Collective Cognition & Sensing in Robotic Swarms via an Emergent Group-Mind

Michael Otte

Aerospace Engineering Sciences Dept.
University of Colorado at Boulder

Abstract. Algorithms for robotic swarms often involve programming each robot with simple rules that cause complex group behavior to emerge out of many individual interactions. We study an algorithm with emergent behavior that transforms a robotic swarm into a single unified computational meta-entity that can be programmed at runtime. In particular, a swarm-spanning artificial neural network emerges as wireless neural links between robots self-organize. The resulting artificial group-mind is trained to differentiate between spatially heterogeneous light patterns it observes by using the swarm's distributed light sensors like cells in a retina. It then orchestrates different coordinated heterogeneous swarm responses depending on which pattern it observes. Experiments on real robot swarms containing up to 316 robots demonstrate that this enables collective decision making based on distributed sensor data, and facilitates human-swarm interaction.

Keywords: Robotic Swarm, Group Mind, Neural Network, Emergent Behavior, Coordination, Distributed Sensing, Multi-Agent System.

1 Introduction, Motivation, Related Work

The human brain is composed of roughly 10^{15} connections between 10^{11} neurons that self-assemble during development as cells respond to local stimuli [1]. While this feat is impressive in the physical sense, it is also compelling from a computational point-of-view. Human and animal societies containing upwards of 10^9 individuals also spontaneously form, have collective intelligence [2], and exert globe-changing collective behaviors that result as the product of countless local interactions [3]. In many ways societies are meta-organisms [4] in which individuals take the roll of cells and communication substitutes for neural connectivity. Science *fiction* authors [5–7] have taken this analogy further — imagining that a group of individuals linked by neural connections might form a collective "group-mind" defined by shared awareness, pooled computational resources, etc. *Non*fictional artificial neural networks (ANNs) were inspired from biology and developed in the 1950s [8], and have proved capable of learning complex tasks across many domains [9–11]. We combine ANNs with swarm robotics and wireless communication to produce an *artificial group-mind* in which a swarm of autonomous robots spontaneously self-assembles into a single computational entity. The resulting entity is trained to distinguish between various heterogeneous

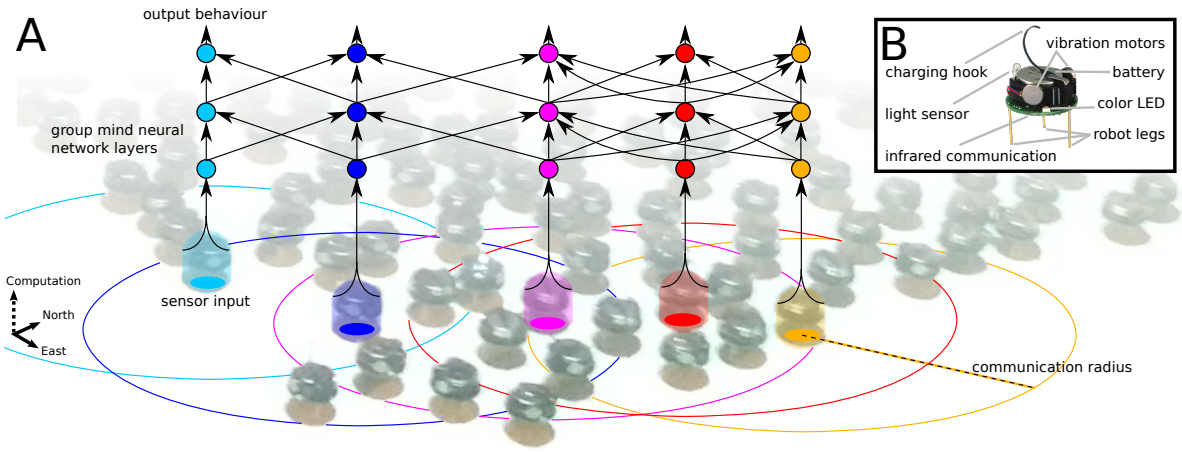

Fig. 1. Emergent group-mind neural network. (**A**) Each robot maintains a slice of neurons (depicted along the vertical axis) and forms neural connections with its neighbors. (**B**) The Kilobot robot that we use.

patterns in the global environment — using the swarm's distributed sensors much like the retina cells in an eye — and then coordinates a collective yet heterogeneous swarm response based on the specific pattern it observes.

Swarm roboticists [12, 13] often design algorithms that leverage *emergence* [14, 15], the idea that simple behaviors performed in concert by a group of interacting individuals can produce complex global behaviors like bird flocking [13, 16], termite nest construction [17], food foraging [18], and collective transport [19, 20]. In robotic self-assembly the emergent product is a robotic megastructure created as robots physically arrange themselves within the environment [21]. Our emergent product is the artificial group-mind itself — a *computational entity* that can be programmed by a human user at run-time to perform explicit complex tasks. "Mind vs. body" is an appropriate metaphor for the distinction between an artificial group-mind and a self-assembling robot; the distinction vs. other emergent swarm behaviors like foraging is akin to that between the nervous system and other distributed bodily systems, e.g., the immune system. Although a robotic swarm is a natural host for an artificial group-mind, intelligent materials [22, 23] could also be used.

2 Technical Approach

The swarm consists of 3.3 cm Kilobots [24] (see Fig. 1). Kilobots locomote via vibration, communicate wirelessly using infrared light (range 10 cm), have ATMega328 microprocessors (8 Mhz 32K memory), visual light-intensity sensors, and a multi-color light emitting diode (LED). A digital light projector mounted above the environment controls environmental light intensity patterns by projecting 50 by 50 pixel grayscale images onto the swarm. Light projections are

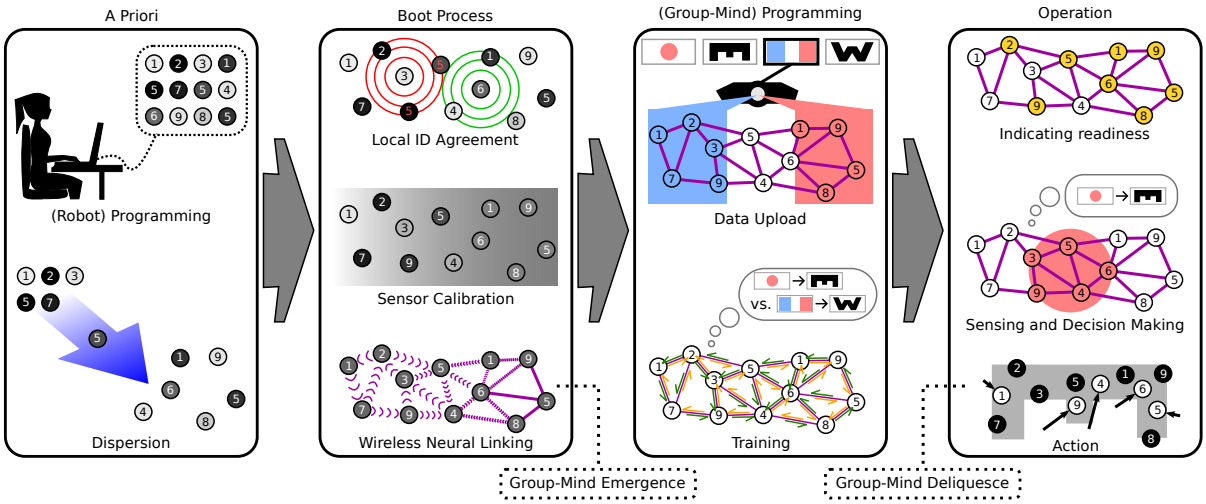

Fig. 2. The group-mind emerges as robots form neural connections with communication neighbors. It is trained to differentiate between global environmental patterns (represented here by the flags of Japan and France) detected across the swarm's collective sensors. The output behavior of the swarm (represented by shapes 'E' vs. 'W') depends on the pattern (flag) that is observed.

used both for human-to-swarm communication and also to create various global light patterns which the swarm is trained to differentiate.

Use of the artificial group-mind is depicted in Fig. 2; pseudocode appears in the Appendix. Robots receive identical individual programming and are distributed in the environment a priori. Although distribution could be accomplished autonomously, see e.g., [24], we manually place robots in the environment to conserve battery life and memory space. The training algorithm assumes that no robot has two or more neighbors with the same identification number. A distributed algorithm is used to ensure this happens with probability 1. The swarm's light sensors are calibrated to correct for imperfections in sensitivity and systematic differences in brightness caused by projector distance.

Two sets of projected images are used to upload data to the swarm. The first is a set of raw environmental feature patterns. The second is a corresponding set of swarm behavioral response patterns. In general, images in both sets may be spatially heterogeneous. Behaviors are encoded via a predefined mapping from normalized grayscale values, and may include simple or complex functionality such as "display red LED" or "move toward brighter light." The group-mind learns to differentiate between the environmental feature patterns during a training phase. After the training phase, the swarm will perform the corresponding behavioral response for whatever feature pattern the group-mind currently sees in the environment.

The artificial group-mind is a swarm-spanning feed-forward ANN that emerges as robots form ongoing wireless neural communications with their neighbors. It

uses a slice-parallel implementation of the backpropogation ANN training algorithm [25] that we have specially modified for use with unreliable wireless communication. Conventional ANNs assume reliable communication, which wireless is not. Instead, we temporarily pause training on any robot that becomes more than B training iterations out-of-sync with any of its neighbors, until those neighbors reestablish communication and catch up. $0 < B < \infty$ is a predefined constant and $B = 100$ in experiments.

Each robots is responsible for maintaining a slice of L neurons within the group-mind (Fig. 1), where L is defined *a priori*. $L = 2$ in experiments, i.e., there is one hidden layer. Connections are established from neurons at layer ℓ on a robot i to those at layer $\ell + 1$ on each of its neighbors $j \in N_i$ for $0 \leq \ell < L$, where N_i is the neighbor set of i, and i is considered a neighbor of itself ($i \in N_i$). The signal values at layer 0 are set by the real-time environmental sensor (light sensor) readings. The output signal from layer L on a particular robot i is used to determine i's behavior at run-time. As with standard ANNs, each neuron's output is calculated by performing a weighted sum over incoming signals and then passing the result through a step-like function (the hyperbolic tangent from [26] is used in our experiments). Training the group-mind via the backpropogation training algorithm involves adjusting link weights to improve performance vs. the training example set. This involves sending update messages in the backward direction. Updates from the final layer L contain the signal error for each training example, and those from internal layers $\ell < L$ encode the cumulative error at L ascribed to local error at ℓ. Given updates from N_i, a node i at layer ℓ can adjust the weights assigns incoming neural signals such that overall ANN performance improves. This results in a form of gradient descent. In practice, a Robot stops training once its local error has fallen below a predefined threshold (5%).

3 Experiments

We experiment with swarms of varying size and two different classes of swarm response behavior (stationary vs. moving robot positions). Tuning parameters are chosen using a tuning data set that is different from the test data set.

Figure 2 shows experiments in which the collective responses involve displaying 2-D color images across the swarm's distributed LED array. Robot behaviors include display "red LED," "blue LED," and "off LED." Thus, the swarm is able to display a yin-yang, wifi symbol, etc., by having different robots perform different behaviors. In these experiments, the swarm may safely use its group-mind as it is being trained, allowing direct assessment of the group-mind's training status via its improving collective behavior. Figure 3 depicts a set of experiments in which the response behaviors require physical movement in order to create one of two different shapes (blue smiley face or red frown face) depending on which environmental feature pattern is observed at run-time (peace and biohazard symbols, respectively). Robot behaviors include: "Random-Search," "Red-Attract," "Blue-Attract," and "Continue-Training." Red-Attract and Blue-Attract cause

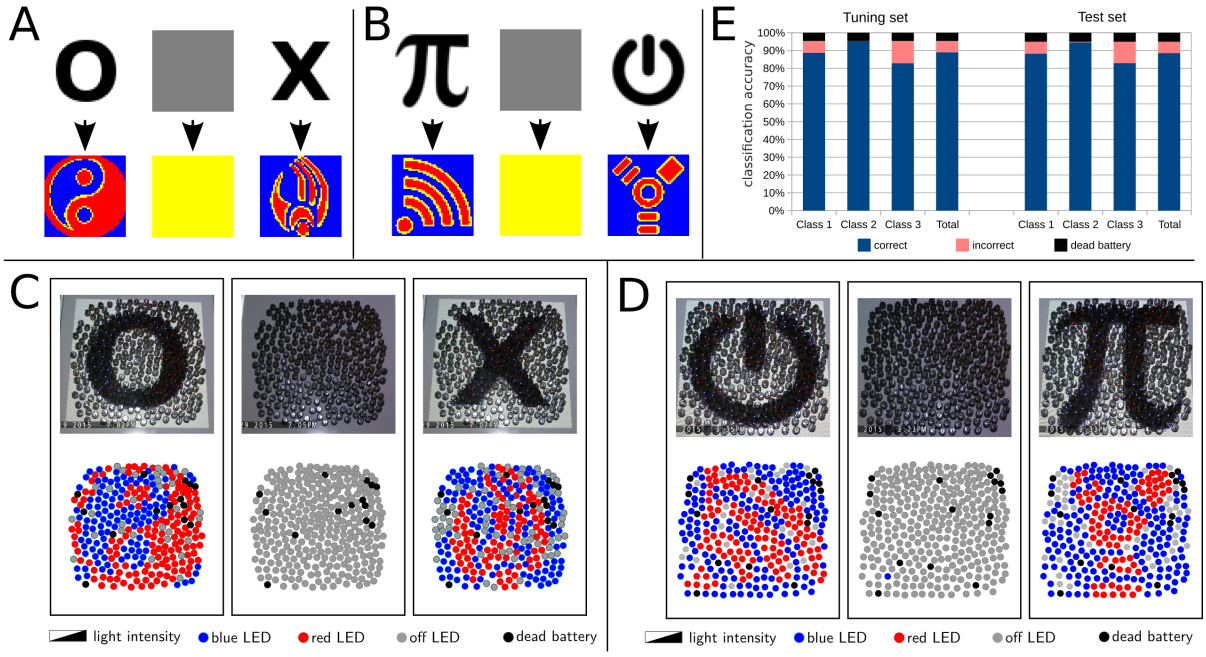

Fig. 3. Experiment without movement. The light pattern tuning set (**A**) and test set (**B**) used for experiments with 303 and 316 robot swarms, respectively. Top rows are raw light intensity feature patterns and bottom rows the corresponding output behaviors (represented by color) the group-mind must learn to perform in response. Feature patterns projected onto the swarm (**C**,**D**) for tuning and test cases, and the behavior that was actually performed as a result. Classification accuracy vs. different input patterns (**E**).

a robot to broadcast "Attract" messages while remaining stationary and displaying red or blue LEDs, respectively. A robot performing Random-Search will move around the environment at random until receiving an "Attract" message sent from closer than 5cm, in which case it halts and displays a white LED. Physical shapes emerges as Random-Search robots move from their original positions to fill the space around attracting robots (or leave the environment).

The "Continue-Training" behavior causes a robot to continue training until its training error has fallen below 5%, and then to display a yellow LED. By training the group-mind to "Continue-Training" in response to a (uniform medium-gray) pattern displayed during training, the overall group-mind training status can be evaluated by observing the proportion of the swarm displaying yellow LEDs.

Physical movement breaks neighborhood connectivity which causes the group-mind to dissolve. Thus, the group-mind must coordinate an organized deliquesce prior to the start of movement. Each robot i continually evaluates the group-mind's calculation of i's output behavior based on the real-time distributed sensor data. If this behavior is not "Continue-Training" for more than a predefined

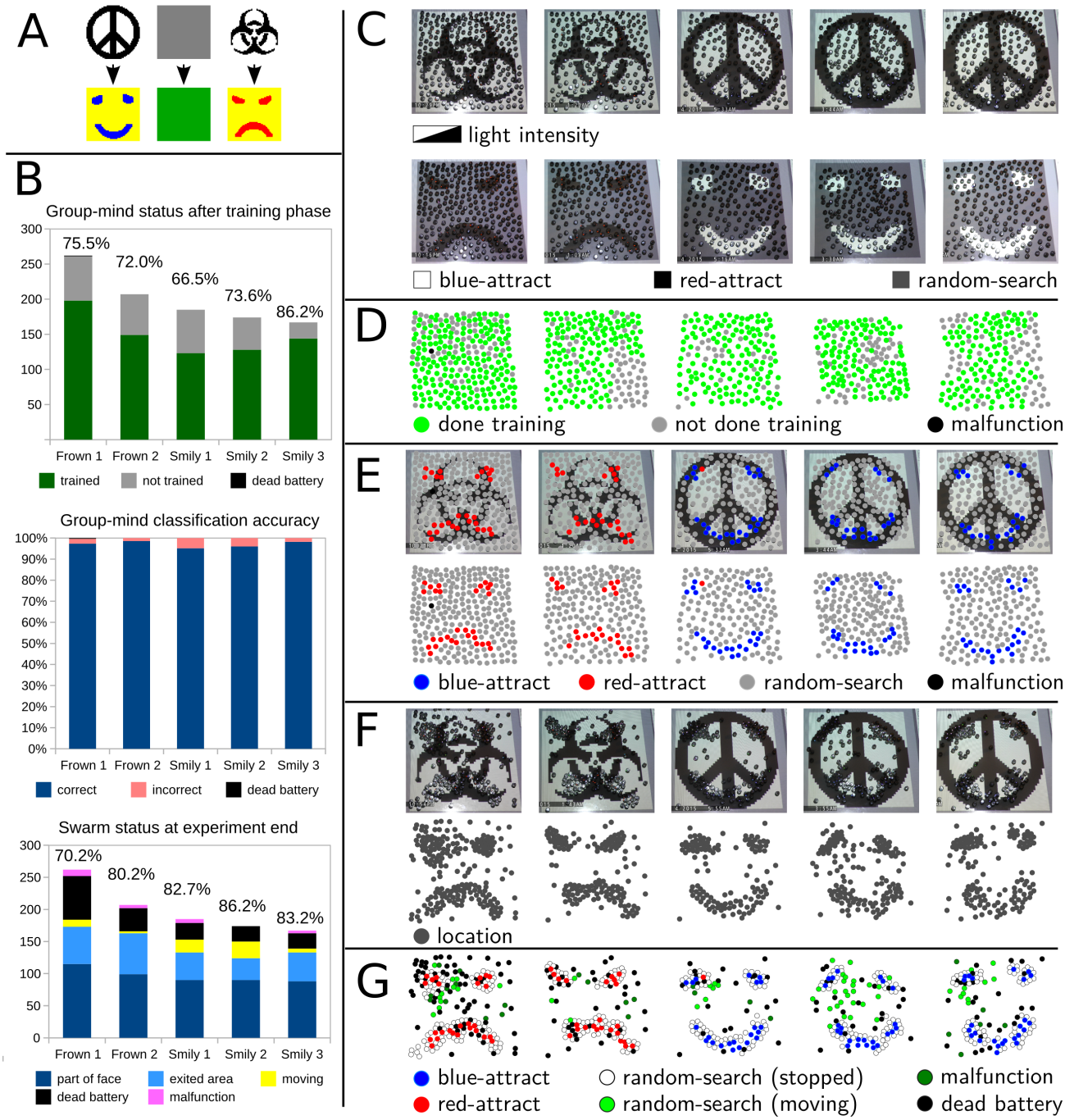

Fig. 4. Experiments with movement. Light intensity pattern test set (**A**) for experiments with movement. (**B**) Results (top to bottom): the swarm's training status when movement started, classification accuracy vs. the raw light intensity pattern that initiated movement, and the breakdown of the swarm's behavior at experiment end. (**C-G**) Columns correspond to experiments. (**C**) Training data (light intensity and output behavior pattern, top and bottom) for the behavior that was eventually chosen. (**D**) Training status when movement started. (**E**) real-time light intensity feature data and resulting output behavior (top, top and bottom). (**F**) Swarm position at experiment end. (**G**) Swarm behavior at experiment end.

length of time (30 seconds), then robot i begins performing the prescribed behavior while broadcasting messages indicating the pattern detected. Any robot $j \neq i$ in a poorly trained subset of the group-mind can calculate its own behavior by combining the data from i's message with its own behavior map. j then performs the appropriate behavior and re-broadcasts the message from i.

4 Main Experimental Insights

Drained batteries were an unexpected difficulty. This problem could be minimized by replacing batteries prior to an experiment and/or modifying the hardware or output behaviors to be more power efficient. However, one reason for using swarms is their robustness vs. partial loss. In experiments where color LED images were the desired output, a dead battery simply meant that a particular robot's LED was dark. In experiments with movement, the desired output shape was discernible despite moderate (up to 26%) loss.

Algorithmically, if a robot looses power during the training process, then its neural signals freeze from a neighbor's point-of-view. If such a frozen signal is detrimental to a neighbor's neural performance then it will be weighted less-and-less over time. However, because robot i to pauses training after becoming B iterations out-of-sync with j, power loss on one robot can potentially pause training across the entire swarm. Although a full-scale failure was not observed in the experiments, this is clearly a weakness of our algorithm. Neighbor pruning could potentially alleviate this problem. For instance, perpetually uncommunicative neurons could have their last known signals treated as fixed input by their neighbors and then subsequently ignored. Although this would technically break the theoretical convergence guarantees of our modified backpropogation training algorithm, these guarantees are also broken whenever a robot becomes permanently uncommunicative (and thus forfeit in the event of power loss anyway).

5 Results

We have performed a variety of experiments on real robot swarms containing up to 316 robots. These provide proof-of-concept that an artificial group-mind can emerge as the result of distributed computation across a robotic swarm, is a useful tool for human-swarm interaction, and enables fine-grained heterogeneous swarm behavior to be programmed at run-time and at a high level by a human user. In particular, the group-mind is capable of detecting and classifying heterogeneous feature patterns across the global environment, and orchestrating a collective heterogeneous swarm response. The simple behaviors exhibited in the experiments could easily be replaced by more sophisticated behaviors with no change to the training and decision making algorithms. Other environmental features (chemical, temperature, acoustic, etc.) could easily be used in place of light intensity. The ad-hoc process in which neural connections form in the group-mind is a departure from traditional ANNs, and echoes similar emergent neural linking in the animal brain.

Acknowledgments. This work would not have been possible without Michael Rubenstein, who designed the Kilobot platform, taught the author how to use it, and provided invaluable feedback on this work. The author is grateful for the knowledge, resources, encouragement, and advice provided by Radhika Nagpal and Melvin Gauci. The author are also grateful to Derek Kingston for providing the time, space, and freedom to pursue this problem. This work was funded by the Control Science Center of Excellence at the Air Force Research Laboratory (CSCE AFRL), the National Science Foundation (NSF) grant IIP-1161029, and the Center for Unmanned Aircraft Systems. This work was performed while Michael Otte was "in residence" at CSCE AFRL.

References

1. Chialvo, D.R.: Emergent complex neural dynamics. Nature physics **6**(10) (2010) 744–750
2. Lévy, P.: L'intelligence collective: pour une anthropologie du cyberspace. Volume 11. La Découverte Paris (1994)
3. Green, D.G.: Emergent behavior in biological systems. In: D.G. Green and T.J. Bossomaier (Editors), Complex Systems: From Biology to Computation. IOS, IOS Press (1993) 24–35
4. Wheeler, W.M.: The ant-colony as an organism. Journal of Morphology **22** (1912) 307
5. Stapledon, O.: Last and first men : a story of the near and far future. Penguin Books London (1937)
6. Asimov, I.: Foundation and Earth. Foundation series. Doubleday (1986)
7. Berman, R., Piller, M., Taylor, J., Taylor, M., Price, A.S., Gaberman, M.: Collective. Star Trek Telivision Series Episode VOY.6.16, Directed by Allison Liddi, Based on Concept by Gene Roddenberry (February 2000)
8. Farley, B., Clark, W.: Simulation of self-organizing systems by digital computer. Transactions of the IRE Professional Group on Information Theory **4**(4) (September 1954) 76–84
9. Pomerleau, D.A.: Efficient training of artificial neural networks for autonomous navigation. Neural Computation **3**(1) (1991) 88–97
10. Cireşan, D., Meierand, U., Gambardella, L., Schmidhuber, J.: Deep, big, simple neural nets for handwritten digit recognition. Neural Computation **22**(12) (2010) 3207–3220
11. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540) (2015) 529–533
12. Amkraut, S., Girard, M., Karl, G.: Motion studies for a work in progress entitled eurnythmy. SIGGRAPH Video Review **21** (1985)
13. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. Computer Graphics **21**(4) (1987)
14. Matarić, M.J.: Interaction and Intelligent Behavior. PhD thesis (1994)
15. Martinoli, A.: Swarm intelligence in autonomous collective robotics: From tools to the analysis and synthesis of distributed control strategies. PhD thesis, Ecole Polytechnique Fédérale de Lausanne (1999)

16. Ferrante, E., Turgut, A.E., Huepe, C., Stranieri, A., Pinciroli, C., Dorigo, M.: Self-organized flocking with a mobile robot swarm: a novel motion control method. Adaptive Behavior (2012)
17. Werfel, J., Petersen, K., Nagpal, R.: Designing collective behavior in a termite-inspired robot construction team. Science **343**(6172) (2014) 754–758
18. Steels, L.: Cooperation between distributed agents through self-organisation. In: Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop on. (Jul 1990) 8–14 supl
19. Becker, A., Habibi, G., Werfel, J., Rubenstein, M., McLurkin, J.: Massive uniform manipulation: Controlling large populations of simple robots with a common input signal. In: Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. (Nov 2013) 520–527
20. Wilson, S., Pavlic, T.P., Kumar, G.P., Buffin, A., Pratt, S.C., Berman, S.: Design of ant-inspired stochastic control policies for collective transport by robotic swarms. Swarm Intelligence **8**(4) (2014) 303–327
21. Groß, R., Bonani, M., Mondada, F., Dorigo, M.: Autonomous self-assembly in swarm-bots. Robotics, IEEE Transactions on **22**(6) (2006) 1115–1130
22. Gilpin, K., Knaian, A., Rus, D.: Robot pebbles: One centimeter modules for programmable matter through self-disassembly. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, IEEE (2010) 2485–2492
23. Butera, W.J.: Programming a paintable computer. PhD thesis (2002)
24. Rubenstein, M., Cornejo, A., Nagpal, R.: Programmable self-assembly in a thousand-robot swarm. Science **345** (2014) 795–799
25. Farber, P., Asanovic, K.: Parallel neural network training on multi-spert. In: Algorithms and Architectures for Parallel Processing, 1997. ICAPP 97., 1997 3rd International Conference on. (Dec 1997) 659–666
26. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.R.: Efficient backprop. In: Neural networks: Tricks of the trade. Springer Berlin Heidelberg (2012) 9–48

Appendix: High-level Pseudo Code

Each robot in the swarm runs identical code. Two different "main" procedures are presented. The first is for situations in which the output behavior of the swarm does not involve movement or other actions that will break the group mind's network connectivity (Algorithm 2). The second is for situations in which the output behavior is expected to break connectivity, and so the group mind must organize an orderly dissolution back to a non-group-mind swarm (Algorithm 3). In addition to the main thread, each robot runs a separate message broadcast thread at approximately 2 Hz (Algorithm 4), and has a callback function to receive incoming messages (Algorithm 5), respectively. Global data is accessible across all threads and functions.

The start-up procedure appears in Algorithm 1 and corresponds to the steps in Figure 2 between "Local ID Agreement" and "Data Upload." Each robot uses a state machine that is initialized to state NOT_YET_TRAINING (line 1). A Boolean value *done_training* is also used to track when training has resulted in an acceptable level of accuracy (on this robot). The battery charge is used to seed a pseudo-random number generator so that different pseudo-random number sequences will be generated on each robot with high probability. A distributed

Algorithm 1: startup_phases()

```
1 state ← NOT_YET_TRAINING
2 done_training ← false
3 seed random number generator with battery charge
4 agree on unique local_id vs. neighbors
5 calibrate light sensors
6 initialize neural network with random weights
7 upload training data
8 state ← TRAIN
```

Algorithm 2: Group Mind Main Thread (without movement)

```
1 startup_phases()
2 loop
3     if training_error() > 5% and not (done_training or out_of_sync()) then
4         backpropagation_training_iteration()
5     else if training_error() ≤ 5% and not out_of_sync() then
6         done_training ← true
7     {τ, behaviour} ← use_group_mind(sample_light())
8     perform(behaviour)
```

algorithm is used to ensure that neighboring robots have unique randomly determined IDs (line 4). Light sensors are calibrated (line 5). Neighbors are discovered and outgoing wireless links to their neurons are created and initialized with random weights (line 6). Data is uploaded to the swarm from a human user via visual light projection following a predefined procedure (line 7). State TRAIN indicates the start-up phase has ended (line 8).

The main thread for non-movement cases appears in Algorithm 2. All signals sent along neural connections are tagged with the number of training iterations this robot has completed. The function out_of_sync() returns **true** whenever this robot has gotten too many training iterations ahead of its neighbors (100 in our experiments). The backpropogation training algorithm is run one iteration at a time (line 4) — but only if the training error needs improvement and this robot is not out-of-sync with its neighbors (line 3). A robot stops training once its local error has fallen below 5% (lines 5-6). This robot uses the subroutine use_group_mind(sample_light()) to both provide its current light sensor reading to the group mind, and to learn the group mind's prediction of the overall swarm behavior τ that should be performed (line 7). The single robot behavior *behaviour* this robot performs as part of τ is also returned, and determined within use_group_mind(sample_light()) by querying a local look-up table with the value of τ. The look-up table is populated with the local mapping from τ to *behaviour* during the data upload portion of the start-up phase.

The main thread used in cases involving movement appears in Algorithm 3. Differences vs. Algorithm 2 (no movement) appear on lines 3 and 8-15. Movement

Algorithm 3: Group Mind Main Thread (with movement)

```
1  startup_phases()
2  loop
3      if state ∈ {TRAIN, CONSIDER} then
4          if training_error() > 5% and not (done_training or out_of_sync()) then
5              backpropagation_training_iteration()
6          else if training_error() ≤ 5% and not out_of_sync() then
7              done_training ← true
8          {τ, behaviour} ← use_group_mind(sample_light())
9          if (done_training and behaviour ≠ CONTINUE_TRAINING) or
           state = CONSIDER then
10             state ← CONSIDER
11             if consideration_time_exhausted() then
12                 state ← ACT
13     else if state = ACT then
14         perform(behaviour)
```

Algorithm 4: Group Mind Send Message Thread

```
1  loop
2      if state ∈ {TRAIN, CONSIDER} then
3          neural_data ← get_neural_data()
4          message ← {state, local_id, neural_data}
5      else if state = ACT then
6          message ← {state, local_id, τ}
7      else
8          populate message as required for calibration, initialization, etc.
9      broadcast(message)
```

destroys the group mind; thus, movement should only start once the group-mind is highly certain it has calculated the correct response behavior. This is facilitated by adding state CONSIDER to the state machine, and also by defining one of the behaviors to be "continue training." In practice, the swarm is trained to continue training in response to a neutral gray light pattern, which is then displayed during the training phase. CONSIDER can only be accessed once a robot believes the desired behavior is no longer "continue training" (lines 9-12). The function consideration_time_exhausted() is used to ensure a robot remains continuously in state CONSIDER for a predetermined amount of time before switching to state ACT to perform the prescribed behavior (lines 11-14). This adds robustness to erroneous outputs from partially trained models.

Algorithm 4 depicts the message broadcast thread. Function get_neural_data() retrieves the neural network data that resides on this robot's portion of the group

Algorithm 5: Group Mind Receive Message Callback

```
1  if state ∈ {TRAIN, CONSIDER} then
2      sender_state ← message{1}
3      if sender_state ∈ {TRAIN, CONSIDER} then
4          {sender_local_id, neural_data} ← message{2 : 3}
5          update_group_mind(sender_local_id, neural_data)
6      else if sender_state = ACT then
7          state ← ACT
8          τ ← message{3}
9          behaviour ← behaviour_map(τ)
10 else if state = ACT then
11     else if sender_state = ACT then
12         {sender_local_id, sender_behaviour} ← message{2 : 3}
13         sender_distance ← calculate_distance(message)
14         behaviour ← modify_behaviour(behaviour, sender_behaviour, sender_distance)
15 else
16     use message for calibration, initialization, etc.
```

mind (line 3). For each training example as well as the real-time environmental sensor input, this includes both the forward neural signals and backpropogation messages (including training iteration number and, for each backpropogation message, the destination ID). Neural data is broadcast, along with this robot's state and ID (line 4). In practice, due to the Kilobots' small message payload size (9 bytes), we must divide each batch of neural network data across multiple messages (not shown). If there is movement behavior such that state ACT is used, then the robot sends this state, its ID, and the swarm behavior class τ output of the neural network vs. real-time environmental data (lines 5-6). To save space we omit the other message passing details necessary to run the standard distributed algorithms that we employ as subroutines during the start-up phase (represented by lines 7-8).

The receive message callback function appears in Algorithm 5. Normal training data is received on lines 2-5. If a neighbor has decided to act (e.g., move) then this robot will join it (lines 6-9); making sure to perform its own prescribed behavior *behaviour* relevant to the overall swarm behavior τ (line 9). The function modify_behaviour(*behaviour*, *sender_behaviour*, *sender_distance*) is used to modify the specific output behavior of this robot during the ACT phase, as a function of interaction with neighboring robots (lines 10-14). This enables more complex swarm behaviors to emerge out of the interactions between robots. For example, the smiley faces in our experiments are created as randomly searching robots stop moving in the vicinity of attracting robots. Lines 15-16 represent other message processing that is used for the distributed subroutines within the start-up phase.