Proceedings of the V Brazilian Conference on Neural Networks - V Congresso Brasileiro de Redes Neurais
pp. 463–468, April 2–5, 2001 - Rio de Janeiro - RJ - Brazil

Neural Mechanisms for Real-Time Pattern Categorization in Robotics Applications

Luiz M. G. Gonçalves
State University of Campinas
CP 6176, CEP 13083-970 Campinas, SP, Brazil
lmarcos@ic.unicamp.br

Cosimo Distante
University of Lecce
Via Monteroni 73100 Lecce Italy
cosimo.distante@unile.it

Antonio A. F. Oliveira
Universidade Federal do Rio de Janeiro
CP 68511, 21945-970, Rio de Janeiro, Brasil
oliveira@lcg.ufrj.br

Abstract

We present behavioraly active, self-growing neural mechanisms for pattern categorization based on multi-feature extraction that can be used in (real-time) robotic applications. By using these mechanisms, robot agents are able to learn a pattern representation for different regions detected in a restricted environment and also to categorize already known (learned) patterns. As a practical result, the robotic agents, doted with an attentional mechanism, are able to perform a visual monitoring task of their underlying space. That is, they are able to construct attentional maps of the environment, and to keep the maps consistent with a current perception of the scene dealing in an efficient manner with new or already known pattern representations.

1. Introduction

We propose the use of neural mechanisms for discriminating objects in a scene based on a model that extract multi-features from input sensory data. Data reduction and abstraction combined with self-growing classifying mechanisms for categorization are the key ingredients of the system, allowing it to be applied in robotic agents that operate in real-time environments. Our interest topics are objects (or sometimes regions of interest) in a restricted robot environment. Attention control and pattern categorization will be employed as a basis to provide interaction between the robot and the environment. By using a previously developed attentional behavior [6], our robotic system is able to foveate (verge) its eyes (cameras) onto a region of interest, to maintain attention on the region as needed, and to shift its attentional focus when the current region is no more of interest. In this work we deal with the problem of improving its performance in pattern categorization behavior. We show experiments and demonstrations involving a visual monitoring task. The goal is to learn how to construct attentional maps of the environment, learning the characteristics of all present objects, dealing with new or already known categories. After such a world representation is constructed the robotic agent can perform other tasks involving atttention and pattern recognition. Based on the pattern categorization behavior, we developed a pure reactive system, that chooses actions based on its reduced and abstracted perceptual state rather than by using a geometric model, as in classical robotics. Moreover, we developed an active behavioral strategy which provides a way to interact with dynamic environments in real-time.

2. Background and Related Work

Several new approaches have been suggested using multi-feature extraction as basis for perception [1, 2]. In general transfer functions gather information from multi-feature maps abstracting data and promoting recognition or identification behaviors. Viola [3] provides a good approach based on Bayesian theory to construct complex features from images in a hierarchical network that, after some training, is able to recognize objects given the projections (images) of them. Rybak et al. [4] describe an interesting approach for recognition, suggesting a sequential search for verifications of image fragments whose contents are processed in parallel. Rao and Ballard [5] provide a set of operators based on Gaussian partial derivatives for feature extraction, which are motivated by biological models. In past work [6] we provided an interesting and *pratically working* feature based model for perception. In relation to the above works our model includes an improved (practically feasible) set of features extracted from real-time sequences of stereo images. For categorization we used appearance-based (semi-invariant) features abstracted from Gaussian operated images plus stereo and motion patterns as input to an associative memory. The last was implemented by using a multi-layer perceptron trained with a back-propagation algorithm (BPNN) [7]. In the current work we present improvements over the previous mechanism used for categorization [6]. Besides the BPNN approach, we include experiments using a self-organizing map (SOM) [8] with a self-growing mechanism. We describe how features are extracted from the images and are mapped into an address of the long term memory, by using both SOM and BPNN approaches. We present comparisons and points to their

main advantages and problems. An interesting characteristic of both mechanisms also introduced is their self-growing capabilities. By using this ability, the system memory starts with no knowledge, discovers new representations in automatic way, and self-updates/grows its network as necessary to deal with a currently known set of objects.

3. Environments and data abstraction

We use sensory data provided by two platforms in the experiments. The first, also used in [6], is a "BiSight" stereo head that can be seen in the top of the humanoid torso *"Magilla"* presented in Figure 1. It consists of two video cameras mounted on a TRC BiSight head providing four mechanical degrees of freedom (DOF): pan, tilt, left vergence, and right vergence. Images from each camera input into a pipelined array (image) processor (IP) from "Datacube". This dedicated device, able to perform image processing operations in real-time (up to 30 frames per second), is used mainly to reduce and abstract data.

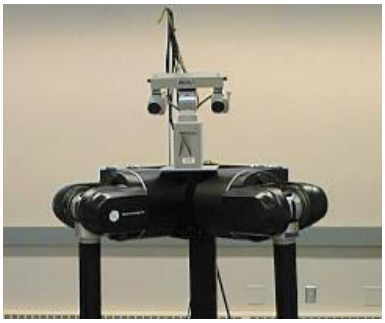

Figure 1: Current configuration of UMASS Torso.

Besides the above data, we include experiments using data from the simulated robot "*Roger-the-Crab*", seen in Figure 2. As can be seen in bottom of Figure 2, Roger has two unidimensional retinas composed of 256 pixels each. Each pixel value is calculated by using a Phong illumination model followed by a Gaussian noise process. This asserts a more natural retinal image and also simulates acquiring errors. Haptics sensory data is also computed in case Roger is grasping an object. An arbitrated object mass and the proprioceptive information relative to the arm configuration is mapped to arm torque and velocity necessary to lift the object (from which the object weight can be calculated).

In this work context, the main experiments involve the construction and maintainance of an attentional map of the environment. Categorization behavior is one of the basic requirements for this task, that is, to identify/recognize the patterns representing each region of interest to fill the maps with correct information. Figure 3 shows the main aspects of the system architecture for this task. Briefly, input from each sensor is abstracted to construct a $perceptual\ buffer$ (feature maps). These features are used as a pattern activation code in a central $associative\ memory$ which matches the properties to a long term memory (LTM) address. This LTM has stored (or will store for new objects) all information about the

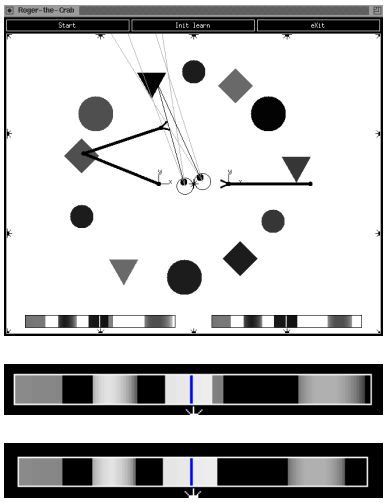

Figure 2: $Roger\text{-}the\text{-}Crab$ and its 1D retinas.

environment/objects. The $learning\ supervisor$ module operates automatically for each new pattern detected in the environment, increasing the associative and long term memories with the new pattern codes. The attentional mechanism operates based on the current task parameters and on results of categorization behavior. It selects a new action, eventually changing the current attention window (topic of interest) and calculating the movements to be applied to the DOFs to attend the new position. The servo controllers use these parameters to effectively bring the system to a new pose. This puts a new set of information in the sensory buffers, re-starting the cycle (feature extraction). We describe mainly the part of the system related with the categorization behavior (associative memory).

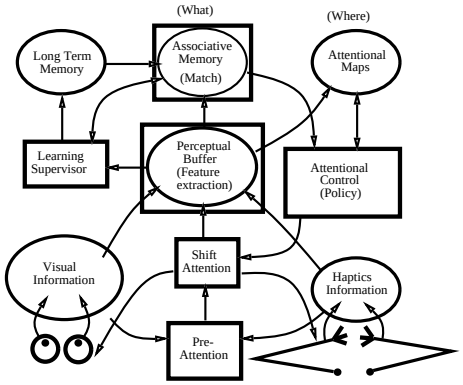

Figure 3: System control architecture

3.1 Data Reduction and Abstraction

We provide data reduction by using a $multi\text{-}resolution$ (MR) representation obtained from our robot retinas. In simulation it is composed of three levels of 32 pixels each and in the stereo head of 4 levels of 64×64. These MR images are computed by applying mean filters on the original captured (or simulated) images. Two MR representations are computed for each eye, one for further extraction of Gaussian features and another, obtained

from the difference between two consecutive frames, for further extracting motion features.

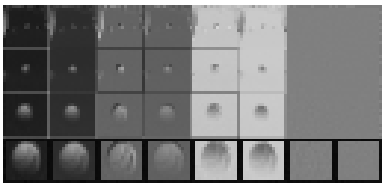

Figure 4: MRMF matrices.

A $multi\text{-}feature$ (MF) representation over the previous MR images provides data abstraction to support high-level behaviors as attention control and categorization. The result is a $multi\text{-}resolution\text{-}multi\text{-}feature$ (MRMF) representation as in Figure 4, for the stereo-head. The first six column images are Gaussian partial derivatives of gray-level intensity images and the last two are derivatives of frames difference representing motion. To generate the Gaussian part of this MF representation, the above MR intensity images are convoluted with the Gaussian derivative kernels given by:

$$\begin{aligned} G^{(0)}(x) &= \lambda e^{ax^2} \\ G^{(1)}(x) &= 2a\lambda e^{a(x^2)}x \\ G^{(2)}(x) &= 2a\lambda e^{ax^2}(2ax^2+1) \end{aligned} \quad (1)$$

where $(x) \in [-s, +s]$; $a = \frac{-1}{2\sigma^2}$, $\lambda = \frac{1}{\sigma\sqrt{2\pi}}$, $\sigma = \{0.7, 1.7, 2.9\}$, and $s = \{3, 5, 7\}$. Our representation for motion features is computed by applying the same first Gaussian derivative to the previous MR motion representation. This helps reducing noise. In the stereo head, the dedicated IP device (Datacube) can compute the MRMF at a rate of 15 frames per second, thus achieving real-time performance necessary in our system.

Following this phase, an attentional process that is not relevant to describe here computes attentional features from the above set and directs the robots to a target region.

3.2 Feature Vectors for Categorization

Once the robot is "looking" to a region, we compute from the above maps the feature set for categorization: $intensity$, $texture$, $shape$, $motion$, $size$, and $weight$. Each $intensity$ vector $I_c^{(k)}$ (the subscript "c" denotes categorization behavior), made for resolution level k, is calculated as an average in the vicinity of the corresponding Gaussian response $G^{(k)}$. Each component of the texture vector $T_c^{(k)}$ is the variance in the vicinity of a Gaussian response. In a similar way, each shape vector $S_c^{(k)}$ is the variance in the vicinity of stereo disparity ($D_a^{(k)}$). Stereo disparity is computed in the previous attentional phase using the above second order Gaussian $intensity$ $G^{(2)}$. Motion $M_c^{(k)}$ is an average in the vicinity of the magnitude $M_a^{(k)}$ of the above MR motion image, also computed in the previous attentional phase. We extract the size of an object by using edge detectors (a segmentation process) in simulation or by using normalized moments in the stereo head images. Analogously, the weight (only in simulation) is extracted from the arms sensors. All of the above features are normalized. As a result of the above averaging and variance, feature matching is more tolerant of scaling, rotation, and shift.

4. Categorization Behavior

We used two types of classifiers in the experiments to provide categorization behavior: a back-propagation neural network and a self organizing map (SOM) [8]. They are self-growing, that means, we set a threshold determined empirically to tell whether a representation is a new one; if the activation for a given pattern is below this value, the $learning\ supervisor$ module can be automatically invoked inserting the new feature set into LTM and updating (creating new nodes/BPNN or neurons/SOM) and retraining the BPNN or SOM classifier. Note that in this situation (no identification at a given trial), an arm or even the eyes can be moved to get a better sensory response. In fact, after a certain (small) number of trials, the representation can be classified as new or identified, then the current position in the attentional maps can be updated and attentional focus changed.

4.1 The Back-propagation Classifier

The BPNN is a feed-forward and totally connected multi-layer perceptron. Activation propagates forward, that means, from the input layer, to the hidden (or intermediate) layer(s), then to the output layer. The acquired knowledge is codified in the synapses (or weights) in each connection between the network unities (nodes). The weights are adjusted by a training process. The activation obtained in the last layer determines an answer representing the activation obtained by the whole network. The BPNN algorithm is currently well defined and more details can be found in works like [9, 10, 7]. Figure 5 shows the structure of the BP net used here. It has one input node for each input feature. The number of nodes in the output layer changes dynamically by using the self-growing mechanism; a new node is created for each new representation detected. A weighted function of the minimum and maximum error during training is used as the threshold to define if a representation is new. The number of hidden nodes is determined empirically. Actually, 1.5 times the number of output nodes gives good results.

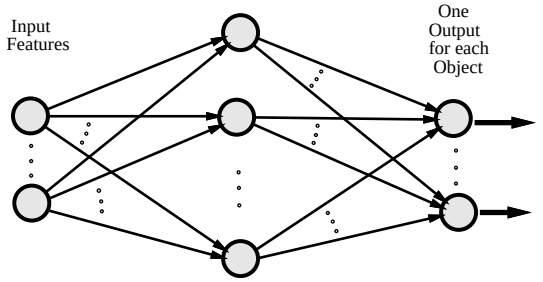

Figure 5: Topology of BPNN used.

4.2 Self Organizing Maps

We further implemented the associative memory using a network based on the self-organizing map introduced by Kohonen [8]. The network embeds a competition paradigm for data clustering by imposing neighborhood constraint on the output units, such that a certain topological property in the input data is reflected in the output's unit weights. The Euclidean distance is considered as the measure of similarity (dissimilarity) and the winning neuron is the one with the largest activation (the lowest distance). The updating procedure takes into account the neighborhood function, in order to self-organize the network around the winner. Inter-neuron connections assure lateral plasticity to preserve the topology of the map (we have chosen a rectangular topology). The Kohonen network moves the neurons towards the input probability distribution. In a first stage (off-line) the net is trained with a few objects by presenting each abstracted feature and selecting the winner neuron. This is made to roughly approximate the input probability distribution of the feature vectors. Then the winner's neighborhood is adapted in order to move each neuron (weighted with a Gaussian function) towards the input vector. During the on-line stage, the quantization error measured on each input vector controls the growing process. Related works on growing Kohonen's map have been proposed in [11, 8]. The allocation process of new recognition codes (*codebooks*) is controlled by a threshold value that is empirically found. The algorithm can be summarized as follows:

1. Initialize randomly all the codebook vectors m_{ij} for $i = 1, ..., N$ and $j = 1, ..., M$ (MxN is the size of the map);
2. Given an input vector x find the winner c as follows:

$$|| x - m_c || = min_i \{|| x - m_i ||\}$$

3. if the distance is less than a certain threshold
   - then adapt the codebook the winner m_c

$$m_c(t-1) = m_c(t) + \alpha(t)[x(t) - m_c(t)]$$

     where α is the learning rate parameter.
   - otherwise find the second best neuron m_k and allocate a new neuron $m_{new} = x$.

Every time a new neuron is allocated, in reality we allocate a new row or column based on the position of the two closest neurons with the input vector x. The codebooks of the other new neurons allocated are initialized based on the interpolation vector found as follows:

$$m_{new} = \lambda m_c + (1 - \lambda) m_k$$

5. Experiments and Results

In the experimental tests, both approaches have been applied to the monitoring task with acceptable results. Basically, many instances and types of objects are placed on a table, for the stereo head, or in the room representing *Roger*'s work-space. By using the attentional behavior, our robotic agents are able to focus attention on selected regions/objects, thus the categorization behavior can learn their characteristics inserting a representation for each one in the system memory and incrementally construct/update attentional maps. In the simulation platform, if an object is not identified only by visual information, an arm movement can be done to try an improvement in the information set (say measuring the object weight).

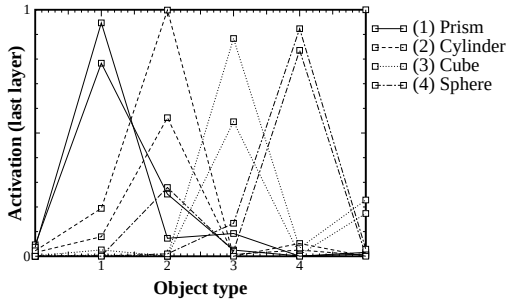

Figure 6: Activations for objects in BPNN.

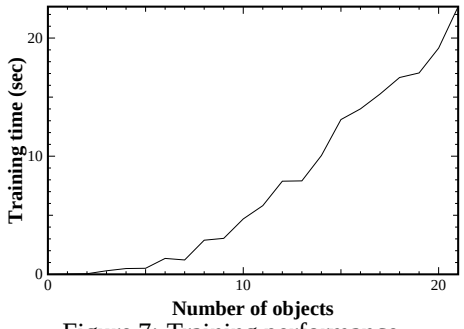

Figure 7: Training performance.

Figure 6 shows confidence for the BPNN (simultaneous activations in its last layer), for the object types seen in Figure 2. For each instance, the upper line is the highest activation (object on ideal or learned pose). The next line is the lowest activation yet allowing identification. Object poses were degraded when viewed from *Roger*'s eyes. We could verify that rotations up to 30 degrees were well supported by the system. We note some partial activation denoting some ambiguity between small-circles and big-circles, but still under a threshold, thus allowing to discriminate them.

Several experiments were done to test the associative memory (BPNN) performance. Figure 7 shows a graph with training time in seconds illustrating performance as the number of objects increases. Actually, this result was obtained using input vectors composed of 112 features in the stereo head platform. The graph shows SThe nean apparently soft exponential function, a characteristic of the BP model. In practice, this issue does not compromise the system performance as a model of short-term, working memory of ten objects (less than 3 sec) seems quite acceptable. Also, in a practical situation, the system would save the synaptic weights and network configura-

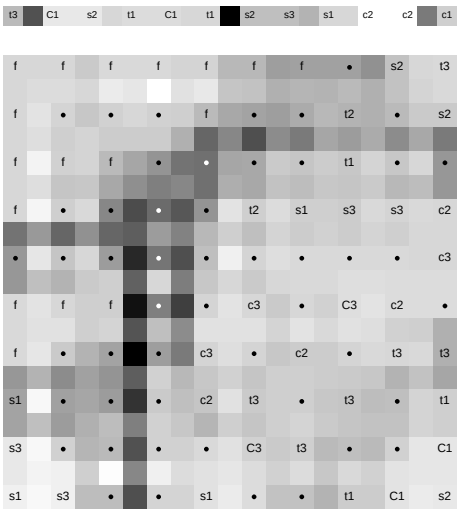

Figure 8: Initial and final (SOM) 1D maps for raw scattered data (due to space reasons, bottom map is shown in 2D).

tion (also the attentional cues) for future use. After a certain time, the probability of finding a new object would be very small, not interfering in system performance.

To test the SOM confidence, we initially conducted some experiments using raw scattered and not well comported data, obtained from $Roger$'s retinas. We used a 1D map in order to grow the network. On the overall evaluation, the SOM has attended our expectations in these initial experiments, with only a few problems. Top of Figure 8 shows the initial phase in which the map is trained with few data. Bottom of Figure 8 shows the final map with 18 neurons after the presentation of the abstracted features for unknown objects. Due to space reasons the resulting 1D map is shown in 2D. Because of using a 1D map and of the bad quality of the data (class "f" has a very sparse distribution), as we can see in the Figure 8, a threshold was not a good compromise to allocate new neurons. As a result, class "f" was broken into several subclasses.

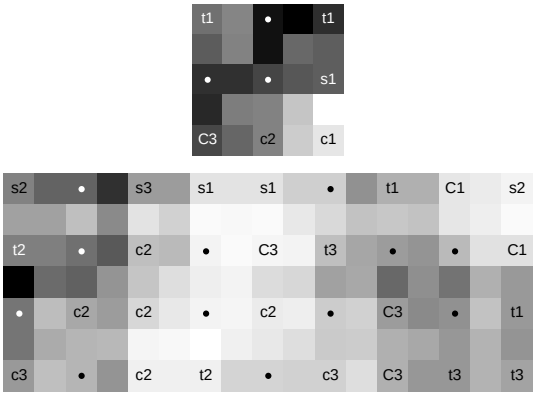

Figure 9: Growing process for the 2D map.

Besides these initial experiments, we conducted other experiments using well comported data. In this case, the SOM worked well. With a time performance significantly better than the BPNN and with almost same precision. Top of Figure 9shows the a first stage in which the map is organized and trained with a few data. Note the clear ordering mechanism of the classified objects. Thereafter, the network grows every time that a new pattern does not satisfy the threshold in the Euclidean space. We use a 2D map to grow the SOM network. Bottom of Figure9 shows the final map (36 neurons) after the presentation of unknown abstracted features. The time spent to learn or classify each new pattern presented is constant and can be neglected. Since the neurons are ordered, just a few iterations for updating an existing codebook is needed to refine it (some 1 or 2 miliseconds).

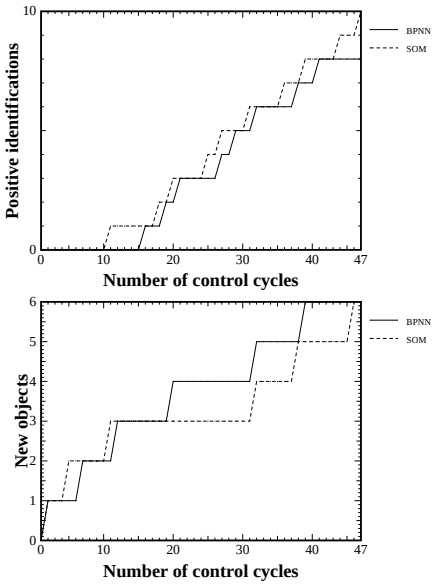

Figure 10: Number of new and known objects detected.

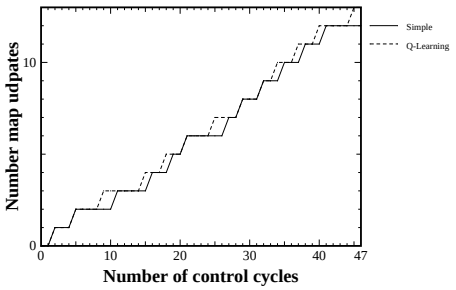

Figure 11: Number of updates in attentional maps

Some evaluation of the algorithms applied to a same environment (with the same light definitions, and the same objects) are shown in figure 10, for the simulator $Roger$. By using the SOM approach, the system has performed a few more positive identifications (of already known objects), and a few less new-object detections than the BPNN approach. Figure 11 shows another evaluation for the number of map updates realized by using both approaches. Slightly different results, with both approaches alternating position in the graphs, were obtained from other experiments in which the system runs until no more new representations were detected. These differences were due to the threshold chosen to decide whether a representation is a new one. As a practical demon-

stration, Figure 12 shows some selected pictures of a sequence in which *Roger* categorize all regions/objects in its environment.

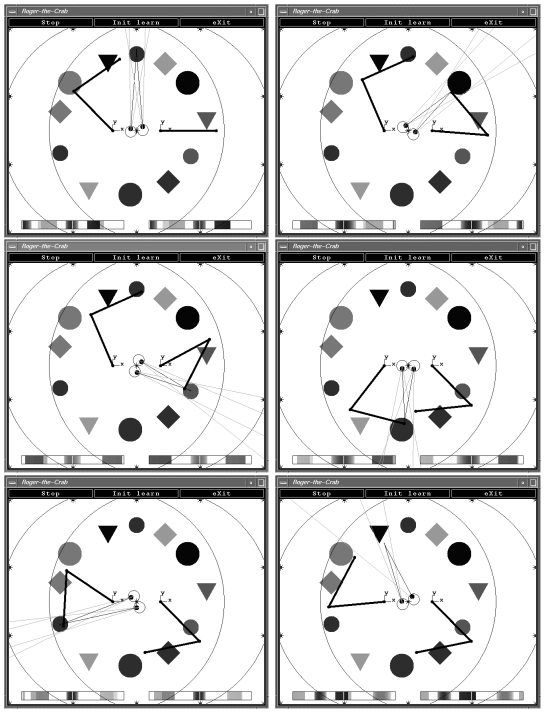

Figure 12: *Roger* constructing attentional maps.

6. Discussion, Conclusion and Future Work

By analyzing the above results, we could see that both methods worked well in our approach for categorization in the real-time monitoring task. At this point, we can not generalize that the SOM approach is better for this function than the BPNN approach, but for sure it has a much better processing time. Other tasks with other types of application (for example, includding top-down attention) have to be tested. The back-propagation network demonstrate to be useful in cases of short memory representations when no more than 10 objects are used. Beyond that, some main problems are the time spent for retraining the network when new categories are found. The BPNN topology used in this work demonstrate be useful on categorization of different pattern representations leading to the same object instance (different views). In this case, it approximates the function as a surjection of $R^n \rightarrow N$. The SOM can accomplish real-time performance even augmenting the number of features (for example, using original features without any sampling for data reduction). SOM are useful because they have a relatively low training time against the resolution of the map. But, some times, a set of features not well defined may cause problems, making our self-growing mechanism break the feature space with introduction of one more neuron, as we have seen above for class "f".

In this work we tried to give a full set of features and let our classifying tools use it for categorization. An improvement could be done by using some learning strategy for feature discriminability. In this way, the best set of features (small) would be used by the system, improving performance. We remark that other top-down attentional tasks (for example, search for an object) can also be formulated using the same model. We have tested and succesfully used the tools in our basic real-time robotic application, thus allowing it to be applied in other general, on-line tasks. Finally, another possibility of future work is to test both approaches using a moving fovea representation (currently, our fovea is defined in the image center). Note that this would save time by avoiding the execution of physical movements, but, a problem is that it may degrade categorization precision, due to more image distortions.

References

[1] P. Van de Laar, T. Heskes, and C. Gielen. Task-dependent learning of attention. *Neural Networks*, 10(6):981–992, August 1997.

[2] L. Itti, J. Braun, D. K. Lee, and C. Koch. A model of early visual processing. In *Advances in Neural Information Processing Systems*, pp 173–179, Cambridge, MA, 1998. The MIT Press.

[3] P. A. Viola. Complex feature recognition: A bayesian approach for learning to recognize objects. AI Memo 1591, Massachusetts Institute of Technology, November 1996.

[4] I. A. Rybak, V. I. Gusakova, A. V. Golovan, L. N. Podladchikova, and N. A. Shevtsova. A model of attention-guided visual perception and recognition. *Vision Research*, 38(2):387–400, 1998.

[5] R. P. N. Rao and D. Ballard. An active vision architecture based on iconic representations. *Artificial Intelligence Magazine*, 78(1-2):461–505, October 1995.

[6] L.-M. Garcia, R. A. Grupen, A. A. Oliveira, D. Wheeler, and A. Fagg. Tracing patterns and attention: Humanoid robot cognition. *The Intelligent Systems and their Applications*, 15(4):70–77, July/August 2000.

[7] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proc. of Int. Conf. on Neural Networks*, pp 123–134. IEEE Press, 1993.

[8] T. Kohonen. *Self Organizing Maps*. Springer Verlag 1997.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning internal representations by error propagation. In Parallel Distributed Processing: Explorations in the microstructure of cognition*, vol 1: Foundations. The MIT Press, Cambridge, Massachusetts, 1986.

[10] P. Werbos. Backpropagation: Past and future. *IEEE International Conference on Neural Networks*, pp 343–353, 1988.

[11] J. Blackmore and R. Miikkulainen. Incremental grid growing: encoding high dimensional structure into a two-dimensional feature map. In *Proc. of the International Conference on Neural Networks (ICNN'90)*, pages p. I–450. IEEE Press, 1990.