Exploratory Data Analysis with the Growing Hierarchical Neural Forest

Esteban J. Palomo[a,*], Ezequiel López-Rubio[a], Francisco Ortega-Zamorano[a], Rafaela Benítez-Rochel[a]

[a]*Department of Computer Languages and Computer Science, University of Málaga. Bulevar Louis Pasteur, 35, 29010 Málaga, Spain*

Abstract

In this paper, a new self-organizing artificial neural network called Growing Hierarchical Neural Forest (GHNF) is proposed. The GHNF is a hierarchical model based on the Growing Neural Forest (GNF), which is a tree-based model that learns a set of trees (forest) instead of a general graph, so that the forest can grow in size. This way, the GHNF faces three important limitations regarding the self-organizing map (SOM): fixed size, fixed topology and lack of hierarchical representation for input data. Experimental results show the goodness of our proposal in terms of self-organization and clustering capabilities. In particular, it has been applied to text mining of tweets as a typical exploratory data analysis application, where a hierarchical representation of concepts present in tweets has been obtained. Moreover, it has been applied to foreground detection in video sequences, outperforming several methods specialized in foreground detection.

Keywords: self-organization, clustering, text mining, image segmentation

1. Introduction

Artificial neural networks have been extensively used for exploratory data analysis applications, such as clustering, data mining, document retrieval or image segmentation. Among these artificial neural networks, self-

*Corresponding author. Department of Computer Languages and Computer Science, E.T.S.I. Informática, University of Málaga, Bulevar Louis Pasteur, 35, 29071, Málaga, Spain. Tel.: +34 952 132 726; fax: +34 952 131 397.

Email address: ejpalomo@lcc.uma.es (Esteban J. Palomo)

organizing neural networks have been successfully applied to unsupervised learning tasks, where a cooperation mechanism among neurons is included besides the Hebbian competition mechanism. The best known self-organizing neural model is the Self-Organizing Map (SOM) (Kohonen, 1982), which maps high-dimensional data into a two-dimensional representation space, usually a rectangular or hexagonal grid. The projection preserves the topology of the data so that similar data items will be mapped to nearby locations on the map. However, some inherent properties of this neural model may be too restrictive in certain applications. First, the size of the map (number of neurons) has to be established in advance. Second, the fixed SOM topology (rectangular or hexagonal grid) can undermine its flexibility when adapting to input data with a complex topological estructure. Finally, the SOM is not designed to represent hierarchical relations that can be present in input data.

The Growing Neural Gas (GNG) (Fritzke, 1995) is a self-organizing model based on the Neural Gas (NG) (Martinetz & Schulten, 1991), which addresses two of the limitations of the SOM. One is related to the fixed topology of the SOM, since both the NG and GNG are based on a graph so that flexibility in the adaptation to the input data is improved. In addition to that, the GNG provides a neuron growth and removal mechanism. Thus, the problem related to the determination of the number of neurons is also solved.

The lack of representation of hierarchical relations when using SOM-based models have been addressed in different works by proposing hierarchical SOMs (Koikkalainen & Oja, 1990; Miikkulainen, 1990; Lampinen & Oja, 1992; Barbalho et al., 2001; Liu & Takatsuka, 2009; Bertolini & Ramat, 2007; Henriques et al., 2012), which have been applied to a wide variety of applications such as vector quantization, image compression, script recognition, image retrieval, object recognition, and geographic information sciences. One outstanding hierarchical model is the Growing Hierarchical Self-Organizing Map (GHSOM) (Rauber et al., 2002), which combines automatic map growing with hierarchical data representation by building a tree of growing SOMs (Alahakoon et al., 1998). However, this approach has still the limitation related to fixed topology since it uses a rectangular map. The Growing Hierarchical Neural Gas (GHNG) was recently proposed as a hierarchical extension of the GNG, which solves the three SOM-related limitations (Palomo & López-Rubio, 2016a).

Moreover, several tree-based self-organizing models have been proposed over the years to represent an input data distribution by means of a tree of

neurons (Astudillo & Oommen, 2014; Rahmel, 1996; Pakkanen et al., 2004; Samsonova et al., 2006). The purpose of these approaches is either to have a more flexible topology, suitable for complex data analysis problems or reduce the computational complexity of the SOM.

In this work, a new hierarchical self-organizing artificial neural network that we call Growing Hierarchical Neural Forest (GHNF) is presented. This model can be considered a hierarchical extension of the Growing Neural Forest (GNF) (Palomo & López-Rubio, 2016b), which is a self-organizing model based on the GNG that modifies its learning algorithm to create a set of trees (forest) instead of a general graph. Therefore, the proposed GHNF not only addresses the SOM limitations related to the fixed number of neurons, fixed topology and lack of hierarchical representation, but it also improves flexibility in the adaptation to input data by creating a hierarchy of forests (GNFs).

The structure of this paper is organized as follows. The proposed model, which we call Growing Hierarchical Neural Forest (GHNF) is presented in Section 2. Experimental results are given in Section 3. In Section 4, a discussion about some key aspects of our proposal is provided. Finally, Section 5 is devoted to conclusions.

2. The model

Hierarchical neural networks consist of multiple neural networks connected and organized as a non-cyclic graph. A tree-structured neural model is a special type of hierarchical network. The Growing Hierarchical Neural Forest (GHNF) model is given by a tree of self-organizing neural forests.

Each neural forest is a set of trees which contain several neurons and connections among them. A Growing Neural Forest is referred to a neural forest where both neurons and connections can be created or destroyed during the learning phase. Once the neural forest has learned, the spanning trees of its connected components are extracted from the graph associated to the model. These spanning trees form a neural forest. Some of the neurons have a child forest, which generates the tree of forests. In the end, the whole model is a networks of networks corresponding to a Growing Hierarchical Neural Forest. An example of a possible GHNF structure is depicted in Figure 2.1.

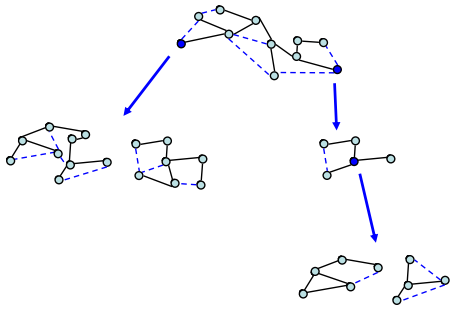

Figure 1: Structure of an example GHNF with four forests. The parent neurons are shown in a darker tone. The connections which are part of a spanning tree are shown as solid lines, and the others are shown as dashed lines.

Next we detail the definition of the GHNF. First we define a self-organizing neural forest (Subsection 2.1), see (Palomo & López-Rubio, 2016b) for more details. Then we specify how some of the neurons of a neural forest are spawned to form a GHNF model (Subsection 2.2).

2.1. Neural forest model

A self-organizing neural forest is a neural model with an adaptive architecture, that is, the learning algorithm inserts and removes neurons and connections as required to adapt to the input distribution. Thus, this model learns a forest, i.e. a set of trees neurons.

At a step time n, a forest contain a number of neurons H. The set of the connections among the neurons of the forest is a undirected graph $A \subseteq \{1, ..., H\} \times \{1, ..., H\}$ from which the spanning tree of each connected component can be extracted. The set of the connection among the neurons of the spanning trees is noted $\hat{A} \subseteq \{1, ..., H\} \times \{1, ..., H\}$ (solid lines in 1), where $\hat{A} \subseteq A$.

As a self-organizing map, each neuron on it will specialize to represent clusters of input data and will content an estimation of the centroid of the cluster $\mathbf{w}_i \in \mathbb{R}^D$

The learning algorithm for the self-organizing forest is as follows:

1. Initialization: Create two units ($H = 2$) joined by a connection.
   In this step:
   (a) Each centroid is initialized to an input sample drawn uniformly at random from the training set $\mathcal{S}$.

(b) The age of the connection is set to zero.
(c) Each neuron will have an associated error $e_i(n+1) = e_i(n) + \|\mathbf{w}_i(n) - \mathbf{x}_n\|^2$, para $n \geq 1$. Clearly, $e_i(0) = 0$, $i = 1, 2$.
(d) The following parameters must be selected: H_{max}, a maximum number of neurons; a_{max}, limit in the age of a non selected edge in the learning process and λ, frequency with a new neuron can be inserted in the network.

2. Draw a training sample $\mathbf{x}_n \in \mathbb{R}^D$ uniformly at random from $\mathcal{S}$.
3. Find the nearest neuron q and the second nearest neuron s in terms of the Euclidean distance from the sample to the centroids:

$$q = \arg \min_{i \in \{1,\ldots,H\}} \|\boldsymbol{w}_i(n) - \mathbf{x}_n\| \tag{1}$$

$$s = \arg \min_{i \in \{1,\ldots,H\} - \{q\}} \|\boldsymbol{w}_i(n) - \mathbf{x}_n\| \tag{2}$$

4. Adaptation
   (a) Increment the age of all connections in A which involve q.
   (b) Recalculate error estimation of neuron q:

$$e_q(n+1) = e_q(n) + \|\boldsymbol{w}_q(n) - \mathbf{x}_n\|^2 \tag{3}$$

   (c) Update q and all its directly connected neighbors in its spanning tree ($\hat{A}$) with learning rate η_b for neuron q and η_v for the neighbors, where $\eta_b > \eta_v$:

$$\eta(n,i) = \begin{cases} \eta_b & \text{iff } i = q \\ \eta_v & \text{iff } (i \neq q) \wedge (i,q) \in \hat{A} \\ 0 & \text{iff } (i \neq q) \wedge (i,q) \notin \hat{A} \end{cases} \tag{4}$$

$$\boldsymbol{w}_i(n+1) = (1 - \eta(n,i))\,\boldsymbol{w}_i(n) + \eta(n,i)\,\mathbf{x}_n \tag{5}$$

5. Growth/Removal mechanism
   (a) If q and s are connected in A, then reset the age of this edge to zero. Otherwise, create the connection.
   (b) Remove all connections in A whose age is higher than a_{max}. Then delete all isolated neurons.

(c) If the time step n is an integer multiple of the parameter λ and the number of neurons H is smaller than the maximum number of units H_{max}, then insert a new neuron as follows. First find the neuron r with the maximum estimated error and the neuron z with the highest estimated error among all directly connected neighbors of r in A. Then insert a new neuron k, create connections linking k with r and z, and delete the original connection between r and z. After that, decrease the error estimations e_r and e_z by multiplying them with a constant α, and setup the error estimation e_k to the new value of e_r. Set the centroid of k to be halfway between those of r and z, as follows:

$$\boldsymbol{w}_k(n) = \frac{1}{2}\left(\boldsymbol{w}_r(n) + \boldsymbol{w}_z(n)\right) \tag{6}$$

Finally, compute the spanning trees by Kruskal's algorithm (Kruskal, 1956) and assign $\hat{A}$ to the set of connections which belong to spanning trees.

6. Growth control
   It consists of a test which stablishes how big the reduction of the mean quantization error must be before accepting the insertion of a new neuron. Let MQE_{old} and $MQE_{new}$the mean quantization error of the backup and the current version of the network, respectively.
   Given a parameter $\tau \in [0, 1]$, the test indicates that the network has converged if the following condition is fulfilled.

$$\frac{MQE_{old} - MQE_{new}}{MQE_{old}} < \tau$$

   In that case, the new version of the graph is not accepted. Otherwise, the graph growth has resulted in a new model that improves the quantization error.
   Note that τ is a parameter that controls the growth process. The higher is τ the smaller the error in the new neural graph must be in order to allow growth.
7. Decrease all error estimations e_i by multiplying them by an error reduction rate, $\delta \in [0, 1]$.
8. If the maximum limit of time steps has been reached, then halt. Otherwise, go to step 2.

Next we specify how the trained forests are combined to form a global hierarchy.

2.2. Hierarchical model

The GHNF is a hierarchical tree of self-organizing neural forests, as stated above. Next we specify how to build such hierarchy of forests. It starts at a root node that corresponds to a neural forest which has been trained previously with the whole set of training samples, as specified in Subsection 2.1. Then each neuron i in the neural network at the root node creates a neural forest however, in this step, the training set will be:

$$\mathcal{S}_i = \left\{ \mathbf{x} \in \mathcal{S} \mid i = \arg \min_{j \in \{1,\dots,H\}} \|\boldsymbol{w}_j - \mathbf{x}\| \right\} \tag{7}$$

It can be noticed that S_i is the region in $\mathcal{S}$ which makes neuron i be the winner according to the expression (1). In this way, we have obtained the child forests. Then, the training process is applied recursively to the neurons in each child node. The branching pattern is controlled by pruning the neural forest with less than three neurons, on the other hand, the level of the hierarchical tree with a pre-specified parameter d, i.e., no forests are created below the depth limit d_{max}.

Next we specify the algorithm to train a GHNF. Given the training set $\mathcal{S}$; the growth control parameter τ and the rest of the GNF parameters λ, η_b, η_v, H_{max}, a_{max}, d we denote the current depth as d; and the maximum depth d_{max}, the steps of the recursive algorithm to train a GHNF on $\mathcal{S}$ are as follows:

1. If $d > d_{max}$ then return an empty GHNF.
2. Train a GNF on training set $\mathcal{S}$ as specified in Subsection 2.1.
3. If $H < 3$ then return an empty GHNF.
4. Invoke recursively this algorithm with the receptive field of unit i as the training set $\mathcal{S}_i$ (Eq. 7) and $d+1$. Then insert all the resulting non empty GHNFs as children of their respective parent units, and return the overall GHNF.

This completes the description of our proposed self-organizing model. The next section reports the results of the experiments that have been performed in order to evaluate our proposal.

3. Experimental Results

Three different kind of experiments have been carried out in order to demonstrate the possibilities of our proposal. First, a set of self-organization experiments has been designed to show the correct unfolding of the GHNF model (Subsection 3.1). Second, a text mining experiment containing an analysis of tweets from American politicians has been proposed as a typical unsupervised learning task, where the hierarchy of the GHNF model can be leveraged (Subsection 3.2). Finally, foreground detection experiments in several video sequences have been performed as a computer vision application (Subsection 3.3).

3.1. Self-organization experiments

This set of experiments is designed to check the self-organization capabilities of our proposal. To this end, several input distributions of different dimensionality D have been taken into account. Concretely, we have selected two-dimensional input distributions ($D = 2$) (a ring and an 'S' letter); two three-dimensional input distributions ($D = 3$) (three clusters and a unit length segment); and hyperspheres in four ($D = 4$), five ($D = 5$), and six ($D = 6$) dimensions.

For each input distribution, a dataset of $M = 10,000$ input samples was extracted. The GHNF training was done for each dataset during $N = 20,000$ time steps (2 epochs). In order to see the effect of the τ parameter in the size of the architecture, three different values for this parameter have been chosen for training (0.2, 0.1, and 0). The rest of parameters has been set to the recommended values in the original GNG paper (Fritzke, 1995). These settings for the GHNF model are given in Table 1. The resulting GHNF hierarchical architectures can be seen in Figure 2. In these plots, input samples are represented by yellow points, neurons are represented by circles and links among neurons are represented by straight lines, where neurons and link colors are darker if they belong to an upper layer. Note that in these plots only 2 layers of the hierarchy can be seen. For hyperspheres, only the first three dimensions have been plotted.

A quantitative evaluation of the GHNF has been also carried out by using three performance measures. The first one is the Mean Squared Error (MSE) (lower is better):

Parameter description	Values
Step size for the winning unit	$\eta_b = 0.2$
Step size for the neighbor unit	$\eta_v = 0.006$
Maximum edge for an edge	$a_{max} = 50$
New units insertion	$\lambda = 100$
Maximum number of units	$H_{max} = 50$
Reduction of the error variables	$\alpha = 0.5$
Error decay rate	$\delta = 0.995$
Growth control parameter	$\tau = \{0.2, 0.1, 0\}$

Table 1: Parameter settings for the GHNF model.

$$MSE_j = \frac{1}{M} \sum_{i=1}^{M} \|\mathbf{w}_j - \mathbf{x}_i\|^2, \quad j = 1, \cdots, K \tag{8}$$

where M is the number of samples in the distribution, K is the number of clusters, $\mathbf{x}_i$ is the i-th input sample and $\mathbf{w}_j$ the prototype of the winning neuron corresponding to $\mathbf{x}_i$.

The second one is the Davies-Bouldin index (DB). It is defined as follows (lower is better):

$$DB = \frac{1}{K} \sum_{i=1}^{K} max_{j \neq i}\{R_{ij}\} \tag{9}$$

where R_{ij} is the within-to-between cluster distance ratio for the jth and kth clusters:

$$R_{jk} = \frac{(MSE_j + MSE_k)}{|\mathbf{w}_j - \mathbf{w}_k|} \tag{10}$$

it takes into account both intra cluster distance and inter cluster distance; MSE_j is the average distance between each point in the ith cluster and the centroid of the jth cluster, $\mathbf{w}_j$ (intra cluster distance) and the distance between the centroids $\mathbf{w}_j$ and $\mathbf{w}_k$ is the inter cluster distance.

Finally, our last considered performance measure is the average silhouette coefficient (S) over all data (Rousseeuw, 1987). The silhouette coefficient is a measure of how how well a point is assigned to a cluster. The silhouette of the i-th input sample (s_i) is defined as follows (higher is better):

$$s_i = \left(\frac{b_i - a_i}{max\{a_i, b_i\}} \right) \tag{11}$$

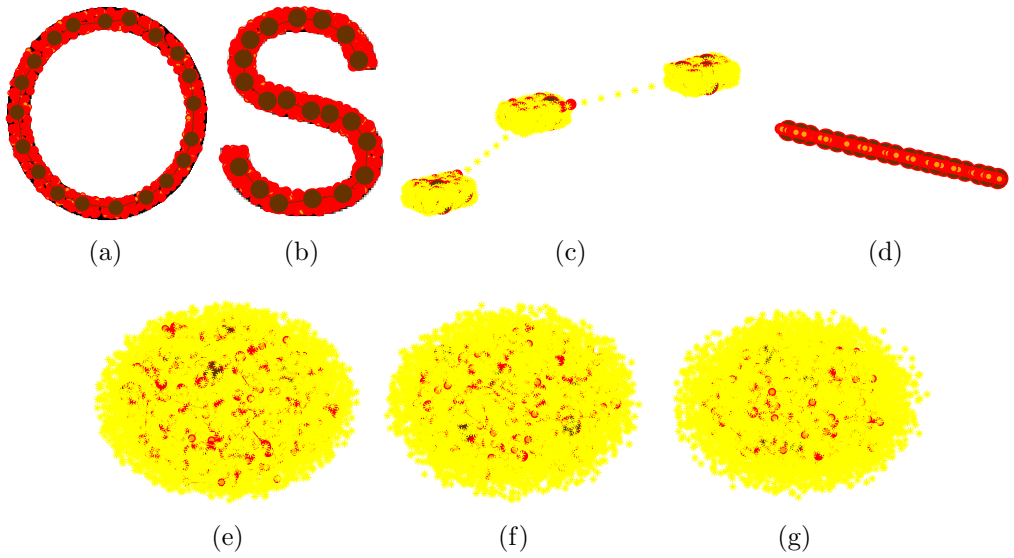

Figure 2: GHNF architectures for $\tau = 0$ and the considered input distributions: (a) ring, (b) 'S' letter, (c) 3D clusters, (d) unit segment, (e) 4D hyperspehere, (f) 5D hypersphere, and (g) 6D hypersphere (for hyperspheres only the first three dimensions have been plotted).

where a_i is the average dissimilarity of the i-th input sample with the rest of samples within the same cluster and b_i is the lowest dissimilarity of the i-th input sample to any other cluster of which is not a member. Note that s_i lies in the range of $[-1, 1]$ where the closer to 1 the better the sample is clustered. These last two performance measures (DB and S) have been widely used for the performance evaluation of different clustering experiments (Ashok et al., 2013; Kim & Billard, 2011; de Amorim & Hennig, 2015; Granados et al., 2015).

The results obtained from GHNF have been compared with those achieved by (Growing Hierarchical Neural Gas) (Palomo & López-Rubio, 2016a), which is another growing hierarchical neural network that is based on the GNG (Growing Neural Gas) (Fritzke, 1995). For purposes of comparison, both the number of generated neurons and the CPU time that training takes have also been included. Trainings have been repeated 10 times to obtain the mean and standard deviations of the considered performance measures. The same values have been chosen for the parameters that both models have in common (see Table 1). These results are given in Table 2-4 for $\tau = 0.2$, $\tau = 0.1$,

Dataset	Model	MSE	DB	S	Neurons	CPU Time
Ring	GHNF	**1.89e-04 (2.57e-05)**	**1.50 (.19)**	.37 (.02)	367.10 (14.38)	4.95 (3.35)
	GHNG	1.96e-04 (2.77e-05)	1.51 (.18)	**.38 (.02)**	355.20 (19.62)	**3.18 (.10)**
S letter	GHNF	**1.24e-04 (6.24e-06)**	**1.35 (.13)**	.39 (.02)	372.00 (13.60)	**3.19 (.05)**
	GHNG	1.29e-04 (1.39e-05)	1.37 (.22)	**.39 (.01)**	356.20 (11.61)	3.24 (.06)
3D Clusters	GHNF	**4.95e-03 (8.87e-04)**	**2.82 (3.19)**	**.24 (.01)**	361.00 (41.98)	**3.05 (.16)**
	GHNG	2.73e-02 (1.28e-03)	3.33 (2.79)	.21 (.01)	363.40 (18.86)	3.20 (.05)
UnitSegment	GHNF	**1.05e-06 (9.12e-08)**	**2.62 (1.36)**	**.61 (.01)**	379.00 (8.64)	**3.23 (.07)**
	GHNG	1.62e-06 (1.97e-06)	11.31 (28.48)	**.61 (.01)**	381.00 (12.16)	3.26 (.10)
Hypersphere 4D	GHNF	**5.46e-02 (2.04e-03)**	**2.29 (.06)**	**.13 (.01)**	381.90 (33.83)	**3.10 (.11)**
	GHNG	5.67e-02 (1.66e-03)	3.11 (1.77)	.12 (.01)	367.90 (37.04)	3.15 (.08)
Hypersphere 5D	GHNF	**9.89e-02 (1.78e-03)**	**2.40 (.06)**	**8.58e-02 (6.85e-03)**	406.10 (19.59)	3.17 (.07)
	GHNG	.11 (.00)	2.47 (.06)	6.87e-02 (1.27e-02)	389.50 (44.13)	**3.09 (.15)**
Hypersphere 6D	GHNF	**.15 (.00)**	**3.52 (1.66)**	**5.43e-02 (9.67e-03)**	414.80 (8.90)	3.16 (.02)
	GHNG	.16 (.00)	4.34 (1.81)	1.38e-02 (6.31e-03)	420.80 (23.62)	**3.15 (.03)**

Table 2: MSE, DB, CS, Neurons and CPU time (in seconds) for self-organization experiments of the GHNF and GHNG models with $\tau = 0.2$. Best results are in bold. Standard deviations are in parenthesis.

and $\tau = 0$, respectively. Note that the lower the τ value, the higher the obtained number of neurons is. Also, the mean number of neurons for the GHNF and GHNG is similar for the same values of τ, although it cannot be the same due to the randomness when drawing an input sample. Generally speaking, the MSE is reduced when increasing the number of neurons, but we can see in these tables how the GHNF can achieve lower MSE values with lower number of neurons compared to the GHNG. Furthermore, DB values for the GHNF are better than those for the GHNG, whereas S values are slightly better for the GHNF. The training CPU time is approximately the same for both models.

3.2. Text mining experiments

In this set of experiments, the GHNF has been applied to text mining of tweets, which represents a typical exploratory data analysis task. These tweets were extracted from four American politicians during 2016, namely, Barack Obama, Donald Trump, Hillary Clinton, and Bernie Sanders. Thus, by using the GHNF the most relevant concepts from these tweets together with their hierarchical structure will be extracted. The number of obtained tweets were $M = 710$, $M = 641$, $M = 673$, and $M = 781$, for Obama, Trump, Clinton, and Sanders, respectively. We have built a common dictionary for the four tweet data sets composed of 994 words ($D = 994$) after removing stop words. A $tf \times idf$ weighting scheme has been used, i.e, term frequency times inverse document frequency (Salton & Buckley, 1988; Aizawa, 2003). This weighting scheme assigns high values to terms that appear frequently within

Dataset	Model	MSE	DB	S	Neurons	CPU Time
Ring	GHNF	1.46e-04 (8.17e-06)	**1.42 (.12)**	**.39 (.01)**	440.20 (13.18)	**3.11 (.03)**
	GHNG	**1.40e-04 (3.99e-06)**	1.45 (.15)	**.39 (.01)**	452.50 (10.01)	3.13 (.04)
S letter	GHNF	**9.96e-05 (1.87e-06)**	**1.33 (.06)**	.38 (.01)	451.50 (6.06)	**3.12 (.02)**
	GHNG	9.98e-05 (1.73e-06)	1.34 (.07)	**.38 (.00)**	460.00 (7.48)	3.20 (.06)
3D Clusters	GHNF	**4.33e-03 (1.81e-04)**	**5.73 (6.57)**	**.23 (.01)**	411.30 (15.28)	**3.16 (.07)**
	GHNG	2.34e-02 (1.82e-03)	7.23 (7.86)	.22 (.01)	425.50 (24.87)	3.21 (.11)
UnitSegment	GHNF	7.61e-07 (5.88e-08)	**12.94 (25.79)**	**.62 (.01)**	425.40 (5.68)	3.21 (.10)
	GHNG	**7.51e-07 (4.15e-08)**	40.03 (81.14)	**.62 (.01)**	427.30 (5.29)	**3.12 (.03)**
Hypersphere 4D	GHNF	4.47e-02 (1.48e-03)	2.03 (.06)	.16 (.01)	445.20 (13.64)	3.26 (.06)
	GHNG	**4.40e-02 (1.22e-03)**	**1.99 (.04)**	**.16 (.00)**	454.10 (15.41)	**3.26 (.05)**
Hypersphere 5D	GHNF	**8.64e-02 (2.01e-03)**	**2.14 (.03)**	**.12 (.00)**	431.10 (19.28)	3.52 (.45)
	GHNG	8.90e-02 (1.97e-03)	2.56 (1.16)	.11 (.01)	433.90 (15.74)	**3.30 (.05)**
Hypersphere 6D	GHNF	**.13 (.00)**	**2.21 (.01)**	**9.78e-02 (3.67e-03)**	431.50 (14.41)	**3.29 (.11)**
	GHNG	.14 (.00)	2.33 (.04)	7.90e-02 (6.05e-03)	446.70 (11.11)	3.32 (.15)

Table 3: MSE, DB, CS, Neurons and CPU time (in seconds) for self-organization experiments of the GHNF and GHNG models with $\tau = 0.1$. Best results are in bold. Standard deviations are in parenthesis.

Dataset	Model	MSE	DB	S	Neurons	CPU Time
Ring	GHNF	1.44e-04 (5.39e-06)	**1.36 (.05)**	**.39 (.01)**	441.80 (9.81)	**3.11 (.07)**
	GHNG	**1.41e-04 (6.60e-06)**	1.43 (.12)	.38 (.01)	457.80 (10.94)	3.15 (.10)
S letter	GHNF	1.02e-04 (3.89e-06)	1.45 (.08)	**.38 (.01)**	448.30 (5.91)	**3.06 (.02)**
	GHNG	**9.85e-05 (2.40e-06)**	**1.34 (.05)**	**.38 (.01)**	462.60 (9.01)	3.11 (.04)
3D Clusters	GHNF	**3.74e-03 (1.30e-04)**	**4.79 (6.34)**	**.25 (.01)**	448.50 (6.87)	**3.14 (.04)**
	GHNG	2.10e-02 (1.18e-03)	12.40 (8.08)	**.25 (.01)**	458.80 (13.97)	3.15 (.10)
UnitSegment	GHNF	**7.43e-07 (5.68e-08)**	**4.28 (6.52)**	.62 (.01)	426.00 (5.23)	**3.06 (.04)**
	GHNG	7.46e-07 (2.49e-08)	39.01 (94.84)	**.62 (.00)**	432.70 (4.37)	3.10 (.03)
Hypersphere 4D	GHNF	4.31e-02 (6.63e-04)	**1.96 (.05)**	**.17 (.01)**	453.20 (4.94)	3.17 (.04)
	GHNG	**4.25e-02 (8.31e-04)**	1.98 (.05)	.16 (.01)	470.10 (8.80)	**3.17 (.03)**
Hypersphere 5D	GHNF	**8.23e-02 (9.75e-04)**	**2.10 (.02)**	**.13 (.00)**	460.70 (6.38)	3.34 (.26)
	GHNG	8.36e-02 (6.99e-04)	2.13 (.03)	.12 (.00)	468.40 (8.37)	**3.20 (.03)**
Hypersphere 6D	GHNF	**.13 (.00)**	**2.19 (.04)**	**.10 (.00)**	460.70 (6.75)	3.24 (.05)
	GHNG	**.13 (.00)**	2.33 (.14)	8.20e-02 (6.37e-03)	466.30 (8.67)	**3.18 (.02)**

Table 4: MSE, DB, CS, Neurons and CPU time (in seconds) for self-organization experiments of the GHNF and GHNG models with $\tau = 0$. Best results are in bold. Standard deviations are in parenthesis.

one tweet, but not within the overall tweet collection. This way, terms that are considered important in describing the contents of a tweet are assigned high values. Data normalization in the range [0,1] has been carried out, where 0 means that a particular term is unimportant to describe the tweet. At the end of the feature extraction process, a Latent Semantic Indexing (LSI) technique has been applied in order to reduce data dimension. This technique is based on Singular Value Decomposition (SVD) and analyzes relationships between a set of tweets and the terms they contain by mapping terms into a set of concepts related to the tweets and terms (Deerwester et al., 1990). Therefore, high-dimensional data is mapped into a lower representation space by yielding 34 concepts ($D = 34$).

The GHNF training was done for each politician data set during 2 epochs. Layer 1 of the resulting GHNF architectures are given in Figure 3. In these plots, each node represents the most frequent concept mapped into that neuron. In order to represent a concept, the four most important terms associated to that concept are shown. We have not chosen more than four terms to avoid cluttered plots. The bigger the node and font size the higher the frequency of the represented concept is. Nodes and edges belonging to layers 1, 2, and 3 are colored in blue, green and red, respectively. Yellow nodes are those that were expanded into a new child forest at next layer. Since the four politicians share 34 concepts, the same concepts can appear in GHNFs belonging to different politicians, however relations among concepts and their occurrences are different, which define the type of tweets written by each politician.

Note that by reading the four terms associated to a node, an idea about the meaning of the represented concept can be obtained, e.g., terms 'violence', 'must', 'gun', and 'leave' can be interpreted as a concept expressing rejection of gun violence; or 'congress', 'add', 'name', and 'actonclimate' can express the conducted efforts to add members of congress in favor of the cause of fight against climate change (in fact 'actonclimate' is a twitter hashtag).

There are other concepts that can be further interpreted by means of the hierarchy. For instance, in the case of Obama the concept represented by the terms 'add', 'name', 'states', and 'united' (see Figure 3a) is represented in a higher level of detail at level 2 since this node (in yellow) created a child forest, which is given in Figure 4a. This new graph is composed of just 3 nodes representing the parent concept and a new concept by using the terms 'doyourjob', 'senate', 'garland', and 'hearing'. The term 'doyourjob' is a twitter hashtag expressing the campaign against Senate Republicans,

whereas the other terms mean the blockade of Merrick Garland, who did not receive a hearing in the Senate Judiciary Committee after his nomination by Barack Obama to serve as an Associate Justice of the Supreme Court, so that this concept represents the criticisms of Barack Obama against Senate Republicans that refused to consider Garland's nomination. The expansion of this concept at level 3 (see Figure 4b) confirms the meaning of this concept by creating three new concepts: 'doyourjob', 'senate', 'garland', and 'hearing'; 'doyourjob', 'american', 'working', and 'must'; and 'help', 'leave', 'scotus', and 'system', where 'scotus' refers to the Supreme Court of the United States.

3.3. Foreground detection experiments

The last set of experiments is devoted to carry out the detection of foreground objects in video sequences for automated surveillance. Self-organizing neural networks have been usually applied to different computer vision tasks such as foreground detection (Bhandarkar et al., 1997; Maddalena & Petrosino, 2008; Gemignani & Rozza, 2016). For checking the performance of the GNF model in this problem see the experiments shown in (Palomo & López-Rubio, 2016b). Here, six color spaces have been taken into account, namely RGB, Lab, Luv, HSV, HSL, and YCbC and we have included six videos in our experiments ('Lobby', 'LevelCrossing', 'Video2', 'Meeting room', 'Water surface', and 'OneShopOneWait1cor'). They are six widely-used video sequences chosen from public repositories, which represent real situations from both indoor and outdoor scenarios. These video sequences are provided with the ground truth information, which consists of a set of manually segmented images. The list of selected video sequences is given in Table 5. They come from VSSN 2006[1], the Institute for Infocomm Research[2], the CAVIAR dataset[3], and the Yaser Sheikh's website[4].

The GHNF model has been trained with several frames from a video sequence in an offline phase in order to learn a background model of the scene. Then the trained model is tested with new frames in an online phase, where foreground objects appear. The training dataset is created by computing the median of 100 frames for each video sequence. Note that by computing

[1] `http://mmc36.informatik.uni-augsburg.de/VSSN06_OSAC/`

[2] `http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html`

[3] `http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`

[4] `http://www.cs.cmu.edu/~yaser/`

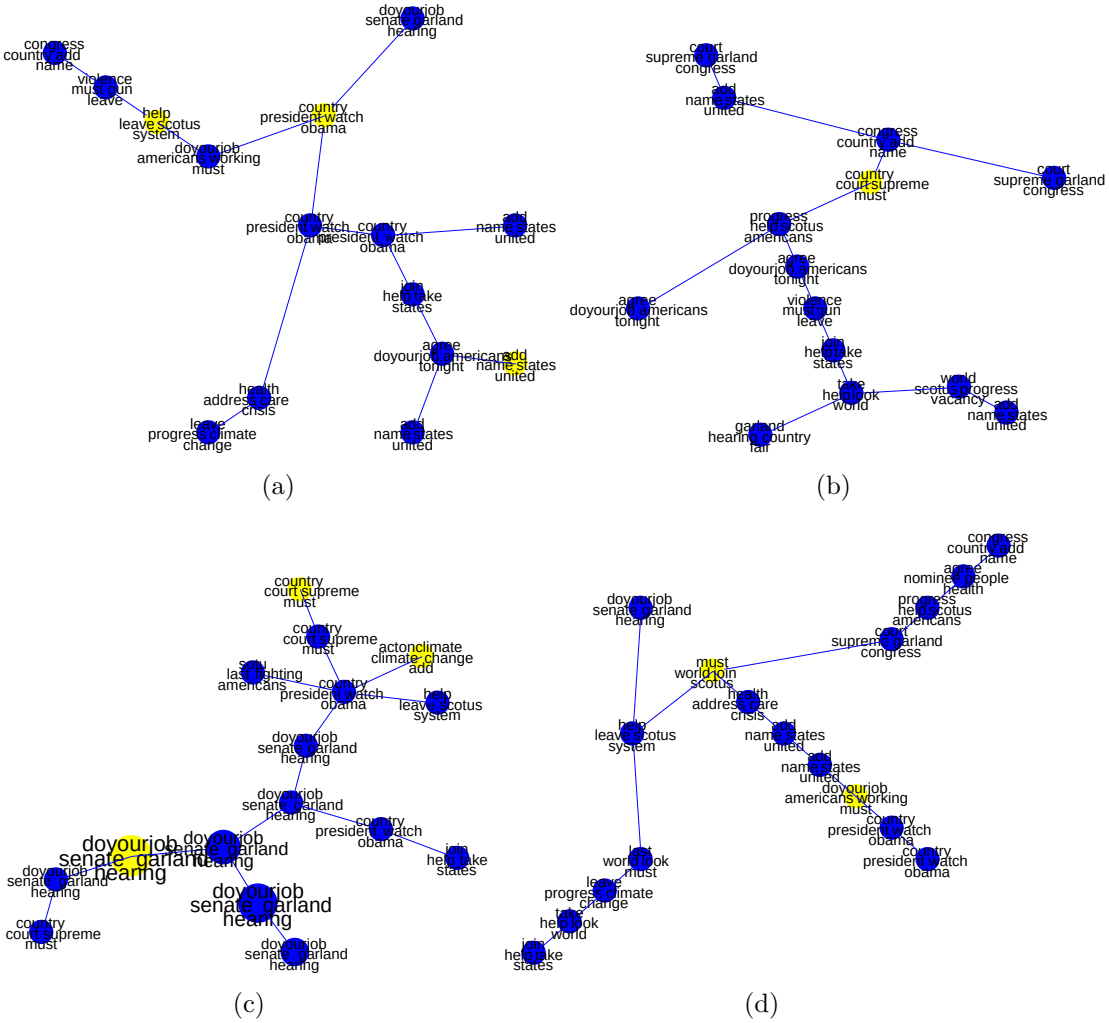

Figure 3: Layer 1 of the resulting GHNF architectures for each considered American politician tweet dataset: (a) Barack Obama, (b) Donald Trump, (c) Hillary Clinton, and (d) Bernie Sanders. Each node represents a concept, which in turn is represented by four terms associated to that concept. Yellow nodes are expanded into a new self-organizing neural forest.

Figure 4: Layer 2 and 3 resulting GHNF architectures from the expansion of the concept represented by the terms 'add', 'name', 'states', and 'united' from layer 1 of the GHNF for Barack Obama tweets: (a) child forest at layer 2, and (b) child forest at layer 3 expanded from the concept represented by the terms 'doyourjob', 'senate', 'garland', and 'hearing' of (a). Each node represents a concept, which in turn is represented by four terms associated to that concept. Yellow nodes are expanded into a new self-organizing neural forest.

Table 5: Video sequences used in the experiments.

Sequence name	Source	Description
Video2	VSSN 2006	3D objects are artificially inserted to detect the foreground easily
Meeting room Water surface Lobby	Institute for Infocomm Research	Camouflage, cast shadows and illumination changes
OneShopOneWait1cor	CAVIAR dataset	Busy corridor
LevelCrossing	Sheikh & Shah (2005)	Presence of nominal camera motion and dynamic textures

the median of several frames, the median pixel values approximate the background of the scene. Then the three color components and the position (i, j) of each pixel are extracted from the median video frame, thus obtaining a five-dimensional dataset ($D = 5$) with M training samples, where M is the number of pixels of the video frame. According to the resolution of each video sequence, the number of training input samples were $M = 92,160$, $M = 20,480$, $M = 20,480$, $M = 20,480$, $M = 110,592$, and $M = 86,400$ for the Video2, Meeting room, Water surface, 'Lobby', OneShopOneWait1cor, and 'LevelCrossing' video sequences, respectively.

Once the GHNF is trained, the foreground detection is performed by comparing the error committed in the training and test phases. Let $\mathbf{x}_k$ be a pixel in a frame and γ_k be the difference between the training phase error and the test phase error, as follows:

$$\gamma_k = ||\mathbf{w}_q^{training} - \mathbf{x}_k|| - ||\mathbf{w}_q^{test} - \mathbf{x}_k|| \tag{12}$$

The pixel $\mathbf{x}_k$ is declared to belong to the foreground of the scene if the absolute value of the difference γ_k is higher than a threshold, as expressed in Eq. (13):

$$\text{PredictedClass}(\mathbf{x}_k) = \begin{cases} \text{foreground} & \text{iff } |\gamma_k| > T_k \\ \text{background} & \text{otherwise} \end{cases} \tag{13}$$

where $T_k = \bar{\gamma_k} + \sigma_k$, that is, the mean of $|\gamma_k|$ plus its standard deviation. As the threshold is unique for each pixel the classifier gets robustness against noise. Foreground detection results achieved by the GHNF model using different color spaces are shown in Figures 5 and 6.

In order to present the results of the previous experiment, six measures that quantify the performance of a model will be considered. Namely false positives (FP), false negatives (FN), precision (PR), recall (RC), F-measure, and accuracy (AC), which are explained next. First the set of pixels corresponding to the foreground A and the set of pixels classified as foreground by the GHNF model B are defined:

$$A = \{\mathbf{x}_k \mid \chi(\mathbf{x}_k) = 1\}, \qquad B = \{\mathbf{x}_k \mid \widetilde{\chi}(\mathbf{x}_k) = 1\} \tag{14}$$

where $\chi(\mathbf{x}_k)$ and $\widetilde{\chi}(\mathbf{x}_k)$ are defined as:

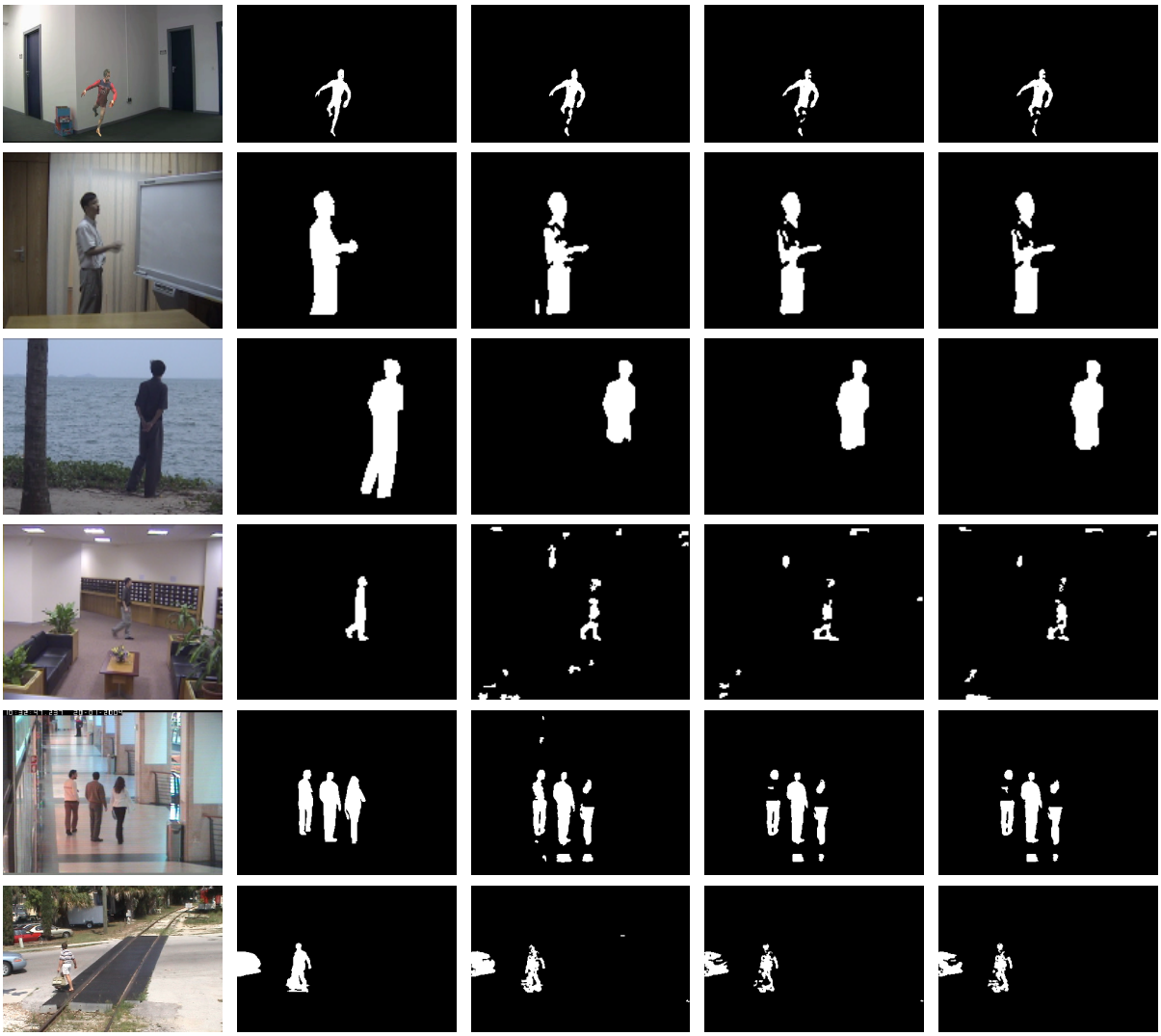

Figure 5: Foreground detection results of the GHNF model for RGB, Lab and Luv color spaces. Rows from top to bottom: Video2 (frame 526), Meeting Room (frame 23242), WaterSurface (frame 1577), Lobby (frame 2265), OneShopOneWait1cor (frame 375), and LevelCrossing (frame 386). The first two columns show the original frame and Ground Truth, respectively. The last three columns show the corresponding tested color spaces.

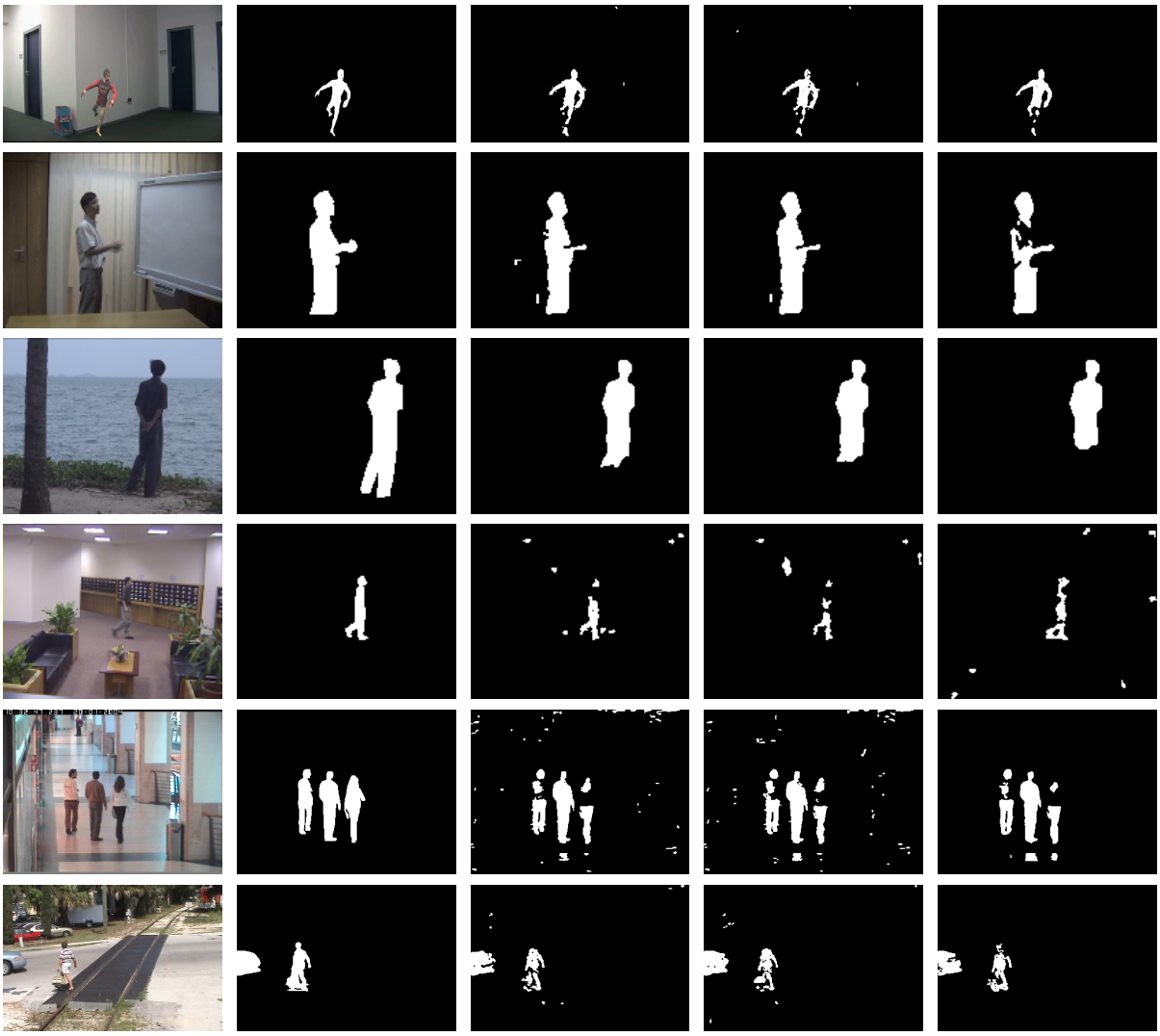

Figure 6: Foreground detection results of the GHNF model for HSV, HSL and YCbCr color spaces. Rows from top to bottom: Video2 (frame 526), Meeting Room (frame 23242), WaterSurface (frame 1577), Lobby (frame 2265), OneShopOneWait1cor (frame 375), and LevelCrossing (frame 386). The first two columns show the original frame and Ground Truth, respectively. The last three columns show the corresponding tested color spaces.

$$\chi(\mathbf{x}_k) = \begin{cases} 1 & \text{iff RealClass}(\mathbf{x}_k) \in \text{foreground} \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

$$\widetilde{\chi}(\mathbf{x}_k) = \begin{cases} 1 & \text{iff PredictedClass}(\mathbf{x}_k) \in \text{foreground} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

where RealClass($\mathbf{x}_k$) and PredictedClass($\mathbf{x}_k$) stand for the real and predicted class labels of pixel $\mathbf{x}_k$, respectively.

If A^c and B^c are the complements of A and B, respectively, the confusion matrix is

		PREDICTION	
		A	A^c
ACTUAL	B	TP	FN
	B^c	FP	TN

where FN stands for False Negatives and FP for False Positives (lower is better in both) otherwise TP means True Positives and TN, True Negatives (higher is better).

Precision (PR), recall (RC) (higher is better) are defined as follows:

$$PR = \frac{TP}{TP + FP}, \quad PR \in [0, 1] \tag{17}$$

$$RC = \frac{TP}{TP + FN}, \quad RC \in [0, 1] \tag{18}$$

and F-measure, which combines PR and RC so higher s better is given by:

$$FM = \frac{2 \cdot PR \cdot RC}{PR + RC} \tag{19}$$

Our last measure is accuracy (AC), which is defined as follows (higher is better):

$$AC = \frac{TP + TN}{FP + FN + TP + TN} \tag{20}$$

The GHNF results for the six performance measures and the CPU time are given in Tables 6-12. The CPU time is the mean of foreground detection running time in the online phase. Best results for each video sequence are in bold. Note that the best color spaces are different for each video sequence.

Table 6: Accuracy obtained by the GHNF model for each video sequence and color space. Best results are in bold. Standard deviations are in parentheses.

Video2	0.75 (0.10)	0.79 (0.09)	0.77 (0.10)	**0.82 (0.07)**	0.80 (0.07)	0.76 (0.10)
Meeting Room	0.75 (0.08)	0.71 (0.08)	0.71 (0.08)	**0.77 (0.08)**	0.75 (0.10)	0.72 (0.08)
Water Surface	0.58 (0.03)	0.62 (0.04)	0.62 (0.05)	**0.75 (0.04)**	0.70 (0.04)	0.61 (0.04)
Lobby	0.44 (0.20)	0.42 (0.20)	0.42 (0.20)	0.43 (0.18)	0.36 (0.16)	**0.53 (0.24)**
OneShopOneWait1cor	**0.67 (0.02)**	0.62 (0.02)	0.62 (0.02)	0.61 (0.02)	0.59 (0.03)	0.64 (0.02)
LevelCrossing	**0.77 (0.09)**	0.62 (0.15)	0.62 (0.16)	0.67 (0.07)	0.62 (0.10)	0.73 (0.10)

Table 7: F-measure obtained by the GHNF model for each video sequence and color space. Best results are in bold. Standard deviations are in parentheses.

	RGB	Lab	HSV	HSL	Luv	YCbCr
Video2	0.85 (0.07)	0.88 (0.06)	0.86 (0.07)	**0.90 (0.04)**	0.88 (0.05)	0.86 (0.07)
Meeting Room	0.85 (0.06)	0.83 (0.05)	0.83 (0.05)	**0.87 (0.05)**	0.85 (0.07)	0.83 (0.06)
WaterSurface	0.73 (0.02)	0.76 (0.03)	0.76 (0.03)	**0.86 (0.03)**	0.82 (0.03)	0.76 (0.03)
Lobby	0.58 (0.24)	0.56 (0.24)	0.56 (0.24)	0.57 (0.23)	0.51 (0.22)	**0.66 (0.27)**
OneShopOneWait1cor	**0.80 (0.01)**	0.76 (0.02)	0.77 (0.02)	0.76 (0.02)	0.74 (0.03)	0.78 (0.01)
LevelCrossing	**0.87 (0.06)**	0.75 (0.12)	0.76 (0.12)	0.80 (0.05)	0.76 (0.08)	0.84 (0.07)

Table 8: Precision obtained by the GHNF model for each video sequence and color space. Best results are in bold. Standard deviations are in parentheses.

	RGB	Lab	HSV	HSL	Luv	YCbCr
Video2	0.96 (0.01)	0.97 (0.01)	**0.98 (0.01)**	0.89 (0.02)	0.89 (0.03)	0.97 (0.01)
Meeting Room	0.96 (0.07)	**0.98 (0.04)**	**0.98 (0.04)**	0.96 (0.03)	0.96 (0.03)	0.97 (0.06)
WaterSurface	0.99 (0.01)	**1.00 (0.00)**	**1.00 (0.00)**	**1.00 (0.00)**	1.00 (0.01)	**1.00 (0.00)**
Lobby	0.57 (0.26)	0.62 (0.28)	0.60 (0.28)	0.57 (0.25)	0.54 (0.24)	**0.70 (0.31)**
OneShopOneWait1cor	0.85 (0.01)	**0.88 (0.02)**	**0.88 (0.02)**	0.79 (0.02)	0.76 (0.03)	**0.88 (0.02)**
LevelCrossing	0.95 (0.03)	0.96 (0.03)	**0.97 (0.02)**	0.91 (0.07)	0.93 (0.05)	0.96 (0.03)

Table 9: Recall obtained by the GHNF model for each video sequence and color space. Best results are in bold. Standard deviations are in parentheses.

	RGB	Lab	HSV	HSL	Luv	YCbCr
Video2	0.77 (0.11)	0.81 (0.09)	0.78 (0.10)	**0.91 (0.08)**	0.89 (0.09)	0.78 (0.11)
Meeting Room	0.77 (0.06)	0.72 (0.07)	0.72 (0.07)	**0.80 (0.09)**	0.77 (0.11)	0.73 (0.07)
WaterSurface	0.58 (0.03)	0.62 (0.04)	0.62 (0.05)	**0.75 (0.04)**	0.70 (0.04)	0.61 (0.04)
Lobby	0.63 (0.20)	0.54 (0.22)	0.57 (0.21)	0.60 (0.19)	0.50 (0.20)	**0.65 (0.23)**
OneShopOneWait1cor	**0.76 (0.02)**	0.67 (0.02)	0.68 (0.02)	0.73 (0.02)	0.72 (0.02)	0.70 (0.02)
LevelCrossing	**0.81 (0.10)**	0.64 (0.16)	0.64 (0.17)	0.72 (0.07)	0.65 (0.11)	0.76 (0.12)

Table 10: False positives obtained by the GHNF model for each video sequence and color space. Best results are in bold. Standard deviations are in parentheses.

	RGB	Lab	HSV	HSL	Luv	YCbCr
Video2	0.03 (0.01)	**0.02 (0.01)**	**0.02 (0.01)**	0.10 (0.02)	0.10 (0.03)	**0.02 (0.01)**
Meeting Room	0.03 (0.05)	**0.01 (0.03)**	**0.01 (0.03)**	0.03 (0.03)	0.03 (0.02)	0.02 (0.04)
WaterSurface	0.00 (0.01)	**0.00 (0.00)**	**0.00 (0.00)**	**0.00 (0.00)**	**0.00 (0.00)**	**0.00 (0.00)**
Lobby	0.35 (0.25)	0.29 (0.27)	0.31 (0.26)	0.34 (0.24)	0.33 (0.21)	**0.25 (0.28)**
OneShopOneWait1cor	0.12 (0.01)	**0.08 (0.02)**	**0.08 (0.02)**	0.16 (0.02)	0.18 (0.02)	0.09 (0.01)
LevelCrossing	0.04 (0.03)	0.03 (0.02)	**0.02 (0.01)**	0.07 (0.06)	0.04 (0.04)	0.03 (0.02)

Table 11: False negatives obtained by the GHNF model for each video sequence and color space. Best results are in bold. Standard deviations are in parentheses.

	RGB	Lab	HSV	HSL	Luv	YCbCr
Video2	0.23 (0.11)	0.19 (0.09)	0.21 (0.10)	**0.08 (0.07)**	0.10 (0.09)	0.21 (0.11)
Meeting Room	0.22 (0.05)	0.28 (0.07)	0.28 (0.07)	**0.20 (0.09)**	0.22 (0.11)	0.26 (0.07)
WaterSurface	0.42 (0.03)	0.38 (0.05)	0.38 (0.05)	**0.25 (0.04)**	0.30 (0.04)	0.39 (0.04)
Lobby	**0.20 (0.06)**	0.30 (0.12)	0.26 (0.09)	0.23 (0.09)	0.31 (0.09)	0.22 (0.07)
OneShopOneWait1cor	**0.21 (0.01)**	0.30 (0.02)	0.30 (0.02)	0.23 (0.01)	0.23 (0.02)	0.27 (0.02)
LevelCrossing	**0.18 (0.09)**	0.36 (0.16)	0.36 (0.17)	0.26 (0.07)	0.34 (0.11)	0.24 (0.12)

Table 12: CPU time (in seconds) obtained by the GHNF model for each video sequence and color space. The CPU time is computed as the mean of foreground detection running time. Best results are in bold.

	RGB	Lab	Luv	HSV	HSL	YCbCr
Video2	3.991	3.706	4.869	4.777	**2.565**	4.838
Meeting Room	**0.003**	**0.003**	0.004	0.004	**0.003**	0.004
WaterSurface	0.016	0.016	0.018	0.019	**0.014**	0.019
Lobby	0.007	0.007	0.008	0.008	**0.006**	0.008
OneShopOneWait1cor	0.072	0.068	0.085	0.081	**0.050**	0.087
LevelCrossing	1.923	1.890	2.407	2.371	**1.277**	2.354

We have compared our results with those achieved by the GNF and its competitors in (Palomo & López-Rubio, 2016b). These competitors are pixel-level methods like the GHNF and GNF, namely WrenGA, GrimsonGMM, GMMV1, GMMV2, FASOM, and MFBM. The first method (Wren et al., 1997) features a single Gaussian. GrimsonGMM (Stauffer & Grimson, 1999), GMMV1 (KaewTraKulPong & Bowden, 2002) and GMMV2 (Zivkovic, 2004) are three mixture Gaussian approaches. FASOM (Maddalena & Petrosino, 2010) is a fuzzy approach to a self-organizing background subtraction (SOBS) algorithm. The last method (MFBM) (López-Rubio & López-Rubio, 2015) is a probabilistic mixture model which handles any number of pixel features and accounts for the correlations among these features. Best results optimizing parameters and feature vectors for each method have been obtained from (López-Rubio & López-Rubio, 2015)[5] and from (Palomo & López-Rubio, 2016b).

Accuracy and F-measure results for the GHNF, GNF, WrenGA, GrimsonGMM, GMMV1, GMMV2, FASOM, and MFBM are given in Tables 13

[5]Reprinted from Computer Vision and Image Understanding, 133, Francisco Javier López-Rubio and Ezequiel López-Rubio, Features for stochastic approximation based foreground detection, 30-50, Copyright (2015), with permission from Elsevier.

Table 13: Accuracy obtained by the GHNF and different pixel-level methods for each video sequence. Best results are in bold. Standard deviations are in parentheses.

Video	WrenGA	GrimsonGMM	GMMV1	GMMV2	FASOM	MFBM	GNF	GHNF
Video2	0.55 (0.09)	0.62 (0.08)	0.43 (0.07)	0.65 (0.08)	0.72 (0.02)	0.73 (0.04)	**0.83 (0.05)**	0.82 (0.07)
Meeting Room	0.54 (0.09)	0.44 (0.09)	0.46 (0.15)	0.54 (0.16)	**0.78 (0.03)**	0.77 (0.03)	0.73 (0.06)	0.77 (0.08)
Water surface	0.77 (0.04)	0.55 (0.20)	0.68 (0.13)	0.53 (0.16)	0.83 (0.01)	**0.91 (0.01)**	0.83 (0.05)	0.75 (0.04)
Lobby	0.21 (0.06)	0.40 (0.06)	0.12 (0.03)	0.44 (0.10)	0.41 (0.05)	0.22 (0.17)	0.36 (0.18)	**0.53 (0.24)**
OneShopOneWait1cor	0.67 (0.04)	0.38 (0.11)	0.53 (0.16)	0.37 (0.11)	0.61 (0.03)	0.66 (0.05)	0.64 (0.02)	**0.67 (0.02)**
LevelCrossing	0.66 (0.06)	0.76 (0.03)	0.53 (0.06)	0.68 (0.08)	0.78 (0.02)	**0.85 (0.01)**	0.54 (0.08)	0.77 (0.09)
Rank sums	31.00	36.00	43.00	33.50	17.50	16.50	23.50	**15.00**

Table 14: F-measure obtained by the GHNF and different pixel-level methods for each video sequence. Best results are in bold. Standard deviations are in parentheses.

Video	WrenGA	GrimsonGMM	GMMV1	GMMV2	FASOM	MFBM	GNF	GHNF
Video2	0.53 (0.04)	0.55 (0.04)	0.45 (0.05)	0.57 (0.03)	0.60 (0.01)	0.84 (0.03)	**0.90 (0.03)**	0.90 (0.04)
Meeting room	0.68 (0.09)	0.57 (0.09)	0.58 (0.16)	0.68 (0.15)	**0.87 (0.02)**	**0.87 (0.02)**	0.84 (0.04)	0.87 (0.05)
Water surface	0.87 (0.03)	0.67 (0.18)	0.80 (0.12)	0.67 (0.15)	0.91 (0.00)	**0.95 (0.00)**	0.91 (0.03)	0.86 (0.03)
Lobby	0.29 (0.08)	0.54 (0.08)	0.18 (0.03)	0.58 (0.11)	0.54 (0.05)	0.28 (0.20)	0.36 (0.18)	**0.66 (0.27)**
OneShopOneWait1cor	0.80 (0.03)	0.54 (0.12)	0.67 (0.16)	0.53 (0.12)	0.76 (0.02)	0.79 (0.03)	0.78 (0.01)	**0.80 (0.01)**
LevelCrossing	0.79 (0.04)	0.86 (0.02)	0.68 (0.06)	0.80 (0.07)	0.88 (0.01)	**0.92 (0.01)**	0.54 (0.08)	0.87 (0.06)
Rank sums	30.00	36.00	42.00	33.00	19.00	17.00	25.00	**14.00**

and 14, respectively. In these tables we can see how the GHNF achieves the best results for 'Lobby' and 'OneShopOneWait1cor' video sequences, whereas its results for the rest of videos are near the best result. This good performance in foreground detection can be confirmed by observing the Rank sums in the tables, where our proposal yields the best overall performance in terms of accuracy and F-measure, also outperforming our previous proposal (GNF).

4. Discussion

In the context of self-organizing neural networks, the GHNF has two outstanding features, namely the unsupervised learning of a hierarchy, and the use of forests as the basic building block of such hierarchy. Next the consequences of these features are examined.

Forests, i.e. sets of trees, are one of the most sparing topologies with the connections among units. In other words, using forests allows learning intricate topologies which have a low average number of connections per unit. An immediate advantage of this is that the computational complexity of the training of a forest is greatly reduced when compared to typical topologies such as regular lattices (as in the standard Self-Organizing Map, SOM) or general graphs (as in the Growing Neural Gas, GNG). Hence forests can be a suitable solution whenever the number of units of the neural model must be large or the computational power of the underlying hardware is limited. Furthermore, forests do not impose a two dimensional structure on

the input dataset, which is the case of regular lattices. Therefore they are more flexible to adapt to complex high dimensional distributions. Thirdly, forests can always be plotted on the plane without intersections among unit connections. This is not the case for general graphs, and ensures that the GHNF can be easily visualized in two dimensions.

The above discussed features of the GHNF are also shared by the GNF. However, the hierarchical structure of the GHNF has additional advantages. At each level of the hierarchy, the size of the training set for each forest is substantially reduced. This means that the deeper forests convey a more detailed representation of smaller regions of the input distribution. This way, the user can select the degree of detail that best suits a particular situation, without having to retrain the model. Moreover, the hierarchical representation can be plotted in two dimensions, as seen in the experiments.

5. Conclusions

A novel self-organizing artificial neural network called Growing Hierarchical Neural Forest (GHNF) has been proposed in this paper. This model can be considered a hierarchical extension of the Growing Neural Forest (GNF). The GNF comes close to the Growing Neural Gas (GNG), but it computes a spanning tree for each subgraph of the overall graph, thereby obtaining a forest structure instead of a graph with no special structure. Therefore, our proposal provides a greater flexibility while at the same time it is able to represent hierarchical relations that can be present in the input data.

In order to demonstrate the performance of the GHNF for exploratory data analysis tasks, three different experiments have been carried out. The first one check the correct unfolding of our proposal when run on input distributions of different dimensions. The obtained quantitative results outperformed those achieved by its competitor called Growing Hierarchical Neural Gas (GHNG). The second experiment shows the capabilities of the GHNF to perform text mining of tweets belonging to American politicians, where hierarchical representations of concepts present in those tweets are provided. Thus, a better understanding of these tweets is gained. Finally, the GHNF has been applied to a challenging task such as foreground detection in video sequences, which is a typical computer vision application. Both qualitative and quantitative results demonstrate its suitability for this task, achieving the best results in terms of accuracy and F-measure compared with other pixel-level methods.

Acknowledgments

This work is partially supported by the Ministry of Economy and Competitiveness of Spain under grant TIN2014-53465-R, project name Video surveillance by active search of anomalous events. It is also partially supported by the Autonomous Government of Andalusia (Spain) under projects TIC-6213, project name Development of Self-Organizing Neural Networks for Information Technologies; and TIC-657, project name Self-organizing systems and robust estimators for video surveillance. All of them include funds from the European Regional Development Fund (ERDF). The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the SCBI (Supercomputing and Bioinformatics) center of the University of Málaga. They also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research. Finally, they are grateful to Mr Haritz Puerto-San-Román for providing the tweets dataset.

Aizawa, A. (2003). An information-theoretic perspective of tf-idf measures. *Information Processing and Management*, *39*, 45–65.

Alahakoon, D., Halgamuge, S. K., & Srinivasan, B. (1998). A self-growing cluster development approach to data mining. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on* (pp. 2901–2906). volume 3.

de Amorim, R. C., & Hennig, C. (2015). Recovering the number of clusters in data sets with noise features using feature rescaling factors. *Information Science*, *324*, 126–145.

Ashok, P., Kadhar, G., Elayaraja, E., & Vadivel, V. (2013). Fuzzy based clustering method on yeast dataset with different fuzzification methods. In *Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on* (pp. 1–6).

Astudillo, C. A., & Oommen, B. J. (2014). Self-organizing maps whose topologies can be learned with adaptive binary search trees using conditional rotations. *Pattern Recognition*, *47*, 96–113.

Barbalho, J., Duarte, A., Neto, D., Costa, J., & Netto, M. (2001). Hierarchical SOM applied to image compression. In K. Marko, & P. Werbos (Eds.),

Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on (pp. 442–447). volume 1.

Bertolini, G., & Ramat, S. (2007). A hierarchical som to identify and recognize objects in sequences of stereo images. In T. Jarm, P. Kramar, & A. Zupanic (Eds.), *11th Mediterranean Conference on Medical and Biomedical Engineering and Computing 2007: MEDICON 2007, 26-30 June 2007, Ljubljana, Slovenia* (pp. 977–981). Berlin, Heidelberg: Springer Berlin Heidelberg.

Bhandarkar, S. M., Koh, J., & Suk, M. (1997). Multiscale image segmentation using a hierarchical self-organizing map. *Neurocomputing*, *14*, 241 – 272.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*, 391–407.

Fritzke, B. (1995). A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*, *7*, 625–632.

Gemignani, G., & Rozza, A. (2016). A robust approach for the background subtraction based on multi-layered self-organizing maps. *IEEE Transactions on Image Processing*, *25*, 5239–5251.

Granados, A., Koroutchev, K., & de Borja Rodriguez, F. (2015). Discovering data set nature through algorithmic clustering based on string compression. *IEEE Transactions on Knowledge and Data Engineering*, *27*, 699–711.

Henriques, R., Lobo, V., & Bacao, F. (2012). Spatial clustering using hierarchical som. In M. Johnsson (Ed.), *Applications of Self-Organizing Maps* chapter 12. Rijeka: InTech.

KaewTraKulPong, P., & Bowden, R. (2002). An improved adaptive background mixture model for real-time tracking with shadow detection. In P. Remagnino, G. A. Jones, N. Paragios, & C. Regazzoni (Eds.), *Video-Based Surveillance Systems* (pp. 135–144). Springer US.

Kim, J., & Billard, L. (2011). A polythetic clustering process and cluster validity indexes for histogram-valued objects. *Computational Statistics and Data Analysis*, *55*, 2250 – 2262.

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

Koikkalainen, P., & Oja, E. (1990). Self-organizing hierarchical feature maps. In *1990 IJCNN International Joint Conference on Neural Networks* (pp. 279–284). volume 2.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, *7*, 48–50.

Lampinen, J., & Oja, E. (1992). Clustering properties of hierarchical self-organizing maps. *Journal of Mathematical Imaging and Vision*, *2*, 261–272.

Liu, Y., & Takatsuka, M. (2009). Interactive hierarchical som for image retrieval visualization. In C. S. Leung, M. Lee, & J. H. Chan (Eds.), *Neural Information Processing: 16th International Conference, ICONIP 2009, Bangkok, Thailand, December 1-5, 2009, Proceedings, Part II* (pp. 845–854). Berlin, Heidelberg: Springer Berlin Heidelberg.

López-Rubio, F. J., & López-Rubio, E. (2015). Features for stochastic approximation based foreground detection. *Computer Vision and Image Understanding*, *133*, 30 – 50.

Maddalena, L., & Petrosino, A. (2008). A self-organizing approach to background subtraction for visual surveillance applications. *Trans. Img. Proc.*, *17*, 1168–1177.

Maddalena, L., & Petrosino, A. (2010). A fuzzy spatial coherence-based approach to background/foreground separation for moving object detection. *Neural Computing and Applications*, *19*, 179–186.

Martinetz, T., & Schulten, K. (1991). A "Neural-Gas" Network Learns Topologies. *Artificial Neural Networks*, *I*, 397–402.

Miikkulainen, R. (1990). Script recognition with hierarchical feature maps. *Connection Science*, *2*, 83–101.

Pakkanen, J., Iivarinen, J., & Oja, E. (2004). The evolving tree - a novel self-organizing network for data analysis. *Neural Processing Letters*, *20*, 199–211.

Palomo, E. J., & López-Rubio, E. (2016a). The growing hierarchical neural gas self-organizing neural network. *IEEE Transactions on Neural Networks and Learning Systems*, *PP*, 1–10.

Palomo, E. J., & López-Rubio, E. (2016b). Learning topologies with the growing neural forest. *International Journal of Neural Systems*, *26*, 1650019.

Rahmel, J. (1996). SplitNet: learning of tree structured Kohonen chains. In *IEEE International Conference on Neural Networks* (pp. 1221–1226). volume 2.

Rauber, A., Merkl, D., & Dittenbach, M. (2002). The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data. *IEEE Transactions on Neural Networks*, *13*, 1331–1341.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53 – 65.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, *24*, 513–523.

Samsonova, E. V., Kok, J. N., & Ijzerman, A. P. (2006). TreeSOM: Cluster analysis in the self-organizing map. *Neural Networks*, *19*, 935 – 949.

Sheikh, Y., & Shah, M. (2005). Bayesian modeling of dynamic scenes for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*, 1778–1792.

Stauffer, C., & Grimson, W. (1999). Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on* (pp. 246–252). volume 2.

Wren, C., Azarbayejani, A., Darrell, T., & Pentland, A. (1997). Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*, 780–785.

Zivkovic, Z. (2004). Improved adaptive Gaussian mixture model for background subtraction. In *Proceedings of the International Conference on Pattern Recognition* (pp. 28–31). volume 2.