ELSEVIER

Image and Vision Computing 19 (2001) 753–761

image AND vision COMPUTING

www.elsevier.com/locate/imavis

View-based route-learning with self-organizing neural networks

F. Hamze*, J.J. Clark

McGill University, Center for Intelligent Machines, Montreal, Canada

Accepted 10 December 2000

Abstract

This paper describes a view-based mobile robot navigation system relying on self-organizing neural networks. *Route navigation* was presumed to consist of a chain of *view–action associations*. A sequence of view images from a test route was obtained, pre-processed, and used to train a system of self-organizing maps. The converged networks consisted of a set of weights representing the learned views and a set encoding the actions to be carried out at those views. A view presented to the trained networks can thus associatively elicit the action coupled to it, allowing autonomous execution of the route. The data were presented to the system of networks using a simple place-dependent scheme, and a context-sensitive decision-maker was used to minimize potential recognition ambiguity during the execution stage.

Keywords: Robot navigation system; Self-organizing neural networks; Trained networks

1. Introduction

Navigation and route-learning are two of the fundamental tasks of mobile robotics. In this work, we outline a system inspired by biological observations of this phenomenon and of learning in general. Many animals demonstrate a highly robust capacity to form internal maps of their surroundings and to navigate within them. In contrast, most artificially engineered systems, which rely heavily on a priori data and modeling techniques, show nowhere nearly the same ability to adapt and generalize to novel surroundings and consequences. The superior performance of biological systems in this respect stems from their *parallel neural organization*. We believe that using a similar strategy, allowing the so-called higher-level features to autonomously emerge from the early processing, will yield a system that is less situation-specific than the constrained methods in fashion today.

1.1. Navigation in biological systems

Much of our knowledge of navigation in animals comes from experiments done with rats in mazes. Tolman [1] postulated that rats navigate using *cognitive maps*, defined as "a series of interconnected places that are systematically linked together through spatial transformation rules" [1]. Rats in radial maze experiments exhibited behavior that suggested the occurrence of *place-learning*, where animals acted with the goal of moving to a certain location, rather than *response-learning*, in which rats simply output a motor action when presented with a sensory stimulus. O'Keefe and Nadel [3] distinguish between *routes* and *maps* in navigation. A route is defined as a list of stimulus–response–stimulus (S–R–S) associations which lead the navigator from one sensory input (e.g. view) to another until a final goal destination is reached. O'Keefe and Nadel point out that, in order to strictly follow a route, the cues must inflexibly appear in the correct order. Consequently, they are sensitive to error; an occluded, distorted, or unattended cue can wreak havoc on the accuracy of navigation. Maps, on the other hand, are representations of parts of space. They are free from the stimulus ordering constraints of routes; novel routes from between places can be computed. The flexibility that arises comes at the expense of a longer processing time and increased storage requirements.

A brain structure known as the *hippocampus* has been found to be crucial to place-learning for mapmaking. The so-called hippocampal *place-cells* are neurons characterized by their responsiveness to particular environmental locations called *place fields*. Place fields can overlap, and cells are known to be responsive to more than one place field [3]. This is highly suggestive of a *distributed* approach to the encoding of place-cell data. In other words, it is unlikely that a place-cell (or population) exists to encode every place experienced. Also, it is now believed that the hippocampus, rather than simply contain the map information, serves to

* Corresponding author.

0262-8856/01/$ - see front matter
PII: S0262-8856(00)00106-2

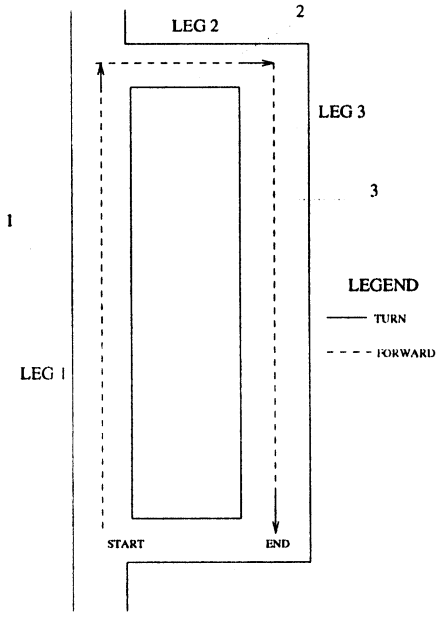

Fig. 1. The route used on the 4th floor of the McGill center for intelligent machines. Note the division into three 'legs', each consisting of zones where FORWARD or TURN (STOP in leg 3) actions are acceptable executions (see Section 2.2).

modulate map storage in cortical regions (called the *association cortex*) [4].

In contrast to mapmaking, route-learning relies on lower brain structures; the *caudate nucleus* is thought to play a large role as evinced by the inability of rats to store the necessary cue information following caudate lesions [5].

Neural network simulations comprising the route and map systems co-acting have been carried out with agreement with the physiological data; specifically, that hippocampal lesions beget a reliance on the route system for navigation [2].

1.2. The Kohonen map

Many populations of biological neurons are known to be arranged in spatial 'map' formations, where cells that are 'close' together tend to optimally respond to stimuli that are 'similar'. Primate cortical area VI, for example, was discovered by Hubel and Wiesel [6] to be organized into blocks containing columns that respond best to specific visual edge orientations. The *inferotemporal cortex* (IT) is also comprised of columns, responding to complex shapes [7]. Within a given column in IT, the cells' preferred shapes are quite similar; across columns, neurons with different selectivities are found.

The neural populations discussed above are said to be *topologically organized*. A neural network architecture designed to simulate this organization is the *Kohonen self-organizing map*(SOM) [8]. Its objective is to cluster into some meaningful pattern, the frequency of occurrence of a set of *training inputs* x_k of dimensionality n (i.e. $x_k = [x_{k1}, x_{k2}, \ldots, x_{kn}]^T$. It consists of an array (usually two-dimensional, although others are possible) of units each defined by a *weight* m_i, also of n dimensions ($m_i = [m_{i1}, m_{i2}, \ldots, m_{in}]$). During *training*, an input x is compared to all the units relative to some pre-defined similarity metric; a common one is the *Euclidean distance*. The *location* of the most similar, or 'winning' m_i is expressible as:

$$c = \arg \min_i \|x - m_i\| \tag{1}$$

During training, the closest matching neuron, along with those in a specified *neighborhood* of it, have their weights updated by the rule:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{2}$$

where t is the discrete-time instant of presentation of input $x(t)$. The weights for the selected neurons have their 'positions' altered toward the input x along a 'line' joining x to the old weight. The term h_{ci} is called the *neighborhood function*. We use a *neighborhood set*, a collection of points, whose size diminishes with time centered around the winning node. Only weights within the set are updated. Thus, we have:

$$h_{ci}(t) = \begin{cases} \alpha(t) & \text{if } i \in N_c \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The parameter $\alpha(t)$ is called the *learning rate*; it is also a decreasing function of training time.

2. The route-learning and following system

We now describe our designed mobile agent route-learning system. Here, we explain how it would learn:

1. Different places based on views from a camera system.
2. The actions associated with those views in order to follow a route.

and subsequently, after training, how it would 'recall' the required actions from a series of views while operating autonomously.

We should point out now that in order to be truly useful, the system must be capable of operating in real time. Owen and Nehmzow [9], who developed and tested an online, sonar/IR-based SOM navigation system, rightfully object that accounting for the real world's tremendous variability is practically impossible through simulation. Our study, however, was a preliminary system designed to test the feasibility of using a SOM for *view-based* navigation, a task considerably more difficult than using the sonar's low-dimensionality data, especially in a non-artificial environment (i.e. without intentionally placed 'objects'). We performed a simulation to test the system's capacity to 'recognize' a view and see whether it merits a real-time implementation.

The target route went through an office hallway

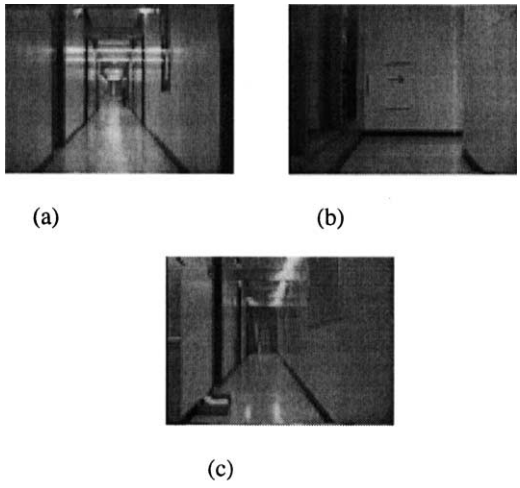

Fig. 2. A sample of the raw image data provided by the camera setup; (a), (b), and (c) are views from locations which roughly correspond to the numbered points in Fig. 1. Note that views (a) and (c) are similar; this observation is shown to be echoed by our network in Section 3.2.

environment, shown in Fig. 1. The path is conceptually divided into different zones; this will be discussed later. A sequence of training images on the route was obtained using a video camera mounted on a manually driven trolley. Masking and electrical tape were placed on the floor and walls at various locations to inform the trolley operator when turns were required; no other artificial cue information was imposed. Five *training* passes were performed, with the camera data stored on video tape. The images were later sampled using a gray-scale frame capture utility. Some raw images from the numbered viewpoints in Fig. 1 are shown in Fig. 2. Five *test* passes were also filmed to assess the autonomous performance of the system.

2.1. Image pre-processing

After the acquisition of training images, a data pre-processing stage reduced the large dimensionality of the input to a tractable yet useful size. This stage also tried to achieve a degree of invariance to transformations that result from changes in vantage.

The initial raw image size was 640×480 pixels, giving a 307,200-element vector. This was reduced to 320×240 (76,800 elements) by subsampling and averaging image regions. The next step was to remove the images' peripheral data and maintain the central portion. The inner part was treated as a single 'object'. Bachelder and Waxman [10] have also used this technique of reducing the peripheral significance in their view-based navigation system. We performed the central windowing with a *mesa filter* [11], generated by the convolution of a disc with a Gaussian:

$$M(u,v) = \left(\frac{\gamma}{f}\right)^2 \exp(-\pi(r\gamma f)^2) \times \Pi\left(\frac{r}{2f}\right) \tag{4}$$

where Π is a unit-valued disc region, γ defines the *sharpness* of the 2D Gaussian, f specifies the disc radius, and r is the radial image-plane coordinate $sqrtx^2 + y^2$. The resulting convolution has the effect of 'blurring' the disc.

Edge detection was then performed; the centroid of the vectorized image was then repositioned to the middle of the image plane. In effect, this allowed us to work on the *scene* center rather than the *image* center, compensating for small translational shifts in viewing position.

The images were then subject to the *log-polar* transform, a *space-variant* mapping that mimics the non-uniform photoreceptor distribution in the primate retina. It consists of sampling an image along a system of circles of exponentially increasing radius. Mathematically, an image point (ρ, θ) is transformed into a point (η, ξ) on the log-polar plane by:

$$\eta = q\theta_j \qquad j \in [1, \ldots, N_{\text{ang}}] \tag{5}$$

$$\xi = \ln\frac{\rho_i}{\rho_0} \qquad i \in [1, \ldots, N_{\text{cir}}] \tag{6}$$

where ρ_0 is the radius of the smallest sampling circle; N_{ang} and N_{cir} are the number of angular and radial subdivisions, respectively. All changes in rotation (θ) and scale (ρ) transform into translations in the log-polar plane. The mapping is periodic in θ, so rotations will merely affect the 'starting point', which results in a vertically 'shifted' image in the log ρ plane. Likewise, changes in scale map into 'delays' on ρ, which is a horizontal translation in log-polar space. The mapped images' log-polar centroids are then centered, taking the periodicity of θ into account. Finally, the images are convolved with a Gaussian kernel and subsampled. The

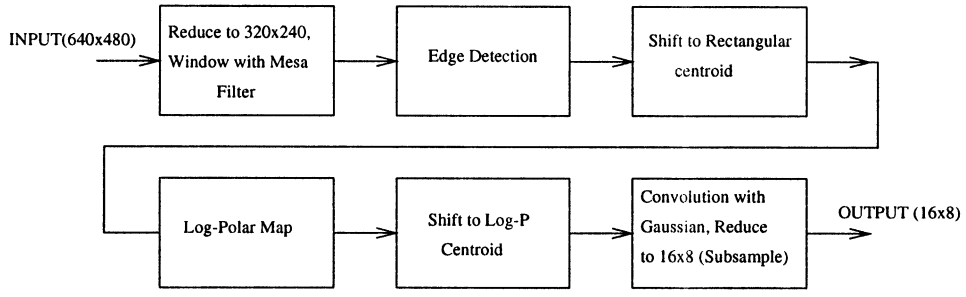

Fig. 3. The steps in the pre-processing, described in the text.

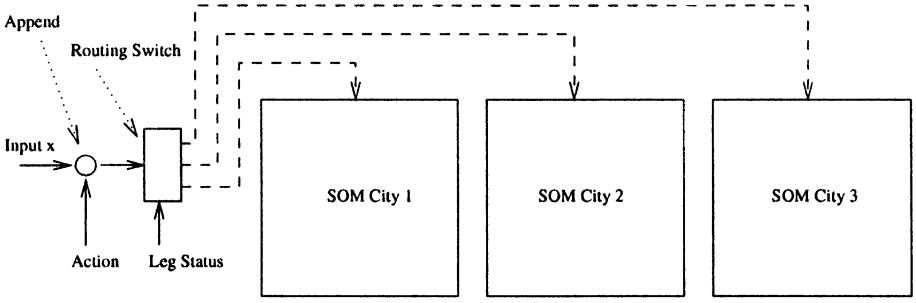

Fig. 4. Schematic of the pathways involved in *training* the network. The view input x is concatenated with the action code as described in the text. The 'leg status' is a supervised input that delineates the current active segment of the whole route. This directs the training data to the right SOM; when in leg 2, for example, SOM city 2 gets trained.

resulting 'image' size is 16×8 (128) elements. A schematic summary of the pre-processing stage is shown in Fig. 3.

2.2. The SOM architecture

After the acquired training images have been manipulated into a usable size, they are presented to the SOM system for training. Note that in a real-time implementation, the acquisition/processing/training would be performed online; in our preliminary study, we executed each step separately. In order to simultaneously allow the SOM network to independently cluster view information and to accept the supervised action inputs, it was necessary to make a slight modification to the Kohonen algorithm. This was done by concatenating the view vectors with externally dictated action codes. A two-element action vector was appended to the pre-processed view data produced by the last stage. Idan and Chevallier [12] have proposed a training algorithm for handwriting recognition, a version of which we use here, that allows a modulation of the influence of the input pattern (view) and the supervised association target (action, in our case). Making the action into an 'input' by appending it to the view data results in the network nodes learning *view–action pairs*, as *routes* have been defined to consist of earlier.

During run-time (after training), pre-processed novel views are compared to the neuron centers *with the learned actions truncated*. The winner of this phase is the neuron whose *view* weights most closely resembles the new test place's. The action required is then extracted from that neuron's action weights. In effect, this is a topologically organized *associative network*.

During preliminary experiments, a single 20×20 Kohonen map was trained on *all* the data; results were quite poor, with many misclassified actions and confusions which would result in catastrophic misorientaion in a mobile robot. The map architecture was then modified due to inspiration from cortical area IT containing shape-responsive columns. As can be seen from Fig. 1, our test route contained three 'legs'. Under the assumption that views from a given leg will be 'similar', a separate Kohonen population was set up for each. We termed these SOM populations 'cities' as an extension of the neighborhoods defined during training. The system's operation was divided into three phases (to be clarified shortly):

1. training;
2. test-orientation;
3. test-execution.

The action appended to the views *within each leg* was either 'Forward' or 'Turn' *relative to the current working leg*. The system relies on *context* to elicit the correct action; it is pre-defined, for example that leg 3 comes after leg 2. If the system 'believes' that it is in leg 2 and has reached its end (i.e. has encountered a *turn* code), then it will expect to begin receiving data corresponding to leg 3.

2.2.1. Training

The *training phase* (Fig. 4) works as follows: the pre-processed view data x_i is appended with the user-defined action vector x_a [1 0] for 'forward' (F) or [0 1] for 'turn' (S). Note that the final 'turn' in leg 3 is in fact a 'stop' command, but this is easily recognized by the contextual decision-maker which 'knows' that the third leg is the last.

The user also issues the 'leg status' to the training mechanism, which directs a data-routing switch to present the view–action vector to the correct city. The data from leg 1 is only used to train city 1. The training method follows [12]:

1. Our view vector $x_v \in \Re^{128}$ and action vector $x_a \in \Re^2$ imply the need of codebook vectors $\in \Re^{130}$. The m_i are structured as $m_{vi} \in \Re^{128}$, the 'view' components of the neuron centers, and $m_{ai} \in \Re^2$, their 'action' components. The distance is:

$$d_i = \frac{1}{1+\lambda}\frac{\|m_{vi} - x_v\|}{128} + \frac{\lambda}{1+\lambda}\frac{\|m_{ai} - x_a\|}{2} \tag{7}$$

   The parameter λ controls the 'emphasis' that view and action components get in the training. We chose $\lambda = 1$ to give the view and action equal weight.
2. As in the standard Kohonen algorithm, the winning

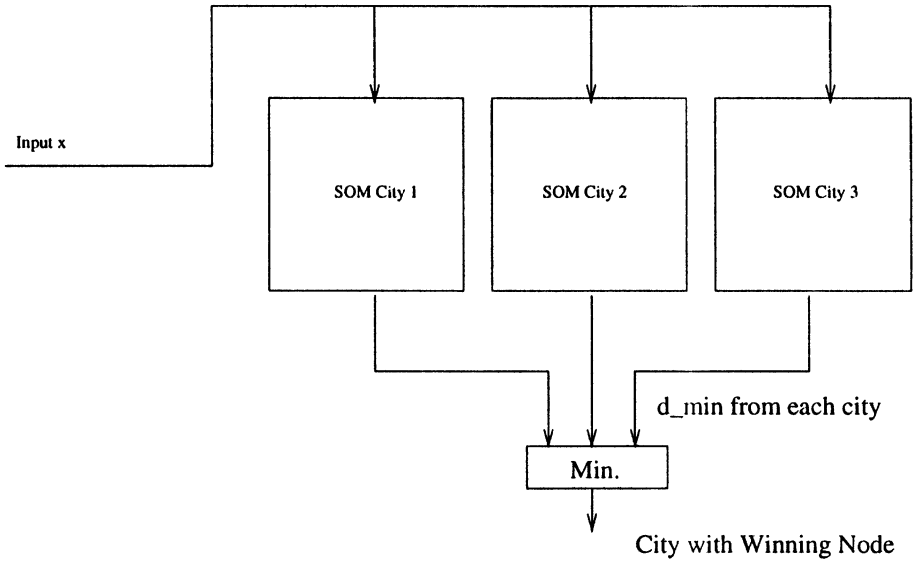

Fig. 5. The orientation system. The view input x is fed to each SOM; the local winners' deviation from the input proceed to a 'final showdown' in the *Min* block. The SOM which sent the least-divergent error measure wins the start-point competition.

neuron c is found by:

$$c = \arg \min_i d_i \tag{8}$$

3. The modification of the weights m_{vi} and m_{ai} are done individually using the same Kohonen method described above:

$$m_{vi}(t+1) = m_{vi}(t) + h_{ci}(t)[x_v(t) - m_{vi}(t)] \tag{9}$$

$$m_{ai}(t+1) = m_{ai}(t) + h_{ci}(t)[x_a(t) - m_{ai}(t)] \tag{10}$$

The 'common winner' determined jointly by x_v and x_a in the last step thus has its components updated.

4. This sequence is repeated for *all* view–action data in the training set.

A total of 2358 images were used in training. The cities were 20×20 square configurations with randomly initialized weights. 150,000 weight updates were performed using the training set. A square neighborhood set $N_c(t)$ was chosen; the size diminished to 1 with training time. The learning rate α implicit in $h_{ci}(t)$ linearly declined to zero.

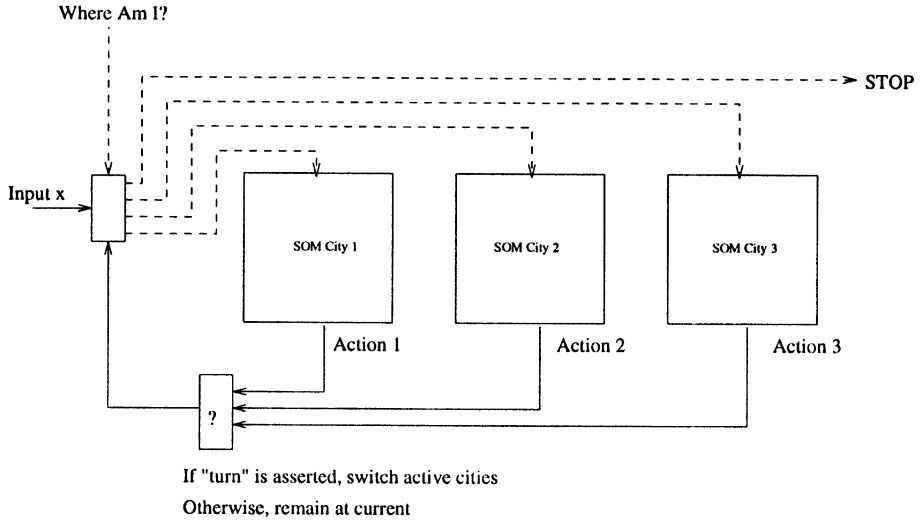

Fig. 6. The method conceived to follow the route once the system has passed the orientation phase with output 'where am I?' This is used as the input to a switch controlling passage of data to the cities. The orientation input is used to *initialize* the routing to the correct city, but after that, it is the outputed actions that signal a map-switch. The '?' on the bottom delivers the 'end-of-leg-x' signal to the routing switch, where x is the leg number. The final end-of-leg results in a STOP signal.

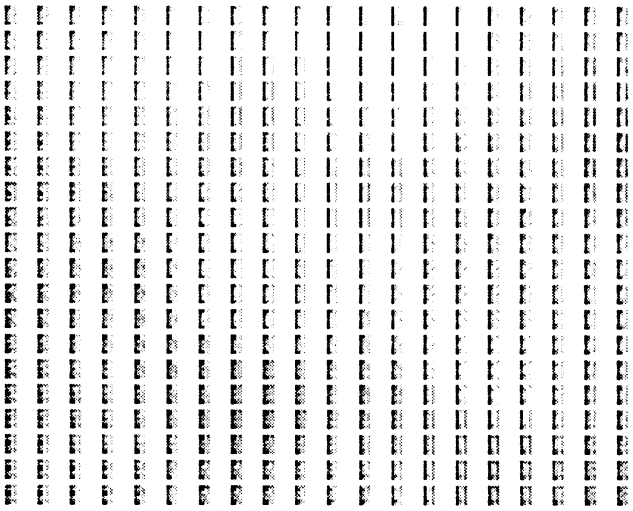

Fig. 7. The converged state of SOM city 1. The individual blocks are the network nodes; each is a 16×8 block of image. The map is an adaptation to the pre-processed input data. Note how the overall organization of the map seems to change gradually along the plane; this is motion in Kohonen's *feature space*. In the Fig. 8, we inspect the topological similarity property further.

2.2.2. Test-orientation

The first step in following a route is to identify your current position. The initial identification is the task of the *orientation* system shown in Fig. 5.

Upon startup, the novel pre-processed image vector x_i is presented to all three cities. The closest match between itself and the *view* codebooks m_{vi} is obtained for each city. The distances corresponding to these 'regional-winners' then proceed to a 'national-championship', with the champion being the minimum of the three city winning distances. The robot's location is thus determined; the problem of resolving ambiguity remains since a scene can be misclassified. (On occasion, subordinate sports teams do 'luck out' and win against the favorites!) We could not actually implement this system since we did not have real-time data contingent on the action available; 'simulation' of this is fairly meaningless. An idea, similar to one by Siebert and Waxman [13], is to rely on multiple views and the learned allowable transitions between the views to corroborate or discount the decision-making. A *short-term memory* buffer of a given number, say 10, of image-decisions is retained while the robot is in the 'confused' or 'ambiguous' state. Once a certain number of correctly defined view-transitions have been executed, the robot enters the confident *test-execution* phase (Section 2.2.3). While the robot is confused, however, a certain 'exploratory' fixed-action pattern is necessary to try to 'catch' a correctly recognizable view.

2.2.3. Test-execution

Once the starting location has been correctly identified, the remainder of the route follows the algorithm described in this section, and shown in Fig. 6. The input 'Where am I?' designates which city the route will start from and directs the next pre-processed image to it.

From then on, the orientation mechanism is passive; the context-dependent nature of the system takes over the decision making as follows:

1. Within the operating city, the closest matching view vector m_{vc} to the image x_{vi} is determined; the corresponding action vector of node c is then executed.
2. The process continues in this 'municipal' fashion until the action m_{ac} dictates a turn (or more accurately, an 'end-of-leg'). The control system issues the required action code *depending on the context*, i.e. current city. In our case, we know that after the first leg, a right-turn is required, and that after the third segment, the route is complete.
3. After executing the 'turn', the system 'expects' to receive input corresponding only to the next city in the sequence. We observe that control is constrained to flow in a *sequential*, *unidirectional manner*.
4. In the final (third for us) city, the 'turn' command is synonymous with 'stop'.

We concede that the decision and orientation systems are far less biologically realistic than the classification and learning methods; they do, however, *functionally* resemble the *cognition* that occurs while following routes. Context is often used while identifying cues; for example, a view of a traffic light on a certain street corner would not cause confusion of being somewhere else just because most traffic lights look similar. The *cognitive context* of 'where am I' prunes the possible influences that visual stimuli can have on place-recognition. The decision-maker presented is useful from an engineering perspective.

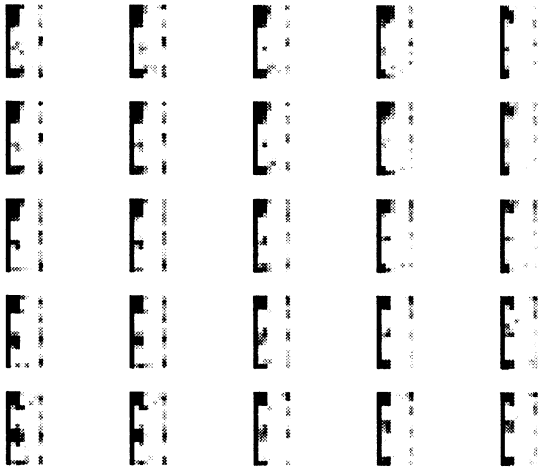

Fig. 8. Here, we show a magnification of a 5×5 top-left portion of Fig. 7. Observe how similar the codebook vectors seem; nodes that are farther apart tend to look different. This smooth transition of the mapping of the input is designed to accommodate the gradually changing view obtained as one proceeds down the route.

3. Simulation results

We now present various experimental findings. The network operation is discussed in the following phases:

1. SOM training;
2. SOM test: initial orientation;
3. SOM test: post-orientation execution.

Table 1
Performance of the initial orientation system. The images represent those that would be used upon initiation of the route. 'Bad' images are those that are classified *outside* of their actual city

Test error: orientation system			
Test pass no.	Total imgs.	Bad classif.	% Error
City 1			
1	55	2	3.64
2	51	4	7.84
3	55	4	7.27
4	27	5	18.51
5	55	6	10.91
City 2			
1	55	0	0.00
2	54	1	1.85
3	55	0	0.00
4	54	3	5.56
5	52	1	1.92
City 3			
1	65	5	7.69
2	63	3	4.76
3	60	9	15.00
4	59	7	11.86
5	64	6	9.38

3.1. SOM training

Fig. 7 shows the converged state of the 20×20 city 1; Fig. 8 shows the 5×5 top left corner of the map for clarity. The network nodes are the 'best-fit' pre-processed views that the weights converged to during training. Note how in Fig. 7, the nodes seem to be 'shaped' to give the whole network a sense of overall order; this is the topological organization of the map.

3.2. SOM test: initial orientation

We now examine the performance of the SOM on the images in the test set. Each test image was fed to the orientation match system individually; an error rate compiled for each leg of the route. These are shown in Table 1.

To show some of the difficulty encountered during orientation, we observe the classification response to two sample test routes (total number of images: 175 and 170). This shows what would have occurred had the route been instigated from that location (viewpoint). In Fig. 9, our a priori knowledge of the *correct* location is plotted against the *actual* orientation decision for *both* routes. It is worth noting that legs 1 and 3 were most frequently confused; views were rarely misclassified as either falsely belonging to 2 or belonging elsewhere, while actually in 2. (Two test passes resulted in 0% error there!) We hypothesize that this is a consequence of the similar appearance of route legs 1 and 3 (see Fig. 2). A human navigator can resolve this discrepancy based on inference from prior knowledge. It is quite conceivable that if one were to 'wake up' in either leg 1 or 3, some confusion about location can initially be present; after all, they are both long hallways with doors on the sides! Subsequently wandering about and thinking relieves the doubt. Our system faces the same ambiguity upon its own 'wake-up'; hence our stated need for an exploratory system. Nonetheless, the performance of this system is promising (isolated worst-case error of 18%).

3.3. SOM test: post-orientation execution

Finally, we examine how well the networks issued appropriate actions *within the given legs of the route*. Once again, the overall results will be shown in tabular form and the functional details of a few sample route executions will be followed.

Table 2 shows the percentage of the forward and turn section for each leg that were correctly assigned an action. Earlier we assigned the vectors [1 0] and [0 1] to represent 'forward' and 'turn', respectively. Prior to classification, the action output elements m_{aij}, $j = 1, 2$, were thresholded as follows:

$$m_{aij} = \begin{cases} 0 & \text{for } 0 \le m_{aij} \le 0.25 \\ 1 & \text{for } 0.75 \le m_{aij} \le 1 \\ m_{aij} & \text{otherwise} \end{cases} \tag{11}$$

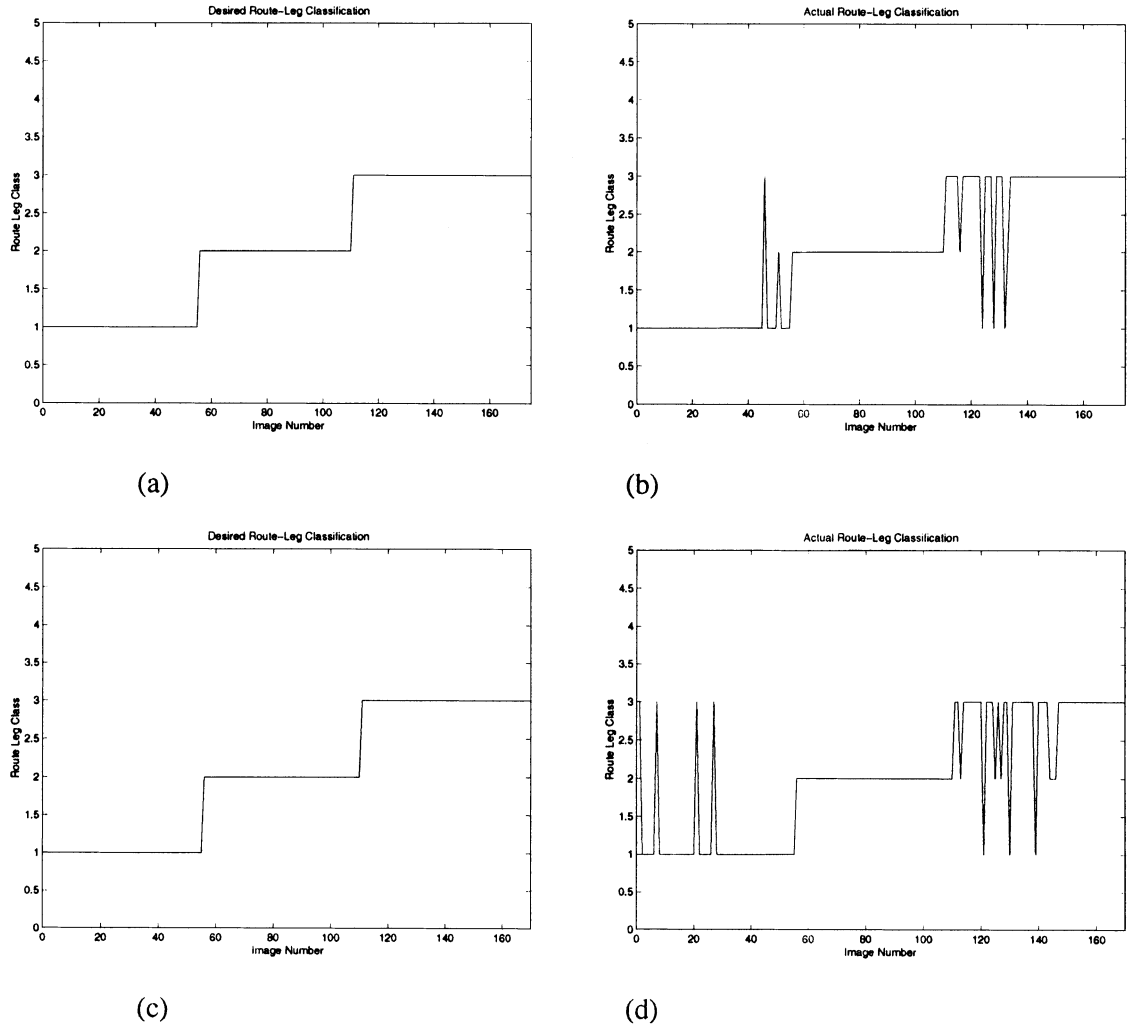

Fig. 9. Performance of the orientation system for two test routes. (a) and (c) are the *desired* orientation results (i.e. the *actual* location of the robot while the test was performed) for the two passes; (b), (d) are the 'believed' decision of our system. 'Spikes' occur where falsely classified images were obtained; note how most spikes are from leg 1 to 3 and vice versa; those segments of the route looked similar.

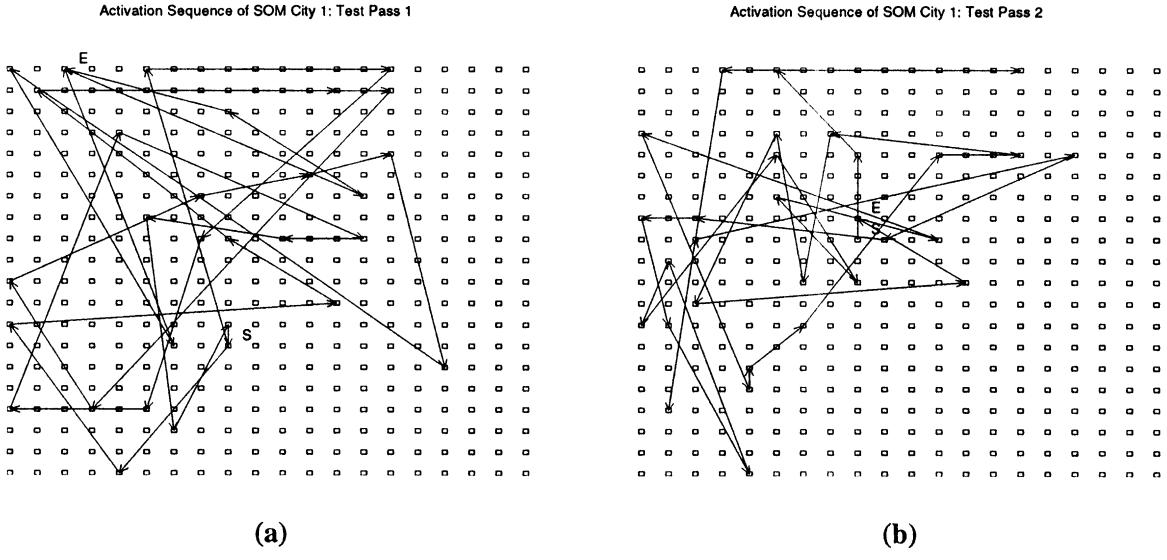

Fig. 10. Activation profiles for SOM city 1 for two test passes of the route; arrows show progression of the most excited neuron as the route is followed. Note that the two patterns are quite disparate.

Table 2
Results of the *action-execution* system, broken down into the three cities and individual route passes for clarity. In this case, the *action* depends on competing neurons *within a given city only*

Test error: execution system			
Test pass no.	Total imgs.	Bad classif.	% Error
City 1			
1	55	9	16.36
2	51	1	1.96
3	55	8	14.54
4	27	3	11.11
5	55	2	3.63
City 2			
1	55	3	5.45
2	54	1	1.85
3	55	1	1.82
4	54	2	3.70
5	52	2	3.85
City 3			
1	65	2	3.08
2	63	2	3.17
3	60	2	3.33
4	59	3	5.08
5	64	3	4.69

Note that outside of the ranges decreed by the above function, the action elements remain the same, and a misclassification subsequently results. The tables thus show the classification of the *thresholded* actions. Note that in this phase, it is *local* similarity or ambiguity between two actually different views that gives misclassification.

The erroneous outputs in this case include meaningless action codes (such as [0 0] or [1 1], which some nodes did converge to). Leg 1 was the worst 'disciplined' of the three; again, we posit that the smoother (and hence more ambiguous) view transition to the 'turn' region resulted in this fact. Legs 2 and 3 had a fairly sharp definition to the entry of their 'stop' zones. Again, though, research into a short-term memory-based confidence system that relies on more than one image to execute is required.

An interesting aspect of the system to look at is the activation sequence of the units within the topologically ordered SOMs. Fig. 10 shows the response sequence of the two sample routes in Fig. 9 in city 1. The arrows show the transition of location of the winning neuron as the view sequence along the route proceeds; nodes corresponding to the start and end locations are labeled 'S' and 'E', respectively. An important observation to draw now is that a geometric relation between two points in space *does not at all imply that the relation will be preserved in the SOM*; the node neighborhoods represent 'perceptually' similar (assuming that the Euclidean norm is a perceptual measure!) locations. Also, two different executions of the route give substantially different activity profiles despite the fact that the views on these two passes look very similar!

4. Conclusions

This paper has looked at a neural view-based route-learning mechanism. It was inspired by the robust performance of biological navigation. The preliminary results appear promising. View data was pre-processed, associated with the required actions, and fed to an SOM network for topological view–action clustering. Subsequently, actions while following the route autonomously at viewpoints along it would be dictated by the system. The order-dependent nature of routes was exploited in the design of the SOM architecture. A 'start-up' orientation system was used to permit execution of the route from any point.

References

[1] E.C. Tolman, Cognitive mapping in rats and men, Psychology Review 55 (1932) 189–208.
[2] N. Schmajuk, A. Thieme, H. Blair, Maps, routes, and the hippocampus: a neural network approach, Hippocampus 3 (3) (1993) 387–400.
[3] J. O'Keefe, L. Nadel, The Hippocampus as a Cognitive Map, Clarendon Press, Oxford, 1978.
[4] N. Schmajuk, Role of the hippocampus in temporal and spatial navigation: an adaptive neural network, Behavioral Brain Research 39 (1990) 205–229.
[5] M.G. Packard, J.L. McGaugh, Double dissociation of fornix and caudate nucleus lesions on acquisition of two water maze tasks: further evidence for multiple memory systems, Behavioral Neuroscience 106 (1992) 439–446.
[6] D. Hubel, T. Wiesel, Receptive fields and functional architecture of monkey striate cortex, Journal of Physiology 195 (1968) 215–243.
[7] I. Fujita, K. Tanaka, Columns for visual features of objects in monkey inferotemporal cortex, Nature 360 (1992) 343–346.
[8] T. Kohonen, Self-Organizing Maps, Springer, Berlin, 1995.
[9] C. Owen, U. Nehmzow, Route-learning in mobile robots through self-organization, Proceedings of Eurobot 96 Workshop on Advanced Mobile Robotics, IEEE Computer Society, 1996.
[10] I. Bachelder, A. Waxman, A view-based neurocomputational system for relational map-making and navigation in visual environments, Robotics and Autonomous Systems 16 (1995) 267–289.
[11] A.B. Watson, The cortex transform: rapid computation of simulated neural images, Computer Vision, Graphics, and Image Processing 39 (3) (1987) 311–327.
[12] Y. Idan, R. Chevallier, Handwritten digits recognition by a supervised Kohonen-like learning algorithm, 1991 IEEE Joint Conference on Neural Networks, vol. 1576, 1991, p. 1581.
[13] M. Siebert, A. Waxman, Adaptive 3D object recognition from multiple views, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (2) (1992) 107–123.