An Adaptive Method For Texture Characterization In Medical Images Implemented on a Parallel Virtual Machine

Socrates. A. MYLONAS
socrates@lim.intercollege.ac.cy
Computer Science, Intercollege, Limassol Campus,
92 Ayias Phylaxeos Str., P.O.Box 51604, CY-3507, Limassol, Cyprus

ABSTRACT

This paper describes the application of a new texture characterization algorithm for the segmentation of medical ultrasound images. The morphology of these images poses significant problems for the application of traditional image processing techniques and their analysis has been the subject of research for several years. The basis of the algorithm is an optimum signal modelling algorithm (Least Mean Squares-based), which estimates a set of parameters from small image regions. The algorithm has been converted to a structure suitable for implementation on a Parallel Virtual Machine (PVM) consisting of a Network of Workstations (NoW), to improve processing speed. Tests were initially carried out on standard textured images. This paper describes preliminary results of the application of the algorithm in texture discrimination and segmentation of medical ultrasound images. The images examined are primarily used in the diagnosis of carotid plaques, which are linked to the risk of stroke.

Keywords—Adaptive Filtering, LMS Algorithm, Medical Ultrasound, Texture Characterization, Image Processing, PVM.

1. INTRODUCTION

Traditional image processing and segmentation techniques detect region boundaries using edges[1]. These techniques rely on the assumption that regions of significance are characterized by abrupt changes in image intensity. Although this is true in a wide range of applications, in areas such as medical imaging and in ultrasound images in particular[2], this might not be the case. These images usually consist of soft tissue and boundaries are not clearly identified as changes in image intensity. Furthermore, they contain noise and interference. Reducing the effect of noise to make visual inspection and interpretation easier has been the subject of research for several years[3,4]. However, the processed images are still hard to segment with traditional automatic techniques, because the boundaries between regions of interest are usually based on more complex characteristics than pure intensity changes. Human observers have an inherent ability to cope with fuzzy images and to use the regularity in image microstructure to distinguish changes of significance. This regularity is often termed as texture and several empirical methods have been used to quantify it. Among these, are the co-occurrence matrix, higher order statistics, wavelet analysis and parametric techniques (like autoregressive filters and Markov random fields). Although there is no clear indication of a single technique being superior to others, autoregressive modelling and wavelets have generally been reported to produce more consistent results[3].

Previous work has indicated that the parameters $\{w_{pq}\}$ of optimum signal modelling techniques[6,7], extracted from small image regions have the ability to characterize texture[4,5] and may therefore be suitable for region discrimination in images where boundaries are not associated with changes in image intensity. In this sense, the estimated parameters are equivalent to an autoregressive model, which produces locally correlated pixel values when applied to the uncorrelated part of the image signal[1, 3, 8].

The work presented in this paper, demonstrates how the observations made from tests on the behaviour of the algorithm on both real and artificial images[5,6,7], were used to develop a method for the description of texture in medical ultrasound images. Such images, taken from the carotid arteries of patients, are used by medical practitioners to evaluate and monitor the risk of stroke associated with each patient. The algorithm requires considerable amount of computation and has been implemented on a Parallel Virtual Machine (PVM) built on top of a Network of Workstations (NoW), using spare computational capacity from any unused computers[5, 6, 7].

The algorithm and its implementation will be described and results on its performance, speed and application will be presented and discussed.

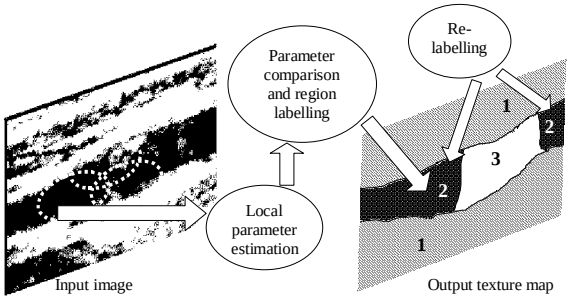

Fig. 1. Conceptual model

2. CONCEPTUAL DESIGN

It is essential to outline the overall concept on which this work is based prior to describing the algorithm, methodology and implementation of the developed system. The method used is based on the fact that adjacent pixel values are correlated[6,7,8] and hence, a given pixel value can be estimated (predicted) from the values of neighbouring pixels and a set of parameters, $\{w_{pq}\}$. These are different for each pixel neigbourhood, because the estimation function in a group of pixels, is expected to be different from that of another area with different appearance. If regions of similar appearance have similar parameter values, it is then possible to use these values to characterize the regions and to ascertain their similarity. Likewise, if regions of different appearance have different parameter values, they can therefore be distinguished based on the values of the parameters. Earlier studies on the use of these parameters to characterize image regions, indicated that the method described in Section 3 of this paper, does produce parameter values, which provide both characterization and discrimination[6,7]. It should be noted that parameter values provide a more robust measure of similarity than direct comparison of pixel values.

As the morphology of an image changes from one area to another, by comparing the estimated parameters extracted from the two image areas, it is possible to identify whether these are similar or not. Hence, each area is assigned a label as follows:

(a) Areas with similar parameters are assigned the same label.

(b) Areas with different parameters are assigned different labels.

The collection of labels for all image regions constitutes the output of the method described. Because the estimated parameters are based on the local image microstructure, they may be considered as texture modelling agents. Thus, the labelled regions can be regarded as a texture map of the image under consideration[3,6,7] (Fig. 1).

From the above, it is evident that any such process must consist of three parts (Fig. 1):

(a) Parameter estimation for each region.

(b) Comparison of the estimated parameters and labelling of adjacent regions.

(c) Relabelling of non-adjacent regions.

The estimation process is carried out on relatively small image blocks, operating on localized data, independent from one region to another. Hence, processing many regions in parallel (simultaneously, rather than sequentially) on several computing nodes was considered in order to reduce the computational time of the algorithm. This is useful in situations where the size of the images (e.g. resolution) is large, resulting in a considerable amount of data to process in a given period of time (e.g. in real time implementations). In these cases, the processing task may be divided into several identical problems with different input data[5, 6, 7].

Similarly, since comparison between the parameter values of adjacent regions is also localized, this process can also be a good candidate for parallel implementation. The relabelling process requires knowledge of regions and parameters over the whole image, but the computational load required is limited. It is evident, that the system is suitable for implementataion on a multiprocessor[5,7].

3. PARAMETER ESTIMATION

The prediction parameters for a small image region can be expressed as a vector, $\mathbf{w} = \{w_{pq}\}$. The objective is to find a set of optimal parameters, such that when they are applied to a group of pixels, $\mathbf{x}_{ij}$ they produce an estimate, $y_{ij} = f(\mathbf{x}_{ij}, \mathbf{w})$, of the pixel value in the centre of the group, d_{ij}. Although $f(\cdot)$ need not be linear, in this implementation it describes a weighted sum ($\mathbf{w}$ expresses the weights to be applied on $\mathbf{x}_{ij}$ to produce

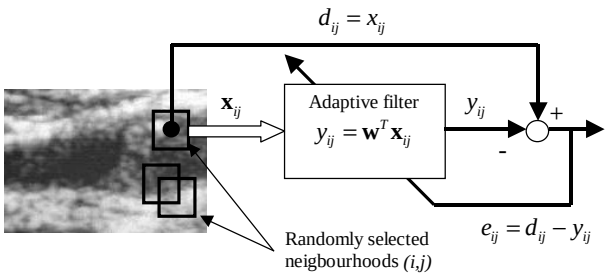

Fig 2. The adaptive process

y_{ij}). This choice has certain advantages, primarily in the existence of well behaving algorithms for the estimation of the optimal $\mathbf{w}$, whose properties are described with general mathematical rigor[9,10,13]. The latter is vital if the algorithm is to operate essentially unsupervised.

For the estimation of $\mathbf{w}$ a range of iterative methods, known as adaptive algorithms have been considered[10,11,12], because of their computational simplicity and stability. The adaptive algorithm used here is the modified two-dimensional LMS algorithm [7, 9, 11, 12,14], which consists of computing the error of the approximation $e_{ij}^{(k)} = d_{ij}^{(k)} - y_{ij}^{(k)}$ on each iteration *k* and using it to modify the weights (parameters) (Fig. 2):

$$w_{pq}^{(k)} = w_{pq}^{(k-1)} + 2\mu^{(k)} e_{ij}^{(k)} x_{i+p,j+q}^{(k)} \qquad (1)$$

In the above algorithm, *p* and *q* indicate the horizontal and vertical displacement from the estimated pixel and $\mu^{(k)} = \frac{\lambda}{\rho^{(k)}}$, where

$$\rho^{(k)} = (1-\alpha) \sum_{p=-m}^{m} \sum_{q=-m}^{m} x_{i+p,j+q}^{2} + \alpha\, \rho^{(k-1)} \qquad (2)$$

The only parameters the algorithm requires is the adaptation rate, λ, whose value is between 0 and 1 and controls how fast the algorithm converges and the forgetting factor, a, which controls the impact of old iterations on Eq. (2).

The neighbourhood over which the algorithm is applied is specified as a square block of MxM pixels. The number of parameters is controlled by *m* which is considerably lower than M.

To ensure that the estimated parameters are not biased, the above algorithm is applied to randomly selected groups of pixels in each block[4,5]. The algorithm generally converges in about 1000 iterations, but if faster convergence is desirable, an adaptive Gram-Schmidt pre-processor[7,10,15] may be used at an additional processing overhead. Initial tests on images were encouraging, but were based on predefined square pixel blocks, which limit the resolution of texture regions to about the block size. The statistical properties of the modelling error, $e_{ij}^{(k)}$, are also estimated during this process.

4. REGION IDENTIFICATION AND LABELLING

To determine the similarity or difference among adjacent image regions, the estimated parameters of each region were compared. Differences in corresponding weights between pairs of image regions were computed and their statistical distribution inspected. Because some parameters are consistently larger than others, the distribution is asymmetric which makes the assumption that they are distributed Normally far from true. For this reason, the Wilcoxon signed rank test of equality of the medians was used[16]. To perform this test, the differences in corresponding parameters, irrespective of sign, must be ranked and the sum of ranks of positive differences, T^+, computed. For a filter of size $n=(2m+1)(2m+1)$, it can be shown that $\mathrm{E}\left[T^+\right] = \frac{n(n-1)}{4}$ and $\mathrm{var}\left(T^+\right) = \frac{n(n+1)(2n+1)}{24}$. From these, the following statistic must be computed.

$$z = \frac{T^+ - \mathrm{E}\left(T^+\right)}{\sqrt{\mathrm{var}\left(T^+\right)}} \qquad (3)$$

This can be tested against a confidence limit, derived from a standard Normal probability distribution. If there is significant difference between the two parameter sets, then the regions are considered different.

The labelling algorithm consists of assigning a unique region identifier to each image region. The parameters of the adaptation are made known to each neighbour, as explained in the following section, which will then perform the test of Eq. (3) for each neighbouring region. For labels to be useful, if a region is found to be similar to another region, it will change its label to the label of the similar region, only if the label is a smaller number. This guarantees that both regions will have the same label.

Finally all labelled regions are gathered for the final labelling process, which inspects them to eliminate any unnecessary ones (due to chaining) and then to identify whether any non-adjacent regions have similar parameters using Eq. (3) and perform final re-labelling.

The labelling process produces an ‘image of labels’, which may be used for further image interpretation.

5. IMPLEMENTATION ON A PARALLEL VIRTUAL MACHINE

A system to model parallel computation has been built using inexpensive personal computers, connected as a Network of Workstations (NoW), as shown in fig. 3. This is based on the widely available Parallel Virtual Machine (PVM) software [17,18,19]. An alternative solution could have employed the Message Passing Interface (MPI)[20] instead of the PVM. Although the MPI solution would have been more efficient[21], PVM is more appropriate when the NoW consists of dissimilar computers and/or software platforms[5,22], which was the case here. It should be noted that one of the requirements for the later use of this and similar techniques is to be deployed on computers, not dedicated to this system, but serving mainly as desktop computers, whose processing power is utilised by the multiprocessor when available.

Performance evaluation[5] indicated that for this type of application, involving short bursts of data over the network and substantial processing on each of the nodes,

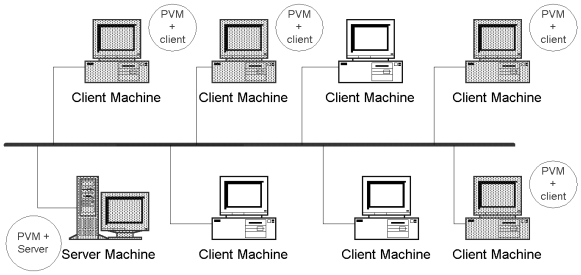

Fig. 3. A network of Workstations running PVM

an almost linear increase in execution speed is possible, even with modest machines and network speed. This is clearly shown in Fig. 4, where the speedup of processing is displayed against the number of nodes in a controlled experiment.

The experiment consisted of noting the time required for the processing of a given image on a multiprocessor consisting of the server and one client machine and then repeating the test by progressively adding client machines to the multiprocessor. Speedup for a given processor configuration is the ratio of the time for processing for a single machine to the time for processing for the corresponding multiprocessor configuration. The system was tested with three image sizes on a virtual machine with 1, 2, 4, 8, and 16 nodes (client machines). Fig. 4 shows five curves, one for each image size (small, S=256x256 pixels, medium, M=512x512 pixels and large, L=1024x1024 pixels. The ideal linear-speedup curve (I) is also shown for reference. The fifth curve corresponds to a small image size using a mixed system configuration (S-mix), which uses a multiprocessor with a mixture of machines with different processing power, to investigate how load balancing operates.

It can be observed that the speedup obtained is near-linear, closely following the ideal curve. For the larger number of nodes, the distance from the ideal curve is larger due to the higher overheads of starting and maintaining a larger virtual machine. The distance from the ideal curve is larger for the large image (L). This is due to the communication cost which is a bottleneck for larger messages on a modest network.

The implementation of the algorithm, described in the previous sections of this paper, on the virtual multiprocessor was based on the client-server paradigm, with one server machine initiating and coordinating activities and each available client machine performing the computational part (Fig. 5).

The server has the image data and it initiates the client processes, one per available machine. It then dispatches to each client process a small section of the image and the

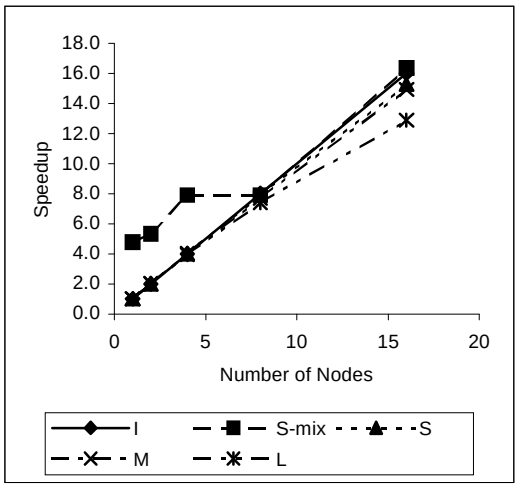

Fig. 4 Speedup as a function of the number of nodes

initial parameter values of the algorithm. The server collects from each client the processed parameter values, sends them to the appropriate clients to compare and label and dispatches another image section to the client.

The client process, upon receiving an image block, performs the adaptation process (Section 3) and then transmits the parameters back to the server. Subsequently, it waits for the server to transmit another image block, or

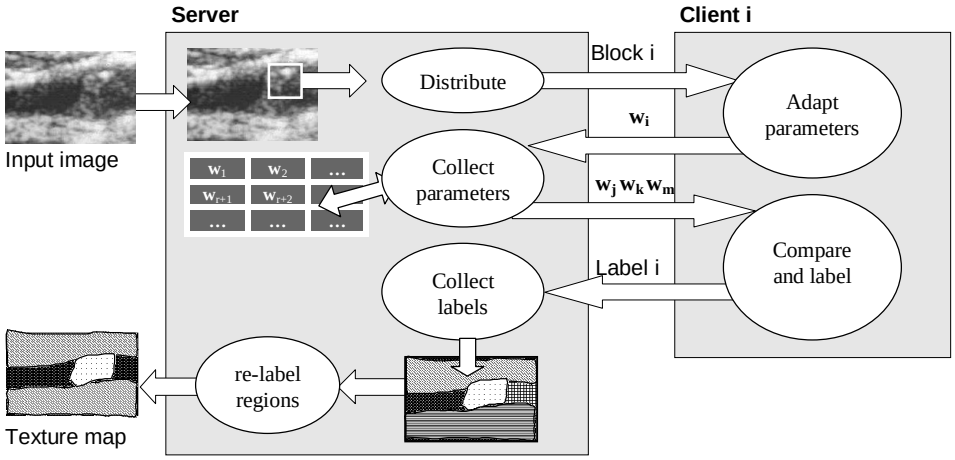

Fig 5. Parallel Implementation

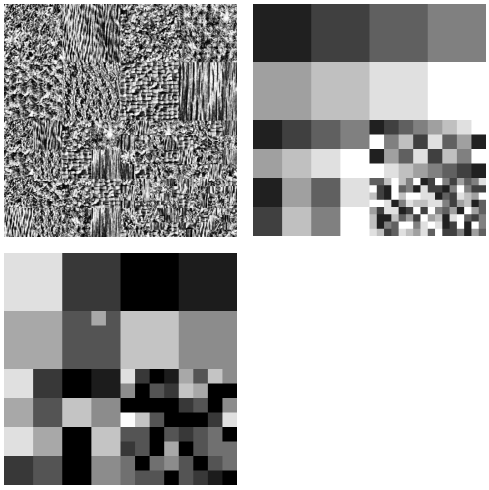

Fig 6. Mosaic, ideal and output texture maps

(when available) parameters from a neighbouring node for comparison. The latter involves the computation and processing of the test described in Section 4, and the initial labelling of the region and informing the server of the outcome.

When the server receives region-labelling information, it determines whether regions have similar parameters and if they do, they are re-labelled. Otherwise, they are simply stored. When the whole image has been processed, the server saves the label map as an image for further processing.

It is important to note the response of the multiprocessor to exceptions. When a machine is added to the multiprocessor (see Fig. 3), the server is notified, a client process is started remotely and the machine is ready to receive and process image data. The server process is also notified when a client machine is about to become unavailable because the desktop user requested it. The server then receives the partial results until that point, which are subsequently sent to another client to complete.

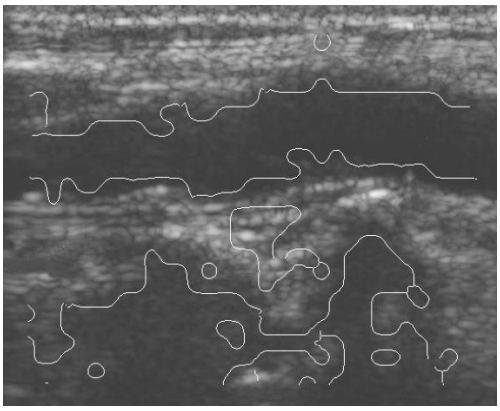

Fig 7. Ultrasound image with detected texture boundaries

6. EXPERIMENTAL RESULTS

The algorithm was initially tested on artificial images and sample textures like the Brodatz Set[23], where the true texture map is known. Once its behaviour was established, it was tested on real medical ultrasound images, used in the diagnosis of the extent of calcification in an artery[2]. As with many real images, there is no objective way to determine a region (texture) map in these cases. Correlation of observations with human experts, who marked different features of significance in each image, was used. After the algorithm completed the labelling process, the results were compared with those of the experts for differences, especially in the boundaries between regions as follows:

Each image was separated into small equal-sized blocks. The number of blocks where the algorithm missed a boundary was counted (this indicates the number of missed features). The number of blocks containing an extraneous texture region was also noted (this indicates 'false' detections). The number of missed features is more significant in practice than extraneous textures, because it is not possible to recover a missing feature with additional processing, whereas further processing of the extraneous texture regions using clustering and other classification techniques may be employed to distinguish those that are medically significant from those that are due to image variability and noise.

Test results appear in Table 1. Each image was tested with two levels of significance 1% and 0.1% for region identification (Section 4). It must be stressed that this paper presents early results on a small sample of images. The algorithm is currently undergoing rigorous testing on a large test bank of medical images. Tests concentrated on determining the algorithm's ability to:

(a) Characterize texture (i.e. classify regions with similar texture in the same category) and

(b) Discriminate texture (i.e. classify regions with different texture in different categories)

The tested system had 11x11 coefficients (m=5), block size 32x32 (M=*32*). Adaptation was repeated for K=1000 iterations with λ=0.1 and α=0.99.

Image	% Missed		% False	
	1%	0.1%	1%	0.1%
1. Brodatz mosaic (512x512)	2%	2%	12%	16%
2. Carotid #1 (300x200)	5%	5%	20%	2%
3. Carotid #2 (242x188)	8%	8%	12%	2%
4. Carotid #3 (515x407)	5%	5%	13%	0%
5. Carotid #4 (465x505)	12%	12%	28%	6%
6. Carotid #5 (568x462)	3%	3%	9%	3%

Table 1. Experimental results

Figure 6 shows the Brodatz *mosaic* together with the output texture map. Figures 7 and 8 show ultrasound

images together with the corresponding texture maps produced by the system.

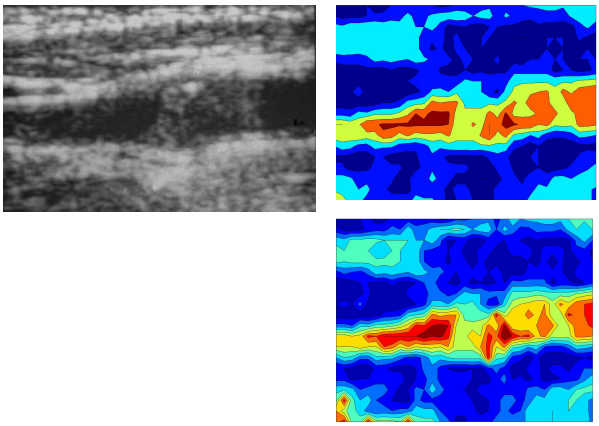

Fig 8. Ultrasound image (left) and texture maps for significance levels of 0.1% (top) and 1% (bottom)

7. COMMENTS AND CONCLUSIONS

Texture Discrimination

As can be seen by the results presented in table 1, the algorithm was fairly successful in quantifying texture in the images considered. This is apparent by the low percentage of missed or misplaced region boundaries, which indicates close correspondence between the expert and the system. It should be stressed at this point that the only parameters used by the system in the description of texture were the prediction parameters, whereas human experts use a variety of cues, often empirical ones, in their classification, based on their experience with other similar images.

The Brodatz *mosaic* was a good test set for the discriminatory power of the algorithm, which performed very well on larger texture blocks, but failed to classify correctly the smaller texture regions in the image. This is primarily attributed to the block size, which controls the region where adaptation takes place. When two separate textured regions appear in the same block, part of the adaptation is performed based on one type of texture and part on another type of texture. As a result the parameters will not correspond to either one of the textures, as they will have values between the parameters of each texture. This is also a potential problem of the algorithm close to the boundary between textures, but the effects there are reduced significantly because of the use of overlapping blocks. The problem may be reduced by selecting smaller block sizes, at the expense of the accuracy of the method, since the parameters extracted from smaller samples will be less accurate.

Texture Characterization

The ability of the algorithm to characterise texture was evaluated by counting the number of additional textures it detected, compared to the 'actual' number indicated by the human experts. As shown in Table 1, for 1% significance level, several extraneous textured regions were detected. This might indicate that the algorithm is not very effective in texture characterization, however, as seen by the images and the corresponding texture maps (Figs. 7 and 8), the algorithm was not imprecise, but rather too specific. Furthermore, many of these additional regions were eliminated (merged with existing ones) for the tests with 0.1% significance level. This effect may be easily interpreted if the significance level is assumed to act as a factor of sensitivity to marginal texture differences. Thus, when increased, more 'detail' in texture is observed. The fact that there were no differences in texture maps between the levels of significance and features marked by a human expert, suggests that the expert probably marked only what was relevant for his/her purposes, rather than label the whole image, as the algorithm did. These additional textures and other medically irrelevant features may be reduced using further processing, such as clustering of similar textures using non-linear classification methods, or predefined classes of relevant features to isolate only those features that are relevant. It is also possible to use the rank sum measure given by Eq. (3) as a measure of similarity between neighbouring regions as an additional measure, for higher level processing, rather than compare it to a fixed confidence limit.

Conclusions

A system for characterising texture in medical ultrasound images has been presented. This was based on a 2-D LMS adaptive modelling algorithm, whose estimated parameter values over small image regions were used to characterize local texture. Parameter values of neighbouring regions were compared using well-established statistical tests to label regions with similar textures. The algorithm requires substantial computational power and has been implemented on a multiprocessor based on a Network of Workstations running PVM. The algorithm is also suitable for clusters consisting of heterogeneous processing nodes and for modest network speeds, as it employs load balancing and is fault-tolerant.

Experimental results indicated that the system was successful in characterizing texture in images and that the level of significance of the comparison used in the labelling process, acts as a factor of sensitivity to the degree of texture similarity.

One of the drawbacks of the algorithm is that if it is applied to extremely small image regions, the parameter values will be over-specialized on the few pixel values on which they were adapted. Using larger regions may lead

to poor texture localization. However, other adaptive algorithms that converge much faster can be used instead of the LMS. A Gram-Schmidt adaptive algorithm[7] has been developed for this purpose.

It must be noted that the proposed algorithm does not rely on the dimensionality of the data and may therefore be easily extended to three-dimensional 'images' (volume) or to spatio-temporal data, such as image sequences (video). In these cases the proposed parallel implementation is expected to be of even greater value.

REFERENCES

[1] E. R. Davies, **Machine Vision, Theory, Algorithms, Practicalities**, **2nd ed**. Academic Press, 1997.

[2] C.I. Christodoulou, C. S. Pattichis, , M. Pantziaris, T. Tegos, A. Nicolaides, T. Elatrozy, M. Sabetai, S. Dhanjil, "Multi-feature texture analysis for the classification of carotid plaques", **Int. Joint Conf. on Neural Networks, IJCNN '99**, vol. 5, 1999, pp. 3591–3596.

[3] T. Randen, J. Håkon Husøy, "Filtering for Texture Classification: A comparative Study", **IEEE Trans. of Pattern Analysis & Machine Intellig.**, vol. 21, No. 4, 1999, pp. 291–310.

[4] S. Omarouayache, S. A. Mylonas, T. J. Ellis, R. A. Comley, "Transputer Implementation of Adaptive Noise Cancelling in Digital Images", **Proceedings, PACTA '92, Barcelona**, 1992, pp. 964-974.

[5] S. A. Mylonas, P. Trancoso, M. Trimikliniotis, "Adaptive Noise Canceling and Edge Detection in Images on a NOW Using PVM", **Proceedings, 10th Mediterranean Electrotechnical Conference (MEleCon 2000)**, 2000, pp. 681-684.

[6] S. A. Mylonas, "Two Adaptive Modelling Algorithms for Texture Characterization on a NOW using PVM", **Proceedings of the 1st IEEE Symposium on Signal Processing and Information Technology (ISSPIT 2001)**, 2001, pp. 60–65, 2001, Cairo, Egypt, IEEE.

[7] S. A. Mylonas, "Texture Discrimination Using Optimum (Adaptive) Modelling on a Parallel Virtual Machine", **Proceedings of the International Conference on Industrial Electronics, Technology & Automation (IETA 2001)**, 2001, pp. 81–86, Cairo, Egypt, IEEE.

[8] J. Mao, A. K. Jain, "Texture Classification and Segmentation using Multiresolution Simultaneous Autoregressive Models", **Pattern Recogition**, vol. 25, No. 2 1992, pp. 173–188.

[9] B. Widrow, S. D. Stearns, **Adaptive Signal Processing**, Prentice Hall, 1985.

[10] S. J. Orphanides, **Optimum Signal Processing: An Introduction**, **2nd ed.**, Mc Graw-Hill, 1990.

[11] B. Widrow, J. M. McCool, M. G. Larrimore, C. R. Johnson Jr., "Stationary and Non-Stationary Learning characteristics of the LMS adaptive Filter", **IEEE Proceedings**, Vol. 64, No. 8, 1976, pp. 1151–1162.

[12] M. M. Hadhoud, D. W. Thomas, "The Two-dimensional Adaptive LMS (TDLMS) Algorithm", **IEEE Trans. Circuits and Systems**, 1988, pp. 485–494.

[13] A. Papoulis, **Probability, Random Variables and Stochastic Proceses**, McGraw-Hill, 1984.

[14] M. Shadayeh, M. Kawamata, "Steady state Analysis of 2-D LMS adaptive filters Using an Independence assumption", **IEICE Trans. Fundamentals**, Vol. EXX-A No 1, 1998.

[15] F. Ling, D. Manolakis, J. G. Proakis, "A Recursive Modified Gram-Schmidt Algorithm for Least Squares Estimation", **IEEE Trans. Acoust. Speech and Signal Processing**, Vol 34, No. 3, 1986, pp 485-494.

[16] J. E. Freund, R. E. Walpole, **Mathematical Statistics 3rd ed.**, Prentice Hall, 1980.

[17] A. C. Arpadi-Dusseau, R. H. Arpadi-Dusseau, D. E. Culler, J. M. Hellerstein, D. A. Patterson "High performance sorting on Networks of Workstations", **SIGMOD '97**, 1997 Tucson, Arizona.

[18] T. J. Sterling, J. Salmon, D. J. Becker, D. F. Savarese, **How to Build a Beowulf: A guide to the Implementation and Application of PC Clusters**, MIT Press, 1999.

[19] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderan, **PVM: Parallel Virtual Machine** MIT Press, 1994.

[20] MPI Forum, "MPI: A message-Passing Interface Standard"; **Int. Journal of Supercomputer Appl.**, Vol 8, 314, 1994, pp. 165-416.

[21] G. A. Geist, J. A. Kohl, P. M. Papadopoulos "PVM and MPI: A Comparison of Features", **Calculateurs Parallels**, Vol. 8, No. 2, 1996.

[22] "Parallel Virtual Machine (PVM)-Version3", http://www.netlib.org/pvm3/index.html.

[23] "The USC-SIPI Image Database", http://sipi.usc.edu/services/database/database.cgi?volume=textur