Getting to Know Your Neighbors: Unsupervised Learning of Topography from Real-World, Event-Based Input

Martin Boerlin
martin_boerlin@bluewin.ch
Institute of Neuroinformatics, University of Zurich and ETH Zurich, CH-8057 Zurich, Switzerland, and Group for Neural Theory, Département d'Etudes Cognitives, Ecole Normale Supérieure, Collège de France, 75005 Paris, France

Tobi Delbruck
tobi@ini.phys.ethz.ch
Kynan Eng
kynan@ini.phys.ethz.ch
Institute of Neuroinformatics, University of Zurich and ETH Zurich, CH-8057 Zurich, Switzerland

Biological neural systems must grow their own connections and maintain topological relations between elements that are related to the sensory input surface. Artificial systems have traditionally prewired such maps, but the sensor arrangement is not always known and can be expensive to specify before run time. Here we present a method for learning and updating topographic maps in systems comprising modular, event-based elements. Using an unsupervised neural spike-timing-based learning rule combined with Hebbian learning, our algorithm uses the spatiotemporal coherence of the external world to train its network. It improves on existing algorithms by not assuming a known topography of the target map and includes a novel method for automatically detecting edge elements. We show how, for stimuli that are small relative to the sensor resolution, the temporal learning window parameters can be determined without using any user-specified constants. For stimuli that are larger relative to the sensor resolution, we provide a parameter extraction method that generally outperforms the small-stimulus method but requires one user-specified constant. The algorithm was tested on real data from a 64 × 64-pixel section of an event-based temporal contrast silicon retina and a 360-tile tactile luminous floor. It learned 95.8% of the correct neighborhood relations for the silicon retina within about 400 seconds of real-world input from a driving scene and 98.1% correct for the sensory floor after about 160 minutes of human pedestrian traffic. Residual errors occurred in regions receiving little or ambiguous input, and the learned topological representations were able to update automatically in response to simulated damage. Our algorithm has applications in the

design of modular autonomous systems in which the interfaces between components are learned during operation rather than at design time.

1 Introduction

Both natural and artificial systems require implicit or explicit knowledge of the topological arrangement of their components to function correctly. This information can be encoded or updated during either the developmental (design) or the sensory (operational) phase of the life of the system. One example of how such information may be encoded during development has been found in mammalian visual systems before eye opening, where spontaneous waves of retinal activity spread across the retinal ganglion cell (RGC) layer of the developing retina (Galli & Maffei, 1988; Meister, Wong, Baylor, & Shatz, 1991; Katz & Shatz, 1996; Wong, 1999) to produce strongly correlated activity among neighboring RGCs. It is believed that correlated activity among RGCs, and hence knowledge about the topological arrangement of the RGCs in the retina, plays a crucial role in topographic map formation in the visual pathway. (McLaughlin, Torborg, Feller, & O'Leary, 2003). Several models have been proposed to explain the formation of topographic maps and ocular dominance stripes in the visual system (for a review, see Swindale, 1996). Many rely explicitly on Hebbian learning rules, whether in the form of local modification rules (von der Malsburg & Willshaw, 1976; Miller, Keller, & Stryker, 1989) or models based on Kohonen's competitive learning rule (Kohonen, 1982; Goodhill, 1993). Most Hebbian learning models are accompanied by a normalization of synaptic weights in order to keep synaptic weight growth bounded. Other models have also been proposed, such as Elliott et al.'s neurotrophic model of activity-dependent competitive synaptic plasticity (Elliott & Shadbolt, 1999; Elliott, Maddison, & Shadbolt, 2001).

Models of topography development describe how topology is preserved from an array of afferent neurons (e.g., the retina) to an array of target neurons (e.g., lateral geniculate nucleus). The topology of the target sheet, which is assumed to be known, is used as a reference for reconstructing the topology of the afferent sheet. Figure 1A illustrates this process for a small system of seven afferent and seven target neurons, where each neuron in the target sheet is best represented by its corresponding neuron in the afferent sheet. The topology of the afferent sheet is then found by assuming that the two afferent neurons that map onto two adjacent target neurons must themselves be adjacent. For this approach to work, it is crucial to know the topology of the target sheet. In real-world systems, however, such knowledge is not necessarily available. It would therefore be desirable to find a method that allows direct unsupervised topology learning of the components of a system to overcome the limitation of prior topology information. In addition, for real-world applications, it is important to know how to set

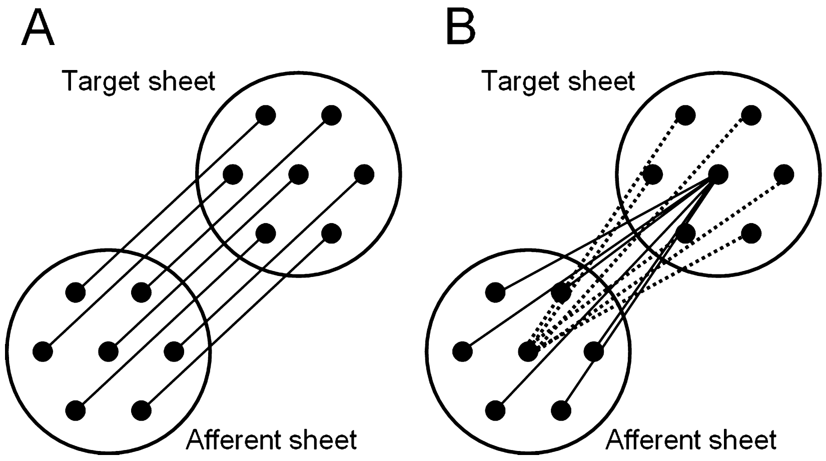

Figure 1: Schematic illustration of the optimal connections from afferent to target sheet of a small system containing seven afferent and seven target neurons. (A) Connections to be learned for models that assume knowledge of the topology of the target sheet. Each afferent neuron projects onto only one target neuron. (B) Connections to be learned for our model, not assuming any prior knowledge of sheet topologies. Each afferent neuron maps onto six target neurons (shown for the central afferent neuron by dotted connections), and each target neuron receives input from six afferent neurons (shown for the central target neuron by solid connections).

or learn important parameters such as learning windows and how to deal with edge effects.

The remainder of this letter details our algorithm for unsupervised learning of the adjacency matrix of a system with an initially unknown configuration of its components, without the above-mentioned assumption of a known target topology. It is designed to deal with real-world noisy input and adapt appropriately to changes in the topology of the system due to damage or component redistribution. To test the algorithm, we show real data recorded from a large, interactive tactile sensory floor (Eng et al., 2003; Delbrück et al., 2007) and an event-based temporal contrast silicon retina (Lichtsteiner, Posch, & Delbruck, 2006, 2008).

2 Topology Discovery Algorithm

2.1 Core Algorithm. Figure 1 graphically depicts the difference between our algorithm and previously published algorithms. As we cannot rely on knowing the topology of the target sheet, it is not possible to use a one-to-one neuron mapping (see Figure 1A). Instead, in our algorithm, each neuron in the afferent sheet maps on to putative neighbor neurons in the target sheet (typically four, six, or eight). Figure 1B schematically visualizes this idea, showing the optimal connections originating from the central neuron in the afferent sheet to its neighbors in the target sheet (dotted connections) as well as the connections between the central target neuron and the six afferent neurons representing it best (solid connections).

The algorithm is based on the idea that the temporal relationships between input stimuli to a system can reflect the underlying spatial relationships among its components. This means that if two sensory components generate input events within a time interval close to the characteristic time the stimulus needs to move from one component to the other, then they are likely to be neighbors. In addition, the algorithm makes the following assumptions:

- Each component of the system is uniquely labeled.
- A salient input at any sensing element generates a unitary event, which can be either an ON (inactive $\rightarrow$ active) or an OFF (active $\rightarrow$ inactive) event.
- The timing of input events is known with a time resolution much better than the rate at which the events occur.
- The number of nearest neighbors of each component is a constant, known number.

The algorithm takes as its input the asynchronous sequence of events, more precisely the sequence of event occurrence times and labels of the corresponding components. We consider a weight matrix $\mathbf{w}$ where each element w_{ij} represents a connection strength from component i to j. Let $\mathbf{w}_j(n)$ denote the weight vector of component j at time step n. The weight matrix is initialized such that

$$w_{ij}(n = 0) = 1. \tag{2.1}$$

When component j receives an input event at time step n, the weight changes $\Delta\mathbf{w}_j$ for weight vector $\mathbf{w}_j$ are calculated using an event-based learning rule with contributions from spike-timing-dependent plasticity (STDP) and Hebbian learning:

$$\Delta\mathbf{w}_j(n) = \Delta\mathbf{w}_j^{STDP}(n) + \Delta\mathbf{w}_j^{Hebb}(n). \tag{2.2}$$

Each of these contributions is explained in more detail below.

2.1.1 STDP-Like Learning. Weight changes are calculated due to the spike time difference $\Delta t = t_j - t_i$ where t_j and t_i denote the times at which components j and i received their last events, respectively. Because of the sequential arrival of events, $\Delta t \geq 0$ as t_j is always larger than t_i. Weight changes are calculated using a gaussian function:

$$\Delta w_{ij}^{STDP}(n) = \frac{1}{M} \exp\left\{ -\frac{[\Delta t - \mu_i(n)]^2}{2\sigma_i^2(n)} \right\}, \tag{2.3}$$

where M is the estimated number of components of the system (the number of components that have already received an event). The parameter μ is the center of the gaussian, corresponding to the characteristic time the stimulus needs to move from a component to its neighbor, and σ is the characteristic width of the gaussian. Both μ and σ are estimated according to characteristics in the movement of the stimulus (see the following section).

2.1.2 Hebbian-Like Learning with a Soft Winner-Take-All Component. The Hebbian-like part of the topology-discovering algorithm is not crucial for the overall functioning of the algorithm. However, it speeds up convergence of the weight matrix. Let $\mathbf{N}$ denote the currently estimated connectivity matrix among components, with N_{ik} referring to the label of the kth neighbor of component i. When component j receives an event, the corresponding weight w_{ij} is increased by a quantity η if component j is currently listed in $\mathbf{N}$ as a neighbor of component i:

$$\Delta w_{ij}^{Hebb}(n) = \begin{cases} \eta & \text{if } N_{ik}(n) = j,\ k = 1, 2, \ldots, m \\ 0 & \text{otherwise} \end{cases}. \tag{2.4}$$

The parameter m is the number of neighbors of a component. $\mathbf{N}$ is calculated with a soft winner-take-all algorithm; that is, the components $k = 1, \ldots, m$ of j are predicted to be those with the highest weights in $\mathbf{w}_j^{sym}$, where $\mathbf{w}_j^{sym}$ is the jth column of the symmetric matrix $\mathbf{w}^{sym} = \mathbf{w} + \mathbf{w}^{\mathbf{T}}$.

The weight update is accompanied by a normalization of the norm of the weight to unity to keep the total length of the weight vectors $\sum_i w_{ij}^2$ constant and prevent runaway growth of individual weights:

$$\mathbf{w}_j(n+1) = \frac{\mathbf{w}_j(n) + \Delta\mathbf{w}_j(n)}{\|\mathbf{w}_j(n) + \Delta\mathbf{w}_j(n)\|}. \tag{2.5}$$

2.2 Sharpening Filter: Recognizing the Border. As presented so far, the algorithm estimates m neighbors for each component. However, for finite systems, there are border components that have less than m adjacent components. To find these components and estimate their actual neighbors, we apply a sharpening filter to the weight matrix $\mathbf{w}$. The idea is to calculate a new weight matrix $\tilde{\boldsymbol{w}}$ for which we set a threshold that separates the weights

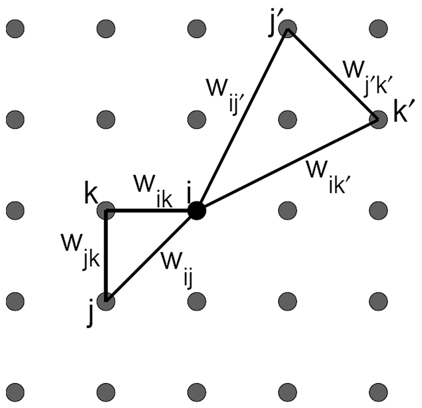

Figure 2: Visualization of weight triangles in equation 2.8. Triangle i–j–k links neighboring components, whereas triangle i–j'–k' links i to its second-order neighbors j' and k' (j' and k' being neighbors). We suppose that $w_{ij} > w_{ij'}$ and $w_{ik} > w_{ik'}$, implying that $v_{kj} > v_{k'j'}$.

of directly adjacent components from those of higher-order neighbors that are farther away.

Due to the asymmetric weight updates of equations 2.3 and 2.4, $\mathbf{w}$ is an asymmetric matrix that has to be transformed into a symmetric one for consistency of bidirectional connections:

$$\mathbf{w}^{sym} = \mathbf{w} + \mathbf{w}^{\mathbf{T}}. \tag{2.6}$$

Then each column of $\mathbf{w}^{sym}$ is modified such that its highest entry is unity:

$$w_j^{rel} = \frac{1}{\max_i w_{ij}^{sym}} w_j^{sym}. \tag{2.7}$$

We now calculate the new connection weight $\tilde{w}_{ij}$ between component j and i. For connection $i \longleftrightarrow j$, we define a vector $\mathbf{v}_j$ whose kth component is given by a weighted sum:

$$v_{kj} = a w_{ij}^{rel} + w_{kj}^{rel} + w_{ki}^{rel}, \tag{2.8}$$

where a is a weighting factor used to potentiate or depress the previously estimated connection strength between components j and i. For reasons of computational efficiency, equation 2.8 is processed only for the highest weights w_{ij} and w_{kj}. Figure 2 visualizes two weight triangles of equation 2.8. Triangle i–j–k links the three directly adjacent components i, j, and k, whereas triangle i–j'–k' links component i to its second-order neighbors j' and k', with j'–k' being direct neighbors. The underlying idea of the sharpening procedure is that the sum of the weights in triangle i–j–k will be higher than the sum of the weights in triangle i–j'–k' because direct neighbors should have higher connection strengths among each other than with higher-order neighbors. In other words, it is likely that $w_{ij} > w_{ij'}$ and $w_{ik} > w_{ik'}$ implying that $v_{kj} > v_{k'j'}$ and hence component j is the most likely neighbor of component i. Averaging over many such weight triangles should consequently increase the weights of actual adjacent components relative to the weights of nonadjacent ones, thus sharpening the weight matrix. After sorting vector $\mathbf{v}_j$, we calculate $\tilde{w}_{ij}$ as

$$\tilde{w}_{ij} = \frac{2}{m} \sum_{k=1}^{\frac{m}{2}} v^*_{kj}, \tag{2.9}$$

where m is the maximum number of neighbors and $\mathbf{v}^*_j$ is the sorted vector $\mathbf{v}_j$ such that $v^*_{kj} \geq v^*_{(k+1)j} \forall k$. We sort $\tilde{w}_j$ to get $\tilde{w}^*_j$ with the property that $w^*_{ij} \geq w^*_{(i+1)j} \forall i$. We also define $x_1, \ldots, x_m$ such that $\tilde{w}_{x_1 j} \geq \tilde{w}_{x_2 j} \geq \cdots \geq \tilde{w}_{x_m j} \geq \tilde{w}_{x_k j} \forall k$. We can then determine the neighbors N_{jk} of component j, requiring that their sharpened weights must lie above a threshold θ and that the relative weight change from one neighbor to the next is below a second threshold ψ:

$$N_{j1} = x_1 \tag{2.10}$$

and

$$N_{jk} = \begin{cases} x_k & \text{if } \tilde{w}^*_{kj} > \theta \quad \text{and} \quad \dfrac{\tilde{w}^*_{(k-1)j} - \tilde{w}^*_{kj}}{\tilde{w}^*_{(k-1)j}} < \psi \\ -1 & \text{otherwise} \end{cases} \qquad \text{for } k = 2, \ldots, m \tag{2.11}$$

An entry of the connectivity matrix equal to -1 refers to a nonexistent component in the system. A component with a -1 entry in its connectivity matrix is therefore expected to be found at the border of the system.

After passing the weights $\mathbf{w}$ through the sharpening filter, we can assume the extracted connectivity matrix to contain only neighbor estimations that are very likely to be correct. We can therefore compare the neighbor estimations for two components and complement the connectivity matrix (if component i contains j as its neighbor but not the other way around, store i as a neighbor of j as well).

The sharpening filter is computationally expensive and requires global knowledge of the weight matrix $\mathbf{w}$. It is therefore applied only after many iterations of the core algorithm of equation 2.2.

2.3 Determining the Learning Parameters. The most appropriate method for determining the center μ and width σ of the gaussian learning time window depends on the size of the input stimulus relative to the input elements. For input stimuli that are small relative to the sensing elements, ON and OFF events are directly correlated. In the example of the sensory tactile floor, a person stepping from one tile to another creates an OFF event that correlates directly with the subsequent ON event. However, for the silicon retina, the ON (dark-to-bright transition) and OFF (bright-to-dark transition) events, analogous to the on-off cells in biological retinas. (Lichtsteiner et al., 2008), are usually generated by stimuli with angular sizes much larger than one pixel. Here we outline the parameter extraction method first for small stimuli (tactile floor), followed by the required changes for large stimuli (silicon retina). For clarity, we refer to the main STDP algorithm as the "algorithm" and the learning window parameter extraction techniques as "parameter extraction methods."

2.3.1 Small Stimuli. For small stimuli with direct per-sensor ON-OFF correlations, we can determine the best learning parameters without using any additional user-specified constants. If we assume uniformly distributed event rates across all components, we can define a global vector **T** that acts as a buffer containing the most recent time intervals over which components were in the ON state. Every time an OFF event is generated, the oldest entry in the buffer is replaced by the latest one representing the time that the component was last ON. From **T**, we extract a global μ and σ:

$$\mu(n) = \tilde{\mathbf{T}} \tag{2.12}$$

$$\sigma(n) = 0.74 \cdot IQR(\mathbf{T}). \tag{2.13}$$

The median operator ($\sim$) ensures stability over extreme scores in **T**, and *IQR* determines the interquartile range of **T**. The factor 0.74 is used to normalize the interquartile range, leading to a robust estimation of the standard deviation of a normal distribution. In the global case, equation 2.3 uses $\mu_i(n) = \mu(n)$ and $\sigma_i(n) = \sigma(n)\ \forall i$.

For more biological plausibility and to deal with cases where the event rates vary by a large amount over the input components, we introduce individual (local) buffers $\{\mathbf{T}_i\}_{i=1}^{M}$ where M is the number of tiles of the system. $\mathbf{T}_i$ contains the most recent time intervals over which tile i has been loaded. We calculate a local $\hat{\mu}_i$ and $\hat{\sigma}_i$ for each tile individually using a similar method to that used for the global case:

$$\hat{\mu}_i(n) = \tilde{\mathbf{T}}_i \tag{2.14}$$

$$\hat{\sigma}_i(n) = 0.74 \cdot IQR(\mathbf{T}_i). \tag{2.15}$$

We then assume that neighboring tiles communicate with each other. A change in $\hat{\mu}_i$ and $\hat{\sigma}_i$ of tile i is influenced by μ_k and σ_k of its estimated adjacent components $k = 1, \ldots, m$ as follows:

$$\mu_i(n) = \hat{\mu}_i(n) - \frac{1}{3m} \sum_{k=1}^{m} [\hat{\mu}_i(n) - \mu_k(n)] \tag{2.16}$$

$$\sigma_i(n) = \hat{\sigma}_i(n) - \frac{1}{3m} \sum_{k=1}^{m} [\hat{\sigma}_i(n) - \sigma_k(n)]. \tag{2.17}$$

It is advantageous to calculate μ and σ according to a local learning rule (see equations 2.16 and 2.17) because no global information on tile occupancy is required as for equations 2.12 and 2.13.

2.3.2 Large Stimuli. For stimuli that are large relative to the sensing element spacing, ON and OFF events are not directly correlated in the same way as for small stimuli. The trailing edge of a large stimulus can generate an OFF event a very long time after the ON event has occurred. This means that we cannot measure the length of time that a sensing element was in the ON state. We apply a more general procedure, inspired by the finding of Zhang, Tao, Holt, Harris, and Poo (1998), showing that in the retinotectal system of the frog, there is a critical window of approximately 40 ms during which the temporal coincidence of spikes arriving at the same tectal cell can lead to cooperation and competition between synapses. We consider, for the ON events only (the same method applies for OFF events), a time interval of size τ. We have an array $\mathbf{T}^{on}$ where each element denotes the last ON event occurring in each component. When a new ON event occurs at time t, we find all the time differences between the new event and the last ON events of the individual components that fall within the time interval τ:

$$\hat{\mathbf{T}} = \left\{\Delta t = t - \mathbf{T}_i^{on} \mid \Delta t < \tau\right\}. \tag{2.18}$$

The learning window parameters μ and σ are then calculated as the mean and standard deviation of $\hat{\mathbf{T}}$. A drawback of this approach is the need for global information of the event times for all components.

We see from this formulation that the large-stimulus method can potentially handle a larger range of inputs than the small-stimulus method, since it does not require each sensing element to have effectively isolated ON/OFF events. However, the price to be paid is the requirement to specify an event buffer time interval τ (see equation 2.18).

3 Results

In sections 3.1 and 3.2, we test the algorithm on examples of stimuli that are small and large relative to the sensor resolution, respectively. Section 3.3 directly compares the small-stimulus and large-stimulus parameter extraction methods.

3.1 Small Stimuli: Interactive Tactile Floor. The interactive tactile floor (Delbrück et al., 2007) was developed for an interactive entertainment environment (Eng et al., 2003). It was considered desirable to devise a topology learning algorithm for the floor to avoid the need for cumbersome manual topology definition each time it was assembled. The floor consists of 360 hexagonal, pressure-sensitive floor tiles, each 0.66 m across, covering a total area of 136 m^2. Since each tile is roughly the size of an average adult human footstep, the stimuli are considered to be small relative to the resolution of the sensors. Each tile contains a local processing unit, and all of the tiles are connected by an industrial automation network to a computer. After software filtering, data are available about the binary occupancy of each tile (whether it was loaded or unloaded) at a rate of about 30 Hz. This data were used to generate tile ON and OFF events for testing our algorithm.

Data were collected from a public exhibition of the interactive environment. Groups of about 25 to 30 visitors interacted with the environment for approximately 5 to 6 min; for our testing, we considered a file of 390,000 recorded tile events corresponding to about 160 minutes of operation. About 7.5% of the tiles were loaded at any time. The parameters for the algorithm were set as follows: (1) $\eta = \frac{0.75}{M}$ in equation 2.4, with M being the estimated number of floor tiles; (2) $a = 1.5$ in equation 2.8; and (3) $\theta = 0.9$ and $\psi = 0.1$ in equation 2.11.

Figure 3A plots the total percentage of correctly estimated adjacency relationships against the number of tile events. The local and global learning rules yield very similar results, suggesting that visitor behavior is fairly uniform across the whole floor. The curves increase monotonically with the exception of a short, rapid decrease at approximately 1.4×10^4 tile events. This bump can be explained by the immature convergence state of the weight matrix and the resulting difficulty of accurate neighbor extraction. After processing all tile events, both methods yield a total of 98.1% correct adjacency estimations.

To reveal information about the overall topological error, we define the topological error distance $D(n)$ after n events as the average distance over all tiles (in units of tile spacings) between a source tile i and its estimated neighbors N_{ij} ($\mathbf{N}$ being the connectivity matrix):

$$D(n) = \frac{1}{mM} \sum_{i=1}^{M} \sum_{j=1}^{m} \left\| \mathbf{r}_{N_{ij}} - \mathbf{r}_i \right\|, \tag{3.1}$$

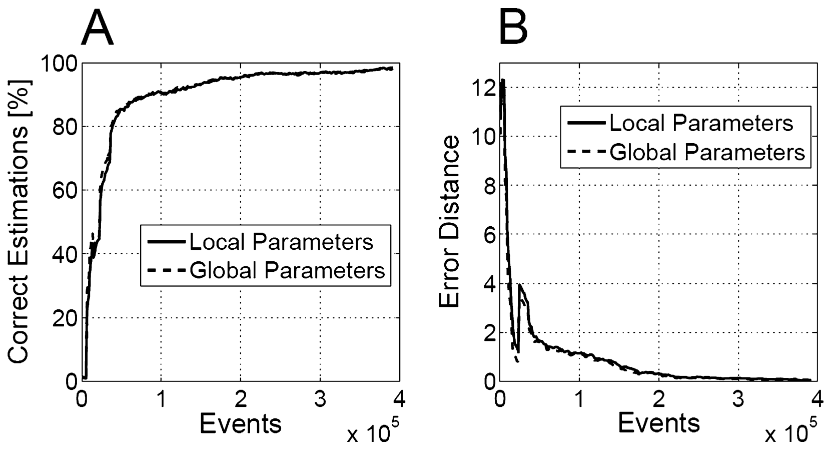

Figure 3: Progression of correct neighbor estimations for data from the interactive tactile floor. (A) Percentage of correct neighbor estimations plotted against number of tile events. (B) Topological error distance (calculated according to equation 3.1) plotted against number of tile events. Global parameters calculated using equations 2.12 and 2.13 and local parameters using equations 2.16 and 2.17.

where m is the number of neighbors of a tile, M is the total number of tiles, and $\mathbf{r}_i$ and $\mathbf{r}_{N_{ij}}$ are the discrete positions of tiles i and N_{ij}, respectively. The quantity $\|\mathbf{r}_{N_{ij}} - \mathbf{r}_i\|$ is the distance between tile i and N_{ij} in units of tile spacings. Figure 3B plots the average topological error distance against the number of processed tile events. The topological error decreases very quickly and approaches zero as the number of tile events increase. This suggests that the wrong estimations are very close to their correct position. The bump observed in Figure 3A can also be seen in the error distance plot. A drawback of this measure is that in cases where a neighbor of a tile is incorrectly thought to be nonexistent (a border tile), the error distance will be zero, although the estimation is incorrect.

Figure 4 shows the performance of the topology-discovering algorithm (local learning rule) using connectivity maps. Each node represents a tile, with estimated adjacent tiles connected by a straight line. A black line in that context stands for a correct connection, a gray line for an incorrect one. The results obtained by the global and local learning rules are very similar.

The algorithm orders an initially random connectivity scheme into an ordered map with very few errors. Most errors and incomplete connections were found at the border of the floor. This deficiency can be understood by the fact that visitors tend to avoid the borders of the floor (Eng, Douglas, & Verschure, 2005), generating correspondingly fewer events (see Figure 5A). Some tiles at the border received fewer than 10 events during the entire

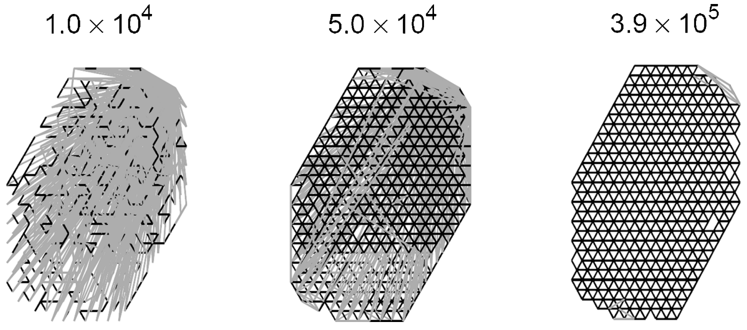

Figure 4: Development of topology for the interactive tactile floor. A black line represents a correct connection, a gray line an incorrect one. Each connectivity map represents the state of the system after a given number of tile events that are indicated above each map.

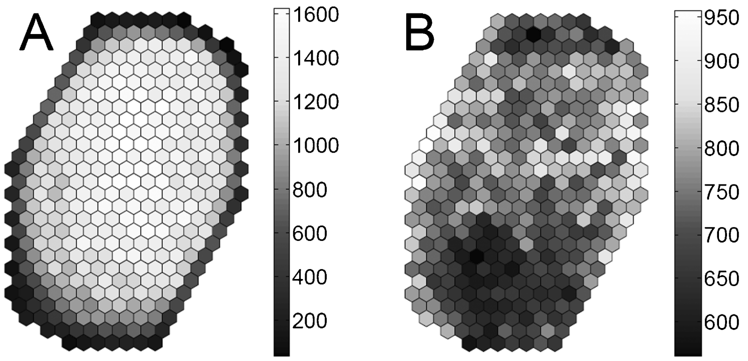

Figure 5: (A) Tile event count for each tile after 3.9×10^5 events. The gray-scale bar indicates the absolute number of events processed per tile. (B) Center μ of the gaussian learning window for individual tiles estimated using the local learning rule at a random time during processing. The gray scale displays the values of μ in units of milliseconds.

recording session, causing estimation difficulties in these regions. Figure 5B shows μ for each tile calculated at a random point in time during learning with the local learning rule. Small neighborhoods with similar values of μ can be seen.

It is useful to know how many events per tile are required in order to get high accuracy estimations. Figure 6 shows the average number of correctly estimated neighbors per tile plotted against the number of events per tile. Approximately 120 events per tile yield five correct neighbor estimations, while about 550 events per tile are required to estimate all neighbors correctly.

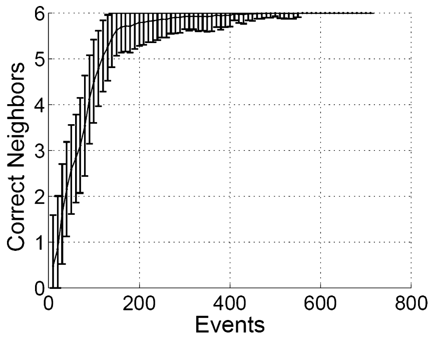

Figure 6: Averaged number of correct neighbor estimations per tile, plotted against the number of processed events per tile. Error bars indicate one standard deviation.

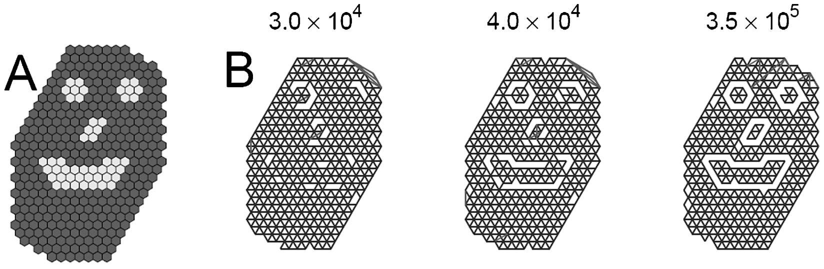

Figure 7: (A) Representation of simulated damage to tiles (in light gray). (B) Development of topology after virtual tile damage. The states of the system are shown after 3.0×10^4, 4.0×10^4, and 3.5×10^5 tile events. Before the start of processing, the weight matrix was in a converged state.

To test the robustness of the topology-discovery algorithm to simulated tile damage, we started with a (nearly) converged weight matrix corresponding to the final connectivity map in Figure 4. The same input data set were then run again, with designated "damaged" tiles (see Figure 7A, light gray tiles) being excluded from receiving any further input. Figure 7B shows the development of adapted topology after tile damage. Visible adaptation starts after about 2.0×10^4 events. After 4.0×10^4 tile events, the connectivity map is recognizably adapted, with the disrupted connections identified. Later, after 3.5×10^5 tile events, the connectivity map has nearly completely adapted.

To test an extreme case of adaptation, we started again from the (nearly) converged weight matrix of Figure 4. All tiles were then randomly relabeled, and the same event sequence was processed again. Figure 8A plots the percentage of correct neighbor estimations plotted against tile

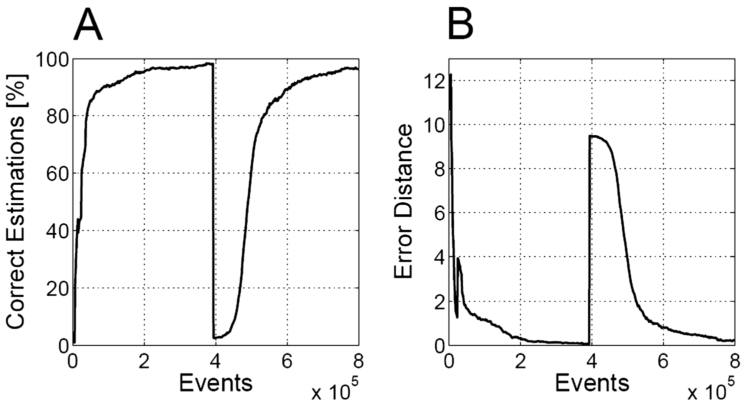

Figure 8: Relearning of nearest neighbors after random relabeling of tiles. The relabeling occurs after 3.9×10^5 tile events. (A) Percentage of correct neighbor estimations plotted against number of processed tile events. (B) Topological error distance, calculated according to equation 3.1, plotted against number of processed tile events.

events, and Figure 8B shows the corresponding averaged topological error calculated according to equation 3.1. Until relabeling (at 3.9×10^5 tile events), the curves are identical to the ones in Figure 3. After relabeling, the algorithm reestablishes correct connections between the relabeled tiles. The curves after relabeling show a similar convergence behavior to the ones before relabeling, but with a slower starting phase required for forgetting of the old incorrect weights. Even in this extremely unrealistic scenario, the algorithm was able to function as expected.

3.2 Large Stimuli: Temporal Contrast Silicon Retina. The temporal contrast silicon retina chip we used contains 128×128 pixels that generate events corresponding to changes in log intensity (Lichtsteiner et al., 2006). Salient objects viewed by the retina typically are several pixels across (at least); hence they are "large" compared to the interpixel spacing, unlike the human input to the tactile floor. Each pixel outputs ON and OFF events, and each event signifies a change by a threshold amount of log intensity since the last event from that pixel (ON-transition = dark-to-bright, OFF-transition = bright-to-dark). The pixel output streams the ON and OFF events as asynchronous binary pulses and hence implements the asynchronous spike-based separation into ON and OFF channels found in the retina or the lateral geniculate nucleus (LGN).

We ran our topology-discovering algorithm on a sequence of pixel events captured by the silicon retina, which was mounted on a car driving around suburban streets. Since the pixels were arranged in a square grid,

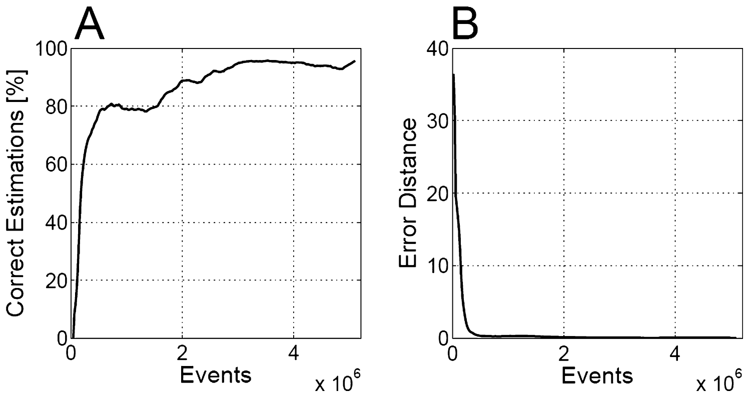

Figure 9: Convergence of retina nearest-neighbor estimations for driving scene input. (A) Percentage of correct neighbor estimations plotted against number of processed pixel events. (B) Topological error distance, calculated according to equation 3.1, plotted against number of processed pixel events.

the algorithm learned the nearest eight neighbors rather than six for the hexagonal floor tiles. To speed up computation without losing generality, we processed only a 64×64 pixel subarea of the visual field. The parameters for the topology learning algorithm were set as follows: (1) $\eta = \frac{0.5}{M}$ in equation 2.4, with M being the currently estimated number of pixels; (2) $a = 0.75$ in equation 2.8; (3) $\theta = 0.94$ and $\psi = 0.03$ in equation 2.11; and (4) $\tau = 35$ ms in equation 2.18.

We use the same measures to analyze the results as previously described for the tactile floor. Figure 9A plots the percentage of correctly estimated adjacency connections against the number of pixel events. Correct estimation increases rapidly up to 80% over the first 5.0×10^5 events, proceeding more slowly in a somewhat step-wise fashion after that. We predicted 95.8% of all nearest-neighbor connections successfully after the available 5.1×10^6 events (representing about 1 minute of driving), but complete convergence was not yet reached. Figure 9B depicts the averaged topological error development. The error distance decreases rapidly over the first 5.0×10^5 events and then gradually approaches zero. The step-wise behavior found in Figure 9A is not observed in the development of topological error.

In Figure 10 the development of topology is represented graphically by means of connectivity maps. Each pixel is represented in the map as a dot; for visual clarity, we show only the incorrect connections between two pixels. Many residual incorrect estimations are found in the upper-right corner of the pixel grid. These are due to partially transparent internal reflections from the inside of the windscreen of the car. The reflections create rapid event sequences, resulting in incorrect neighbor estimates (long

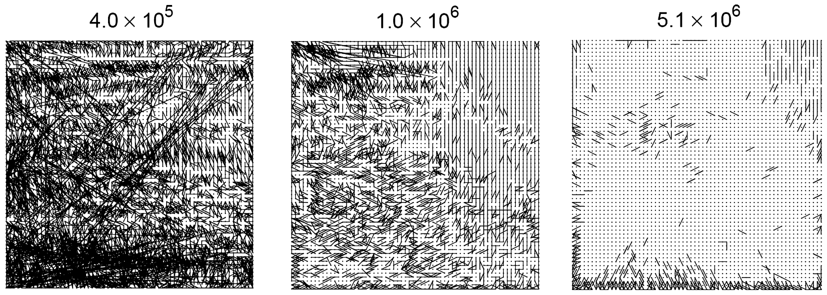

Figure 10: Development of topology for the temporal contrast silicon retina (64×64 pixels). Pixels are represented as dots, and incorrect connections are represented by lines. Correct connections are not shown for visual clarity. Depicted are the states of the system after 4.0×10^5, 1.0×10^6, and 5.1×10^6 events.

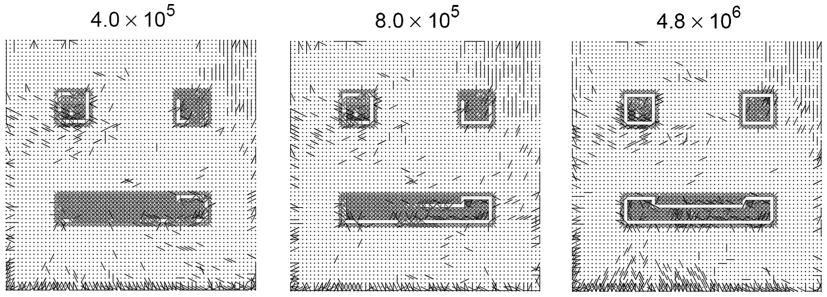

Figure 11: Development of retinal topology after simulated pixel damage. The states of the system are shown after 4.0×10^5, 8.0×10^5, and 4.8×10^6 events. At zero events, the weight matrix was in a converged state. Within the damaged regions, both the correct (gray) and incorrect (black) connections are shown.

vertical lines) that require many more input events to correct than the rest of the scene. Another problematic region is the lower border of the pixel grid, where reflections from the hood of the car were common. The consequence was that border pixels were not recognized as such, leading to incorrect connections originating from these pixels to higher-order neighbors. Further training with real-world input would eventually resolve these problems.

To test the robustness of the algorithm to simulated pixel damage, we started with the connectivity matrix obtained after processing all 5.1×10^6 pixel events (see Figure 10). We then "damaged" a number of pixels and reprocessed the same event data again. The results are shown in Figure 11. In order to visualize the disruption of connections between functioning and damaged pixels, both correct (gray lines) and incorrect (black lines) connections are shown in the damaged area. After 4.0×10^5 events, the first connections have been disrupted, and the border around the damaged

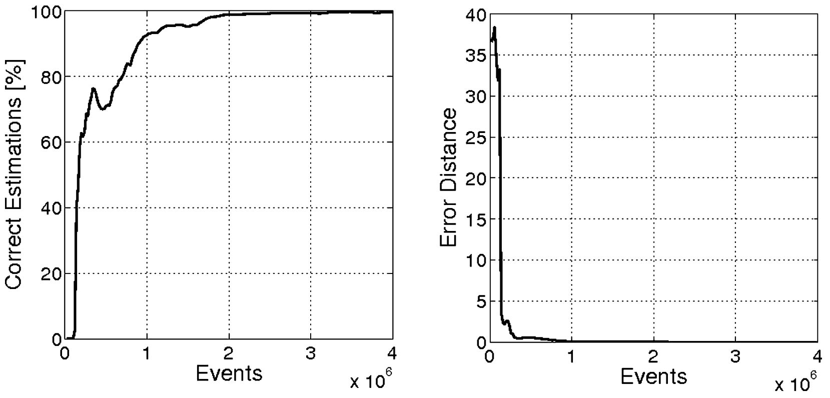

Figure 12: Convergence of retina nearest-neighbor estimations for simple grating input. (Left) Percentage of correct neighbor estimations plotted against number of processed pixel events. (Right) Topological error distance, calculated according to equation 3.1, plotted against number of processed pixel events.

pixels begins to emerge. Rematuring of the connectivity maps is largely completed after 2.0×10^6 events.

Because the highly irregular features in the driving scene retina input made it difficult to assess its potential performance under more ideal conditions, we tested the algorithm again using a simple real-world grating stimulus consisting of sharp black-and-white edges. These edges were moved in front of the retina in various directions to generate correlated input for learning. The learning parameters were the same as for the driving scene, except that the window size (see equation 2.18) was set to $\tau = 20$ ms for the grating stimulus compared to 35 ms for the driving scene to allow for the reduced uncertainty in the input. After 4.0×10^6 events, 99.7% of the connections were estimated correctly (final error distance 0.0035), a clearly better result than for the natural stimuli (see Figure 12). Convergence also occurred faster for the simpler stimulus, with 99% of the connections correctly identified after 2.2×10^6 events. All but two of the incorrect connections occurred at the border. Even when considering only the border elements, 96.7% of border neighbors were recognized correctly, a result still better than the overall performance for the driving scene input.

3.3 Comparing Small-Stimulus and Large-Stimulus Parameter Extraction Methods. To directly compare the small-stimulus and large-stimulus parameter extraction methods, we took a floor test data set containing 69,000 floor events. New synthetic data sets were then generated, each consisting of the original floor events plus N simultaneous new events for the N tiles immediately surrounding each original event ($N = 1..6$). These new data sets approximate the input provided by a person with

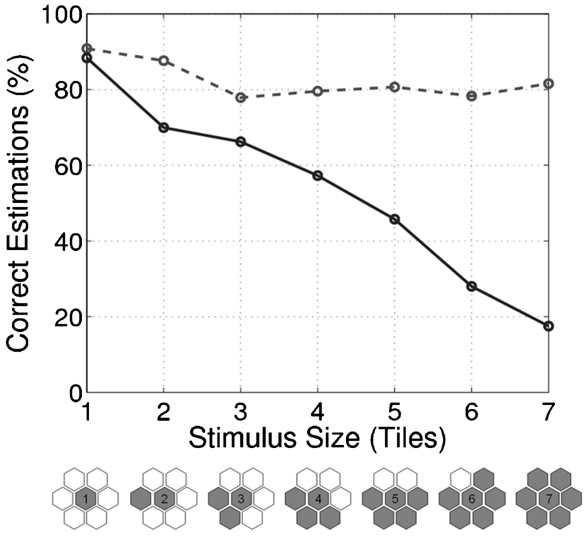

Figure 13: Variation of floor topology learning algorithm performance with simulated increase in stimulus size. The increase in stimulus size was simulated by simultaneously activating N of the directly adjacent tiles to each original input event, where N ranges from 0 to 6. Solid line: Small-stimulus parameter estimation method with local event buffers (see section 2.3.1). Dashed line: Large-stimulus parameter estimation method (see section 2.3.2) with time interval 1.0 s.

progressively larger feet walking on the floor. For the large-input parameter extraction method, we chose a time interval $\tau = 1.0$ s. The results (percentage of correct connections) of testing the new data sets using both parameter extraction methods are summarized in Figure 13. For the original "small" input, the algorithm using both the small-stimulus and large-stimulus parameter estimation methods performs equivalently (about 90% correct). However, for "larger" input, the large-stimulus method maintains approximately constant performance, while the small-stimulus method performs much worse (below 20% correct for a seven-tile-sized stimulus). Thus, we can conclude that for the synthetic tests we conducted, the large-stimulus parameter extraction method performs better than the small-stimulus method. Note that it was not possible to repeat the direct comparison for the retina input data, since there is no simple way to generate meaningful synthetic small-stimulus data from existing large-stimulus data.

4 Discussion

We have presented a topology-discovery algorithm for unsupervised adaptive learning of the adjacency matrix for systems with initially unknown component configurations. The requirements for learning the configuration are the number of nearest neighbors for the components and the timings of labeled events from each component. The algorithm was shown to satisfactorily handle real-world noise and irregularities of complex real-world input in two event-based systems. For both systems, most adjacency connections (more than 95%) could be found after a reasonable number of events in real time, considering that in most cases, learning from scratch need only be carried out once. We also demonstrated the ability of the algorithm to adapt to extreme topological changes in the system (e.g., damage of components, random rearrangement of components).

Given the primary constraints of the unsupervised learning problem to be solved (i.e., unknown geometry, unknown borders, and event-based input), it is difficult to imagine any completely different alternative algorithms that do not take advantage of the timing of the input events. Rather, any alternative or improved algorithms will take the form of modifications to the basic STDP concept. The STDP learning window used was a symmetric gaussian function with no negative branches to model long-term depression (LTD) as found in naturally occurring STDP (Bi & Poo, 1998). The weights were kept within reasonable limits by a global normalization step, which in the future could be replaced by on STDP learning window including LTD. Such a change could remove the biologically unrealistic need for knowledge of the sum of the weights, at the cost of requiring careful tuning of the learning window to prevent the weights from heading toward zero or infinity. An LTD component could also help to accelerate learning by depressing weights for events with very short time intervals. Any number of alternative learning window shapes could also be used to give better learning performance if more a priori information is known about the statistical properties of the input.

Problems with determining neighbors in the silicon retina were caused by large variations in the speed of stimulus movement due to reflections. We view this problem as a strength of our real-world testing process. It is unlikely that we would have detected this issue if we had used only standard or simulated moving grating stimuli. The variations in the stimulus movement speeds necessitated a larger value of τ compared to that used for the simple grating stimulus. One way to improve the handling of different speeds could be to learn the connectivity with multiple parallel systems running at different timescales. This process, which is equivalent to learning not just nearest neighbors but also higher-order neighbors, would provide not just the connectivity but also an idea of the most commonly occurring stimulus speeds and directions.

An alternative method for dealing with large local differences in stimulus speed, although applicable only in the small-stimulus case, could be to change the number of local buffers (see equations 2.16 and 2.17) to be an intermediate value between 1 and M. In this case, each buffer refers to a cluster of sensing elements, the members of each cluster being assigned during learning by the current neighborhood estimator. Choosing an intermediate value provides the opportunity to trade off the degree of localization of information against the computational cost of calculating μ and σ. Using an intermediate number of buffers would require complex modifications to equations 2.16 and 2.17, since the neighborhood clusters update dynamically during learning. Such a method could be seen as being equivalent to clustering the sensing elements into regions of different characteristic speeds.

Our primary reason for testing our algorithm with real data collected from hardware, rather than using simulations, was to avoid making the extra assumptions inherent in using simulations. We could therefore be more confident that our results can be directly applied to real-life problems in robotics and human-machine interaction with minimal extra parameter tuning. For the tactile floor data, although several models of pedestrian movement exist (Helbing, Farkas, & Vicsek, 2000; Dijkstra, Jessurun, & Timmermans, 2001), they usually treat pedestrians as two-dimensional unit circles. In particular, they do not include individual footsteps that would be required to produce realistic simulated tactile input data. For the retina, in principle it would have been possible to generate simulated input, but that would have required us to generate and process synthetic high-speed images using huge amounts of computational effort and data storage. Normal high-speed video cameras combined with off-line processing could also have been used, but the silicon retina achieves an equivalent result in real time using several orders of magnitude less power.

The differences between the methods used to determine the size of the learning window were necessitated by the lack of correlation between ON and OFF events for large stimuli. The large-stimulus parameter extraction method also works for small stimuli, but at the cost of using an extra parameter τ (see equation 2.18). We showed that the small-stimulus parameter extraction method works only as well as the large-stimulus method when the stimulus is the same size as (or smaller than) the sensing elements (see Figure 13). However, the higher generality of the small-stimulus method in requiring no extra parameters makes it attractive for situations for use in fully unsupervised situations.

Despite its generally superior performance, one other disadvantage of the large-stimulus parameter extraction method is that it requires global information of the spike times to learn the most appropriate learning window parameters. However, this is not necessarily a problem for biological plausibility. While the automatic learning window adjustment feature is useful for artificial systems, biological systems may have evolved hard-wired learning windows inherent in the biophysical properties of the relevant intracellular mechanisms. The differences between the parameter extraction methods could also point to differences in the mechanisms of natural topology formation between mammalian retinas (larger stimulus relative to element size) and insect retinas (smaller stimulus relative to element size, and compound eyes).

Another aspect of the algorithm that does not map obviously onto a biologically plausible implementation is the edge detection sharpening filter, which requires global knowledge of the connectivity matrix. This feature is useful for artificial systems, but it seems unlikely that special mechanisms are employed to take care of small edge effects in mammalian retinas with tens of millions of elements. However, edge effects may be more important for insect retinas with only a few hundred or thousand elements, and with small numbers of elements, it may become more reasonable to postulate the existence of global signaling processes for handling edge effects during development.

In the tactile sensory floor, limitations were found for regions that received very small numbers of events (as few as about 10 events in 2.5 hours in some cases). Since the floor could also provide luminous output, a way of actively improving this situation could be for the floor to output a confidence level for its own topology. Brightly lit tiles could indicate tiles with little cumulative traffic, inviting pedestrian traffic to the area to dim and eventually extinguish the tiles. Damaged tiles, no longer providing input to the system, would cause their neighbors to grow brighter and then progressively dimmer again as the system weights adapted to the missing component.

A future possible development of the algorithm includes superimposing multimodal input streams (e.g., visual and tactile input) for sensor fusion. Such an algorithm could then form the basis of a system for multimodal tracking. Direct applications of the algorithm include automatic topology learning for distributed sensor networks, automatic detection and compensation of defects in imaging arrays, and adaptive learning of somatotopy or retinotopy in neural implants.

Acknowledgments

We acknowledge the assistance of the reviewers for their detailed and insightful feedback on the letter. In addition, we thank the developers of the tactile sensory floor: Rodney Douglas, Adrian Whatley, Paul Verschure, Klaus Hepp, and the rest of the Ada exhibit team. We also thank Patrick Lichtsteiner for his development work on the latest version of the silicon retina. This work was supported by the Institute of Neuroinformatics at the University of Zurich and ETH Zurich.

References

Bi, G.-q., & Poo, M.-m. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience, 18*(24), 10464–10472.

Delbrück, T., Whatley, A. M., Douglas, R. J., Eng, K., Hepp, K., & Verschure, P. F. M. J. (2007). A tactile luminous floor for an interactive autonomous space. *Robotics and Autonomous Systems, 55*, 433–443.

Dijkstra, J., Jessurun, A. J., & Timmermans, H. J. P. (2001). A multi-agent cellular automata model of pedestrian movement. In M. Schreckenberg & S. D. Sharma (Eds.), *Pedestrian and evacuation dynamics* (pp. 173–181). Berlin: Springer-Verlag.

Elliott, T., Maddison, A. C., & Shadbolt, N. R. (2001). Competitive anatomical and physiological plasticity: A neurotrophic bridge. *Biological Cybernetics, 84*, 13–22.

Elliott, T., & Shadbolt, N. R. (1999). A neurotrophic model of the development of the retinogeniculocortical pathway induced by spontaneous retinal waves. *Journal of Neuroscience, 19*(18), 7951–7970.

Eng, K., Baebler, A., Bernardet, U., Blanchard, M., Costa, M., Delbruck, T., et al. (2003). Ada—intelligent space: An artificial creature for the Swiss Expo.02. In *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation*. Piscataway, NJ: IEEE.

Eng, K., Douglas, R. J., & Verschure, P. F. M. J. (2005). An interactive space that learns to influence human behaviour. *IEEE Transactions on Systems, Man and Cybernetics Part A, 35*(1), 66–77.

Galli, L., & Maffei, L. (1988). Spontaneous impulse activity of rat retinal ganglion cells in prenatal life. *Science, 242*(4875), 90–91.

Goodhill, G. J. (1993). Topography and ocular dominance: A model exploring positive correlations. *Biological Cybernetics, 69*(2), 109–118.

Helbing, D., Farkas, I., & Vicsek, T. (2000). Simulating dynamical features of escape panic. *Nature, 407*, 487–490.

Katz, L. C., & Shatz, C. J. (1996). Synaptic activity and the construction of cortical circuits. *Science, 274*, 1133–1138.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics, 43*, 59–69.

Lichtsteiner, P., Posch, C., & Delbruck, T. (2006). A 128 × 128 120dB 30mW asynchronous vision sensor that responds to relative intensity change. In *Proceedings of the International Solid State Circuits Conference* (ISSCC 2006). Piscataway, NJ: IEEE.

Lichtsteiner, P., Posch, C., & Delbruck, T. (2008). A 128 × 128 120dB 15us latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits, 43*, 566–576.

McLaughlin, T., Torborg, C. L., Feller, M. B., & O'Leary, D. D. M. (2003). Retinotopic map refinement requires spontaneous retinal waves during a brief critical period of development. *Neuron, 40*, 1147–1160.

Meister, M., Wong, R. O. L., Baylor, D. A., & Shatz, C. J. (1991). Synchronous bursts of action potentials in ganglion cells of the developing mammalian retina. *Science, 252*(5008), 939–943.

Miller, K. D., Keller, J. B., & Stryker, M. P. (1989). Ocular dominance column development: Analysis and simulation. *Science, 245*(4918), 605–615.

Swindale, N. V. (1996). The development of topography in the visual cortex: A review of models. *Network: Computation in Neural Systems, 7*, 161–247.

von der Malsburg, C., & Willshaw, D. J. (1976). A mechanism for producing continuous neural mappings: Ocularity dominance stripes and ordered retino-tectal projections. *Exp. Brain. Res. Suppl., 1*, 463–469.

Wong, R. O. (1999). Retinal waves and visual system development. *Annu. Rev. Neurosci., 22*, 29–47.

Zhang, L., Tao, H. W., Holt, C. E., Harris, W. A., & Poo, M. M. (1998). A critical window for cooperation and competition among developing retinotecal synapses. *Nature, 395*, 37–44.

Received June 20, 2007; accepted April 7, 2008.