Cent. Eur. J. Comp. Sci.
DOI: 10.2478/s13537-012-0004-x

VERSITA

Central European Journal of **Computer Science**

Analysis of swarm behavior using compound eye and neural network control

Research Article

Wolfgang Kramper[1], Ralf Wanker[2], Karl-Heinz Zimmermann[1]*

1 *Institute of Computer Technology, Hamburg University of Technology, 21071 Hamburg, Germany*

2 *Department of Biology, University of Hamburg, 20146 Hamburg, Germany*

Received 11 November 2011; accepted 10 February 2012

Abstract: The emergent collective intelligence of groups of simple agents known as swarm intelligence is a new exiting way of achieving a form of artificial intelligence. This paper studies a formal model for swarm intelligence inspired by biological swarms found in nature. Software agents are used to model the individuals of a swarm. Each agent is controlled by a neural network that processes position data from the others in its visible zone given by a compound eye and in this way navigates in 3D space. An additional input parameter is used to represent the agent's motivation to form a swarm. Simulations with different motivation parameters exhibit remarkable agent formations that can be considered as biologically plausable. Several ways to improve the model are discussed.

Keywords: software agent• artificial neural network

1. Introduction

The aggregate motion of a school of fish, a flock of birds, or a herd of land animals is beautiful to watch and intriguing to contemplate. Swarming is a prominent example of emergent behavior: A number of simple entities (agents) operate in an environment forming a more complex behavior as a collective. This complex behavior is not a property of any single such agent, nor can it easily be predicted or deduced from the behavior of these agents. The emergent property of swarming provides qualities that are not directly traceable from the swarm's individual agents and is referred to as strong emergence [4]. In a biological sense, swarming, flocking or shoaling may be adaptive for the participating individuals because of reduced predator risk [5, 9], increased foraging success [34], reduced competition for resources [16], and increased rates of social learning [32, 34].

* *E-mail: k.zimmermann@tuhh.de*

Braitenberg vehicles illustrate the abilities of simple agents that can autonomously move around [3]. A Braitenberg vehicle is equipped with primitive sensors that are directly connected to effectors which immediately produce the movement of wheels. In a complex environment with several sources of light and shadow, Braitenberg vehicles will exhibit a goal-oriented, flexible and adaptive behavior, like zigzagging trajectories with accelerations and slowdowns, moving as fast away as possible from any strong light sources and exploring the surroundings until a deep pocket of shadow is found where they can rest.

Computer animation aims to describe and control all types of motion. Swarm motion can be animated by a particle system with the simulating agents being the particles and the laws of the simulated physics ruling the motion including a set of behaviors programmed into it by an animator, as described by Reynolds [27] and Heppner and Grenander [13]. The model is based on simulating the behavior of each agent independently. The aggregate motion of the swarm results from the dense interaction of the relatively simple behavior of the individual simulated agents. The animations showing simulated flocks built from this model seem to correspond to the observer's intuitive notion of a flock-like motion. However, it is difficult to objectively measure how valid these simulations are.

Similarly, individual-based models of attraction, alignment, and avoidance among agents have been used to investigate schooling among fish [6, 11, 14, 17, 22, 28]. Several patterns of swarming in these models resemble those found in real swarms [12, 15, 18]. However, these models are based on strong assumptions such as control of speed (fixed with random error), range of perception (too large to be considered local), and school size (too small). The significance of these models remains partly unclear when they are compared with empirical data of real fish [25].

Swarm intelligence describes the collection of decentralized, self-organized systems that appear *in vivo* or *in silico* [2]. Swarm intelligence systems are typically made up of a population of simple agents (boids). Artificial swarm intelligence is a modern artificial intelligence discipline that aims to design multi-agent systems with applications, e.g., in robotics and optimization. The design paradigm for these systems is fundamentally different from more traditional approaches. Swarm intelligence is based on many unsophisticated entities that cooperate in order to exhibit a desired behavior. The inspiration for the design of such systems is taken from the collective behavior of insects like ants, termites, bees, and wasps, or from the behavior of other animal societies like flocks of birds or schools of fish. Even though the single members of these societies are unsophisticated individuals, they are able to achieve complex tasks in cooperation. Coordinated behavior emerges from relatively simple actions or interactions between the individuals. These systems are not only decentralized and self-organized but often can be rather robust and flexible.

Swarm intelligence provided new approaches in optimization. In particular, there are two popular swarm inspired methods used in computational intelligence: ant colony optimization (Aco) and particle swarm optimization (Pso). Aco is a probabilistic search technique inspired by the behavior of ants and allows to tackle computational problems that can be reduced to finding good paths in networks [7]. Pso is a population based stochastic optimization technique inspired by social behavior of bird flocking or fish schooling which can be used to find an approximative solution to an objective (fitness) function in a multi-dimensional search space [20].

Multi-agent systems are also employed to explicitly represent co-ordination structures that implement dynamic team structures and co-ordinated team behavior. The BattleModel framework [10] was initially developed to support operational studies in military. It allows to integrate fidelity models of physical systems and provides agent-environment and agent-agent interactions. RescueModel is a simulation framework that is based on BattleModel and can be used to explore social issues in agent oriented systems [1]. Ekemas is an agent-based geo-simulation framework that supports continual planning in the real world [29].

One of the most prominent software packages for swarm simulation is Swarm [1]. It provides a tool box for implementing, observing, and conducting experiments on agent-based models. Swarm provides a conceptual framework, conventions, and a code library to design the software.

This paper provides a model of swarming that vastly differs from the existing models which are largely based on repulsion, attraction, and alignment. Each boid of a swarm is controlled by a neural network that processes position data from the comrades in its visible area given by an apposition eye and in this way navigates in the 3D space. This work is based on a research cooperation of the authors and particularly involves the PhD thesis of the first author [21].

One inspiration of building a control system for a software agent introduced here is the elementary neural system of

[1] *Swarm Development Group, University of Michigan, www.swarm.org/index.php/Main_Page*

primitive species. Eric R. Kandel [19] has shown in his work concerning *Aplysia californica*, also known as the California sea slug, that a nervous system needs only a few neurons for controlling the essential life system. E.g., *Aplysia* requires only four neurons for regulating its heart beat. One of our goals has been to keep the agent's control system as simple as possible just like Occam's razor. In contrast to this, one can find large neural networks which have to reproduce a reaction to a predictive signal. In our approach the precision of a reaction is of smaller importance. The more important issue has been to build the swarming behavior principles in a simple neural net based agent.
Swarm simulations have eventually to take in account a huge population of individuals. Simulations of fish schools use 10.000 and more individuals to study their behavior. In our simulations we deal with a smaller number of individuals but we are convinced that our results will scale up well to larger populations.
In 1987, C. Reynolds presented a computer model of coordinated animal motion such as bird flocks and fish schools [27]. His flocking model is based on three simple steering rules that describe the movement of flockmates: separation (i.e., steer to avoid crowding local flockmates), alignment (i.e., steer towards the average heading of local flockmates), and cohesion (i.e., steer to move towards the centre position of local flockmates). This model of computer animated flying objects has been considered, modified and extended by behavioural biologists in order to analyse and understand the complex mechanisms of natural animal behaviour [18, 36].
Couzin et al. [6] reproduced the three behavioural patterns introduced by Reynolds on spherical areas, which can be captured by the agent located in the centre of the sphere. A founding object (an obstacle or another agent), detected in one of the three areas, triggers off a different behavioural reaction in the agent system for interact with the environment. They defined three zones: the inner *zone of repulsion* where objects avoid each other, the middle *zone orientation* where the moving direction is adapted to other objects and the outer *zone of attraction* where the object steers towards other flockmates. The overall behaviour of this approach is defined by a set of the three "zone equations" and the influence of each equation is given by the detection of objects in the zones. Two significant swarm characteristics are shown: a torus shape where all swarm members move around in a circle at an empty centre point and a parallel configuration where all members move in the same direction in a compact group. Modelling behaviour in this way is a substantial step to model swarming from a physical view as particle motion in a fluid media [24, 33].
A further step of modelling swarm behaviour is to leave the spherical zones and find a shape more adapted to the physiognomy of sensory apparatus. Hemelrijk et al. [11] defined non-spherical zones and took into account that dead areas behind an animal do not contribute any information. Also the steering forces were extensively studied. Seven single steering forces build the behaviour reaction including a weighting factor for all of them to scale or inhibit the influence of every steering force [11]. They discussed some weaknesses of the model two which will be considered later on.
In the following a single member of a virtual swarm will be denoted as *software agent* or simply as *agent*. An agent exhibits a behavioural reaction to the presence of objects in its visible area. This area may be divided into distinct spaces like the three zones of Couzin's model. Each agent records the space around it, whereby a small section behind the agent is not considered. One can imagine an agent as a virtual bird or fish.

2. Methods

2.1. Software agent model

In the agent model suggested, the agents move in open three-dimensional space. The agent has a main propulsion from which it can move sideways in all directions. For this purpose the agent knows its own direction and speed. In the concept of the software agent, the agent is understood as an object with the attributes of position, location, and speed. In this way, the model based on software agents differs from those models in which swarm objects are merely elements of a global representation matrix.
Two concepts of our swarm model differ significantly from existing models that are based on the Reynolds' principles: the sensoric recording of the environment through virtual compound eyes and the implementation of behavioural attributes through a neuron network. Both concepts will be explained in detail further on.

Parameter	Unit
Number of Agents	
Min.	20
Max.	2,000
Regular Speed	1 U/s
Max. Detection Radius	180°
Max. Agent's View D_{max}	1,000 U
Max. Moving Agility	30°
Dimensions of an Agent	
Length	4.5 U
Width	0.5 U
Heigth	2.5 U

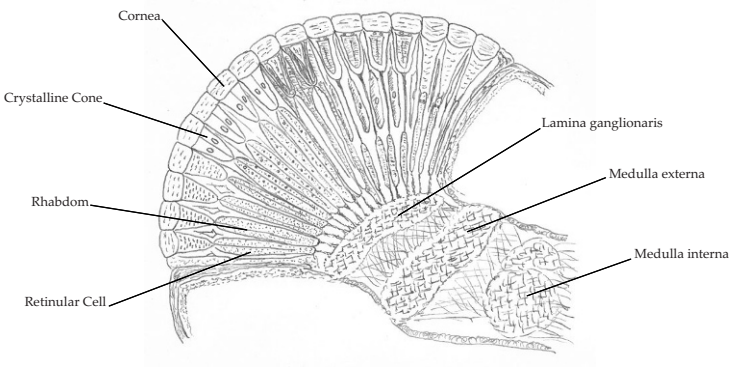
Table 1: Summary of agent's model parameter (U = virtual unit).

Our software is written in C++ and we use the Qt framework[2] as a library for the abstraction layer to the operating system. Using Qt makes it possible to port the software easily on all the usual operating systems. However the software was only tested on Windows and Linux. We also implement the agents and the neural network directly in our program, because none of the checked libraries and frameworks were suitable for our purposes. For instance, an agent library is given in Swarm Development Group[3] but the development ended in 2005. Other libraries do not fit either in programming language, operating system or age of project.
The simulation system depicts all agents as three-dimensional objects. After creating a number of agents (at least three) at distributed randomly positions in space the session will start. In an iterative manner the objects visible to each agent are determined and these data are given to a socalled behavioural control system. This system provides in turn a behavioural reaction which affects the position, direction, and speed of the agents. This leads to a new configuration of the virtual swarm and completes one step of simulation.
The simulation program enables the observation point given by the swarm centre to move along with the swarm itself, as if a "flying observer" were travelling with the swarm. The centre of the swarm is also the fixed focus point in the centre of the screen. This centre is marked with a red cross. The agent's shape depicted in Figure 6 has no relevance whatsoever to the recognition of other agents.
Each agent comes with the following data:

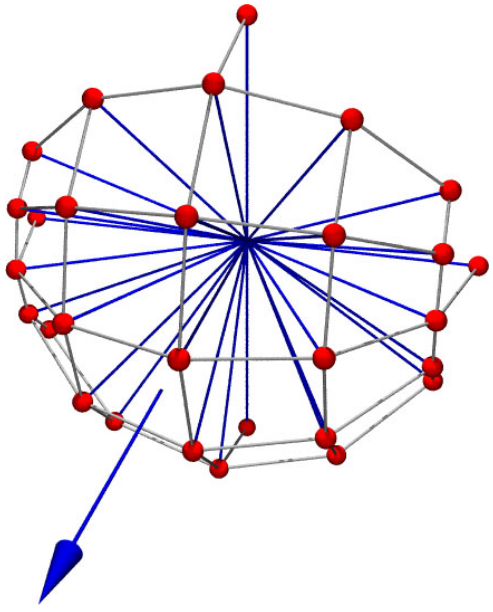
1. The absolute position $\boldsymbol{p} \in \mathbb{R}^3$ in an unbounded 3D space,
2. the current orientation which is the present moving direction $\boldsymbol{d} \in \mathbb{R}^3$, where $\|\boldsymbol{d}\| = 1$,
3. and the current agent velocity $v \in \mathbb{R}$.

These individual movement characteristics make each agent a unique object in the swarm. Every update to these parameters also updates a global log file where every direction step of each agent is recorded. The analysis of these data allows to create the trajectories such as those depicted in Figure 7.
Table 1 summarises the model parameters. We used 20 agents to verify the behavioural attributes. In order to investigate our model with a greater number of simulation objects, we increased the number to 2,000. The runtimes of one simulation step for various sizes of the swarm are illustrated in table 3. It appears that the runtime increases linearly with the size of the swarm.

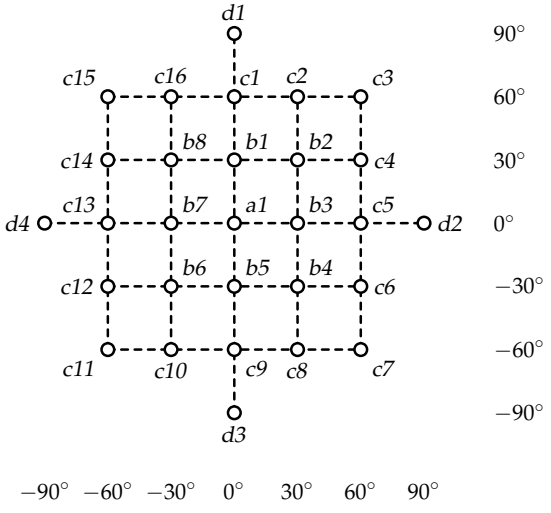
[2] *Nokia Corporation, The Qt home site, http://qt.nokia.com/products*
[3] *University of Michigan, www.swarm.org/index.php/Main_Page*



(a) Cross section of compound eye [35].



(b) Spatial arrangement (the arrow indicates the principal axis of the agent).



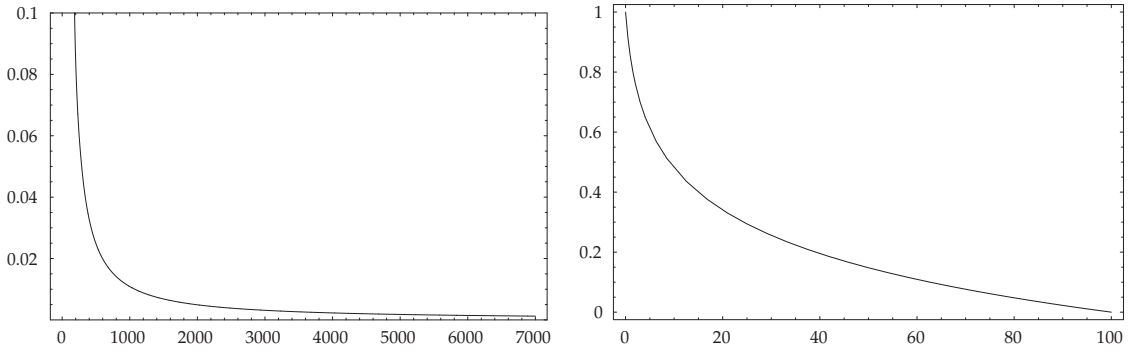
(c) Angle distances and designations (the designations refer to Table 2).

Figure 1: Cross section of a natural compound eye and the virtual compound eye of an agent.

2.2. Detection using virtual compound eye

As stated in the introduction, we use a model in which the agent records its environment visually. We believe that modelling the sensory information is necessary in order to make the swarm simulation more biologically plausible and hence to reproduce the inherent mechanisms in a more realistic way. The biological example can be explained with help of the principle of the compound eye of arthropods [31].

Figure 1a shows a cross-section of a compound eye. It consists of a variety of rod-shaped single eyes, the ommatidia. The number of these single eyes varies from 700 (the fruit fly *Drosophilia*) to 28,000 (the American dragonfly *Anax junius*) [23]. Every single ommatidium is an optical device, consisting of a lens for focusing the light, the cornea; the



(a) Learning curve of sensor network.

(b) Input function $f(d)$ for $D_{\max} = 100$.

Figure 2: Learning curve and sensitivity of agent's input visibility.

crystalline cells found beneath it; through to the rhabdom, connected at the lower end which in turn is surrounded by a hem of photoreceptors (Fig. 1a).

The light coming in through the lens is concentrated there and travels via the crystalline cells to the rhabdom, where the attached photoreceptors are stimulated. Every ommatidium thereby transmits a pixel in the same intensity as the light entering its optical axis.

In order for the insect to see a complete image, several single eyes must be arranged in a way that their respective optical axes depict a section of the image without overlapping. A complete picture is comprised of single images fitting together like a mosaic. The definition of the compound eye is thus dependent on the number of single eyes. The corneas form a regular domed hexagonal shape on the surface which is described as a compound eye.

Some types of insects have a differentiated eye system in which the single eyes are summarised or in which the dispersion of the optical axes of neighbouring ommatidia are taken into account. This is irrelevant as far as the compound eye serves solely as inspiration for the sensor system.

In our method the agent has a virtual compound eye with which it can record a section of its immediate environment. Due to practical considerations and limitations of the arithmetical complexity available for calculations, during each step in the sequence of an agent, it is not possible to implement hundreds or even thousands of single eyes.

In order to keep the complexity of the compound-eye based neural networks within reasonable bounds, we restricted the consideration to few—here 29—single eyes. Figure 1b shows the semicircular arrangement of the virtual compound eye. It provides an 180° field of vision in two directions. Due to the chosen number of single eyes, each sensor records, as depicted in Figure 1c, a section of $\pm 15°$. The arrangement is located in the middle at the front of the agent on its principal axis with a central front sensor a1. In this way it records all possible areas in which it can manoeuvre but not those areas which are directly behind it or at its sides towards the rear.

A realistic copy of the light entering the cornea should sum up the angle dependent light strengths in order to create an intensity value [23]. We have decided to do without this and have implemented a simplified process which draws the eye to the nearest object in the area of the virtual ommatidium. That means that ray tracing from a virtual light source does not take place, rather a sensor merely records objects along its optical axis including those in an area $\pm 15°$ of this axis. In our method the closest object is used, all other are ignored for object recording.

Since recorded objects can be far away and very close, we have limited the maximum visual area to D_{max} and calculate an intensity value

$$d(\delta) := \frac{\log(\delta + 1)}{\log(D_{max} + 1)}$$

where δ with $0 \leq \delta \leq D_{max}$ is the distance to the object. Hence distant objects lead to $d \to 0$ and close ones to $d \to 1$. The intensity values of all sensors are then transmitted to the behavioural control system. Hence the sensors are able to detect other agents within a limited distance (we set this distance to $D_{max} = 100$).

No.	Input Sensor	Outputs ($d_m = 0$)			Outputs ($d_m = 1$)		
#	Label	s_a	s_m	s_v	s_a	s_m	s_v
1	a1	0.00	0.00	1.00	0.00	0.00	1.00
2	b1	0.00	0.33	1.00	0.00	-1.00	1.00
3	b2	0.33	0.33	1.00	-1.00	-1.00	1.00
4	b3	0.33	0.00	1.00	-1.00	0.00	1.00
5	b4	0.33	-0.33	1.00	-1.00	1.00	1.00
6	b5	0.00	-0.33	1.00	0.00	1.00	1.00
7	b6	-0.33	-0.33	1.00	1.00	1.00	1.00
8	b7	-0.33	0.00	1.00	1.00	0.00	1.00
9	b8	-0.33	0.33	1.00	1.00	-1.00	1.00
10	c1	0.00	0.66	1.00	0.00	-0.50	1.00
11	c2	0.33	0.66	1.00	-1.00	-0.50	1.00
12	c3	0.66	0.66	1.00	-0.50	-0.50	1.00
13	c4	0.66	0.33	1.00	-0.50	-1.00	1.00
14	c5	0.66	0.00	1.00	-0.50	0.00	1.00
15	c6	0.66	-0.33	1.00	-0.50	1.00	1.00
16	c7	0.66	-0.66	1.00	-0.50	0.50	1.00
17	c8	0.33	-0.66	1.00	-1.00	0.50	1.00
18	c9	0.00	-0.66	1.00	0.00	0.50	1.00
19	c10	-0.33	-0.66	1.00	1.00	0.50	1.00
20	c11	-0.66	-0.66	1.00	0.50	0.50	1.00
21	c12	-0.66	-0.33	1.00	0.50	1.00	1.00
22	c13	-0.66	0.00	1.00	0.50	0.00	1.00
23	c14	-0.66	0.33	1.00	0.50	-1.00	1.00
24	c15	-0.66	0.66	1.00	0.50	-0.50	1.00
25	c16	-0.33	0.66	1.00	1.00	-0.50	1.00
26	d1	0.00	1.00	1.00	0.00	0.00	1.00
27	d2	1.00	0.00	1.00	0.00	0.00	1.00
28	d3	0.00	-1.00	1.00	0.00	0.00	1.00
29	d4	-1.00	0.00	1.00	0.00	0.00	1.00

Table 2: Training table for the neural net from Figure 3.

2.3. Neural behavioural control system

For the implementation of the behavioural control system we chose a neural network in order to increase the biological plausibility and because we believe that this aspect of existing models has not been considered sufficiently [8].
If one looks at Fig. 1a more closely, one can see how the cutout of retinular cell, in which the photoreceptors are embedded, leads to the lower end of the first optical ganglion, the Lamina ganglionaris. Further neuron clusters are connected there, first the Medulla externa and then the Medulla interna. Thus it seems plausable to formulate the behavioural reaction of the agent to optical stimulation, if one accepts artificial neural networks as a model, in order to thereby depict natural neurons.
Our neural network has 29 inputs given by the virtual photoreceptors. In addition we have allowed for a further input in order to be able to control the behavioural function. The agent can thus exhibit a search and an avoidance behaviour. The type of its basic motivation is set by a *motivation signal* via this input.
The output of the neural network directly impacts the directional controls of the agent. The output provides signals for meridian (up/down) and azimuthal (left/right) control. A third output signal sets the speed of the agent. In this way, the external dimensions of the network are defined.
When one designs a neural network, one or more intermediate levels are introduced internally which play an important role in modelling of the behavioural function. These intermediate levels give the network a certain depth and enable greater plasticity of the behavioural function. When dimensioning one is unfortunately dependent on trial and error in order to determine the optimum network size, as there is no process known which allows to determine an optimum network size from the problem size a priori.
The network structure determined is shown in Figure 3. Here we have defined a feed forward network in which the

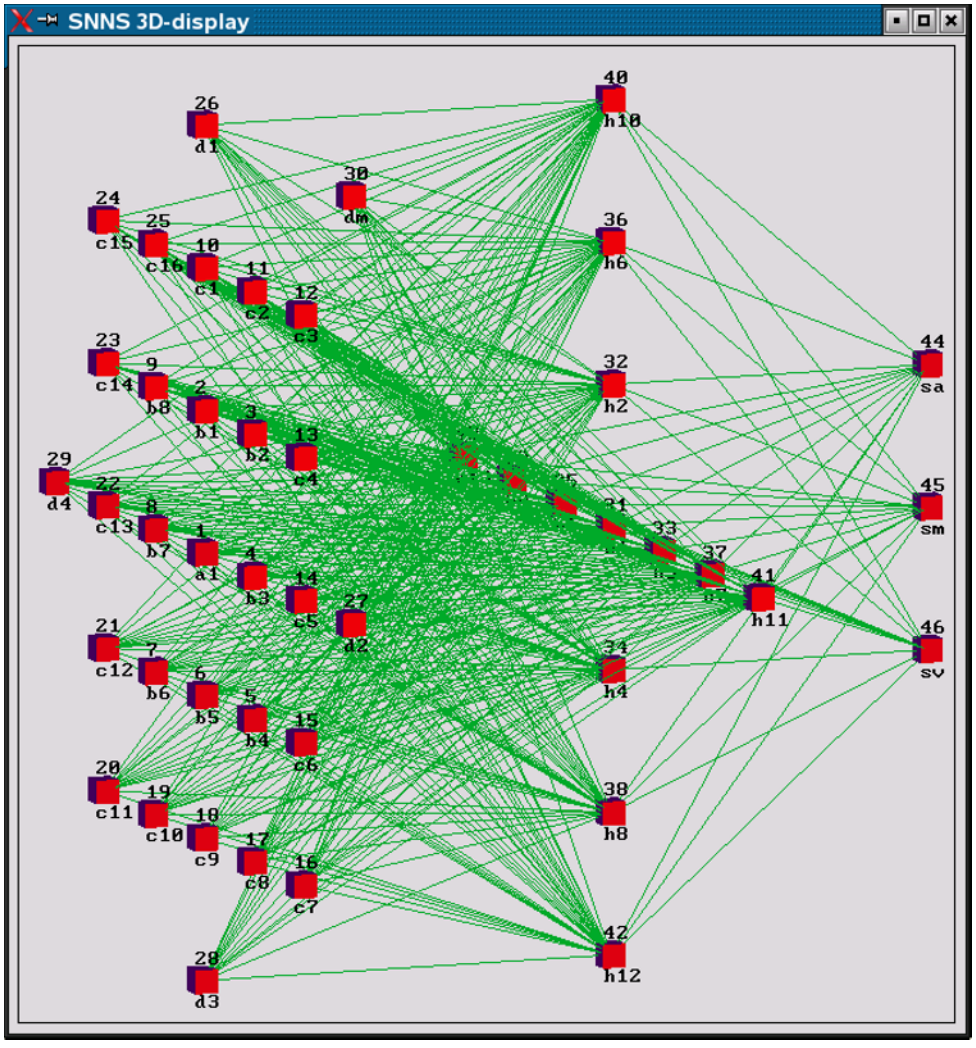


Figure 3: Neural net of the behavior control (Screen shot of SNNS).

information flow is led from the input neurons (depicted as red squares on the left in this figure) to an intermediate level (neurons in a cross-shaped arrangement) and from there to three output neurons. The figure shows a window of the program with which the network is parameterised. For this purpose we used the SNNS[4].

The figure also shows that all input neurons are connected with all neurons of the intermediate level and these are then connected with all three output neurons. As these connections are already fixed, the desired transmission behaviour of the network cannot be based in this structure. Rather it lies hidden in the scaling values of the connections—socalled weights—and the parameters of the activation function, which every artificial neuron has and which must be determined as a whole.

The behaviour of a neural network is determined by learning until the desired behaviour has been reached. The corresponding behavioural function is listed by a so-called *learning table* in which input and desired output values are given. The search for a behavioural function that fulfils the learning table exactly or approximately is described as

[4] *Stuttgart Neural Network Simulator, http://www-ra.informatik.uni-tuebingen.de/SNNS*

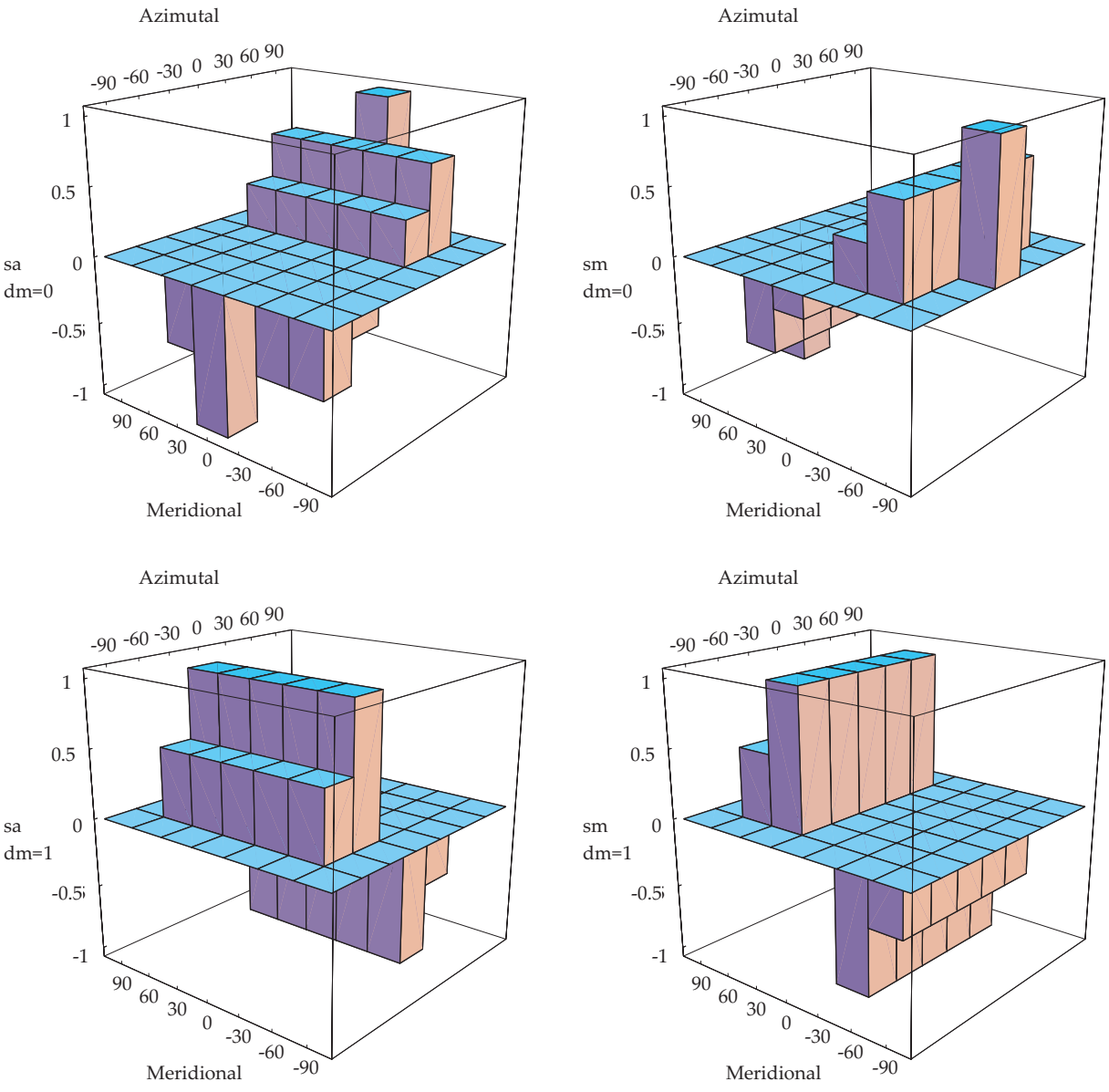


Figure 4: Output values with representation of the excitation of the input sensors.

training [37].

This implementation of the behavioural function cannot be based on equations or physical constructions such as the "centre of gravity" of the swarm. To create the learning table one must consider various behavioural reactions and define them as control values of the agent. Table 2 shows these values. One can clearly recognise that it is possible to differentiate between two motivational conditions: avoidance ($d_m = 0$) and search ($d_m = 1$). The learning table provides sampling points of a behavioural function which will be interpolated by the learning process. This will be due to the network's internal layer—provided that no errors (e.g., over-training, inappropriately selected intermediate level) have been made [37].

The intuitive definition of a behavioural function may appear strange at first, which is why we would like to expand upon this point. The diagrams in Figure 4 should help to further illustrate this issue. An intensity value of 0 means that the sensor has not detected anything and a value of 1 means an immediate collision with an object. Increasing intensity

Algorithm 2.1 SwarmSim(N, T, $D_{\max}$, d_m)

Input: initial distribution of boids in space, boid i has position $\boldsymbol{p}_i^{(0)}$ and orientation $\boldsymbol{d}_i^{(0)}$.
Output: time evolution of swarm, boid i has position $\boldsymbol{p}_i^{(t)}$ and orientation $\boldsymbol{d}_i^{(t)}$ at time t.

1: **for** $t \leftarrow 0$ to T **do**
2: **for all** boids B_i **do**
3: **if** there is no boid visible to B_i **then**
4: $\boldsymbol{d}_i^{(t+1)} \leftarrow \boldsymbol{d}_i^{(t)}$
5: $\boldsymbol{p}_i^{(t+1)} \leftarrow \boldsymbol{p}_i^{(t)} + s_{vn}\boldsymbol{d}_i^{(t+1)}$ {s_{nv} is the free running speed, default value is unit speed s_v}
6: **else** {there are boids visible to B_i}
7: $\boldsymbol{d}_i^{(t+1)} \leftarrow \boldsymbol{0}$
8: $\boldsymbol{p}_i^{(t+1)} \leftarrow \boldsymbol{0}$
9: **for all** boids B_j visible to B_i **do**
10: $\boldsymbol{d}_{ij}^{(t+1)}, \boldsymbol{p}_{ij}^{(t+1)} \leftarrow \text{newData}(B_i, B_j, t)$
11: $\boldsymbol{d}_i^{(t+1)} \leftarrow \boldsymbol{d}_i^{(t+1)} + \boldsymbol{d}_{ij}^{(t+1)}$
12: $\boldsymbol{p}_i^{(t+1)} \leftarrow \boldsymbol{p}_i^{(t+1)} + \boldsymbol{p}_{ij}^{(t+1)}$
13: **end for**
14: $\boldsymbol{d}_i^{(t+1)} \leftarrow \frac{1}{N_{it}}\boldsymbol{d}_i^{(t+1)}$ {compute averages; N_{it} is the number of boids B_j visible to B_i}
15: $\boldsymbol{p}_i^{(t+1)} \leftarrow \frac{1}{N_{it}}\boldsymbol{p}_i^{(t+1)}$
16: **end if**
17: **end for**
18: **end for**

values therefore imply the approach of an object. Evasive action therefore requires a movement in the opposite direction to the incoming stimulus. This is depicted in the two diagrams above in Figure 4. The controlling signal possesses a positive and a negative level for both drive directions (a positive drive direction is represented by a bar pointing upwards and a negative drive direction by a bar pointing downwards). In particular, a level 0 means no movement. A search reaction requires the opposite behaviour. The drive movement must be made towards the approaching object. The two diagrams below depict this in Figure 4.
In our implementation every agent has the same parameters—their behavioural responses do not differ in principle. This made it possible to create the network only once, as a static entity, and enabled each agent to use it to calculate the iterative steps.
Table 2 also reveals that a modulation of the velocity does not occur. Variations in speed had no visible effect on the swarm behaviour. Therefore, we did not model it and set the value of s_v to 1.

2.4. Swarm simulation

A swarm consists of a finite set of boids located in three-dimensional Euclidean space. Each boid is described by three parameters: absolute position $\boldsymbol{p} \in \mathbb{R}^3$, orientation $\boldsymbol{d} \in \mathbb{R}^3$ with normalized value $|\boldsymbol{d}| = 1$, and propulsion $v \in \mathbb{R}$; for simplicity, all boids move with unit speed of $v = 1$ m/sec. The orientation vector points into the boid's moving direction and thus determines the boid's visual zone (blue arrow in Fig. 1c). We assume that all boids are aware of a (virtual) geotaxis such that the movement along the positive y-axis is considered to be on the top; this axis is the top direction of the OpenGL reference coordinate system [30]. Furthermore, we assume that each boid is equipped with the sensor apparatus introduced above. More specifically, all boids are furnished with identically trained neural networks (i.e., networks with the same weights).
The algorithm for swarm simulation called SwarmSim is illustrated in Alg. 2.1. It is time-discrete and parametrized by the number of boids N, the number of time steps T for simulating the swarm, the maximal distance $D_{\max}$ up to which objects can be detected within the boid's visual hemisphere, and the motivation parameter d_m.
Initially, at time step $t = 0$, the boids are randomly distributed in space. In each time step $t \geq 0$, the current location of the swarm is used to compute the new location at the next time step $t + 1$. For this, the boids are considered one after the other at time step t. For each boid at time step t, it is first determined whether there is another boid in its visible zone which is not farther away than $D_{\max}$; these boids are visible to the boid considered. If not, the boid changes only its position by using the free running speed s_{nv}, whose default value is given by the unit speed s_v (lines 4

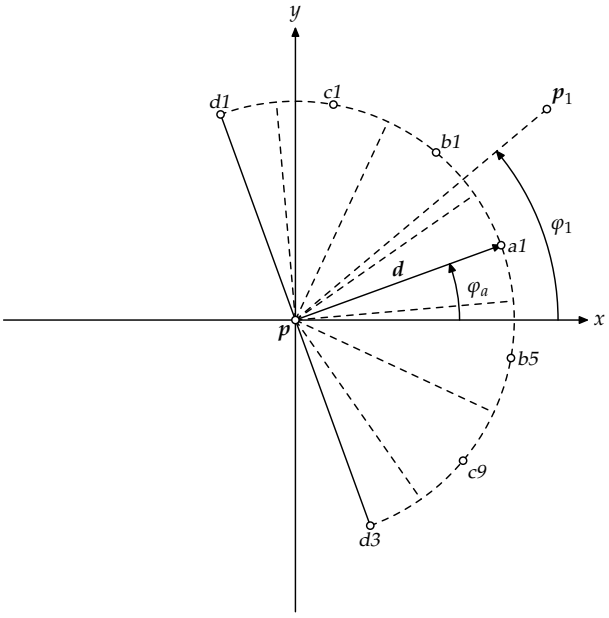


Figure 5: Computation of meridional tilting angle $\varphi = \varphi_i - \varphi_j$.

and 5). Otherwise, the boid changes its orientation and position according to its neural network (lines 7 to 15). This is accomplished by the procedure newData (line 10) that works as follows:

1. Compute the meridional and azimuthal tilting angles φ and ψ, respectively, in dependence of the orientation of B_i and the location of B_j (Fig. 5).

2. Determine the rectangular section and thus the sensor on the hemisphere of B_i that corresponds to the location of boid B_j by using the tilting angles.

3. Use the distance $d = \|\boldsymbol{p}_i^{(t)} - \boldsymbol{p}_j^{(t)}\|$ between the boids to provide an input to B_i's neural network: The value for the sensor (input neuron) determined in step 2 is

$$f(d) = 1 - \frac{\log(d+1)}{\log(D_{\max}+1)},$$

   while all other sensor input values are set to zero. This function increases the boid's sensibility for narrower objects (Fig. 2b).

4. The boid's neural network provides the response values s_a, s_m, and s_v, where s_v is set to unit speed of 1 m/sec.

5. Put $\boldsymbol{d} = \boldsymbol{d}_i^{(t)}$. Compute the respective azimuthal and meridional tilting vectors

$$\widetilde{\boldsymbol{d}_a} = \begin{cases} (1-s_a)\boldsymbol{d} + s_a\boldsymbol{d}_l & \text{if } s_a \geq 0, \\ (1+s_a)\boldsymbol{d} - s_a\boldsymbol{d}_r & \text{otherwise,} \end{cases}$$

   and

$$\widetilde{\boldsymbol{d}_m} = \begin{cases} (1-s_m)\boldsymbol{d} + s_m\boldsymbol{d}_u & \text{if } s_m \geq 0, \\ (1+s_m)\boldsymbol{d} - s_m\boldsymbol{d}_d & \text{otherwise,} \end{cases}$$

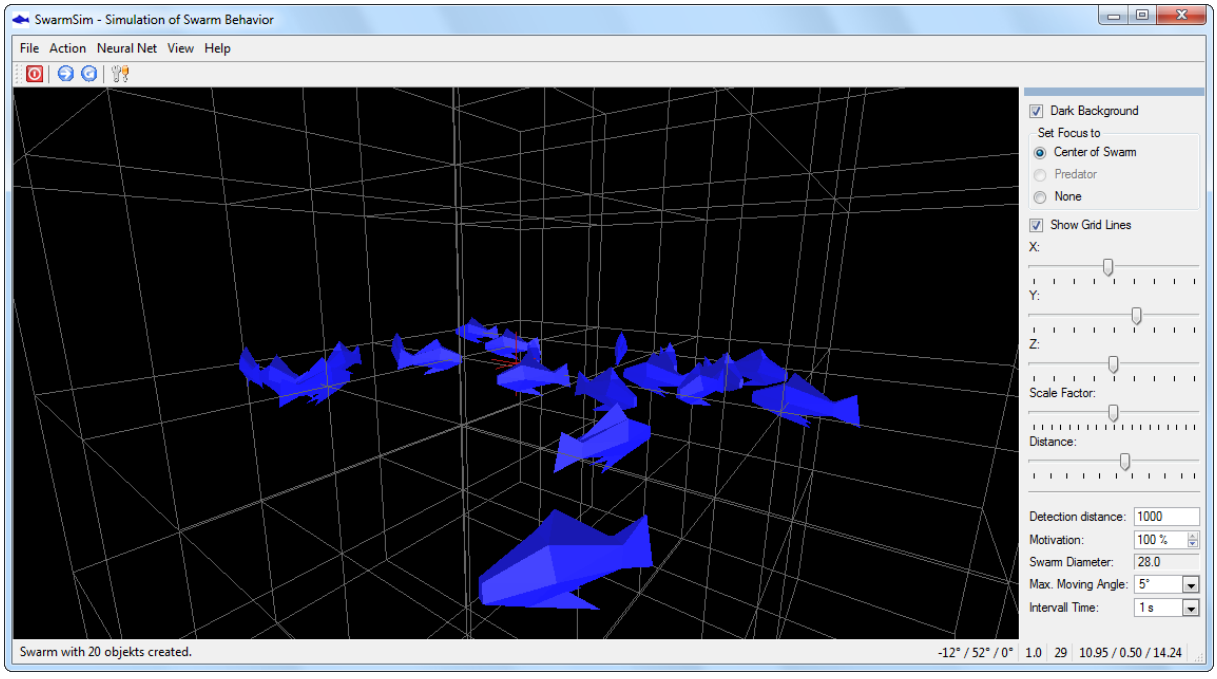


Figure 6: Screen shot of simuation program.

where in view of the virtual geotaxis, the tilting axis is given by the vector product

$$\boldsymbol{v}_{\text{pitch}} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \times \boldsymbol{d},$$

and $\boldsymbol{d}_u = \boldsymbol{d} \times \boldsymbol{v}_{\text{pitch}}$ (dorsal), $\boldsymbol{d}_d = -\boldsymbol{d}_u$ (ventral), $\boldsymbol{d}_r = -\boldsymbol{v}_{\text{pitch}}$ (lateral, right), and $\boldsymbol{d}_l = -\boldsymbol{d}_r$ (lateral, left).

6. Compute the boid's new direction depending on the tilting vectors,

$$\boldsymbol{d}_{ij}^{(t+1)} = \boldsymbol{d}_i^{(t)} + \frac{\widetilde{\boldsymbol{d}_m} + \widetilde{\boldsymbol{d}_a}}{\|\widetilde{\boldsymbol{d}_m} + \widetilde{\boldsymbol{d}_a}\|},$$

and the new position with respect to orientation and propulsion value,

$$\boldsymbol{p}_{ij}^{(t+1)} = \boldsymbol{p}_i^{(t)} + s_v \boldsymbol{d}_{ij}^{(t+1)}.$$

7. Return $\boldsymbol{d}_{ij}^{(t+1)}$ and $\boldsymbol{p}_{ij}^{(t+1)}$.

Note that the new orientation and position of a boid is calculated by averaging over all boids in its visual hemisphere (lines 14 and 15). In particular, the new position of the boid B_i is given by the former position, the new orientation and the unit speed as follows,

$$\boldsymbol{p}_i^{(t+1)} = \boldsymbol{p}_i^{(t)} + s_v \boldsymbol{d}_i^{(t+1)}.$$

Number of agents	Runtime in ms		
	min	average	max
20	27	66.3182	888
100	142	449.577	2144
500	632	1011.29	3859
1000	1614	2221.29	43104
2000	4763	5255.64	20105

Table 3: Runtime measurement for an iteration step of all agents over at least 200 cycles with $d_m = 0.5$.

3. Results

At first it was of interest for us to find out whether the values 0 and 1 at the motivational input of the network achieved the desired behavioural patterns in the agents. In order to check this, every test sequence begins with 20 agents, which are randomly distributed in 2D space. This allows the basic behavioural forms to be better observed. Figure 6 shows a screen shot of the simulation program with such an initial distribution. The agents are depicted as expanded geometrical objects, however only the geometrical midpoint of the agent—its position value—is actually relevant for object recognition. Everything else would require a complicated ray tracing method of all geometrical bodies and would greatly increase the amount of calculations required.
If the signal $d_m = 0$ is set, avoidance behaviour occurs in the agent. From the initial distribution all agents strive to move apart radially. Each agent strives to put the greatest possible distance between itself and the others (Figure 7 upper left).
On the other hand, if the signal $d_m = 1$ is set, search behaviour becomes effective and each agent moves towards the other. A stable final state is always one or more formations in which all agents are in a row. The swarm usually decays into several fragments when this happens (Figure 7 upper right). These two states represent the extreme values of behavioural control.
An interesting behaviour occurs when $d_m = 0.5$ is set: The agents circle each other as if they were in a vortex! This formation is stable; nearly every agent from the initial position remains in the circular swarm. Couzin et al. [6] report similar behaviour despite the fact that a completely different swarm model is used there. This vortex essentially stays in the area $0.2 \leq d_m \leq 0.8$ and only finally falls apart outside this range.
The two diagrams below in Figure 7 show the cases $d_m = 0.35$ and $d_m = 0.65$. While for the smaller d_m-value some agents leave the edge of the swarm, as they would do for $d_m \to 0$, one recognises the beginning of the fragmentation of the swarm at the higher value, as with $d_m \to 1$.
The plasticity of the neural network makes it possible to vary the motivational parameters and to change the behavioural characteristics of the network seamlessly. Thus the behaviour can be controlled by the domain of the signal d_m.
We are aware of the fact that these short sequences do not in themselves lead to conclusions on the real swarm formation. Yet we have shown:

1. that our model masters principal swarm mechanisms, such as contraction and rapid expansion,
2. that an arrangement of 29 virtual sensors is sufficient to detect other objects and initiate simple behavioural reactions,
3. and finally that a neural controlling behaviour is capable of implementing these two basic behavioural states and that all intermediate states can eventually be realized.

Our model requires greater differentiation for more extensive simulation studies. Therefore, we consider this approach rather as a “proof of concept” so that the principles we have introduced may be helpful for such modelling.
Of course we were interested in how large simulated swarms can be in order to still be moved fluidly on a standard PC. We have therefore increased the number of agent configurations and determined the time the system requires for an interval step of all agents. In one interval step all agents are considered one after another. The objects in view are determined and the resulting sensory input values are given to the agent. The resulting network reaction is stored in an

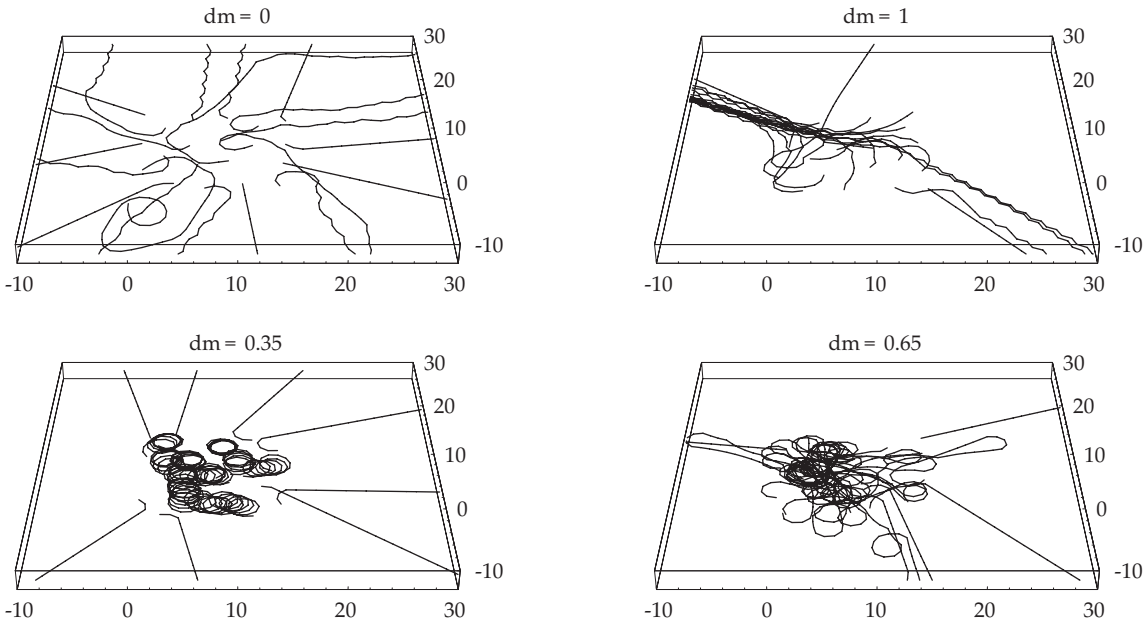


Figure 7: Trace lines after a start with coincidentally selected positions (it are represented the first 50 iteration steps in each case).

intermediate variable. The agent is then reset and the same process takes place for the next agent. We have tested this on a relatively new PC with an Intel Core i5 CPU with 2.67 GHz on a system with Windows 7 (Windows Performance Index 5.9).
The runtimes specified in Table 3 must be considered as approximate values, since Windows 7 is not a real-time system and therefore execution times are not guaranteed. The runtime indications stated were determined by the function *QueryPerformanceCounter()* where at least 200 cycles were recorded. The data indicate that at 500 agents the real-time display starts to falter. However we were not striving for a high real-time performance, otherwise we would have connected the tasks in parallel, i.e. have put every agent into a separate thread, for example, and hence taken advantage of modern multi-core processors. By that alone one could have increased the speed by a factor of 4.
Figure 8 shows a swarm with 2,000 agents, which are moving freely in space. Here the motivational signal $d_m = 0.5$ is selected and the agents form a vortex in which they circle one another.
All simulation results are provided at our web site[5]. This includes the source code of the simulation program, a binary installer for Windows, the SNNS setup files and screen shots of running simulations.

4. Prospects

Naturally it is essential to increase the number of sensors in order to conduct a more differentiated examination of swarms. In addition the visible area must be expanded and conducted partly to the rear. It is also questionable whether capturing the closest object only is really sufficiently plausible. Objects which are further away along the same optical axis represents a useful piece of information, that should not be ignored. However, a compound eye does not have a lens mechanism which allows it to transmit information regarding the relative displacement of the object in focus whilst

[5] *Source file and material of simulations, http://kramper.de/swarms*

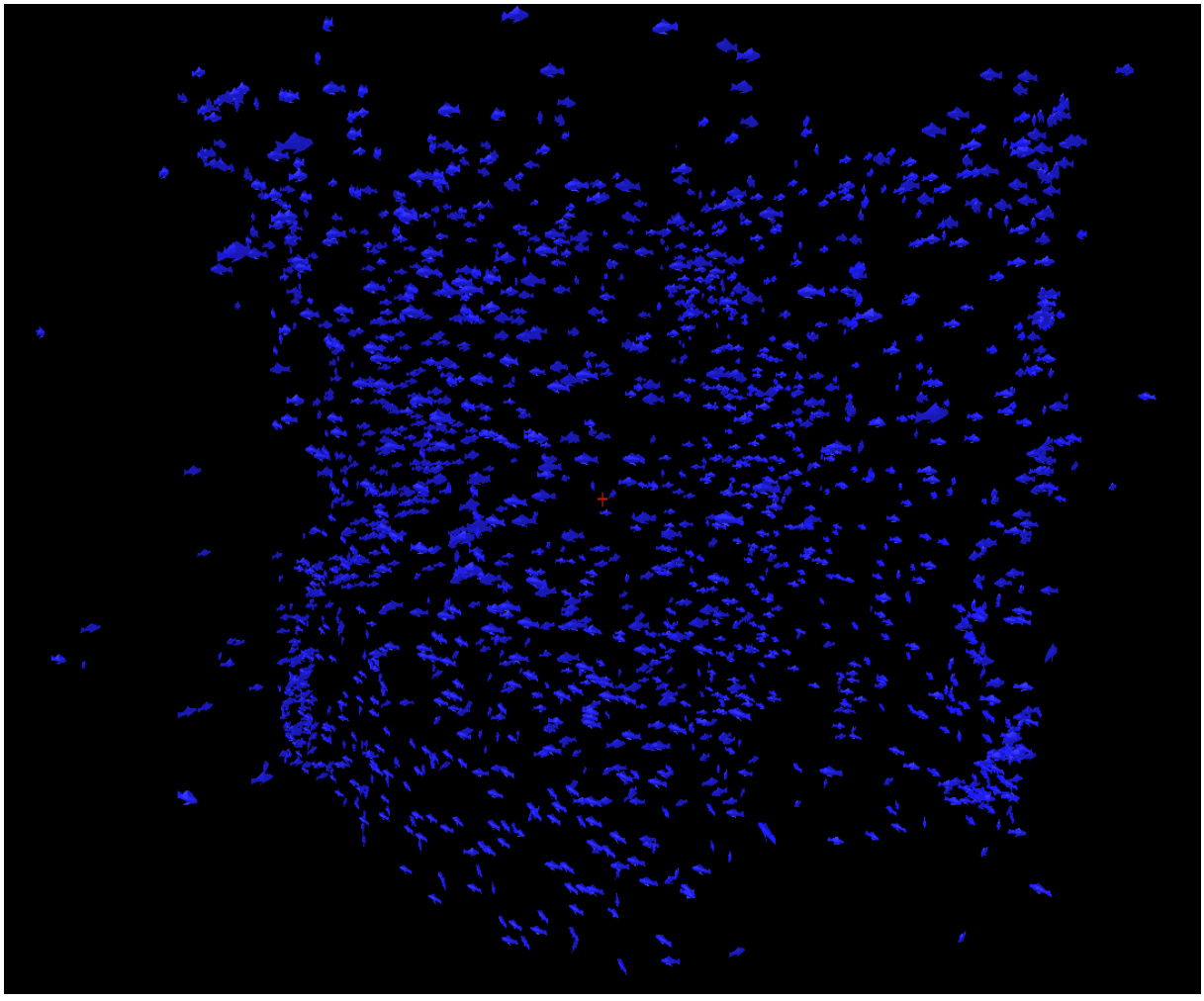


Figure 8: Swarm with 2,000 agents with $d_m = 0.5$.

focussing. Typical swarm animals like fish or birds have a lens eye and can differentiate between the distances of objects in the same line of sight.

Naturally a differentiated behavioural model also requires an extensive neural control system. The two conditions we implemented are insufficient for this purpose. The cross section of a compound eye in Figure 1a shows the visual processing steps in the lower right part of the image. The optical information is transmitted from the photoreceptors via neuron clusters which are positioned one after another. We could imagine a model with a cascade of neural networks, which transmit the optical sensor information to the connecting level, depending on controlling input neurons. Every processing level in this cascade model would react to certain visual patterns and initiate behavioural reactions accordingly.

Such a separate processing unit would also implement the motivational signal, which was controlled by the user in our model. This network would for example, detect the presence of an enemy or an attacker. If the attacker is far away, it would emit a lower value of d_m and the agent would circle this value in large radii, however on approach of the attacker this value would swell, leading to contraction. Thus the swarm model would simulate an attacker-prey situation. With a differentiated behavioural control system, avoidance behaviour as observed in nature might be possible.

References

[1] Au G., Goss S., Heinze C., Pearce A.R., RescueModel: A multi-agent simulation of bushfire disaster management, LNCS, 2019, 285-290, 2001

[2] Blum C., Merkle D., Swarm Intelligence – Introduction and Applications, Springer, New York, 2008

[3] Braitenberg V., Vehicles: Experiments with Synthetic Psychology, MIT Press, Cambridge, MA, 1984

[4] Chalmer, D.J., Clayton P., Davis P., Strong and weak emergence, The Re-Emergence of Emergence, Oxford Univ. Press, Oxford, 2006

[5] Chivers D.P., Brown G.E., Smith R.J.F., Familiarity and shoal cohesion in fathead minnows Pimephales promelas: implications for antipredator behavior, Can. J. Zool., 73, 955-960, 1995

[6] Couzin I.D., Krause J., James R., Ruxton G.D., Franks R.N., Collective Memory and Spatial Sorting in Animal Groups J. Theor. Biol., 218, 1-11, 2002

[7] Ebling M., Di Loreto M., Presley M., Wieland F., Jefferson D., Ant foraging model implemented on the time warp operating system, Simulation Series, 21, 21-26, 1989

[8] Freeman J.A., Skapura D.M., Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, Reading, MA, 1991

[9] Griffiths S.W., Brockmark S., Höjesjö J., Johnsson J.I., Coping with divided attention: the advantage of familiarity, Proc. Biol. Sci., 271, 695-699, 2004

[10] Heinze C., Goss S., Josefsson T., Bennett K., Lloyd I., Murray G., Oldfield J., Interchanging agents and humans in military simulation, AI Magaz., 23, 37, 2002

[11] Hemelrijk C.K., Hildenbrandt H., Self-organized shape and frontal density of fish schools, Eth., 114, 245-254, 2008

[12] Hemelrijk C.K., Kunz H., Density distribution and size sorting in fish schools: an individual-based model, Behav. Ecol., 16, 178-187, 2005

[13] Heppner F., Grenander U., Krasner S., A stochastic nonlinear model for coordinated bird flocks, The Ubiquity of Chaos, AAAS Publications, Washington, DC, 1990

[14] Hiramatsu K., Shikasho S., Mori K., Mathematical modeling of fish schooling of Japanese medaka using basic behavioral pattern, J. Faculty Agricult., 45, 237-253, 2000

[15] Hoare D.J., Couzin I.D., Godin J.-G.J., Krause J., Context-dependent group size choice in fish, Anim. Behav., 67, 155-164, 2004

[16] Höjesjö J., Johnsson J.I., Petersson E., Jarvi T., The importance of being familiar: individual recognition and social behavior in sea trout Salmo trutta, Behav. Ecol., 9, 445-451, 1998

[17] Huth A., Wissel C., The simulation of the movement of fish schools, J. Theor. Biol., 156, 365-385, 1992

[18] Huth A., Wissel C., The simulation of fish schools in comparison with experimental data, Ecol Modell., 74-75, 135-146, 1994

[19] Kandel E.R., Biology of Aplysia, A Contribution to the Comparative Study of Opisthobranch Molluses, W. H. Freeman and Company, San Francisco, 1979

[20] Kennedy J., Eberhardt R., Particle swarm optimization, Proc. IEEE Int. Conf. Neural Network, 1942-1948, Piscataway, NJ, 1995

[21] Kramper W., Simulation von Schwarmverhalten, Mensch & Buch, Berlin, Hamburg University of Technology, 2010

[22] Kunz H., Hemelrijk C.K., Artificial Fish Schools: Collective Effects of School Size, Body Size, and Body Form, Artif. Life, 9, 237-253, 2003

[23] Neumann T.R., Modelling Insect Compound Eyes: Space-Variant Spherical Vision, In: Proceedings of the 2nd International Workshop on Biologically Motivated Computer Vision, Springer-Verlag, Berlin, 2002

[24] Olfati-Saber R., Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory, IEEE Trans. Autom. Control, 2004

[25] Parrish J.K., Viscido S.V., Hemelrijk C.K., Traffic rules of fish Schools: A review of agent-based approaches, Self-Organization and Evolution of Social Behavior, Cambridge Univ Press, Cambridge, 2005

[26] Partridge B.L., Pitcher T., Cullen J.M., Wilson J., The three-dimensional structure of fish schools, Behav. Ecol. Sociobiol., 6, 277-288, 1980

[27] Reynolds C., Flocks, Herds, and Schools: A Distributed Behavioral Model In: Proceedings of SIGGRAPH '87, 21, 25-34, 1987
[28] Romey W.L., Individual differences make a difference in the trajectories of simulated schools of fish, Ecol. Modell., 92, 65-77, 1996
[29] Sahli N., Moulin B., Ekemas, Appl. Intell., 31, 188-209, 2009
[30] Shreiner D., Woo M., Neider J., Davis T., OpenGL Programming Guide, Addison Wesley, Amsterdam, 2007
[31] Snyder A.W., Acuity of compound eyes: Physical limitations and design, J. Comp. Physiol., 116, 161-182, 1977
[32] Swaney W., Kendel J., Capon H., Brown C., Laland K.N., Familiarity facilitates social learning of foraging behavior in the guppy, Anim. Behav., 62, 591-598, 2002
[33] Toner J., Tu Y., Flocks, herds, and schools: A quantitative theory of flocking, Am. Phys. Soc., 58, 4828-4858, 1998
[34] Ward A.J.W., Hart P.J.B., Foraging benefits of shoaling with familiars may be exploited by outsiders, Anim. Behav., 69, 329-335, 2005
[35] Weber H., Grundrißder Insektenkunde, Gutav Fischer Verlag, Stuttgart, 1974
[36] Wood A.J., Ackland G.J., Evolving the selfish herd: emergence of distinct aggregating strategies in an individual-based model, In: Proceedings of the Royal Society, 274, 1637-1642, 2007
[37] Zell A., Simulation neuronaler Netze, R. Oldenbourg Verlag, München Wien, 1997