Modified Dendrite Morphological Neural Network Applied to 3D Object Recognition

Humberto Sossa and Elizabeth Guevara

Instituto Politécnico Nacional - CIC, Av. Juan de Dios Batiz, S/N, Col. Nva. Industrial Vallejo, Mexico, D. F. 07738, Mexico

Abstract. In this paper a modified dendrite morphological neural network (DMNN) is applied for recognition and classification of 3D objects. For feature extraction, the first two Hu's moment invariants are calculated based on 2D binary images, as well as the mean and the standard deviation obtained on 2D grayscale images. These four features were fed into a DMNN for classification of 3D objects. For testing, COIL-20 image database and a generated dataset were used. A comparative analysis of the proposed method with MLP and SVM is presented and the results reveal the advantages of the modified DMNN. An important characteristic of the proposed recognition method is that because of the simplicity of calculation of the extracted features and the DMNN, this method can be used in real applications.

Keywords: Dendrite morphological neural network, efficient training, 3D object recognition, classification.

1 Introduction

Recognizing an object from an image is an important task in computer vision due to it having a variety of applications in many areas of artificial intelligence including, for example, content-based image retrieval, industrial automation or object identification for robots [1].

In recognition systems, objects are represented in a suitable way for the processing; such representations (features) can be considered as patterns, so the correct classification of these patterns is an essential part of these systems.

Besides the pattern recognition approach that uses low-level appearance information for recognizing an object, there are feature-based geometric approaches that construct a model for the object to be recognized and match the model against the image [2]. In this paper, we describe a method to recognize 3D objects from 2D images through a pattern recognition approach using a modified DMNN for classification [3].

The proposed method of object recognition system has two phases: training and testing phase. During the training phase, the images are given as input to the system, the image is preprocessed and the feature vector is generated. The feature vector is stored with the image label and the DMNN is trained. During testing phase, the test image is given to the system; the features are extracted

J.A. Carrasco-Ochoa et al. (Eds.): MCPR 2013, LNCS 7914, pp. 314–324, 2013.

from the preprocessed image and then the classifier is employed to recognize the object.

The system was evaluated on the COIL-20 dataset [4] and on a set of images captured without controlled lighting condition. The system was trained using a small number of images of each object class, and as we show experimentally, it is then able to identify objects under other non trained transformations.

The rest of the paper is organized as follows. Section 2 describes the feature extraction process. Section 3 explains the modified DMNN. Section 4 presents the proposed object recognition method. Section 5 is focused to present the experimental results where the classification method used in the recognition process is tested and compared with other classifiers. Finally, Section 6 is oriented to provide the conclusions and directions for further research.

2 Feature Extraction Process

Feature extraction is a process for converting the input data into a set of features that extract the relevant information from the data in order to perform a task, in this case, recognize an object. In this paper, we use Hu's moment invariants and the mean and standard deviation of the distribution of the pixels to represent the object.

2.1 Moment Invariants

The geometric moment invariant was introduced by Hu based on the theory of algebraic invariants [5]. Image or shape feature invariants remain unchanged if the image undergoes any combination of the following transformations: translation, rotation and scaling. Therefore, the moment invariants can be used to recognize the object even if the object has changed in certain transformations.

The 2D moment of order $(p+q)$ of a digital image $f(x,y)$ of size $M \times N$ is defined as [6]

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q f(x,y) \tag{1}$$

where $p, q = 0, 1, 2, ..$ are integers. The corresponding central moment of order $(p+q)$ is defined as

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x-\bar{x})^p (y-\bar{y})^q f(x,y) \tag{2}$$

for $p, q = 0, 1, 2, ..$ where $\bar{x} = \frac{m_{10}}{m_{00}}$ and $\bar{y} = \frac{m_{01}}{m_{00}}$.

The normalized central moments, denoted η_{pq}, are defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{3}$$

where $\gamma = \frac{p+q}{2} + 1$ for $p + q = 2, 3, ...$

A set of seven invariant moments can be derived from the second and third moments

$$\phi_1 = \eta_{20} + \eta_{02} \tag{4}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{5}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - 3\eta_{03})^2 \tag{6}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{7}$$

$$\begin{aligned}\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + \\ (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]\end{aligned} \tag{8}$$

$$\phi_6 = (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{9}$$

$$\begin{aligned}\phi_7 = (3\eta_{21} - \eta_{30})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + \\ (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]\end{aligned} \tag{10}$$

In this paper we only use the first and second invariant moments to represent the object.

3 Dendrite Morphological Neural Networks

The dendrite morphological neural networks (DMNN) were first described by Ritter and colleagues in [7] and [8]. DMNN emerge as an improvement of classical morphological neural networks (MNN), originally introduced by Davidson in [9] and then re-discussed by Ritter and Sussner in [10]. Algorithms and applications of MNN can be found in [11], [12], [13], [14], [15], [16], [17], [18] and [19]. Morphological perceptrons with competitive learning, a variation of standard morphological perceptrons are discussed in [20]. Processing at the level of dendrites and not only at the level of the cell body allows neurons to power their processing capacities [21]. This fact is taken into account by Ritter and colleagues in the DMNN proposal.

A key issue in the design of a DMNN is its training; this is in the selection of the number of dendrites and the values of synaptic weights for each dendrite. Diverse algorithms to automatically train a DMNN can be found in [7], [8], [22], [23], [24], [25] and [26].

A novel algorithm for the automatic training of a DMNN was proposed in [3] and it is applied in this work for object recognition.

3.1 Basics on Dendrite Morphological Neural Networks

Morphological neural networks are closely related to Lattice Computing [27], which can be defined as the collection of Computational Intelligence tools and techniques that either make use of lattice operators *inf* and *sup* for the construction of the computational algorithms and replace the multiplication operator of the algebra of the real numbers $(\mathbb{R}, +, \cdot)$ by the addition. Therefore, morphological neural networks use lattice operations $\vee$ (maximum), or $\wedge$ (minimum), and $+$ from the semirings $(\mathbb{R}_{-\infty}, \vee, +)$ or $(\mathbb{R}_{\infty}, \wedge, +)$ where $\mathbb{R}_{-\infty} = \mathbb{R} \cup \{-\infty\}$ and $\mathbb{R}_{\infty} = \mathbb{R} \cup \{\infty\}$. The computation at a neuron in a MNN for input $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is given by

$$\tau_j(\mathbf{x}) = a_j \bigvee_{i=1}^{n} b_{ij}(x_i + w_{ij}) \tag{11}$$

or

$$\tau_j(\mathbf{x}) = a_j \bigwedge_{i=1}^{n} b_{ij}(x_i + w_{ij}) \tag{12}$$

where $b_{ij} = \pm 1$ denotes if the ith neuron causes excitation or inhibition on the jth neuron, $a_j = \pm 1$ denotes the output response (excitation or inhibition) of the jth neuron to the neurons whose axons contact the jth neuron and w_{ij} denotes the synaptic strength between the ith neuron and the jth neuron. Parameters b_{ij} and a_j take $+1$ or -1 value if the ith input neuron causes excitation or inhibition to the jth neuron.

The computation performed by the kth dendrite can be expressed by the formula:

$$D_k(\mathbf{x}) = a_k \bigwedge_{i \in I} \bigwedge_{l \in L} (-1)^{1-l} \left(x_i + w_{ik}^l\right) \tag{13}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ corresponds to the input neurons, $I \subseteq \{1, \ldots, n\}$ denotes to the set of all input neurons N_i with terminal fibers that synapse on the kth dendrite of a morphological neuron N, $L \subseteq \{0, 1\}$ corresponds to the set of terminal fibers of the ith neuron that synapse on the kth dendrite of N, and $a_k \in \{-1, 1\}$ denotes the excitatory or inhibitory response of the kth dendrite.

Clearly, $I \neq 0$ y $L \neq 0$ since there is at least one axonal fiber coming from at least one of the input neurons with synapse dendrite k. The activation function used in a MNN is the hard limiter function that assigns 1 if the input is greater or equal to 0 and assigns 0 if the input is lesser than 0. A more detailed explanation can be found in [7] and [25].

Figure 1 shows a dendrite morphological neural network with an input layer that separates two classes: C^1 and C^2. The neurons of the input layer are connected to the next layer via the dendrites. The black and white circles denote excitatory and inhibitory connection respectively. The geometrical interpretation of the computation performed by a dendrite is that every single dendrite defines a hyperbox which can be defined by a single dendrite via its weight values w_{ij} as the example shows.

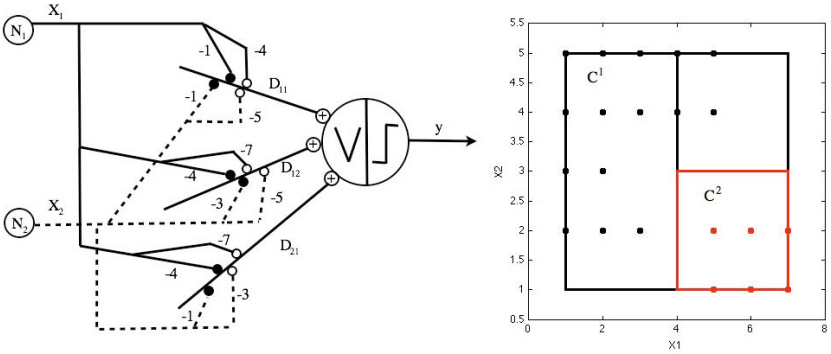

Fig. 1. Morphological neural network for solving the two classes separation problem that appears on the right side of the figure. Points of C^1 are shown as black solid dots and points of C^2 are shown as red solid dots. Points of class C^1 are enclosed by the two black boxes and C^2 by the red box generated by the dendrites.

3.2 The Training Algorithm

The proposed training algorithm for a DMNN in [3] is summarized below. Given p classes of patterns, C^k, $k = 1, 2, ..., p$, each with n attributes, the algorithm applies the following steps:

Step 1) Select the patterns of all the classes and open a hyper-cube HC^n (with n the number of attributes) with a size such that all the elements of the classes remain inside HC^n. The hyper-cube can be one whose coordinates match the patterns of class boundaries; it can be called the minimum hyper-cube *MHC*. For having better tolerance to noise at the time of classification, add a margin M on each side of the *MHC*. This margin is a number greater or equal to zero and is estimated as a function of the size T of the *MHC*. If $M = 0.1T$ then the new hyper-cube will extend that ratio to the four sides of the *MHC*.

Step 2) Divide the global hyper-cube into 2^n smaller hyper-cubes. Verify if each generated hyper-cube encloses patterns from only one class. If this is the case, label the hyper-cube with the name of the corresponding class, stop the learning process and proceed to step 4.

Step 3) If at least one of the generated hyper-cubes (HC^n) has patterns of more than one class, then divide HC^n into 2^n smaller hyper-cubes. Iteratively repeat the verification process onto each smaller hyper-cube until the stopping criterion is satisfied.

Step 4) Based on the coordinates on each axis, calculate the weights for each hyper-cube that encloses patterns belonging to C^k. By taking into account only those hyper-cubes that enclose items C^k and at this moment the DMNN is designed.

To improve the classification results, an evolutive algorithm was used to determine the optimal value of M and the hard limiter activation function of the original DMNN was changed by the maximum function.

4 Object Recognition System Overview

The flowchart of the framework for object recognition is shown in Fig. 2. For recognition purpose, a modified DMNN is used as classifier supported by features extracted from the given images. Hu's moment invariants and the mean and standard deviation of the distribution of the pixels are the features that represent the object. The original images are in grayscale, therefore the preprocessing phase converts the gray image into binary image necessary to calculate the moment invariants. The binarization is performed using the Otsu algorithm. The mean and the standard deviation are obtained directly from the grayscale images.

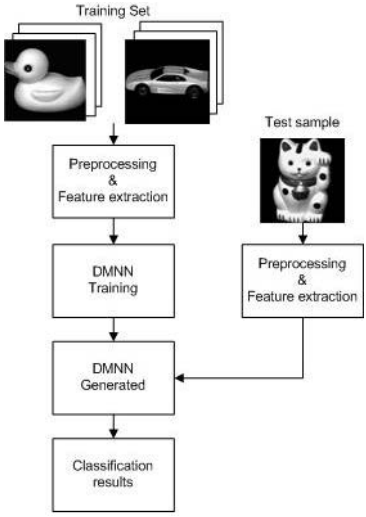

Fig. 2. Object recognition process

5 Experimental Results and Discussion

To test the performance of the object recognition approach, we use a set of images captured without controlled lighting condition and the COIL-20 dataset. The experimental dataset consists of 5 objects with 10 images each one. The images were captured using a RGB camera with a resolution of 320x240 but the region of interest of the image was of 250x200 pixels. Figure 3 a) shows the objects of the dataset.

The COIL-20 dataset [4] consists of grayscale images of 20 different objects with black background, each one is rotated with 5 degree angle interval in vertical axis. Hence for every object there are 72 images. The size of the images is of 128x128 pixels. A part of the gallery used is shown in Fig. 3 b).

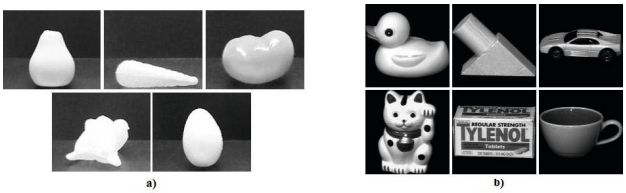

Fig. 3. a) Set of 5 objects, b) Sample objects from COIL-20 dataset

The proposed object recognition process (Fig. 2) was implemented for training and testing online using a platform developed in C#. All the algorithms were implemented on a desktop computer with Intel i7 2.2 GHz processor, with 8GB in RAM. Figure 4 shows an image of the graphic interface of the application. The 5 objects dataset was generated with this interface.

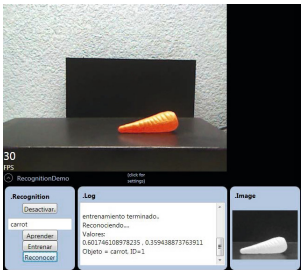

Fig. 4. Graphic interface for online training and testing for object recognition

Figure 5 presents some taken images of the carrot; on these images it is obvious that the carrot has shadows because these images were not taken under strict illumination controlled condition; in spite of this, results were satisfactory.

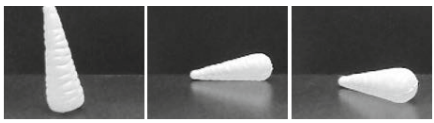

Fig. 5. Sample images of the carrot

For object representation, we use four features: the first and second Hu's moments obtained from binary images and the mean and standard deviation of

the distribution of the pixels calculated from gray images. The binary images were generated in the preprocessing stage by the Otsu algorithm.

The system was trained with 3 random images of each object and the rest of the images (7) were used for testing. To display some results MATLAB 7.11 was used; Figure 6 presents a graphic of attributes from the 5 objects, it only displays three features (first moment, second moment and mean), in this way, the patterns are in the 3D space and the results can be viewed graphically.

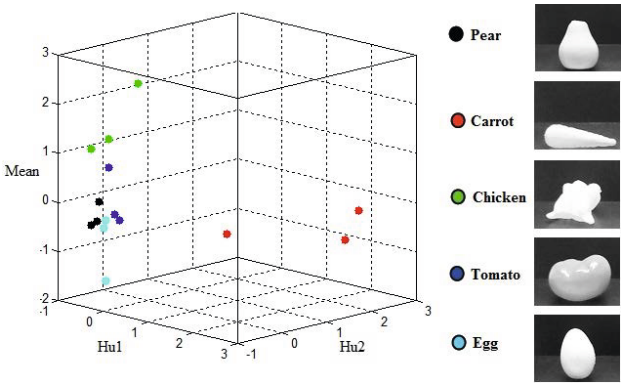

Fig. 6. Three features from the 5 objects dataset

Figure 7 a) presents the result of the training and Figure 7 b) shows the classification results. In these figures, solid dots represent the training points for the classes. Empty diamonds represent the test samples and asterisks represent the test samples classified in each class. The algorithm achieved a recognition rate of 88.57% with M=0.2861 and 10 dendrites. These results were compared with a Multilayer Perceptron (MLP) and Support Vector Machine (SVM).

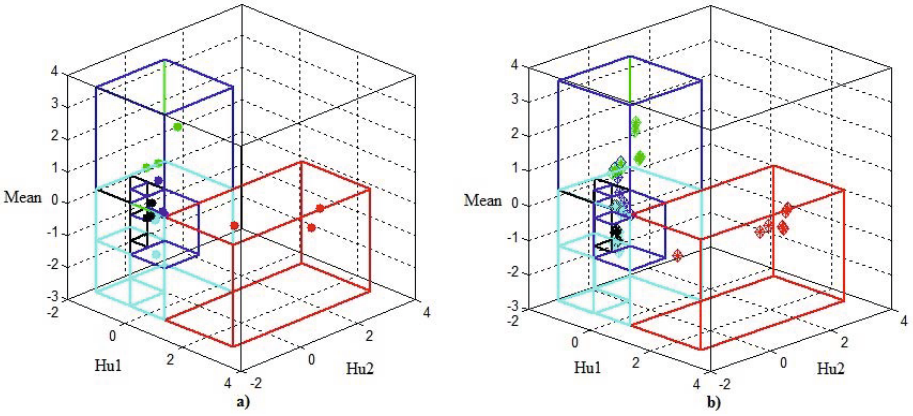

Fig. 7. a) Training result for 3 features, b) Classification results

The MLP and SVM were applied using the software Weka 3-6-8. For the MLP with one hidden layer, the training parameters were established as: learning rate=0.3 and momentum=0.2, the activation function used was the sigmoid. For the SVM, a Polynomial Kernel of degree 2 was used. Table 1 presents the results of the comparison, as it can be seen, the modified DMNN improves the results of the MLP and the SVM, so the performance of the proposed algorithm [3] is satisfactory for object recognition.

Table 1. Comparison table of the MLP, SVM and modified DMNN for the 5 objects dataset

MLP		**SVM**	**DMNN**		
# Neurons	% of Recognition	% of Recognition	# Dendrites	M	% of Recognition
9	80	85.71	10	0.2861	88.57

For the COIL-20 dataset, we used the images at 0, 45, 90, 135, 180, 225, 270 and 315 degrees for training on each of the 20 classes. Testing was done on the remaining images. Table 2 presents the recognition results for the COIL-20 dataset obtained by each one of the algorithms. As can be seen, also for this problem, the error obtained with the modified DMNN improves the results of the MLP and minimally to the SVM.

Table 2. Comparison table of the MLP, SVM and modified DMNN for the COIL-20 dataset

MLP		**SVM**	**DMNN**		
# Neurons	% of Recognition	% of Recognition	# Dendrites	M	% of Recognition
32	79.76	82.10	75	0.2876	82.27

To have an estimated error generalization, 10 experiments were realized with training and testing samples randomly selected for both datasets. Table 3 shows on the left, the average of the percentage of recognition for the 5 objects dataset, and on the right, the average for the COIL-20 dataset is presented. For the 5 objects dataset, 3 images were randomly selected for training and the rest were used for testing; and for the COIL-20 dataset, 8 images of each object were randomly selected (160 samples) for training and the others 1280 samples were used for testing. For the COIL-20 dataset, the results show that is convenient to include images of specific views of the object in the training dataset to improve the recognition percentage.

These results reveal that the performance of the modified DMNN for object recognition improves the results obtained with the MLP and SVM; however, it is necessary to find better features that describe to objects for improving these recognition rates.

Table 3. Average percentage of recognition of the MLP, SVM and modified DMNN for both datasets

	MLP	SVM	DMNN
5 objects dataset			
% of Recognition	80.57	83.42	86.85

	MLP	SVM	DMNN
COIL-20 dataset			
% of Recognition	75.77	79.16	79.36

6 Conclusions

In this paper a modified DMNN was applied for 3D object recognition using 2D images. By using 2D moments and characteristics of the pixel distribution, the proposed method does not require complex features calculation, thus reduces processing time in feature extraction stage. Comparisons of the modified DMNN with the MLP and SVM demonstrated the advantages of the DMNN. The characteristics of the proposed method allowed its implementation in a real application with good results.

Future work will be focused on the improvement of the results using features more adequate for 3D images as color and depth characteristics. Furthermore, it is necessary the implementation of DMNN training algorithm on a parallel architecture for evaluating the method in larger databases with more describing features.

Acknowledgments. H. Sossa would like to thank SIP-IPN and CONACYT under grants 20121311, 20131182 and 155014 for the economical support to carry out this research. E. Guevara thanks CONACYT for the scholarship granted to pursuit her doctoral studies.

References

1. Wöhler, C.: 3D Computer Vision: Efficient Methods and Applications. Springer (2012)
2. Grauman, K., Leibe, B.: Visual Object Recognition. Morgan & Claypool (2011)
3. Sossa, H., Guevara, E.: Efficient training for dendrite morphological neural networks. Submitted to Neurocomputing - Elsevier Journal
4. Nene, D., Nayar, S., Murase, H.: Columbia object image library: COIL (1996)
5. Hu, M.K.: Visual pattern recognition by moment invariants. IRE Transactions on Information Theory 8, 179–187 (1962)
6. González, R., Woods, R.: Digital Image Processing. Pearson (2007)
7. Ritter, G.X., Iancu, L., Urcid, G.: Morphological perceptrons with dendritic structure. In: 12th IEEE International Conference in Fuzzy Systems (FUZZ 2003), vol. 2, pp. 1296–1301 (2003)
8. Ritter, G.X., Urcid, G.: Lattice algebra approach to single-neuron computation. IEEE Transactions on Neural Networks 14(2), 282–295 (2003)
9. Davidson, J.L., Hummer, F.: Morphology neural networks: An introduction with applications. Circuits Systems Signal Process 12(2), 177–210 (1993)
10. Ritter, G.X., Sussner, P.: An introduction to morphological neural networks. In: Proceedings of the 13th International Conference on Pattern Recognition, vol. 4, pp. 709–717 (1996)

11. Sussner, P.: Morphological perceptron learning. In: IEEE ISIC/CIRA/ISAS Joint Conference, pp. 477–482 (1998)
12. Piñeiro Colón, R.C., Ortiz, J.L.: Evolutionary Training of Morphological Neural Networks. PhD thesis, Electrical and Computer Engineering Department. University of Puerto Rico, Mayagüez Campus
13. Lima, C.A.M., Coelho, A.L.V., Silva, M.E.S., Gudwin, R.R., Von Zuben, F.J.: Hybrid training of morphological neural networks: A comparative study. Technical report, DCA-FEEC-Unicamp
14. De Aráujo, R., Madeiro, F., De Sousa, R.P., Pessoa, L.F.C.: Modular morphological neural network training via adaptive genetic algorithm for designing translation invariant operators. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2006), vol. 2, pp. 873–876 (2006)
15. Raducanu, B., Graña, M., Sussner, P.: Morphological neural networks for vision based self-localization. In: IEEE International Conference on Robotics and Automation, pp. 2059–2064 (2001)
16. Nong, Y., Hao, W., Changyong, W., Fanming, L., Lide, W.: Morphological neural networks for automatic target detection by simulated annealing learning algorithm. Science in China (Series F) 46, 262–278 (2003)
17. Villaverde, I., Graña, M., D'Anjou, A.: Morphological neural networks for localization and mapping. In: IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA 2006), pp. 9–14 (2006)
18. Roberson, C., Dankel II, D.D.: A morphological neural network approach to information retrieval. In: Twentieth International Florida Artificial Intelligence Research Society Conference, pp. 184–185 (2007)
19. Cheng-Tian, S., Ke-Yong, W.: Image target detection using morphological neural network. In: International Conference on Computational Intelligence and Security, pp. 234–236 (2009)
20. Sussner, P., Esmi, E.L.: Morphological perceptrons with competitive learning: Lattice-theoretical framework and constructive learning algorithm. Information Sciences 181, 1929–1950 (2011)
21. Segev, I.: Dendritic processing. In: The Handbook of Brain Theory and Neural Networks, pp. 282–289 (1998)
22. Barrón, R., Sossa, H., Cortés, H.: Morphological neural networks with dendrite computation: A geometrical approach. In: Sanfeliu, A., Ruiz-Shulcloper, J. (eds.) CIARP 2003. LNCS, vol. 2905, pp. 588–595. Springer, Heidelberg (2003)
23. Valle, M.E.R., Pimenta, M.A.: Perceptrón morfológico de camada única. Technical report, Department of Mathematics of the State University of Londrina, Brazil (2005)
24. Ritter, G.X., Schmalz, M.S.: Learning in lattice neural networks that employ dendritic computing. In: IEEE International Conference on Fuzzy Systems, Vancouver, BC, Canada, pp. 7–13 (2006)
25. Ritter, G.X., Urcid, G.: Learning in lattice neural networks that employ dendritic computing. Computational Intelligence Based on Lattice Theory 67, 25–44 (2007)
26. Chyzhyk, D., Graña, M.: Optimal hyperbox shrinking in dendritic computing applied to Alzheimer's disease detection in MRI. In: Corchado, E., Snášel, V., Sedano, J., Hassanien, A.E., Calvo, J.L., Ślęzak, D. (eds.) SOCO 2011. AISC, vol. 87, pp. 543–550. Springer, Heidelberg (2011)
27. Graña, M.: Special issue on: Lattice computing and natural computing. Neurocomputing 72(10-12), 2065–2066 (2009)