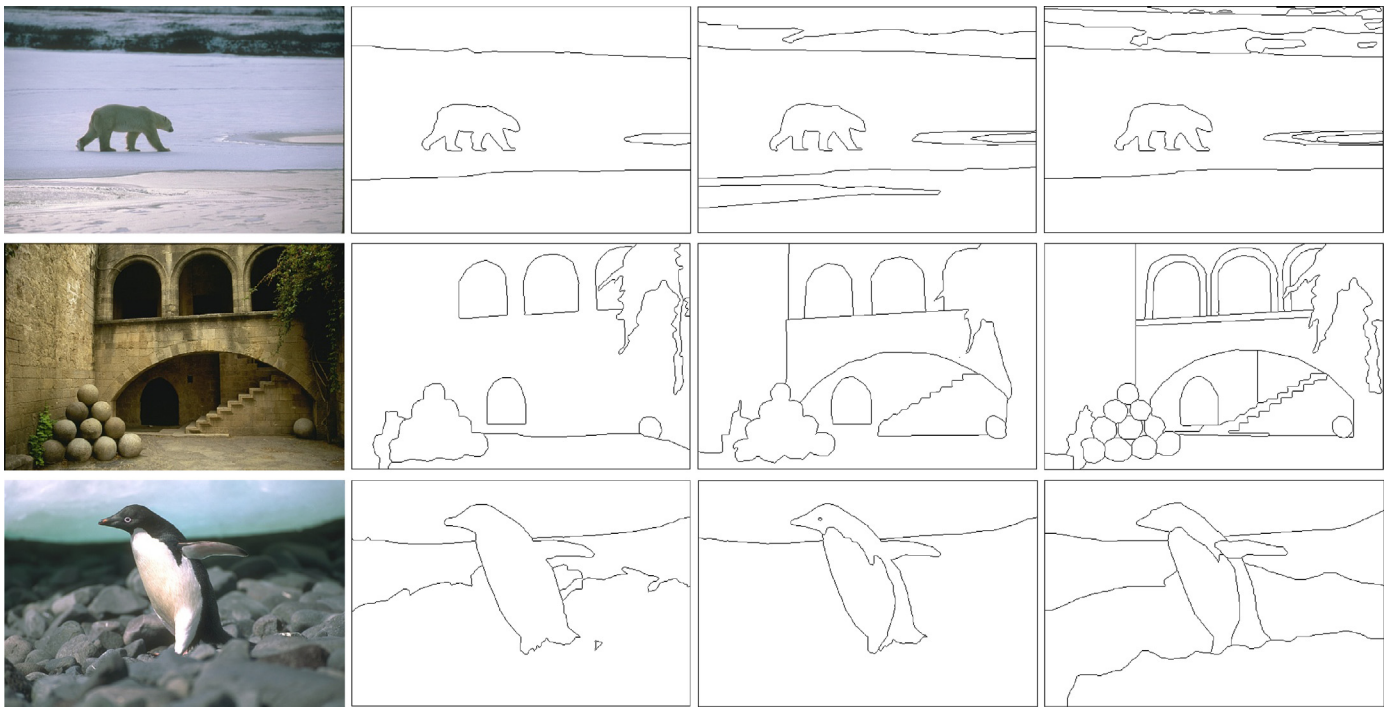

Fig. 1. Three sample images from the Berkeley Image Segmentation Dataset (BSDS 500) [3], each along with three different human-marked ground-truth images.

In this paper, we propose an algorithm that uses the concept of Swarm Intelligence to detect contours in images. We believe that the idea of interacting distributed agents to migrate from local information to global boundary estimates is a promising alternative to existing methods.

While a variety of Swarm Intelligence inspired algorithms already exists (refer to Section 2), their swarms often seek to find one global optimal solution in a search space, as these algorithms descend from the field of optimization.

However, it is our goal to detect a non-predefined number of boundaries with arbitrary positions, sizes, and shapes in an image, which all have to be considered as equal or even independent solutions. Therefore, we established a novel algorithm based on distributed agents being explicitly tailored to the demands of contour detection. The main contribution of this paper are the development of (1) a novel movement and positioning rule, along with (2) local information gathering mechanisms, and (3) a set of fitness functions and communication schemes.

This paper is organized as follows: Section 2 gives an overview on approaches in the fields of image segmentation and contour detection, which utilize Swarm Intelligence. Section 3 investigates the principles of Swarm Intelligence and Section 4 provides details on our algorithm. Section 5 shows results of our algorithm tested on the task of boundary detection in natural images.

2. Related work

In recent years, algorithms descending from the field of Evolutionary Computing, Genetic Algorithms, and also Swarm Intelligence are increasingly applied to the image segmentation problem. These optimization tools have mostly been used to improve the performance of important steps of segmentation algorithms. In [15,16], evolutionary and genetic algorithms were used to enhance the clustering process in segmentation. In [17], the authors combined high-level features generated with a visual attention model with low-level features to guide region growing algorithm, where the optimal thresholds of the region growing process were detected using the Particle Swarm Optimization (PSO) algorithm.

In [18], the PSO was applied to find the optimal fuzzy entropy thresholds to segment images into foreground and background. A related approach was introduced in [19], where PSO was used to tune thresholds in 2D-histograms, maximizing the entropy to segment infrared images.

SI was also utilized to improve Segmentation based on clustering approaches, which can get stuck in local optima, depending on their initialization. In [20], the authors developed a hybrid combination of the Ant Colony Optimization algorithm (ACO) and PSO to make K-Means clustering and Single Competitive Learning more independent from their initial cluster centers, and learning rate, respectively.

Besides deploying SI techniques to optimize specific steps in existing segmentation approaches, SI algorithms were also applied to the tasks of edge detection and contour detection, akin to our proposed algorithm.

The authors of [21] proposed ant-based correlation for edge detection. Their method is capable of performing feature extraction for edge detection and segmentation, generating less distortion in the presence of noise, as compared to classical edge detectors, like Sobel, Prewitt, and Canny.

In [22], a variant of the PSO algorithm is utilized for finding a proper edge detection filter size in noisy images. While combining edge filters of different sizes can improve the edge detection in noisy images, it is generally computationally expensive when applied to the entire image. Thus, the heuristic capabilities of PSO were used to adaptively decide, where on the image to use filters of various sizes. The evaluation of edge intensity and the proposed movement principle in this approach has analogies to our algorithm, as they divide the edge regions into sets and use averaged intensities. However, the major difference is the usage of discrete PSO instead of our communication scheme where the agents share their local information.

3. Principles of Swarm Intelligence

Swarm intelligence constitutes a fast-growing field with researchers from various disciplines from social and natural sciences to computational engineering engaged. In nature, phenomena of SI can be found in multiple examples, including fish schooling, ant colonies, bird flocking, animal herding, and bacterial growth, as well as human crowd behavior.

SI is based on self-organized individuals, generally called agents, whose actions and interaction add up to intelligent global behavior. This occurrence is commonly referred to as emergence. Thereby, a key point of SI is the absence of a centralized control instance. Instead, each agent acts according to some basic principles, which are

- *autonomy*, i.e. no centralized control,
- *local view*, i.e. incomplete knowledge of the environment and also no or little concern about the global behavior of the system,
- *movement*, i.e. an agent's possibility to discover its environment,
- *communication*, i.e. interaction between agents,
- *memory*, i.e. knowledge of prior conditions,
- *fitness function*, i.e. a measure to evaluate an agents current position or solution.

Since the first appearance of SI in a technical context [23], researchers utilized its principles on various applications, for optimization, and data analysis problems [24]. In recent years, SI-based methods proofed to be capable of managing complex systems and tasks, from route-planning and control of unmanned vehicles to mobile telecommunication networks.

While SI-based algorithms mostly descend from the field of optimization, they were also successfully applied to computer vision and image processing tasks, ranging from edge detection [25], over feature extraction and object recognition [26,27], to three-dimensional tracking in stereo-video [28]. All these algorithms offer robustness alongside low computational costs.

Two prominent and widely used examples for realizations of the SI-principles are the Particle Swarm Optimization (PSO) [29], and the Ant Colony Optimization ACO [30]. These two instances show the manifold possibilities of interpretation of the SI-concept. While they follow the same principles, their realizations illustrate significant differences. To give an example, the communication of the ACO algorithm is indirect and location-based, i.e. the ants leave a pheromone-trail at each location, which describes the position with respect to the task. On the opposite, the particles of PSO use a one-to-all broadcast-communication, which causes the other particles of the swarm to move closer to the currently best solution.

Algorithms of the SI-field have been already applied to image segmentation in various ways, e.g. for finding adaptive thresholds of regions using PSO [31], or on color segmentation via an ACO-based algorithm using a specialized swarm of scouts and workers [32]. In [33], an agent-based diffusion-breeding concept was introduced to image segmentation, in which the agents move over an image and connect homogeneous regions via an interplay of creating offspring and deactivating agents.

In this work, we present a novel algorithm based on the principles of Swarm Intelligence, that is particularly tailored to the task of contour or boundary detection in images. Boundaries in images can be arbitrarily distributed over the entire image, hence an algorithm should be able to detect and localize them independent of their position, size, shape, orientation, and appearance. The distributed nature of the swarms established in this algorithm suits to this circumstance. The details of our algorithm are explained in the following section.

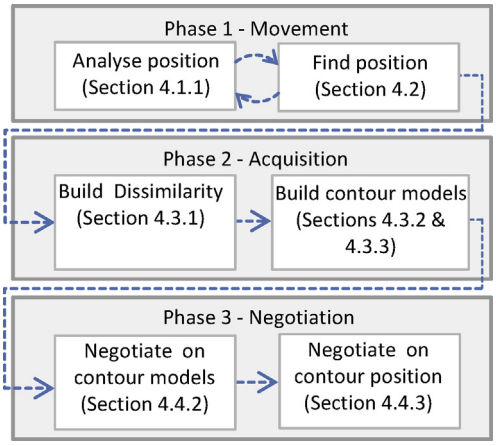

Fig. 2. Overview of the three phases of the agents. The single steps are explained in the corresponding subsections.

4. SI contour detection algorithm

Following the principles of Swarm Intelligence mentioned in the last section, we utilize a set of N distributed autonomous agents. Instead of being user-defined, the number of agents N is determined by the image size and its content. The goal is to ensure, that the entire image is covered with agents, so that all information anywhere on the image is considered. As each agent locks a patch of size 5×5 around its final position (see Section 4.1), N ranges between $(H \times W)/5 \leq N \leq (H \times W)/3$, with H and W being the image height and width.

The algorithm begins with the movement phase, in which the agents iteratively re-position themselves until they have found the locally most meaningful position (see Section 4.2). Once an agent has reached its final position, it analyzes the surrounding pixels to gather local information (see Section 4.1.1). In the end of the movement phase, the final value of N is obtained. In the next phase, the agents acquire information of their neighbored agents and build an initial estimation of the position and orientation of a contour within their local neighborhood (see Section 4.3). In the last phase, the agents interact to iteratively adjust their model to their neighbors' and thus grow from local models towards global contour estimates (see Section 4.4). Fig. 2 shows an overview of the procedural flow of the algorithm.

For communication purposes, each agent is assigned a unique *ID*, from $1 \ldots N$.

4.1. Local view

For information acquisition, positioning, and contour estimation, each agent features three different windows, an analyzing window W_A of size 3×3, a positioning window W_P of size 5×5, and a communication window W_C of size 11×11, each centered around the agent's position. Fig. 3 illustrates the extent of each of the three windows, centered around an agent. The choice of the window sizes allows a special interplay between the agents, which will be explained in detail in the following subsections.

4.1.1. Analyzing window

Every agent ag can directly access image data inside its analyzing window W_A, which is centered at position (xc_{ag}, yc_{ag}). The size of W_A is considerably small to allow accurate localization and discovery of arbitrary and capillary contour shapes.

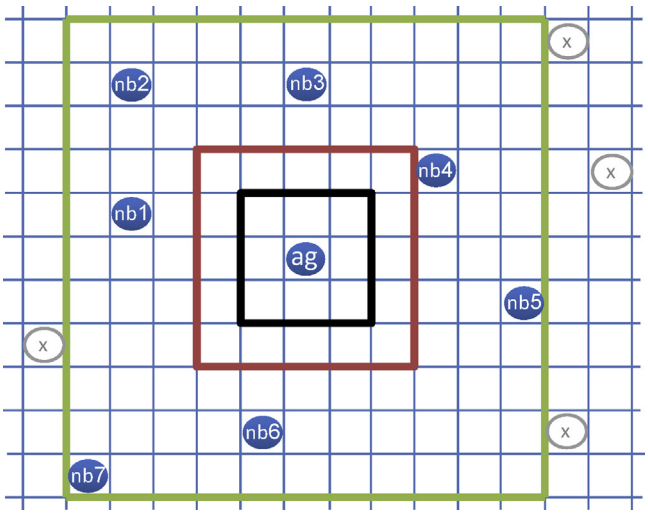

Fig. 3. Local view of an agent *ag* with its three windows, the analyzing window W_A (black), the positioning window W_P (red), and the communication window W_C (green). Within W_C the agent can interact with its neighbors *nb*1 to *nb*7. Agents outside W_C are displayed with an *x* and cannot be contacted by *ag*. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

The information each agents is gathering is firstly the *arithmetic mean* $M^{ch}_{W_A}$ of the nine pixels inside W_A via

$$M^{ch}_{W_A}(ag) = \frac{1}{9}\sum_{i=-1}^{1}\sum_{j=-1}^{1} I_{ch}(xc_{ag}+i, yc_{ag}+j), \tag{1}$$

for each color channel *ch*, with $I_{ch}(x, y)$ being the image values of channel *ch* at the pixel position *x*, *y*. Secondly, each agent calculates the *mean edge strength* E_{W_A} using

$$E_{W_A}(ag) = \frac{1}{9}\sum_{i=-1}^{1}\sum_{j=-1}^{1} G(xc_{ag}+i, yc_{ag}+j), \tag{2}$$

with $G(x, y)$ being the magnitude of the image gradient at the pixel position *x*, *y*, computed by

$$G(x,y) = \sqrt{I_x(x,y)^2 + I_y(x,y)^2}. \tag{3}$$

The values of I_x and I_y are the discretized partial derivative images in *x* and *y* direction, which are computed on the grayscale image via some standard approach like Sobel, or Prewitt.

4.1.2. Positioning window

The positioning window W_P describes the agents' movement capability, which will be explained in Section 4.2. Once an agent has discovered its final position, it blocks its area in the size of W_P, prohibiting any neighbored agent to move closer than 3 pixels towards the agent. The neighbored agent therefore is forced to search its final position outside the blocked area. Thus the size of W_P ensures non-overlapping W_A windows. Fig. 4 shows the possible distances between an agent *ag* and its neighbor *nbx* in *x*-direction. It is plain to see that, if neighbor *nbx* settles on a position with a distance between 3 and 5 pixels, it will stay a direct neighbor of *ag* (examples *a* to *c* in Fig. 4). A distance ≥6 pixels in one direction will cause another neighbor *nby* to move in the gap between the positioning windows of *ag* and *nbx* (example *d*). This positioning schema forces agents to settle along the contours everywhere in the image, thus making contours describable by the agents.

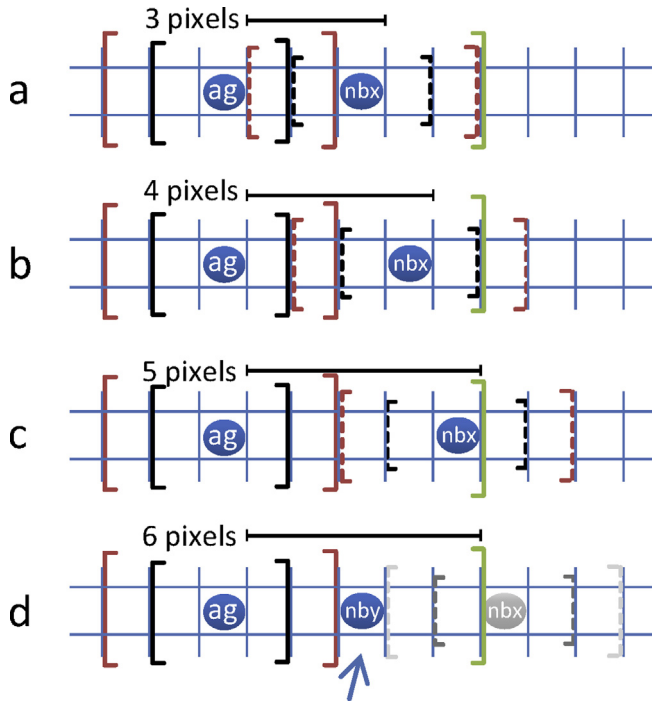

Fig. 4. Possible pixel distances of neighbors in *x* direction, minimum 3 and maximum 5. The black, red, and green solid lines refer to W_A,W_P, and W_C of the agent. The black and red dashed lines show the W_A and W_P of a neighbor *nbx*. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

4.1.3. Communication window

Gathering image information within small windows like W_A is generally prone to the typical errors in contour or boundary detection described in Section 1. We tackle this shortcoming via the communication window, that enables the agents to interpret the small local information in a broader context to eventually robustify the agents' collective decision. Furthermore, the communication principle allows to establish global contour estimations from the local information. The size of W_C descends from the size of W_P, as its size ensures that there is at least one neighbor, i.e. a communication partner in each main direction surrounding an agent. As shown in Section 4.1.2 and Fig. 4, the distance between an agent and a neighbor in any direction is 5 pixels at most, which coevally marks the needed extent of W_C in one direction.

The communication window W_C limits the area, in which an agent can find and contact neighbored agents. Each agent can communicate with all agents inside its communication window to access their model information. Any agent also collects the neighbored agents' *IDs* and their positions to relate them geometrically to its own position. The geometric information is utilized when the contour models are built. The access of neighbored agents constitutes a communication that resembles a spatially limited blackboard communication. To affirm their final connections, the agents furthermore use a direct communication, checking if their connection requests, i.e. their contour estimates, are acknowledged by the desired neighbor. Only connections that are confirmed by all connected agents are taken into account in the final contour image.

4.2. Movement

Initially the agents are distributed equally spaced over the entire image. The movement follows the goal of positioning the agents on image locations, which are possibly meaningful for contour detection. As a hint towards the final contours, the mean edge strength

information E_{W_A} of Eq. (2) is utilized. Every agent moves over the image iteratively checking the E_{W_A} values inside its W_A, centered at each pixel position (xc_{ag}, yc_{ag}) within its W_P. Subsequently each agent moves to the maximum inside W_P and re-centers the W_P, until it discovers a local maximum edge strength. Let t denote the current iteration, the movement rule can be expressed by

$$(xc_{ag}, yc_{ag})^{(t)} = (xc_{ag} + i^*, yc_{ag} + j^*)^{(t-1)}, \tag{4}$$

with i^*, j^* computed by

$$(i^*, j^*) = \underset{i,j \in \mathbb{Z}^{[-2,2]}}{\arg\max}\ E_{W_A}(xc_{ag} + i, yc_{ag} + j). \tag{5}$$

A local maximum is detected, if the position does not change, i.e.

$$(xc_{ag}, yc_{ag})^{(t)} = (xc_{ag}, yc_{ag})^{(t-1)} \tag{6}$$

is fulfilled. When an agent reaches the final position, it locks the area of the size of W_P around its position, so any other agent cannot get closer towards the locked position.

When the entire image is covered by agents, each holding a local maximum position regarding the edge strength, the movement phase stops.

4.3. Initial contour model

Each finally positioned agent tries to build an *initial contour model*. It therefore estimates the *course* and the *position* of a possible contour or boundary within its communication window. Furthermore, it evaluates the strength or probability of the existence of the assessed contour. For this purpose, the agent calculates three major fitness values, i.e. the agent dissimilarity value D, contour model fitness MF, and the contour position fitness PF.

4.3.1. Agent dissimilarity value

Each agent calculates the agent dissimilarity value $D_{ag}(nb_i)$ to each neighbored agent nb_i, $i = 1, \ldots, n$, with n being the number of neighbored agents within W_C. This measure describes the agent's dissimilarity to the neighbor utilizing the information gathered in each agent's W_A. A high $D_{ag}(nb_i)$ value means a high dissimilarity.

The agent dissimilarity value is calculated via

$$D_{ag}(nb_i) = \alpha \cdot \frac{1}{CH} \cdot \sum_{ch}^{CH} \frac{(M^{ch}_{W_A}(nb_i) - M^{ch}_{W_A}(ag))^2}{M^{ch}_{W_A}(nb_i) + M^{ch}_{W_A}(ag)} + \beta \cdot \frac{(E_{W_A}(nb_i) - E_{W_A}(ag))^2}{E_{W_A}(nb_i) - E_{W_A}(ag)}, \tag{7}$$

with $M_{W_A}(j)$ and $E_{W_A}(j)$ being the mean image value and the mean edge strength value of the W_A of agent j, as defined in Section 4.1.1, ch being the current color channel, running from $1 \ldots CH$, and CH being the number of image channels. In RGB color images $CH = 3$ and in grayscale images $CH = 1$. The factors α and β allow weighting of color and edge information.

The dissimilarity values $D_{ag}(nb_i)$ enable the agent to compare its image information to that of each neighbor. In Eq. (7) we adapt a formula, which is widely used as a dissimilarity measure for histograms (see e.g. [34]). This method emphasizes small image discontinuities in smooth regions, while it attenuates dissimilarities in textured regions.

4.3.2. Contour model fitness

Using its neighbored agents, each agent establishes several *contour models*, which describe an estimation of a contour's possible orientation or course. This is exemplarily shown in Fig. 5. To build a contour model m, the agent first connects itself to two neighbored agents. The connected agents, marked as C^m_{ag}, are supposed to have a considerably small agent dissimilarity value. Such a connection splits the agent's communication window into two regions A^m_{ag} and B^m_{ag}, with an arbitrary number of neighbored agents positioned inside each of them. The agent allocates each neighbor to a region using the relative position and geometry information. In a second step, the D_{ag} values of the neighbors in A^m_{ag} and B^m_{ag} are used to determine the region most separated to the agent and its two connected neighbors. Hence, high dissimilarity values in a region advert to a separation.

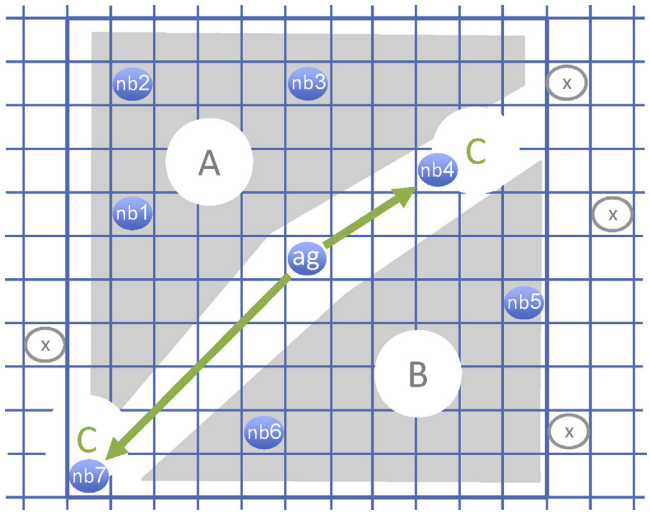

Fig. 5. A contour model m of agent ag, with all neighbors $nb1 \ldots nb7$ marked either as connected C^m_{ag}, or as belonging to region A^m_{ag}, or respectively region B^m_{ag}.

Consequentially, a contour model fitness MF is calculated by

$$MF^m_{ag} = max\ (MD_{B^m_{ag}}, MD_{A^m_{ag}}) - MD_{C^m_{ag}}, \tag{8}$$

with m being current model running from $m = 1, \ldots, M$. $MD_{B^m_{ag}}$, $MD_{A^m_{ag}}$, $MD_{C^m_{ag}}$ are the mean of the D_{ag} values of the neighbors, positioned either in A^m_{ag}, B^m_{ag} or marked as connection C^m_{ag}. The maximum number of possible models M for each agent is a user-defined number that can be used to limit the computational amount. Only the most promising models, estimated via the D_{ag} value of the connected neighbors are taken into account. From the set of M models, all agents pick the model with the maximum MF as a starting point.

Eventually, each model m is described with

- the set C^m_{ag}, which consists of the IDs of the two connected neighbors,
- the sets A^m_{ag} and B^m_{ag}, holding the agents' IDs that lie in the two regions separated by C^m_{ag},
- the information, which region is the most separated one, and
- the fitness MF^m_{ag} evaluating the quality of each model.

Finally, we want to mention that C^m_{ag}, A^m_{ag} and B^m_{ag} are non-intersecting sets, i.e.

$$C^m_{ag} \cap A^m_{ag} = C^m_{ag} \cap B^m_{ag} = A^m_{ag} \cap B^m_{ag} = \emptyset \tag{9}$$

4.3.3. Contour position fitness

As stated above, the *contour model* predominantly describes how a contour or boundary is orientated within an agent's neighborhood, and which region is separated from the agent. However, each agent additionally has to estimate the contour's exact location within the neighborhood, i.e. whether the assessed contour goes through the agent and its two connections, or possibly parallel

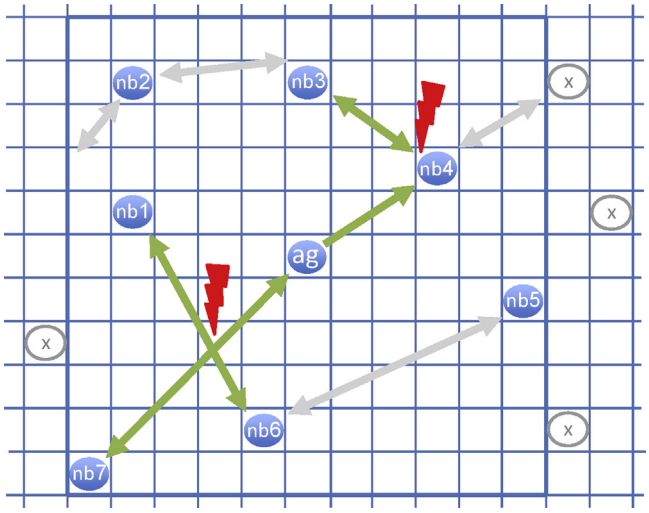

Fig. 6. Example of the possible conflicts of the agents' models. The arrows mark the connection requests of the agents. Multiple connection requests between *ag*, *nb*4, and *nb*3, and crossing contours between the connections $ag \leftrightarrow nb7$ and $nb1 \leftrightarrow nb6$.

through the neighbored agents in the separated region. As the dissimilarity values D_{ag} and the mean edge strength E_{W_A} are highest in each direction right upon a contour, each agent calculates a contour position fitness *PF*, simply by

$$PF_{ag} = \frac{1}{n} \cdot \sum_{i=1}^{n} D_{ag}(nb_i) + E_{W_A}(ag), \tag{10}$$

with $D_{ag}(nb_i)$ being the agent dissimilarity values of each neighbor nb_i, $i = 1, \ldots, n$, and n being the number of neighbored agents.

4.4. From local models to global contours

The use of the neighbored agents' information natively causes a dependance between the *initial contour models* of each agent. However, in certain situations the initial model picked by an agent can be erroneous due to misleading local image information. Experiments showed that in those cases the models picked by the agents only have slightly higher *MF* values than the models that are globally desired, or respectively regarded as correct. Furthermore, there is a subset of surrounding agents, which initially prefer the correct course of the contour. Thus, in a further step, the agents align and adjust their individual models to those of their neighbors. This exhibits the second aspect of communication besides each agent's access to its neighbors.

To accomplish the alignment of the contour estimations, the agents use communication, i.e. they share and compare their estimates with each other and negotiate about which model to follow. Thus, the final aligned models are the best compromise within the neighborhood.

When aligned, the contour model yields agents located on contours to connect with neighbors along this contour. Additionally, it forces agents near this contour to construct models with connections parallel to that contour. Therefore, a contour is estimated not only by the agents placed on it, but also by the agents placed parallel to it in its surrounding. This is a key point of this algorithm, increasing robustness in ambiguous image regions.

4.4.1. Conflicts between contour models

While the agents' contour models are unaligned, two main conflicts arise between the agents, which are illustrated in Fig. 6.

The first conflict is referred to as *multiple connection requests*. It originates, when two or more agents want to connect to the same neighbor, which is shown in Fig. 6. Here, both *ag* and *nb*3 request *nb*4 for a connection, but *nb*4 initially connects to *nb*3. This can be stated as

$$nbi \in C_{ag}^{m_k} \wedge \quad ag \notin C_{nbi}^{m_l}, \tag{11}$$

with *C* being the connection sets described in Section 4.3.2, and *k* and *l* being the currently chosen models of *ag* and *nbi*, respectively. In this example *nbi* is *nb*4.

The second conflict which appears is called *crossing contours*, and describes the situation, where two connections, and thus two contour estimates cross each other. As both connection estimations also imply that there are two regions which are not connected to each other, crossing connections must display a conflict. This is illustrated in Fig. 6 with the connection of *ag* and *nb*7 crossing the connection of *nb*1 and *nb*6. For a neighbor *nbi* this can be stated as

$$nbi \in q_{ag}^{m_k} \wedge \quad C_{nbi}^{m_l} \in r_{ag}^{m_k}, \tag{12}$$

with $q, r \in [A, B]$, $q \neq r$, and *k* and *l* being the currently chosen models of *ag* and *nbi*, respectively. In this example *nbi* is *nb*6 and *nb*1.

4.4.2. Negotiation on contour models

In order to solve the conflicts described in Section 4.4.1, each agent involved searches in its set of possible models $m = 1, \ldots, M$ for the best model that is not conflicting. Thus, in case of a *multiple connection request* between *ag* and its neighbor *nbi*, the agent *ag* takes the model m_o^*, which holds

$$m_o^* = \underset{m_o \mid \; nbi \notin C_{ag}^{m_o}}{\arg\max} \; (MF_{ag}^{m_o}). \tag{13}$$

Analogously, *nbi* chooses the model m_p^*, which holds

$$m_p^* = \underset{m_p \mid \; ag \in C_{nbi}^{m_p}}{\arg\max} \; (MF_{nbi}^{m_p}). \tag{14}$$

In Fig. 6, *nbi* is *nb*4. The *crossing contours* conflict follows the same mechanism, which means e.g. for *ag* and *nb*6 in Fig. 6 that both search for the best model m_o^* and m_p^*, which do not meet the condition stated in Eq. (12).

The agents corporately decide, which agent has to change its model, by comparing the model fitness of the old models m_k and m_l to those of the new models m_o^* and m_p^*. Therefore, each involved agent calculates the model fitness difference $\delta_{ag} = MF_{ag}^{m_k} - MF_{ag}^{m_o^*}$ and $\delta_{nbi} = MF_{nbi}^{m_l} - MF_{nbi}^{m_p^*}$. The agent with the smaller difference has to change from the old to the new model, so if $\delta_{nbi} < \delta_{ag}$, then $m_l \rightarrow m_p^*$, and $m_k \rightarrow m_o^*$ otherwise.

This procedure ensures that the overall fitness within the neighborhood is maximized. If an agent cannot find a model to solve the conflict, the value $MF_{ag}^{m_o^*}$, or respectively $MF_{nbi}^{m_p^*}$ is set to zero, causing the maximum possible difference. In this way, agents change their models until no conflicts occur any more. Agents with no acknowledged connections, or no possible model left to choose will be regarded as deactivated and no longer be considered for contour detection in the subsequent stage. This desired effect takes place within regions where agents either cannot agree to a final contour model, as there is none, or their final contour strength is negligibly small compared to the correct contours or boundaries. Hence, the algorithm is self-regulating to a certain degree.

4.4.3. Negotiation on contour position

When all conflicts are solved, the agents will either have established aligned contour models, which run parallel along region borders, or be deactivated. In order to decide for the exact position, the remaining active agents sum the *PF* values of themselves and their connections and compare it to the parallel model that lies

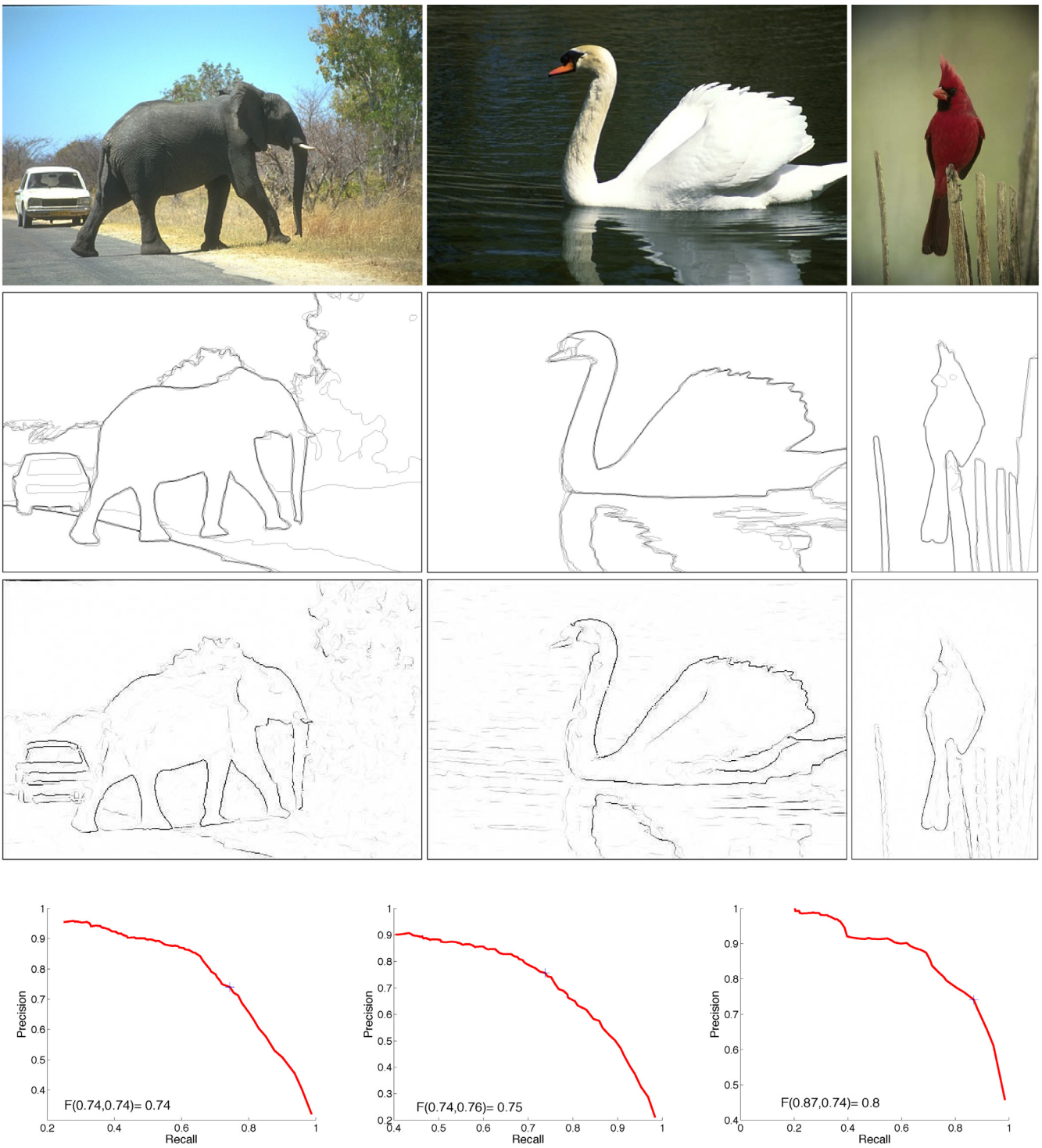

Fig. 7. Three sample images with distinctive content from the Berkeley Image Segmentation Dataset (BSDS 500) [3]. From top to bottom: the original images, the summed ground-truth images, the result images and the *precision-recall curves* of our algorithm.

in the direction of their most separated region. Each agent adds its final MF_{ag}^{m} to the agent with the maximum *PF*, which can also be the agent itself, and its two connections. Thus, parallel models support each other and strengthen the final position.

The final contour image corresponds to the summed position fitness of each agent and its connected neighbors. As stated above, only agents with connections that are without conflict and acknowledged by the respective connection partners are taken into account.

5. Experiments and results

In this section we show some results of our algorithm achieved in task of boundary detection. We will first comment on evaluation

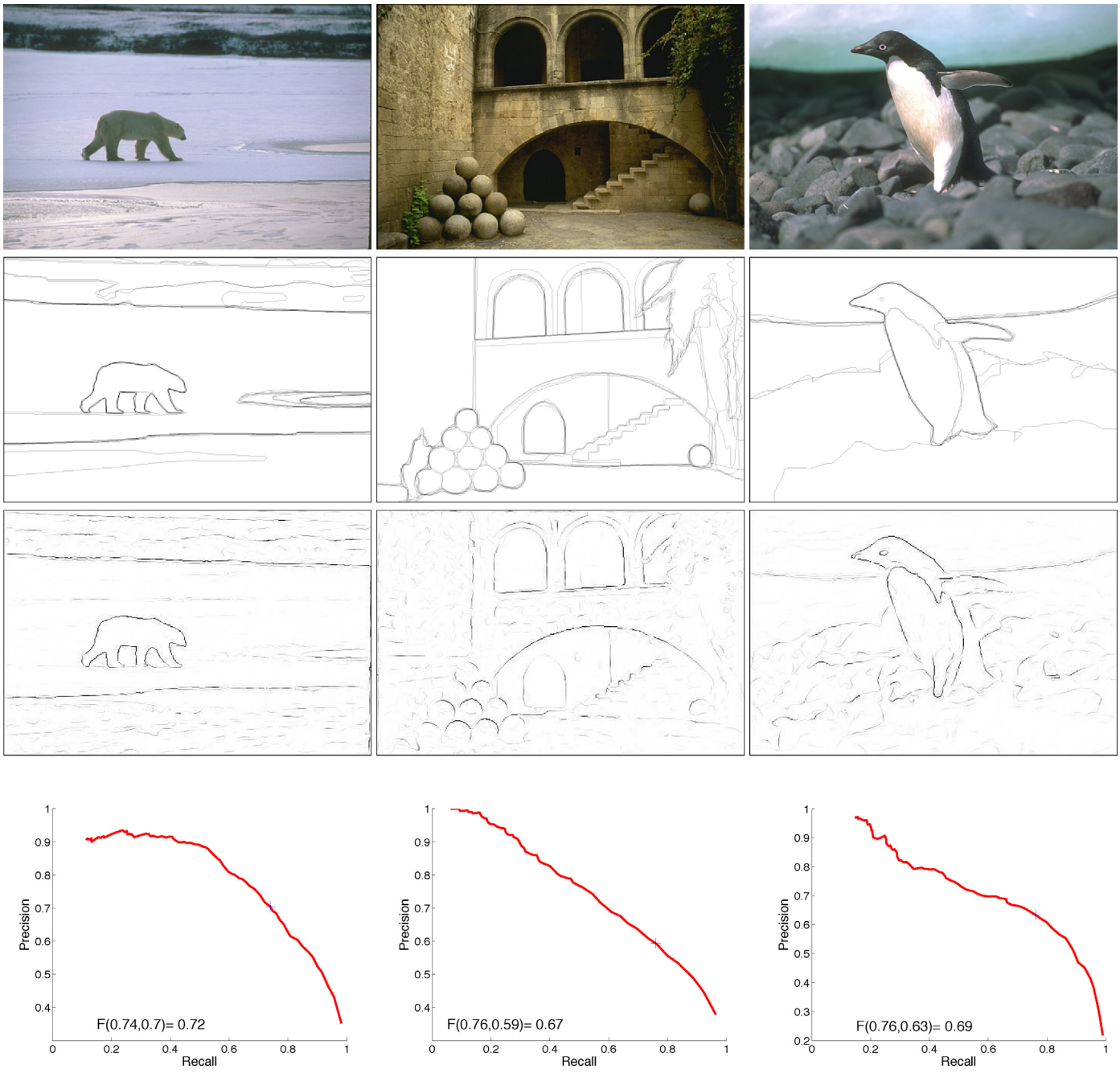

Fig. 8. Three sample images from Section 1, which yield to average performance with regard to the overall results. Taken from the Berkeley Image Segmentation Dataset (BSDS 500) [3]. From top to bottom: the original images, the summed ground-truth images, the result images and the *precision-recall curves* of our algorithm.

metrics and state our decision on the metrics that we applied. Then we will investigate and discuss the results of our algorithm in detail utilizing example images. Furthermore, we will deliver a comparison to methods from the literature which used the same evaluation metrics, and in the end, we will discuss the computational efficiency of our method.

5.1. Evaluation dataset and metrics

As described in Section 1, a major problem when it comes to evaluate boundary detection or image segmentation algorithms, is the need of a ground truth, and coevally the lack of uniqueness of such. One of the most extensive and widely used image collections for this purpose is the Berkeley Segmentation Dataset (BSDS) [1], which provides a set of 500 natural images of size 481×321, each along with 5–10 human-generated segmentations, which serve as ground-truth.

General evaluation metrics for image segmentation are for example the Probabilistic Rand Index [35], the Jaccard index, or the Object-level Consistency Error OCE [36]. Such metrics are usually based on the amount of overlap between ground-truth segments and segments detected by the algorithm. They differ for example in the used distance measures, the punishment for under- or oversegmentation, or the way the overlap is weighted with regard to the segment sizes. However, the metrics mentioned above demand image regions with closed boundaries that are fully segmented from each other, so any pixel in the image can be assigned to one segment exclusively. Therefore, those metrics cannot be

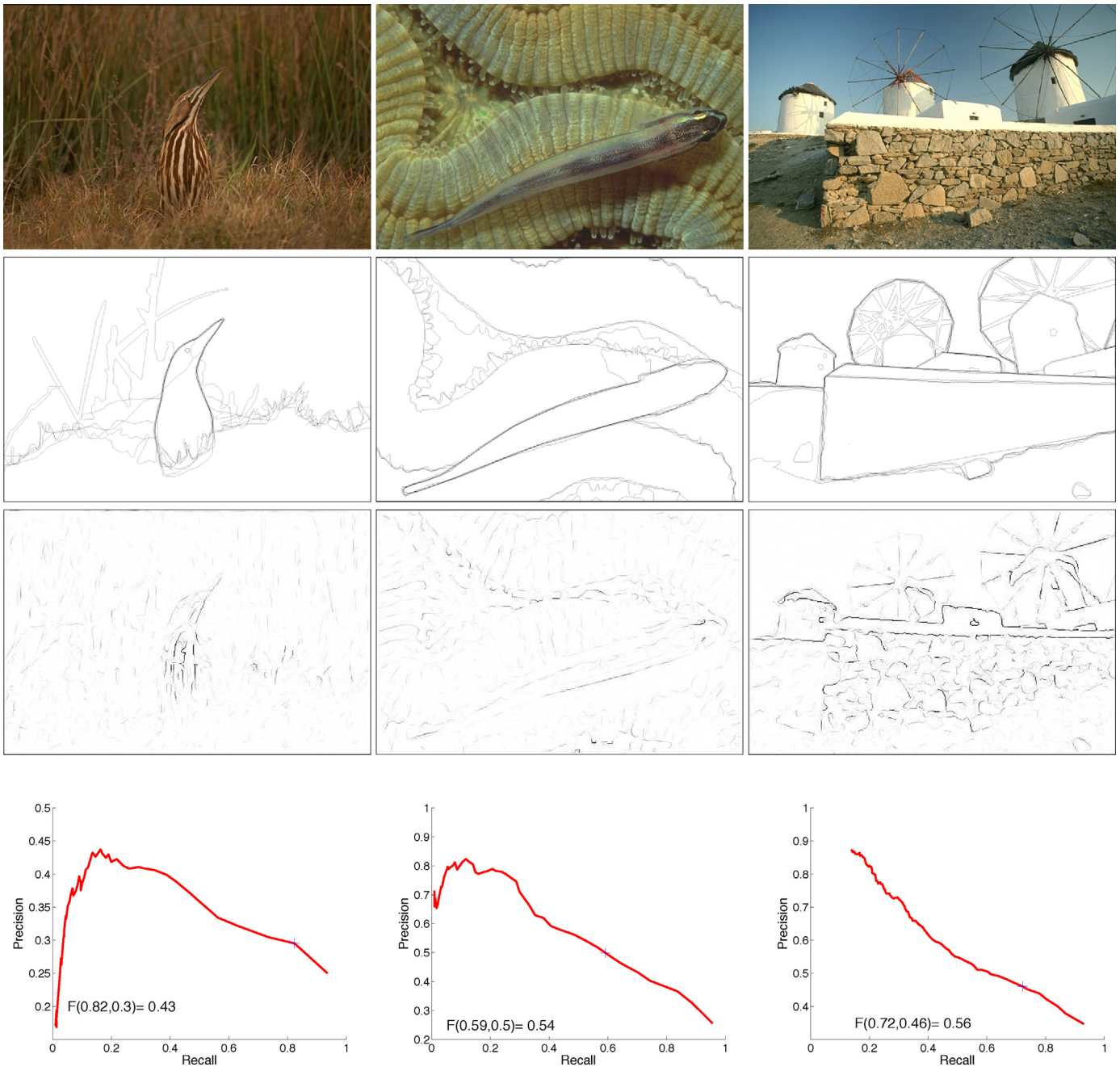

Fig. 9. Three sample images containing strong texture. Taken from the Berkeley Image Segmentation Dataset (BSDS 500) [3]. From top to bottom: the original images, the summed ground-truth images, the result images and the *precision-recall curves* of our algorithm.

applied to the topic of boundary detection, as such algorithms produce potentially non-closed contour estimates rather than segmented regions. Nevertheless, boundary based evaluation metrics exist, like *Pratt's Figure of Merit (FoM)* [37], *Precision-Recall Curves (PRC)* [1], and *Receiver-Operator-Characteristics (ROC)* [38], whereas the *PRC*-based benchmark system in [1] allows for comparison of region-based segmentation and contour detection methods in the same framework. Those methods usually measure the coincidence and the accuracy of boundary or edge pixels compared to ground-truth boundaries.

Pratt's Figure of Merit is a distance transform between the ground-truth and the evaluated boundary map binarized at varying thresholds. It measures the detection, localization, and spurious response, which means that the score is influenced by all edges being found, all edges being placed in the correct location, and false alarms. The *FoM* is computed by

$$FoM = \frac{1}{max(N_{GT}, N_{res})} \sum_{k=1}^{N_{res}} \frac{1}{1 + ad(k)^2},$$

where N_{GT} is the number of ground-truth edge pixels, N_{res} is the number of edges pixels detected by the algorithm, $d(k)$ denotes the distance from the kth detected edge pixel to the nearest ground-truth edge pixel, and a is a scaling constant, set to 1/9 as in Pratt's work. The metric ranges from 0 to 1, with 1 being the best obtainable result, which means that the ground-truth and the detected result are identical. As the threshold is varied, the best result obtained is used for evaluation [37].

Along with the BSDS image database [1], delivers a benchmark, in which the ground-truth data is compared to machine generated output using *Precision-Recall Curves*. *Precision* measures the probability that a machine-generated boundary pixel is a true boundary pixel, while *Recall* gauges the probability that a true boundary pixel is detected, i.e. the amount of ground-truth captured by an algorithm. Again, in order to obtain binarized edge maps from the algorithm's edge fitness output, *Precision* and *Recall* are calculated for a set of varied thresholds, which captures the trade-off between the accuracy and noise, i.e. the false-positives and the true-negatives.

To approximate the *Precision-Recall Curves'* optimal trade-off, the *F-Measure* is calculated for each threshold. The *F-measure* is the harmonic mean of *Precision* and *Recall*, calculated by

$$F = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}.$$

Another well-known metric in image evaluation is the *Receiver-Operator-Characteristic*, or *ROC-Curves*, whose axes are called *Fallout* and *Recall*. These metrics are comparable to the *Precision-Recall* metric. The *Recall* is defined identically in both metrics. *Fallout* measures the probability that a true negative was labeled a false positive. Just as the *PR-Curve*, the *ROC-Curve* is a graphical plot that displays the performance of a binary classifier system with a varied binarization threshold. However, the definition of the *Fallout* makes the *ROC-Curves* dependent on the image resolution, while the *Precision-Recall Curves* have the advantage of being scale-invariant [39] and are therefore preferred.

The BSDS benchmark delivers three major quantities: first the Optimal Dataset Scale (ODS), which is best F-measure on the dataset for a fixed scale or threshold, second the Optimal Image Scale (OIS), which is the aggregate F-measure on the dataset for the best scale in each image, and third the Average Precision (AP) on the full *Recall* range, which is equivalently the area under the *Precision-Recall Curve*.

5.2. Examples and discussion

The BSDS image dataset is partitioned in images for training, testing, and validation. For the evaluation, we ran the algorithm on the test- and validation-set of 300 images overall. We used the training set to empirically find the optimal parameters for the weighting factors α and β of the agent dissimilarity values, which were set to $\alpha = 0.35$ and $\beta = 0.65$.

Initial experiments revealed that the *negotiation mechanism* delivers an overall improvement of 0.01 for the *ODS* and *OIS* values compared to the initial connection estimates. This indicates that the *initial* Boundary Model estimates, relying on the Dissimilarity Value (see Section 4.3.1) and on the rule that only connections acknowledged by the neighbors are used, are generally sound. While the negotiation does not significantly improve the overall result in the complete dataset, we could observe slight improvements in ambiguous image areas, where the initial estimates of the agents were misleaded by e.g. textures.

Figs. 7–10 present example images categorized into four different categories, each along with the ground-truth, our resulting boundary images, and the corresponding *Precision-Recall Curves* with the optimal *F-measure*. Fig. 7 shows example images, where the algorithm performs well, as the picture content is quite distinguishable and consists of salient regions. The performance on the images displayed in the introduction (see Fig. 8) is similar to the average performance over the entire image set. The image content includes a balanced mixture of strong edges, salient regions, and also cluttered or textured regions.

The three example images of Fig. 9 contain a considerable amount of textured regions, which leads to a significant performance loss of our algorithm. This is due to the fact, that textures are not explicitly considered when calculating the boundary estimates. An improvement is to expect, when the agents are extended to incorporate texture cues like Textons, or Local Binary Patterns into the Dissimilarity Value formula. Furthermore, examining the image information inside an agent's neighborhood for repeating patterns texture-like structures seems promising. We will investigate both approaches in future work.

In [39] the authors argue that contour detection, while being closely related, is not perfectly the same as boundary detection. They state that contours, which might be important in terms of the image content do not necessarily have to be region boundaries, and might thus be excluded in boundary detection. The three example images in Fig. 10 illustrate the difference between boundaries and contours. It is obvious that our algorithm detects contours rather than boundaries, e.g. the single windows in the left picture, or the black and white stripes of the zebras. While extracting these salient contours might be desirable in some applications, it significantly degrades the performance of our algorithm with regard to boundary detection, as can be seen in the *PR-Curves*. To a certain extent, these misleading detections might supposedly be alleviated by considering textures, as stated above. For further elimination, additional information, e.g. in form of object or appearance models, knowledge, or training might become necessary.

Table 1
Results of Pratt's Figure of Merit (FoM) measure for the gPb-owt-ucm method described in [3], our algorithm and the Canny detector, mean and standard deviation calculated over the BSDS 500 test dataset.

Method	Mean FoM	Std. Dev.
gPb-owt-ucm [3]	0.6815	±0.11422
Our algorithm	0.50242	±0.11611
Canny	0.49208	±0.13758

5.3. Comparison

Table 1 shows the results for the FoM metric of our boundary detector, in comparison with the method from [3], and the Canny Detector. It displays the mean FoM and standard deviation, calculated over the BSDS test dataset. The method from [3] expectably performs best in terms of the *FoM* measure, our algorithm operates slightly better than the Canny edge detector.

The lower part of Table 2 confirms the trend of the *FoM* measure, with the method of [3] performing best, followed by our algorithm, which works better than the Canny method. Table 2 additionally

Table 2
Boundary benchmark results on the *BSDS*300 and *BSDS*500, for seven different segmentation methods in the upper table and two contour detectors plus our algorithm in the lower table, taken from [3]. As our method is a boundary detector, we added its results in the lower table. The values represent the F-measures using an optimal scale for the entire dataset (ODS) or per image (OIS). The area-precision is expressed in (AP).

	BSDS300			*BSDS500*		
	ODS	OIS	AP	ODS	OIS	AP
Human	0.79	0.79	–	0.80	0.80	–
gPb-owt-ucm [3]	0.71	0.74	0.73	0.73	0.76	0.73
Mean Shift [9]	0.63	0.66	0.54	0.64	0.68	0.56
NCuts [6]	0.62	0.66	0.43	0.64	0.68	0.45
Canny-owt-ucm [3]	0.58	0.63	0.58	0.60	0.64	0.58
Felz-Hutt [5]	0.58	0.62	0.53	0.61	0.64	0.56
SWA [40]	0.56	0.59	0.54	–	–	–
gPb	0.70	0.72	0.66	0.71	0.74	0.65
Our algorithm	**0.60**	**0.63**	**0.52**	**0.63**	**0.65**	**0.57**
Canny	0.58	0.62	0.58	0.60	0.63	0.58

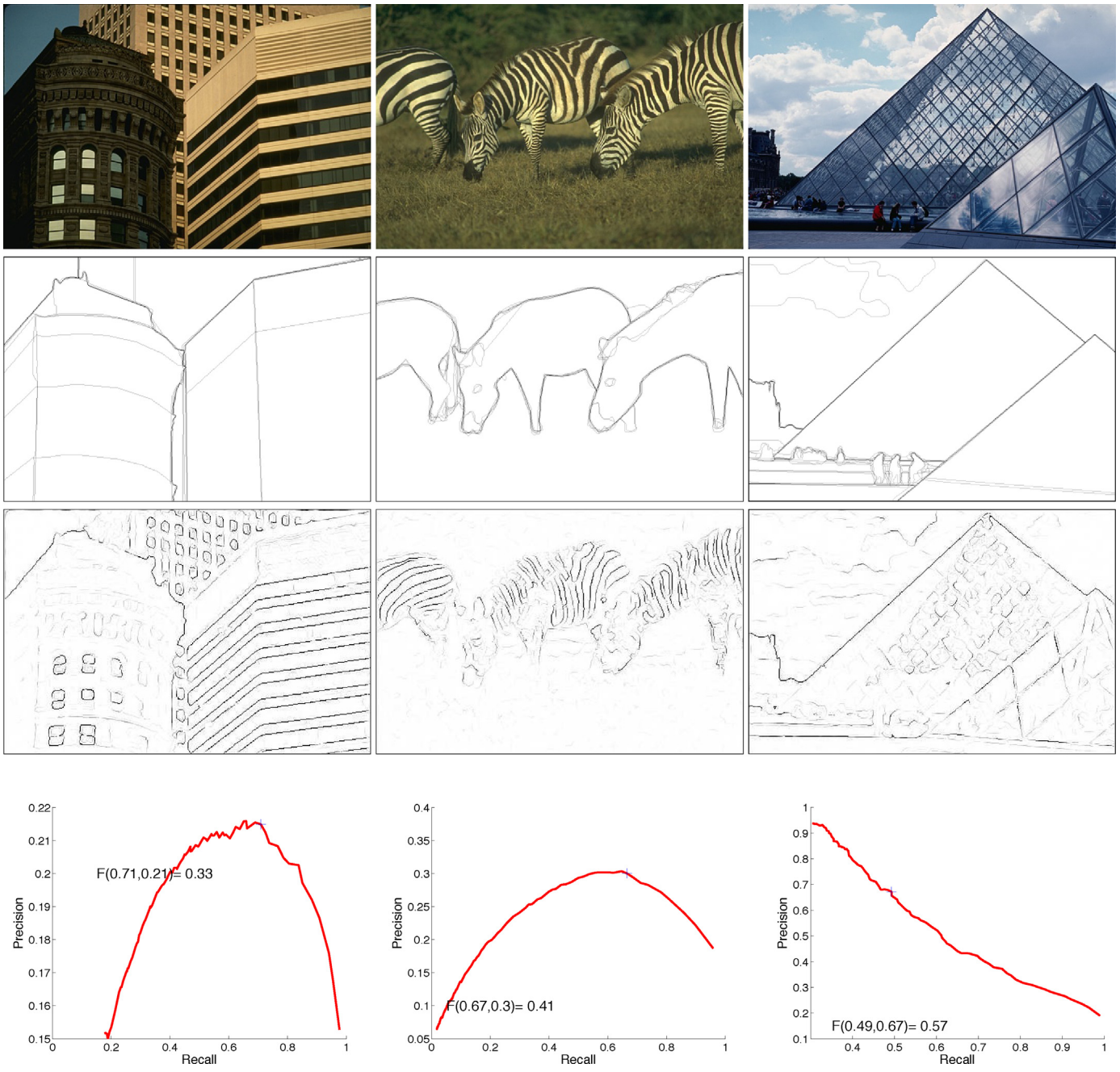

Fig. 10. Three sample images containing strong contours that are not boundaries. Taken from the Berkeley Image Segmentation Dataset (BSDS 500) [3]. From top to bottom: the original images, the summed ground-truth images, the result images and the *precision-recall curves* of our algorithm.

shows a comparison of other segmentation methods, provided by [3]. The compared methods besides *gPb* are originally described in [40,5,6,9].

Our tests reveal results that are quite promising, keeping in mind, that e.g. the *gPb* method of [3] is a highly advanced and sophisticated concept, developed and refined over several years, which explicitly considers e.g. texture information and also benefits from training phases.

Similarly, most of the methods compared in Table 2 utilize further image cues. As stated in Section 5.2, our method does not yet explicitly apply texture information but we are confident, that we could further improve our results by integrating more image information.

5.4. Computation time

Two parameters have principal influence on the computation time of our algorithm. The first is the number of agents N, which in turn depends on the image size, resulting in $\frac{1}{3} \times H \times W$ as worst case. The second influence is the number of negotiations, where in a worst case scenario negotiations would run until any agent has checked all its possible models. The number of negotiations directly depends on the maximum number of possible models M. Thus, the runtime of the algorithm is linearly dependent as $O(N \cdot M)$.

We ran our algorithm on an Intel Core 2 Quad CPU Q6600, 2.39 GHz and 3 GB of RAM, implemented in MATLAB. We used a mex-File to implement the negotiation part in *C*, besides that no

further code optimization was utilized. Our algorithm ran with 5.74 seconds on average per 481 × 321-image. For comparison, the *gPb*-method from [3] demands around four minutes of computation time for one image of the dataset using a regular *C* implementation. As stated in [41], the computation time of the *gPb*-method was reduced to 1.8 seconds utilizing parallelized GPU implementation. Thus, we believe that the computational speed of our method is advantageous when it comes to time-dependant applications.

6. Conclusion

In this work, we present a novel algorithm that uses a set of distributed agents and the principles of Swarm Intelligence to detect contours in images. Instead of optimizing a single existing task, our method seeks to analyze image content from scratch.

In a nutshell, the main contribution of our approach include (1) a moving rule, which by distributing the agents on the entire image in locally potentially meaningful positions creates a dynamic mesh; (2) a set of local analyze- and communication-tools that help agents to cooperatively decide on image contours based on their local environment; (3) a set of novel fitness functions, and an interactive negotiation rule leading to an overall best decision between the agents.

Contrary to recent methods, the algorithm does neither need any explicit object models nor training phases, or user-defined prior knowledge. The promising results of our experiments in the field of boundary detection indicate that the interaction of agents can lead from local segments to robust globally valid contours.

Further effort has to be spent on the agents' capability to deal with ambiguous regions, like texture and smooth transitions of regions. We will seek for methods to push the algorithm from contour to towards boundary detection. In order to step from boundary detection to image segmentation, we want to establish a version of the algorithm, in which agents can explicitly deal with contour junctions and also seek to close contours.

References

[1] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: International Conference on Computer Vision, 2001, pp. 416–423.

[2] D. Martin, C. Fowlkes, J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (5) (2004) 530–549.

[3] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (5) (2011) 898–916.

[4] S. Thilagamani, N. Shanthi, A survey on image segmentation through clustering, International Journal of Research and Reviews in Information Sciences 1 (1) (2011) 16–19.

[5] P.F. Felzenszwalb, D.P. Huttenlocher, Efficient graph-based image segmentation, International Journal of Computer Vision 59 (2) (2004) 167–181.

[6] T. Cour, F. Benezit, J. Shi, Spectral segmentation with multiscale graph decomposition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2005, pp. 1124–1131.

[7] W. Cui, Z. Guan, Z. Zhang, An improved region growing algorithm for image segmentation, in: International Conference on Computer Science and Software Engineering, 2008, pp. 93–96.

[8] J. Chang, J. Fisher, Efficient MCMC sampling with implicit shape representations, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 2081–2088.

[9] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 4 (5) (2002) 603–619.

[10] J. Chen, T. Pappas, A. Mojsilovic, B. Rogowitz, Adaptive perceptual color-texture image segmentation, IEEE Transactions on Image Processing 14 (10) (2005) 1524–1536.

[11] N. Senthilkumaran, R. Rajesh, Edge detection techniques for image segmentation – a survey of soft computing approaches, International Journal of Recent Trends in Engineering 1 (2) (2009) 250–254.

[12] Y. Ramadevi, T. Sridevi, B. Poornima, B. Kalyani, Segmentation and object recognition using edge detection techniques, International Journal of Computer Science & Information Technology 2 (6) (2010) 153–161.

[13] W. Ma, B.S. Manjunath, Edgeflow: a technique for boundary detection and image segmentation, IEEE Transactions on Image Processing 9 (8) (2000) 1375–1388.

[14] H. Wang, J. Oliensis, Generalizing edge detection to contour detection for image segmentation, Computer Vision and Image Understanding 114 (7) (2010) 731–744.

[15] K. Hammouche, M. Diaf, P. Siarry, A multilevel automatic thresholding method based on a genetic algorithm for a fast image segmentation, Computer Vision and Image Understanding 109 (2) (2008) 163–175.

[16] S. Das, S. Sil, Kernel-induced fuzzy clustering of image pixels with an improved differential evolution algorithm, Information Sciences 180 (8) (2010) 1237–1256.

[17] C.-Y. Lee, J.-J. Leou, H.-H. Hsiao, Saliency-directed color image segmentation using modified particle swarm optimization, Signal Processing 92 (1) (2012) 1–18.

[18] L. Li, D. Li, Fuzzy entropy image segmentation based on particle swarm optimization, Progress in Natural Science 18 (9) (2008) 1167–1171.

[19] D. Feng, S. Wenkang, C. Liangzhou, D. Yong, Z. Zhenfu, Infrared image segmentation with 2-D maximum entropy method based on particle swarm optimization (PSO), Pattern Recognition Letters 26 (5) (2005) 597–603.

[20] S. Saatchi, C.-c. Hung, Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, Itech Education and Publishing, Vienna, Austria, 2007, Ch. Swarm Intelligence-based image segmentation, pp. 163–178.

[21] S.A. Etemad, T. White, An ant-inspired algorithm for detection of image edge features, Applied Soft Computing 11 (8) (2011) 4883–4893.

[22] M. Setayesh, M. Zhang, M. Johnston, Edge detection using constrained discrete particle swarm optimisation in noisy images, in: IEEE Congress on Evolutionary Computation (CEC), 2011, 2011, pp. 246–253.

[23] G. Beni, J. Wang, Swarm Intelligence in cellular robotic systems, in: NATO Advanced Workshop on Robots and Biological Systems, 1989, pp. 26–30.

[24] W. Yong, C. Jun, Using ant Swarm Intelligence for data clustering analysis, in: 2nd IEEE International Conference on Computer Science and Information Technology, 2009, pp. 429–432.

[25] A.V. Baterina, C. Oppus, Image edge detection using ant colony optimization, WSEAS Transactions on Signal Processing 6 (2) (2010) 58–67.

[26] A. Amali Asha, S. Victor, A. Lourdusamy, Feature extraction in medical image using ant colony optimization: a study, International Journal on Computer Science and Engineering 3 (2) (2011) 714–721.

[27] E. Lakehal, A Swarm Intelligence based approach for image feature extraction, in: International Conference on Multimedia Computing and Systems, 2009, pp. 31–35.

[28] U. Kirchmaier, S. Hawe, K. Diepold, Dynamical information fusion of heterogeneous sensors for 3D tracking using particle swarm optimization, Information Fusion 12 (4) (2011) 275–283.

[29] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.

[30] A. Colorni, M. Dorigo, V. Maniezzo, Distributed optimization by ant colonies, in: Proceedings of the First European Conference on Artificial Life, 1992, pp. 123–142.

[31] F.M.A. Mohsen, A new optimization-based image segmentation method by particle swarm optimization, International Journal of Advanced Computer Science and Applications Special Issue (Image Processing and Analysis) (2011) 10–18.

[32] C. White, G. Tagliarini, S. Narayan, An algorithm for Swarm-based color image segmentation, in: IEEE SoutheastCon, 2004, pp. 84–89.

[33] J. Liu, Y.Y. Tang, Adaptive image segmentation with distributed behavior-based agents, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (6) (1999) 544–551.

[34] X. Liu, D. Wang, A spectral histogram model for textons and texture discrimination, International Joint Conference on Neural Networks 2 (2001) 1083–1088.

[35] R. Unnikrishnan, C. Pantofaru, M. Hebert, Toward objective evaluation of image segmentation algorithms, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (6) (2007) 929–944.

[36] M. Polak, H. Zhang, M. Pi, An evaluation metric for image segmentation of multiple objects, Image and Vision Computing 27 (8) (2009) 1223–1227.

[37] W.K. Pratt, Digital Image Processing, 3rd edition, John Wiley & Sons Interscience, New York, 2001.

[38] K. Bowyer, C. Kranenburg, S. Dougherty, Edge detector evaluation using empirical roc curves, Computer Vision and Image Understanding 84 (1) (2001) 77–103.

[39] G. Papari, N. Petkov, Edge and line oriented contour detection: state of the art, Image and Vision Computing 29 (2-3) (2011) 79–103.

[40] E. Sharon, M. Galun, D. Sharon, R. Basri, A. Brandt, Hierarchy and adaptivity in segmenting visual scenes, Nature 42 (7104) (2006) 810–813.

[41] B. Catanzaro, B.-Y. Su, N. Sundaram, Y. Lee, M. Murphy, K. Keutzer, Efficient, high-quality image contour detection, in: IEEE 12th International Conference on Computer Vision, 2009, 2009, pp. 2381–2388.