Hierarchical Overlapped Growing Neural Gas Networks with Applications to Video Shot Detection and Motion Characterization

Xiang Cao and P. N. Suganthan
School of Electrical and Electronic Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798
p148482455@ntu.edu.sg, epnsugan@ntu.edu.sg

Abstract – This paper describes a hierarchical overlapped architecture (HOGNG) based upon the Growing Neural Gas (GNG) network [1]. The proposed architecture combines the unsupervised and supervised learning schemes in GNG. This novel network model was used to perform automatic video shot detection and motion characterization. Experimental results are presented to show the good classification accuracy of the proposed algorithm on real MPEG video sequences.

I. INTRODUCTION

Recent advances in the storage, communication and compression technologies have made digital video materials more and more available and pervasive. There is an increasing need to efficiently index, browse, search and retrieve the visual information from video databases. Generally accepted, the first step towards automatic video indexing is to break up video into temporal homogeneous segments called shot. Shots can be further divided into sub-shots that exhibit consistent property in terms of camera and object motion, which is also called motion-based shot characterization [13]. Moreover, recognition of camera motion is useful to distinguish gradual shot transitions from the false positives due to camera operations.

In this paper we present a neural network based scheme to detect shot boundary and characterize shots by camera motion. A hierarchical overlapped incremental neural network called HOGNG is proposed to detect the scene change and classify the different camera motion in shots. This architecture is based on Fritzke’s GNG neural network [1]. We combine the unsupervised and supervised GNG networks into the hierarchical structure. A degree of overlap in the upper-level GNGs enables us to make a more accurate final decision by fusing the individual classifications produced by the proposed architecture.

The rest of the paper is organized as follows. In the next section, we discuss the theoretical aspects of the GNG model and present our new HOGNG algorithm. Section III describes the shot detection and camera motion characterization procedures using HOGNG. Some promising experimental results and comparisons are presented in Section IV. The paper is concluded in Section V.

II. HOGNG

A. The Growing Neural Gas Algorithm

Fritzke proposed an incremental self-organizing network with variable topology, known as Growing Neural Gas (GNG) [1], for clustering and vector quantization a few years ago. The GNG has its origin in the Neural Gas (NG) [2] algorithm. It combines the growth mechanism inherited from the Growing Cell Structures (GCS) [3], with the topology generation of Competitive Hebbian Learning (CHL) [4]. The GNG is an unsupervised network model, which is capable of generating and removing both lateral connections and neurons. Hence, no determination of the network dimensionality is required in advance. In particular, starting with a few units (generally, two), new units are inserted successively near the unit featuring the largest local error measurement. Similar to Kohonen’s self-organizing maps (SOM) [5], GNG determines the best-matching unit (bmu) for the current input signal, increases matching at the bmu and its topological neighbors. Unlike SOM, the adaptation strength in GNG is constant over time and only the bmu and its direct topological neighbors are adapted.

Through a combination of the original GNG with radial basis functions (RBF), the Supervised GNG (SGNG) was proposed [6]. In the SGNG, each unit in the GNG is regarded as a RBF unit and associated with a Gaussian function. The number, diameter and position of RBF units are determined automatically through the growth process, which can be stopped as soon as the network performance is good enough. Since positioning of RBF units and supervised training of connection weights are performed in parallel, the current local error measurement can be used to determine where to insert new RBF units. Thus, the network can generalize well and have a relatively small size. The original GNG and SGNG algorithm are summarized in Table I and Table II respectively.

B. Hierarchical and Overlapped Architecture

The structure of the HOGNG network architecture is an adaptation of the hierarchical overlapped architecture developed for SOMs by Suganthan [7]. First, the network is initialized with just one layer which is called the base layer. The base layer is trained using the unsupervised GNG algorithm. Owing to the insertion and removal strategy employed in the GNG, we need not perform the additional merging and removing operation as in [7]. Having completed the unsupervised GNG learning in the base layer, a new
TABLE I
GNG ALGORITHM

GNG1: Start with two nodes \(a\) and \(b\) at random positions \(w_a\) and \(w_b\) in \(\mathbb{R}^n\).
GNG2: Present an input vector \(v\) from the training dataset.
GNG3: Determine the winner \(s_1\) and the second-nearest node \(s_2\), if \(s_1\) and \(s_2\) are not connected by an edge, create it. Set the age of the connection between \(s_1\) and \(s_2\) to zero (“refresh” the edge).
GNG4: Increase the age of all edges emanating from \(s_1\).
GNG5: Increase the local error of \(s_1\) according to:
\[
\Delta E_{s_1} = \left| w_a - v \right|
\]
GNG6: Increase matching for \(s_1\) and its direct topological neighbors by learning rates \(\varepsilon_b\) and \(\varepsilon_n\) respectively (\(\varepsilon_b \gg \varepsilon_n\)).
GNG7: Remove edges whose age is larger than \(a_{\text{max}}\). If this results in nodes having no emanating edge, remove such nodes.
GNG8: Insert a new node every \(\lambda\) adaptation steps as follows (or using other insertion criterion):
1) Determine the node \(q\) with the maximum accumulated local error.
2) Insert a new node \(r\) halfway between \(q\) and its neighbor \(f\) with the largest error variable in the direct topological neighborhood of \(q\).
3) Insert edges connecting the new node \(r\) with \(q\) and \(f\), remove the original edge between \(q\) and \(f\).
4) Interpolate the local error of the new node \(r\) with the local errors of \(q\) and \(f\), decrease \(E_q\) and \(E_f\) by a fraction \(\alpha\).
GNG9: Decrease all local error variables by multiplying them with a fraction \(\beta\).
GNG10: Repeat GNG2-9 until a stopping criterion (e.g., net size or the maximum number of epochs or some performance measure) is fulfilled.

TABLE II
SGNG ALGORITHM

SGNG1: As GNG1.
SGNG2: Create \(m\) linear output units and a weighted connection \(w_{ij}\) from each competitive node \(i\) to each output unit \(j\) (\(j \in \{1,2,\ldots,m\}\)).
SGNG3: Associate every competitive node with a Gaussian function.
SGNG4: Load I/O-pair (\(\xi, \zeta\)) from training data.
SGNG5: As GNG3.
SGNG6: As GNG4.
SGNG7: As GNG6.
SGNG8: As GNG7.
SGNG9: Compute activation \(d_i\) for every competitive node \(i\) according to the Gaussian function. The standard deviation \(\sigma_i\) is defined as the mean length of all edges emanating from \(i\).
SGNG10: Compute the vector \(\sigma=(\sigma_1,\ldots,\sigma_m)\) of all output unit activations by the product of \(w_j\) and \(d_i\) with a sigmoid activation function.
SGNG11: Perform one delta-rule learning step for the output weights by learning rate \(\eta\).
SGNG12: Increase the local error of \(s_1\) according to:
\[
\Delta E_{s_1} = \left| e - \sigma \right|
\]
SGNG13: As GNG8, and interpolate the output weights of the new node \(r\).
SGNG14: As GNG9.
SGNG15: Repeat SGNG4-14 until a stopping criterion (e.g., classification error low enough or the maximum number of epochs) is fulfilled.

SGNG network is created for each node in the base layer, which forms the second level of the hierarchical structure. As depicted in Fig. 1, network \(A', B', C'\) are networks in the second level that are created for the base layer units \(A, B, C\). The SGNG network can be seen consisting of a hidden layer and an output layer. The training algorithm of the SGNG is based on two parts: the adaption of weight vectors in the hidden layer and the adaption of the weights from the hidden to the output layer. The hidden layer of each SGNG network in the second level is initialized using the values of its base layer unit (i.e., root unit) and the direct topological neighbors of the base layer unit, with their connections duplicated. Binary-valued class indicator outputs are used as the desired output values for the output layer, as shown by
\[
d_{ij} = \begin{cases} 
1 & \text{if input vector } X_j \text{ belongs to class } k, \\
0 & \text{otherwise}
\end{cases}
\]
where \(k=1,2,\ldots,m\) and \(m\) is the number of classes. Thus the number of output units is equal to the number of classes and the values of the output units form an output vector for each sample.

![Fig. 1. The HOGNG architecture showing three overlapped units A, B, C from the base GNG network being expanded to the second level. The edges between units are omitted.](image-url)
makes use of the same training sample to train the second level SGNG networks grown from those units. For example, in Fig. 1, the overlapped SGNG network A’ grown from unit A is trained on all the training samples presented to the base layer network for which base unit A is either the winner or one of the first few runners-up. Each of the other networks in the second level is trained in a similar fashion. In other words, if we have an overlap of \( k+1 \) (i.e., a winner and \( k \) runners-up) for the training samples, then each training sample is being used to train \( k+1 \) different second level SGNG networks. The testing samples are also duplicated, but to a lesser degree. Hence the testing samples fit well inside the feature space spanned by the winner and several runners-up in the training data. In addition, this duplication of the samples yields the HOGNG network to generate multiple independent classifications for every sample.

In order to combine the outputs of overlapped SGNG networks, we employed the idea of confidence values \([8]\). Different from \([8]\), we directly take the output vector of the output layer in the second level SGNG network as the confidence vector. That is, without calculation, each element in the output vector is used as the confidence value for every sample in the corresponding class. We can define the output vector of an overlapped second level SGNG network as

\[
C' = \{c_{k}^{i} \mid k = 1,2,..,m \} \quad (i = 1,2,..,n)
\]

where \( m \) is the number of classes and \( n \) is the number of overlaps considered. Then the overall confidence vector \( C^* \) of the HOGNG architecture can be calculated by summing up the individual output vectors of the overlapped SGNG networks, as

\[
C^* = \{c_{k}^{*} = \sum_{i=1}^{n} c_{k}^{i} \mid k = 1,2,..,m \}
\]

We can then assign the class label of the test data according to the maximum entry in the overall confidence vector. More sophisticated decision fusion techniques such as the fuzzy integral \([8]\) may be employed to improve the final decision based on these multiple overlapped SGNGs. The summary of the HOGNG network algorithm is given in Table III.

### III. VIDEO SHOT DETECTION AND CHARACTERIZATION

A shot is defined as a sequence of frames captured from a unique and continuous record from a camera. The process of identifying such a sequence is known as the temporal video segmentation, shot boundary detection or scene change detection. After a video sequence is segmented into shots, one or a few representative frames commonly referred to as key frames can be selected to represent each shot. Since motion is an important component of the underlying scene as well as the camera work present in a shot, shot content characterization based on motion has received a large attention in video databases research. In addition, the clues obtained by camera motion detection are useful to key frame selection. For example, when the camera zooms, the frames at the beginning and end of the zoom may be considered as representative of the entire shot.

For shot boundary detection, the reader is referred to our paper \([9]\) for reviews of the various available methods. In this paper, we mainly discuss about shot motion characterization. In the past, several approaches have been developed for camera motion characterization in spatial domain. Boutheimy et al. \([10]\) proposed a robust statistical method to detect shot change and camera motion. As nowadays video is increasingly stored and transmitted in compressed format (e.g. MPEG), many researchers have shifted to investigate methods for video processing directly in compressed domain. Such solutions have many advantages, such as reduced computational complexity, smaller storage, and faster operation due to the lower data rate. Further, some of the intermediate information (e.g. block motion vectors) for shot characterization is readily available in compressed video for direct use. The use of a 3-parameters motion model and a least squares technique for estimation of camera operation was proposed in \([11]\). Zhang et al. \([12]\) computed measures based on the direction and components of the motion vectors and then used thresholds to determine if a certain camera movement has occurred. A potential disadvantage of this approach is its sensitivity to the suitable choice of thresholds. Patel and Sethi \([13]\) applied decision trees (DTs) built through a process of supervised learning to perform the frame motion classification which relieved the need of thresholding. However, DTs delineate the concept by a set of axis-parallel hyperplanes which constrains their accuracy in realistic problems \([15]\).

In this paper, we present an approach for shot detection and motion characterization based on the proposed HOGNG neural network. In particular, for motion characterization, the frame-by-frame motion vectors are extracted and reconstructed from the encoded MPEG bit stream, and a
suitable set of features is evaluated for each frame of the video sequence. Then the HOGNG structure is trained using these features in order to correctly classify several different types of video data. Finally, shots can be characterized and sub-divided by detecting discontinuities in the camera or object motion properties.

A. Data Pre-processing

A video is first segmented into shots with the use of a shot boundary detection algorithm similar to the one we proposed in [9]. Instead of using GNG, in this paper, we adopt the proposed HOGNG and compare the performance with the GNG.

After segmentation, frames in the shot are classified into several motion classes. In the present simulations, we consider five predefined motion classes [13]: stationary (stationary camera and little scene motion), panning/tilting (camera rotation around its horizontal/vertical axis), zooming (focal length change of the lens of a stationary camera), tracking (moving objects being tracked by the camera) and object motion (stationary camera and large scene motion).

Each of the motion classes described above can be characterized by a specific field in the field of motion vectors of consecutive frames. An MPEG stream consists of a sequence of group of pictures (GOP). The GOP is composed of three types of frames: I frames (intra-coded), P frames (forward motion predicted), and B frames (bi-directional forward and backward motion predicted). Both the forward and backward motion vectors are generally computed by block matching over a search window around each macroblock. Therefore, they can be seen as an approximation of the optical flow. However, these vectors are not defined over a continuous time scale, since they correspond to displacements from the closest I/P frames (reference frames), with bi-directional predictions and the possibility of skipping several frames. Furthermore, no motion vectors are provided for I frames. Thus, to find the motion from frame to frame, an approximation of the frame-by-frame forward motion vectors must be computed using the available ones, as shown in Fig. 2. Based on the rule that only if there is a similar type of vector (forward, backward or both) present in the present frame and the past frame can the frame to frame motion be found, some simple approximations similar to [11] are developed here: For I frame, reverse vectors of the backward motion vectors for the previous B frame are considered; For the P frame or B frame preceded by either the I frame or P frame, only the forward vectors can be used; For the P frame or B frame preceded by a B frame, both forward and backward vectors can be used. In this case, we choose the differential vector of the forward motion vectors or backward motion vectors in the two consecutive frames as the frame-by-frame motion vector, depending on which pair is available. If both pairs exist, the average of these two differential vectors is chosen.

A more sophisticated approach to reconstruct the step-wise motion vectors is using the heterogeneous transcoding method proposed in [14]. Including the ones we computed above, a set of frame-by-frame candidate motion vectors is found. All the estimated motion vectors are then compared, and the one that gives the least coding error in term of sum of absolute differences (SAD) is chosen. Here we assume the motion between the frames is uniform, such that the forward and the reverse motion vectors are images of each other, or an interframe motion vector is a scaled version of a larger frame distance and so on.

![Fig. 2. The original sequence and the converted sequence with frame-by-frame forward motion prediction. The frames are shown in the display order, but numbered in the encoding order.](image)

After the frame to frame motion vectors are extracted, the $3 \times 3$ vector median filter [16] is applied to smooth the reconstructed motion vectors. Fig. 3 shows examples of such motion patterns with the reconstructed and smoothed motion vectors. We can see that some noises and random oriented motion vectors caused by the uniform background are suppressed by the spatial filter.

B. Feature Extraction

Using the smoothed motion vectors, a 28-dimensional feature vector is generated for each frame. The first component of this feature vector is the fraction of the frame macroblocks with no motion, which measures how static the frame is. The reconstructed vector field for each frame is then subdivided into $3 \times 3 = 9$ super-blocks. Three parameters are computed for each super-block: the average of the motion vectors directions, the standard deviation of the motion vectors directions, and the average of motion vectors magnitudes. Note that the computations exclude the macroblocks with zero motion. These parameters of all the super-blocks form the rest components of the feature vector.

To get rid of the discontinuity of the angles at $0/360^\circ$, when calculating motion vectors directions we employ a method to measure the distance of directions which takes into account the “similarity” between motion vectors directions. For example, $5^\circ$ and $355^\circ$ should be recognized as similar with a distance of $10^\circ$ instead of $350^\circ$. The method proposed...
in [15] is a rather accurate solution, while it has a high computational complexity of $O(k^n)$ when there are $n$ directions and $k$ values for each direction. In this paper, for each super-block, we take the direction of the mean vector of the motion vectors as the average direction. This operation also further reduces the effects of some small outlier vectors. All angles are normalized in $(0 \leq \theta < 360^\circ)$. Then in calculating the standard deviation of the directions, we use the following equation to compute the distance of two directions:

$$D(\alpha, \beta) = \begin{cases} |\alpha - \beta| & \text{if } |\alpha - \beta| < \pi \\ 2\pi - |\alpha - \beta| & \text{otherwise} \end{cases}$$  \hspace{1cm} (4)$$

Note that this distance measure for the directions can also be used in the neural network learning described below.

C. Frame and Shot Motion Classification

After the feature vector is extracted for each frame, the frame motion classification is performed by the proposed HOGNG network. The HOGNG is trained by a set of feature vectors with known classification (i.e., the training samples). For testing, the HOGNG network will associate each frame with a motion class label. To suppress some suspect motion discontinuities, a temporal window is introduced to filter the sequence of frame classes. The class of each frame is redefined according to a voting scheme across the neighboring frames.

After each frame in a shot is classified as one of the five motion classes, the consecutive frames that fall in the same class are grouped together into sub-shots. The partition with less than a minimum number of frames of the dominant class is split into two halves and merged into its neighbors so that each new partition contains at least the minimum number of frames of the dominant class. Finally, adjacent sub-shots with the same dominant class are grouped to form a longer sub-shot. The class label that is present more than 50% of the times in a shot is then selected as the shot motion characterization label. A shot is labeled as ambiguous if no motion label is found to occur more than 50% of the times in the shot.

IV. EXPERIMENTAL RESULTS

In this section we provide the experimental validation of the shot boundary detection and motion characterization methods described above. All of the experiments were conducted on a PIII450 PC running Windows-NT system.

For shot boundary detection, we processed on the same data set used in our previous work [9]. Then a comparison can be conducted. Table IV summarizes the segmentation performance of the GNG and the proposed HOGNG. Due to the robust decision mechanism introduced in [9], HOGNG got the similar results with a little improvement.

For motion characterization, first we manually labeled 2100 frames of motion vectors exhibiting different motion types (420 for each class). One feature vector was extracted from each frame as discussed above. Then Approximately 50% of the data set was used to train the neural networks, while the remaining was used as testing samples. Table V shows the classification results for frame motion characterization using SGNG and HOGNG. To better reveal the classification capabilities of the neural networks, the classification accuracies of individual classes are reported in Table VI. It may be seen that while some types of motion are easily identified (e.g., zooming and panning), recognition of object motion is the most challenging problem.

After the frames in one shot were labeled as one of the motion classes by the neural networks, shot motion characterization and sub-shots were obtained by the method described above. A total of 59 shots of different variety were selected from the output of the shot detection system. The
overall accuracies for the shot classification are 94.92% and 96.61%, by SGNG and HOGNG respectively.

In these experiments, for the HOGNG, the degree of overlap is seven for training data and five for testing data. As can be seen in these results shown above, neural networks perform well in the tasks of shot boundary detection and motion characterization. In particular, the proposed HOGNG architecture improves further on the high classification accuracy provided by the original SGNG.

V. CONCLUSION

In this paper, we have proposed a hierarchical overlapped neural network model and a new scheme based on this model for video shot boundary detection and motion characterization. This model combines the unsupervised learning and supervised learning algorithms, and final classification is obtained by fusing the individual classifications generated by the upper-level networks. Thus, with the carefully designed features extracted from the video sequences, the proposed HOGNG network architecture produced a better classification.

Currently, we are investigating the possibility of exploiting other useful information to further improve the classification capabilities and using the proposed HOGNG network to perform further operations in video processing (e.g., spatio-temporal video objects segmentation). Another future work will concentrate on enhancing the proposed neural network architecture (e.g., finding better growing and stopping criteria in the GNG algorithm) and evaluating its performance on other public data sets.

REFERENCES


TABLE IV

<table>
<thead>
<tr>
<th>Video Samples</th>
<th>Detected</th>
<th>False</th>
<th>Missed</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>G</td>
<td>229</td>
<td>6</td>
<td>97.45</td>
<td>94.24</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>230</td>
<td>6</td>
<td>97.46</td>
<td>94.65</td>
</tr>
<tr>
<td>Testing</td>
<td>G</td>
<td>421</td>
<td>30</td>
<td>93.35</td>
<td>93.97</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>422</td>
<td>29</td>
<td>93.57</td>
<td>94.20</td>
</tr>
<tr>
<td>Total</td>
<td>G</td>
<td>650</td>
<td>36</td>
<td>94.75</td>
<td>94.07</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>652</td>
<td>35</td>
<td>94.91</td>
<td>94.36</td>
</tr>
</tbody>
</table>

G – using GNG, H – using HOGNG

TABLE V

<table>
<thead>
<tr>
<th>Classification Accuracy on Frame Motion Characterization</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGNG</td>
</tr>
<tr>
<td>Training samples</td>
</tr>
<tr>
<td>Testing samples</td>
</tr>
<tr>
<td>Total</td>
</tr>
</tbody>
</table>

TABLE VI

<table>
<thead>
<tr>
<th>Recognition of the Individual Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGNG</td>
</tr>
<tr>
<td>Training</td>
</tr>
<tr>
<td>Stationary</td>
</tr>
<tr>
<td>Panning</td>
</tr>
<tr>
<td>Zooming</td>
</tr>
<tr>
<td>Object Tracking</td>
</tr>
<tr>
<td>Object Motion</td>
</tr>
</tbody>
</table>