Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Evolution of multisensory integration in large neural fields

Benjamin Inden, Yaochu Jin, Robert Haschke, Helge Ritter

2012

Preprint:

This is an accepted article published in Lecture Notes in Computer Science. The final authenticated version is available online at: https://doi.org/[DOI not available]

LEGAL NOTICE

This accepted author manuscript is only for personal use.
The final publication will be available soon at springerlink.com:

Proceedings of Evolution Artificielle, Lecture Notes in Computer Science 7401, Springer-Verlag 2012, http://www.springerlink.com.

Evolution of multisensory integration in large neural fields

Benjamin Inden[1], Yaochu Jin[2], Robert Haschke[3], and Helge Ritter[3]

[1] Research Institute for Cognition and Robotics, Bielefeld University
binden@cor-lab.uni-bielefeld.de
[2] Department of Computing, University of Surrey
[3] Neuroinformatics Group, Bielefeld University

Abstract. We show that by evolving neural fields it is possible to study the evolution of neural networks that perform multisensory integration of high dimensional input data. In particular, four simple tasks for the integration of visual and tactile input are introduced. Neural networks evolve that can use these senses in a cost-optimal way, enhance the accuracy of classifying noisy input images, or enhance spatial accuracy of perception. An evolved neural network is shown to display a kind of McGurk effect.

1 Introduction

Animals take advantage of different sensory facilities to react appropriately to their environment. Among these facilities are visual, auditory, tactile, olfactory, gustatory, and electric senses. Researchers are typically interested in questions like: How are percepts from different modalities perceived as a single entity? What happens if there is conflicting input from different modalities? How can top-down influences (attention) bias the process of integration? [1,2]

Here we are particularly interested in evolutionary aspects of multisensory integration. Advantages of multisensory integration may include increased spatial accuracy of perception, increased classification accuracy, and minimizing costs associated with different forms of perception. The purpose of this study is to introduce a number of sufficiently understandable tasks that require multisensory integration to be solved, to verify that artificial neural networks can evolve to solve these tasks, and to get a rough look at how these networks do it.

Studying the integration of input from different modalities has already been possible for some time in the context of evolutionary robotics. One important difference between earlier studies and what we present here is that the techniques for evolving neural networks have advanced significantly in the meantime [3]. Previously, only a few contact or distance sensors were typically used. If there was camera input, it was usually pre-processed to reduce its dimension [4]. It is now possible to evolve very large neural networks that take input from not just a few sensors, but from whole sensor arrays, e.g. cameras. The NEAT-fields method used here for neuroevolution is particularly suited to these tasks

because it evolves not just the connectivity between single neurons, but also the connectivity between complete neural fields. How this method solves a range of different problems and how its implementation details and default parameter settings were determined is discussed in detail elsewhere [5,6,7]. We expect that some other recent neuroevolution methods could also solve the tasks presented here. A comparison between NEATfields and these other methods has been done previously [7] and confirms that using NEATfields here is a reasonable choice. Of course, multisensory integration has also previously been studied using large hand designed networks (e.g. [8]). However, an advantage of evolving neural networks is that often solutions are found that are free from human bias, or beyond human imagination.

2 Methods

2.1 The NEATfields method

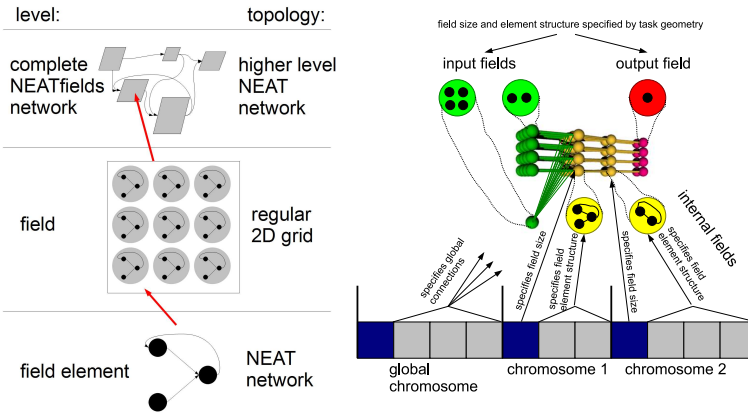

Fig. 1. (a) The three levels of architecture in NEATfields. (b) Construction of a NEATfields network from its genome (bottom). The balls in the central network represent field elements. Their contents are shown in the circles above and below. In these circles, black dots represent the individual (input, output or hidden) neurons.

Neural networks NEATfields is based on the well known NEAT method [9] that simultaneously evolves the topology and the connection weights of neural networks. NEATfields networks have three levels of architecture (see fig. 1 (a)). At the highest level, a network consists of a number of fields that are connected just like individual neurons are in a NEAT network. At the intermediate level, fields are collections of identical (or similar) subnetworks with a two dimensional

topology. At the lowest level, these subnetworks, termed field elements, are recurrent NEAT networks composed of individual neurons. A complete NEATfields network consists of at least one internal field as specified by the genome, and fields for network input and output as specified by the given task. There can be several input and output fields with different dimensions. For example, a bias input can be provided as an input field of size 1×1. Within the NEATfields network, connections can be local (within a field element), lateral (between field elements of the same field), or global (between two fields). It should be noted that connections between field elements or fields are in fact connections between individual neurons in these field elements or fields. How they are established will be described below. Evolution is started with a single internal field of size 1×1 that is connected to all input and output fields. So there is initially only one field element, and it contains one neuron for each different output.

The activation of the neurons in NEATfields is a weighted sum of the outputs of the neurons $j \in J$ to which they are connected, and a sigmoid function is applied on the activation: $o_i(t) = \tanh(\sum_{j \in J} w_{ij} o_j(t-1))$. Connection weights are constrained to the range $[-3, 3]$. There is no explicit threshold value for the neurons. Instead, a constant bias input is available in all networks.

Encoding The parameters for an individual field are encoded in the genome on a corresponding chromosome (see fig. 1 (b)). The first gene in a chromosome specifies the field size in x and y dimensions. After that node and connection genes for one field element are specified. All genes contain a unique reference number that is assigned once the gene is generated through a mutation. In addition, connection genes contain a connection weight, a flag indicating whether the connection is active, and the reference numbers of the source and target neurons.

There is a special chromosome that contains genes coding for global connections. Global connections contain reference numbers of a source and a target. These can be reference numbers either of neurons or of inputs and outputs as specified by the task description. They must be in different fields — global connections between two neurons in the same field are never created by the NEATfields method. Due to the higher level architecture of NEATfields, a neuron with a given reference number will be present n times in a field with n field elements. A single global connection gene implicitly specifies connections for all these neurons. If a global connection is between neurons in fields with the same sizes, every field element in the target field will get a connection from the field element in the source field that has the same relative position (in x and y dimension) in its field. Their connection weights are all the same because they are all derived from one gene. If field sizes are different in a dimension, then the fields will still be connected using a deterministic and topology preserving method (see fig. 2 (a)): if the source field is smaller than the target field, each source field neuron projects to several adjacent target field neurons, whereas if the source field is larger than the target field, the target field neurons get input from a number of adjacent source field neurons, while the genetically specified connection weight

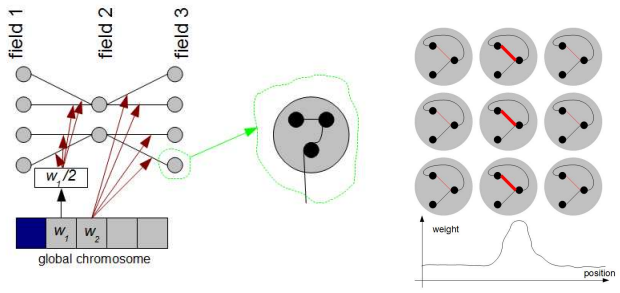

Fig. 2. (a) An example of how the global connections between neurons in fields of different sizes (shown here as one dimensional) are created using a deterministic and topology preserving method. The genetically specified weights are automatically scaled if necessary. As shown in detail on the right side, connections go in fact to individual neurons within the field elements as specified by the connection gene. (b) Dehomogenization of neural fields by the focal areas technique. The thickness of connections here symbolizes their weights.

is divided by that number. That way, field sizes can mutate without changes in the expected input signal strength.

Mutation operators for the field elements The NEATfields method uses mutation operators that are very similar to those of the original NEAT implementation for evolving the contents of the field elements. The most common operation is to choose a fraction of connection weights and either perturb them using a normal distribution with standard deviation 0.18, or (with a probability of 0.15) set them to a new value. The application probability of this weight changing operator is set to 1.0 minus the probabilities of all structural mutation operators, which amounts to 0.8815 in the experiments reported here. In general, structural mutations are applied rarely because they will cause the evolutionary process to operate on larger neural networks and search spaces. A structural mutation operator to connect neurons is used with probability 0.02, while an operator to insert neurons is used with probability 0.001. The latter inserts a new neuron between two connected neurons. The weight of the incoming connection to the new neuron is set to 1.0, while the weight of the outgoing connection keeps the original value. The existing connection is deactivated but retained in the genome where it might be reactivated by further mutations. There are two operators that can achieve this: one toggles the active flag of a connection and the other sets the flag to 1. Both are used with probability 0.01.

Evolving network topology on higher levels For evolving higher level topology, NEATfields introduces some new operators. At the level of a single field, one operator doubles the field size along one dimension (with a probability of 0.001)

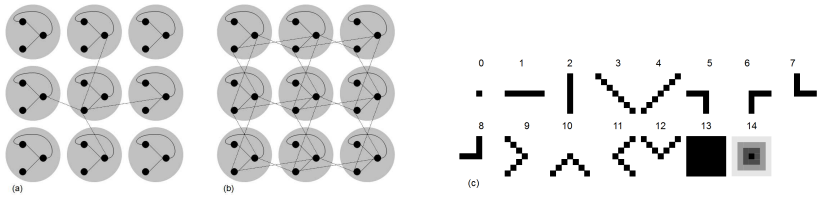

Fig. 3. (a) Lateral connections are established between a neuron in a field element and another neuron in each of the up to four neighbor field elements. There are less neighbors if it is at the border of the field. Lateral connections are only shown for the central element as dotted lines here for clarity. (b) All lateral connections constructed from a single gene. (c) Hand-designed patterns for local feature detectors. Black denotes a connection weight of 1.0, white a connection weight of -1.0, and the gray scales denote connection weights between 0.0 and 1.0.

and another changes the size (for both dimensions independently) to a random value between its current size and the size of the largest field it is connected to (with a probability of 0.005). NEATfields networks can also have lateral connections between field elements of the same field. Lateral connections are like local connections: they are between two neurons in the NEAT network that describes the field elements. However, the connection from the source neuron does not go to a neuron in the same field element, but to the corresponding neurons in the up to four neighbor field elements instead (see fig. 3). The gene coding for a lateral connection specifies source and target neuron reference numbers just as genes coding for local connections do; it is also located in the same chromosome. However, it has a lateral flag set to 1, and is created by a lateral connect operator (with a probability of 0.02).

By default, corresponding connections in different field elements all have the same strength so they can be represented by one gene. The same is true for the global connections between field elements of two fields. For some tasks, it is useful to have field elements that are slightly different from each other. One way to realize this is to make a connection weight larger in a neighborhood of some center coordinates on the field. The connection weight is scaled according to $\exp(-\epsilon(\frac{distance}{field\,size})^2)$ (in our implementation, this is done separately for the x and y dimensions), where $\epsilon = 5.0$ (see figure 2 (b)). The center coordinates are specified in the genome. There is a mutation operator that (at a probability of 0.03) sets these properties for a single connection gene.

At the level of the complete NEATfields network, there is an operator that inserts global connections (with a probability of 0.01) and an operator that inserts a new field into an existing global connection (with a probability of 0.0005). The size of the new field is set randomly to some value between 1 and the larger of the sizes of the two fields between which it is inserted. This is done independently for both dimensions. An existing field can also be duplicated, where all elements of the new field receive new reference numbers. The new

field can either be inserted parallel to the old one (in this case, the outgoing connection weights will be halved to prevent disruption of any existing function) or in series with the old one (in this case, every neuron in every field element in the new field gets input from the corresponding neuron in the corresponding field element in the old field, while the output from the new field goes to where the output from the old field went previously). The serial duplication operator is also applied on input fields, in which case an internal field is created that contains one neuron for every input in the input field. Both mutations occur with probability 0.0005.

Building local feature detectors NEATfields also has another kind of global connections that serve to detect local patterns by connecting to a number of adjacent field elements in another field. For example, if the input field contains camera data, they can detect lines or edges in particular orientations. These global connections have an additional number on the connection gene that refers to one of at most 16 prespecified connection weight patterns. The patterns have been designed by hand with the aim of providing useful building blocks for local feature detection (see fig. 3 (c)). A value from the pattern is multiplied by the original connection weight specified in the gene to get the weight of the connection from the field element at the respective position. The patterns have a size of 7×7. The central element of the pattern corresponds to the weight modification of the connection between exactly corresponding field elements. The feature detectors do not have to use an entire pattern. Instead, the maximal offsets between the field element positions in the source and target fields can be between 0 and 3 separately for the x and y dimensions, and are also specified on the genome. There is a mutation operator (used with a probability of 0.02) that mutates the pattern reference and maximal offsets of an already existing connection. Here we use only patterns 1 to 4 from figure 3 (c) with equal probabilities, and the maximal offsets in x and y direction are 0 or 1 with equal probabilities. Comparisons to other parameter settings and a number of alternative approaches can be found in earlier work [7].

2.2 Selection methods

We use speciation selection based on genetic similarity as described previously [7] for the spatial accuracy task. For the other three tasks, we use a hybrid selection method that combines fitness based tournament selection and a recent approach known as "novelty search". Half of the population is chosen based on fitness, and the other half is chosen based on novelty. Each half has its own elite of size 10 that is copied to the next generation unchanged. This method has also be described elsewhere [6]. The behavioral novelty measures used are described in the sections that describe the respective task to be solved. In all cases, a population of 1000 individuals is used.

3 Tasks

3.1 The "Scan/Lift" task

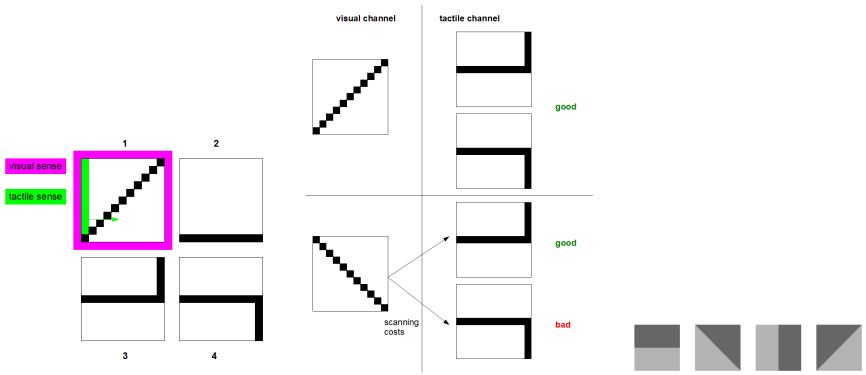

Fig. 4. Input patterns for multisensory integration tasks. (a) The Scan/Lift task. There are four episodes in this task with four different images. The visual sense sees the whole image after a lift action was executed, whereas the tactile sense can scan the image column-wise. (b) The Conditional Scan task. There are four episodes with different combinations of visual and tactile input. Tactile scanning is only required in the lower two episodes to disambiguate the input. (c) The four classes of gray scale images presented to neural networks in the third task.

This classification task can be optimally solved by integrating information from different sensory channels. The network sees a different image of size 11×11 in each of the four episodes (see fig. 4a). These patterns have to be assigned to two classes depending on whether the line goes up or down on the right side of the image. The neural network has four outputs: one classification output for each class, and two outputs for active perception. Every 10 time steps, the output with the highest value is considered to be an action. If it is a classification action, the episode terminates, and the reward is determined. The active perception outputs control one of the two sensory modalities each. The first controls the "visual" modality. If it is active, a "lift" command is executed, which means the whole pattern becomes visible in the visual input field. At the beginning of an episode, the pattern is not visible, so all visual inputs are 0.0. The second active perception input controls the "tactile" input, which can look at the same image in a different way. If it is active, a "scan" command is executed, which means that the scanning apparatus (which provides a single column of the image as input) moves one column to the right. At the beginning of an episode, it shows the leftmost column. If it has already been moved to the rightmost column, it will not move further.

The neural network gets 35 fitness points for every correct classification. In addition, it gets $15 - 5a_L - a_S$ fitness points in every episode, where a_L denotes whether the lift action was invoked (0 or 1), and a_S how often the scan action was invoked. So lifting is more expensive than scanning a row, but it is cheaper than scanning the whole image. This means that a network can get the optimal fitness if it uses scanning once for images 1 and 2 (where the line begins in the bottom of the image), but lifting for images 3 and 4 (where the line begins in the middle of the image). In that case, it gets a total fitness of 188 points. The behavioral distance between individuals is calculated as follows: If the corresponding commands of the two individuals at time step t are unequal, $\frac{10}{t+1}$ points are added to the distance. Times where only one of the individuals is still alive are disregarded. So this measure assigns the greatest weights to commands given at the beginning.

3.2 The "Conditional Scan" task

This task (see fig. 4b) is like the previous one but now the "visual" and "tactile" channel both have a size of 11×11, and they are provided with different input images. The visual channel is always active, but the tactile channel is active only after a tactile scan is performed. Evolution starts with a network that has two completely separate partitions, one consisting of the visual input field, an internal field, and the output field, and the other partition consisting of the tactile channel, another hidden field, and another output field, the state of which is ignored. Evolution can subsequently add more fields or connect them. There are three possible actions for the network: scan, assign to the "good" class, or assign to the "bad" class. "Scan" makes the whole image appear at once on the tactile channel. The neural network gets 10 fitness points for correct classification of a "bad" pattern, 5 points for correct classification of a "good" pattern, and 1 point if the scan action was not used in an episode. So the highest obtainable fitness is 27. Behavioral distance is calculated as for the previous task.

3.3 A multisensory classification task

In this task the networks have a visual input channel of size 11×11. 32 patterns of 4 different classes are displayed in separate episodes (see fig. 4c). However, the patterns are very noisy. The pixels in the area with the brighter color are generated from a normal distribution with mean 0.1 and variance 0.3, while those in the darker area are generated from a normal distribution with mean -0.1 and variance 0.3. The network has four classification outputs and an output to generate a scan command. Its tactile sense is of size 11×1 and looks at a single row of another image of the same class. Initially, the scanner is at position $x = -1$, so no input is perceived. Scanning does not incur any costs. There is a control condition were all rows are empty (i.e., the tactile input is 0.0). The neural network gets 1.0 fitness points for every episode in which it classifies the image correctly. If the classification is wrong, the network can get up to 0.5 fitness points according to $f = 0.5 - 0.25(o_a - o_c)$, where $o_a$is the highest output

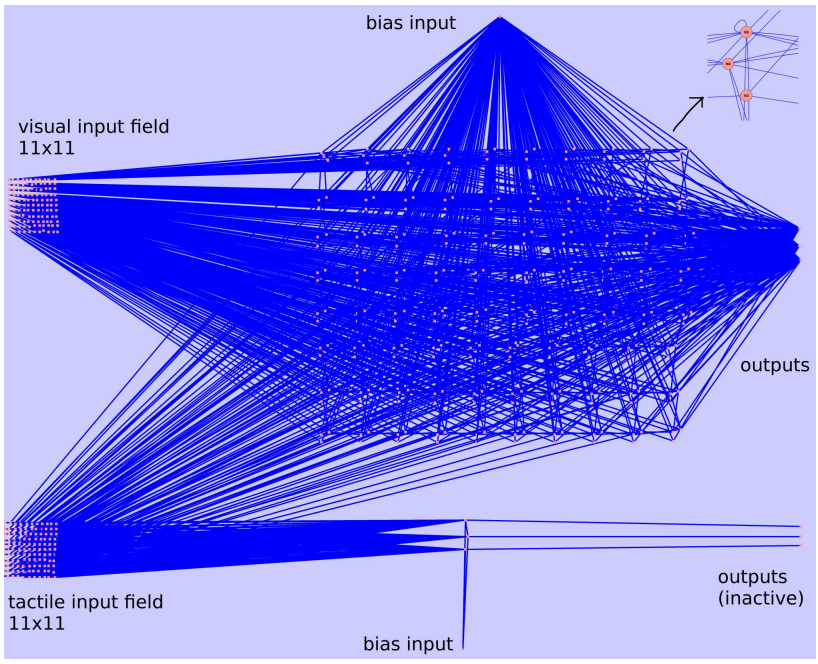

Fig. 5. A neural network solving the "Conditional Scan" task. In the upper left corner, a field element of the large internal field is shown magnified.

activation, and $o_c$is the activation of the correct output. Behavioral distance is calculated as for the other tasks.

3.4 Increasing spatial accuracy by sensorimotor integration

In this task 64 randomly generated images each containing a spatial cue are presented to a network in separate episodes. The input value is 1.0 at the position of the cue, and 0.0 elsewhere. The network can perceive the images through a visual and a tactile channel, each of size11×11. The positions of the cues on the input fields randomly deviate from the true position of the cue by up to ± 2 in both dimensions. For both input fields there is each an internal field and an output field. These two pathways are completely separated in the common ancestor. However, the network can gain higher fitness if it has evolved connections to average over the positions of the cues in the two input fields. The output fields are also of size 11×11. In each episode, the network gets a reward of $2 - (d(c, o_1)^2 + d(c, o_2)^2)/s^2$, where d is Euclidean distance, c is the true position of the cue, and the o_i are the positions on the two output fields with the highest activation. Behavioral distance between two individuals is defined as the sum over all episodes of the squared distances between the positions with highest activation on the corresponding output fields.

4 Results

20 runs of each of the four tasks are performed. The "Scan/Lift" task is solved perfectly in 75% of the runs using 349883 evaluations on average. In 15% of the runs, the classification is done correctly by the winner, but the "lift" action is used in all episodes, which results in a fitness of 180 points only. In 10% of the runs, the fitness is between 180 and the maximum 188 points. The "Conditional Scan" task is solved perfectly in 85% of the runs using 329705 evaluations on average. The other 15% runs converge on a solution that never uses the "scan" action and always classifies the ambiguous visual pattern as "bad", which results in a fitness of 24 points. One of the networks solving the "Conditional Scan" task perfectly can be seen in fig. 5. The internal field that is connected to the visual input has grown to a size of 10×8 by evolution, while the internal field connected to the tactile input has not grown. Instead, evolution has connected the first internal field to the tactile input as well. It can be seen that evolution uses one recurrent connection within each field element. Further recurrence is achieved through mutual lateral connections between some neurons in different field elements. It should be noted that solutions from other runs look very much different. In particular, some solutions for these tasks use several internal fields.

In the multisensory classification task, 31.55 fitness points are reached on average, while in its unisensory control, 30.74 fitness points are reached on average. This difference is slight but significant ($p < 0.001$, t-test). Looking at the 9 evolved solutions from the multisensory condition that solved the task perfectly, a number of different strategies can be found. One solution does not use tactile scans at all. Therefore, it still works perfectly when it is evaluated under control conditions. Seven solutions use tactile scans in between 8 and 28 of the 32 episodes. When they are evaluated under control conditions, they never classify the patterns that they normally used the scanner on, but perform scans until their lifetime is over. One solution uses the scanner in 17 episodes. Under control conditions, it stills classifies correctly in 10 of these episodes.

In the spatial accuracy task, 125.1 fitness points are reached on average, while in its unisensory control, 123.4 fitness points are reached on average. This is a significant difference ($p < 10^{-7}$, t-test). When having a closer look at the evolved network with the highest fitness, we found that it displayed a kind of McGurk effect. This effect originally refers to the perception of an intermediate phoneme when a subject simultaneously sees a video of a person producing one phoneme and hears a recording of that person producing another phoneme. It demonstrates that speech perception is based on multimodal information [1]. Our network was examined under two conditions. The first was to provide it with a new set of 100 randomly chosen locations and record the distances to the cues and the distances to another set of randomly chosen cues that were not displayed. The second condition was to test it with the same 100 locations again, but this time the tactile cues were from the other random set, so they were different from the visual cues. The distance to both cues was recorded. The network displayed a kind of McGurk effect because the distances to the cues in the second condition both were significantly larger than the distance to the cue

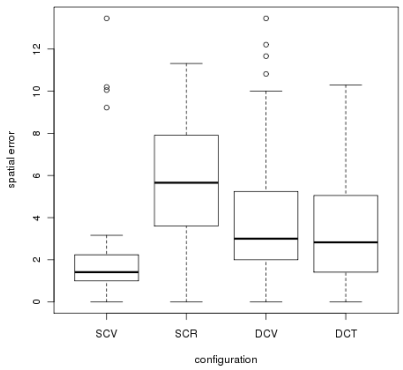

Fig. 6. Errors in the spatial accuracy task for four configurations: SCV - same cue condition / visual cue; SCR - same cue condition / suppressed cue; DCV - different cue condition / visual cue; DCT - different cue condition / tactile cue. All differences, except between DCV and DCT, are significant.

in the first condition, and were both significantly smaller than the distance to the non-visible cue in the first condition (see fig. 6) That means that the response was intermediate between both cues. The network is shown in fig. 7.

5 Discussion

We have shown that multisensory integration can evolve for cost-optimal use of sensors or increased classification or spatial accuracy. Importantly, this has been done not just using a few sensors, but using complete images. Further research will be on using real world input data. Besides, tasks with more emphasis on the temporal dimension on multisensory integration could be studied. If learning mechanisms are added to neuroevolution methods, the interaction of evolution and learning in multisensory integration can also be considered. In general, evolved neural networks that can process high-dimensional input in non-trivial ways are not just relevant for the goal of automatic construction of controllers, but also provide a useful tool for neuroscience research [10].

Acknowledgments Benjamin Inden gratefully acknowledges the financial support from Honda Research Institute Europe.

References

1. Spence, C.: Multisensory integration, attention and perception. In: Signals and Perception — The Fundamentals of Human Sensation. palgrave macmillan (2002) 345–354

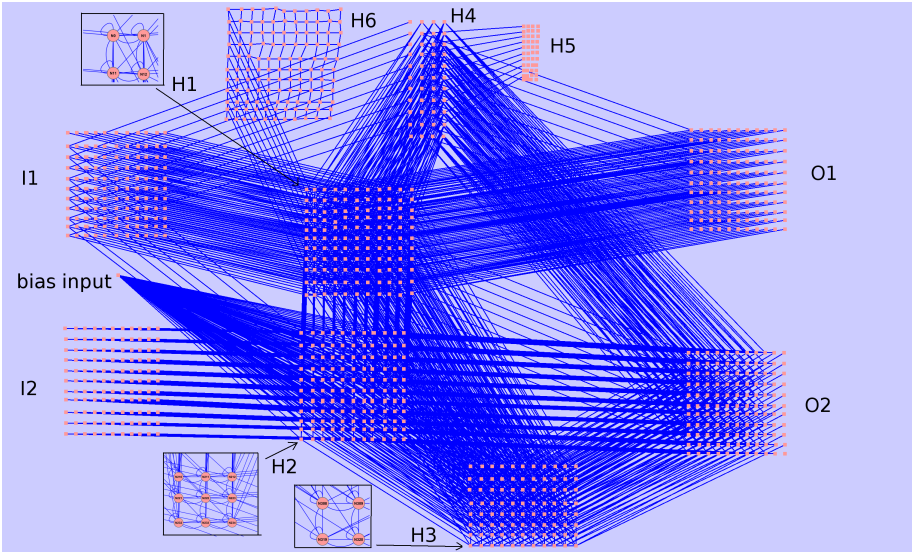

Fig. 7. An evolved network for the spatial accuracy task. "I" denotes an input field, "H" a hidden/Internal field, and "O" an output fields. The fine structure of three internal fields is shown in magnification. Field H4 only produces constant output, while fields H5 and H6 do not produce outputs because the respective connections have been deactivated by evolution. Multisensory integration is achieved by mutual connections between H1 and H2. Furthermore, H3 processes inputs from both sensor fields, but only contributes output for O2.

2. Stein, B.E., Stanford, T.R.: Multisensory integration: current issues from the perspective of the single neuron. Nature Rewievs Neuroscience **9** (2008) 255–266
3. Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: from architectures to learning. Evolutionary Intelligence **1** (2008) 47–62
4. Nolfi, S., Floreano, D.: Evolutionary Robotics — The Biology, Intelligence, and Technology of Self-Organizing Machines. MIT Press (2000)
5. Inden, B., Jin, Y., Haschke, R., Ritter, H.: Neatfields: Evolution of neural fields. In: Proceedings of the Conference on Genetic and Evolutionary Computation. (2010)
6. Inden, B., Jin, Y., Haschke, R., Ritter, H.: How evolved neural fields can exploit inherent regularity in multilegged robot locomotion tasks. In: Third World Congres on Nature and Biologically Inspired Computation. (2011)
7. Inden, B., Jin, Y., Haschke, R., Ritter, H.: Evolving neural fields for problems with large input and output spaces. Neural Networks **28** (2012) 24–39
8. Westermann, G.: A model of perceptual change by domain integration. In: Proceedings of the 23rd Annual Conference of the Cognitive Science Society. (2001)
9. Stanley, K., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10** (2002) 99–127
10. Ruppin, E.: Evolutionary autonomous agents: A neuroscience perspective. Nature Reviews Neuroscience **3** (2002) 132–141