2-D ITERATIVELY REWEIGHTED LEAST SQUARES LATTICE ALGORITHM AND ITS APPLICATION TO DEFECT DETECTION IN TEXTURED IMAGES*

Ruşen Meylani[1], Cenker Öden[2], Ayşın Ertüzün[1] and Aytül Erçil[3]
[1]Department of Electrical and Electronic Engineering, Boğaziçi University, Bebek, İstanbul, Turkey 34342, [2]Department of Systems and Control Engineering Boğaziçi University, Bebek, İstanbul, Turkey 34342, [3]Faculty of Engineering and Natural Sciences
Sabancı University, Tuzla, İstanbul, 81474,Turkey
e-mail: ertuz@boun.edu.tr

Abstract - In this paper, a 2-D iteratively reweighted least squares lattice algorithm, which is robust to the outliers, is introduced and is applied to defect detection problem in textured images. First, the philosophy of using different optimization functions that results in weighted least squares solution in the theory of 1-D robust regression is extended to 2-D. Then a new algorithm is derived which combines 2-D robust regression concepts with the 2-D recursive least squares lattice algorithm. With this approach, whatever the probability distribution of the prediction error may be, small weights are assigned to the outliers so that the least squares algorithm will be less sensitive to the outliers. Implementation of the proposed iteratively reweighted least squares lattice algorithm to the problem of defect detection in textured images is then considered. The performance evaluation, in terms of defect detection rate, demonstrates the importance of the proposed algorithm in reducing the effect of the outliers that generally correspond to false alarms in classification of textures as defective or nondefective.

Key words: Robust Least Squares Lattice Algorithm, 2-D Lattice Filters,Texture Analysis, Defect Detection

I. INTRODUCTION

The field of multidimensional digital signal processing has become increasingly important in recent years due to number of trends in digital signal processing. Two-dimensional (2-D) lattice filter structures have found numerous applications in 2-D prediction, 2-D spectral estimation, signal modeling, 2-D filter design, image processing such as image compression and coding, restoration and noise cancellation and texture analysis. The 2-D lattice filter structures in literature [1-3] combine one forward and a number of backward prediction error fields into a single structure and they all are modular and can be obtained by cascading identical stages. Each stage is a multi-input/multi-output structure defined in terms of reflection coefficients. The reflection coefficients can be calculated either directly solving normal equations [1-3] or recursively by adaptive methods [4-8]. The algorithms developed for the adaptation of the lattice parameters are either gradient-based [4-6] or

*This work has been partially supported by Turkish Technology Development Foundation under contract number TTGV- 169.

recursive least squares (RLS) type [7-8] algorithms. Ffrench *et.al.*[7] have developed a recursive least squares lattice (RLSL) type adaptive algorithm for the twelve-parameter 2-D lattice filter and applied it on the detection of small objects in correlated clutter. They have shown that RLSL algorithm provides the exact least squares solution for a single stage lattice filter.

Motivated by the success of the RLSL algorithm in [7], we developed an iteratively reweighted least squares lattice (IRLSL) algorithm which is robust to outliers using the concepts of robust estimation methods [11]**.** Even though robust estimation methods are not new in literature, to the best of our knowledge they have not been used in the context of 2-D lattice filters which have many attractive features and are common structures in signal and image processing applications.

The proposed IRLSL algorithm is developed for the twelve-parameter 2-D lattice filter structure that is the most general structure in the sense that no spectral symmetry assumptions are imposed on the input data. However with small modifications, this algorithm can easily be applied to various 2-D lattice structures.

The organization of this paper is as follows: The background material on robust estimation is given in Section II. The general concepts related to the twelve-parameter 2-D lattice filter structure are elaborated in Section III. In Section IV, the IRLSL algorithm for the 2-D twelve-parameter lattice filter is presented. Section V is devoted to a brief explanation of the texture defect detection scheme and then focuses on the practical use of the proposed algorithm for the texture defect detection problem. The experimental results are also presented in this section. Finally, the conclusions are drawn in Section VI. In the Appendix, the 1-D robust regression algorithm is also presented for reference.

II. BACKGROUND MATERIAL ON ROBUST ESTIMATION

The method of least squares is a model dependent procedure used to solve linear filtering problems. This method can be viewed as a deterministic alternative to Wiener filter theory. Method of least squares estimates the unknown parameters of the model:

$$\mathbf{y} = \mathbf{c_0} + \mathbf{X}\mathbf{c} + \boldsymbol{\varepsilon} \tag{2.1}$$

where $\mathbf{y}$ is an n-by-1 vector of responses, $\mathbf{X}$ is an n-by-p matrix of known predictor or independent variables, and $\boldsymbol{\varepsilon}$ is an n-by-1 vector of errors. The objective is to estimate the unknown intercept $\mathbf{c_0}$ and the p-by-1 parameter vector $\mathbf{c}$.

The resulting estimator from the method of least squares is given as:

$$\hat{\mathbf{c}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{2.2}$$

This estimator is equivalent to minimizing a performance index that consists of sum of error squares:

$$J = \sum_{i=1}^{n} \varepsilon_i^2 \tag{2.3}$$

where ε_i's are the elements of the estimation error vector $\boldsymbol{\varepsilon}$.

The method of least squares estimates the unknown parameters directly using Eq. (2.2) or recursively using the RLS algorithm [9]. The least squares estimator, whether it calculates the unknown parameters directly or recursively, is known to be unreliable when the observations contain outliers in the data [10, 11]. The outliers may be present as a result of nonnormal errors. The classical equation of (2.3) can be made robust in a straightforward way; instead of minimizing a sum of squares, we minimize a sum of less rapidly increasing functions of the residuals:

$$J = \sum_{i=1}^{n} \rho(\varepsilon_i / d) \tag{2.4}$$

where ε_i's are the elements of the estimation error vector $\boldsymbol{\varepsilon}$, d is a robust estimator of scale and $\rho(.)$ is some appropriately chosen function which down-weights observations with large residuals. Different types of ρ function can be used to reduce the effects of outliers. The solution to Eq. (2.4) can be found using the iteratively reweighted least squares algorithm [12]. The algorithm for the numerical solution of 1-D robust regression is given in Appendix. The idea in this algorithm is iteratively to update the unknown parameters c_i's i.e. the elements of vector **c** and the parameter d until convergence for both is achieved.

III. 2-D LATTICE FILTERS

A 1-D lattice filter [9] is a prediction error filter where the input to the first stage is the signal of interest and the output is the error between the predicted (either into the future or into the past) and the true values of the input. The error obtained in predicting into the future is referred to as the forward prediction error and the error obtained in predicting into the past is called the backward prediction error. The forward and the backward prediction error filtering are combined into a single structure in the lattice filter. The filter is arranged as a cascade of stages; the input and the output of stages are forward and backward prediction errors. Each stage is characterized by a parameter called the reflection coefficient. The reflection coefficients can be used to represent the input signal as an *autoregressive* (AR) model. There is a one-to-one correspondence between the reflection coefficients and the AR parameters. It is much simpler to use the reflection coefficients instead of the AR parameters to have an AR model of the input signal. Lattice filter has attractive features like modularity, arithmetic insensitivity; and the local optimizations of the stages lead to global optimization of the whole structure. Theory of 1-D lattice filter has been well developed for several applications. It is used in spectral analysis, speech synthesis and system modeling.

2-D lattice filter is a digital filter that has been developed extending the structure of 1-D lattice filter to 2-D. With the attractive features stated [1-8], the 2-D lattice filter has been the center of interest in this study as an effective tool for modeling the input data as an AR process.

2-D lattice filter structures consist of concatenated multi-input/multi-output stages that are defined in terms of the reflection coefficients. The inputs and the outputs of each stage are forward and backward prediction error fields that are generated simultaneously.

Among many different lattice filter structures present in the literature, the twelve-parameter lattice filter, being the most general quarter-plane filter, where no assumptions on spectral symmetry conditions of the input data have been made, will be used in this work.

In twelve-parameter lattice filter, there are four quarter-plane filters which are designed independently. No restrictions are imposed on their design since no spectral assumptions are required [3, 4, 7]. The input-output relation of the *n*-th stage can be explicitly written as follows:

$$\begin{bmatrix} e_{00}^{(n)}(i,j) \\ e_{10}^{(n)}(i,j) \\ e_{11}^{(n)}(i,j) \\ e_{01}^{(n)}(i,j) \end{bmatrix} = \begin{bmatrix} 1 & -k_1^{(n)} & -k_2^{(n)} & -k_3^{(n)} \\ -k_4^{(n)} & 1 & -k_5^{(n)} & -k_6^{(n)} \\ -k_7^{(n)} & -k_8^{(n)} & 1 & -k_9^{(n)} \\ -k_{10}^{(n)} & -k_{11}^{(n)} & -k_{12}^{(n)} & 1 \end{bmatrix} \begin{bmatrix} e_{00}^{(n-1)}(i,j) \\ e_{10}^{(n-1)}(i-1,j) \\ e_{11}^{(n-1)}(i-1,j-1) \\ e_{01}^{(n-1)}(i,j-1) \end{bmatrix} \tag{3.1}$$

Here *(i,j)'s* are the pixel indices, $k_i^{(n)}$'s are the reflection coefficients of the *n*-th stage where $(n)=(n_1,n_2)$ with $(n+1)=(n_1+1,n_2+1)$ and $n_1=1,...,N_1, n_2=1,...,N_2, n=1,...,N$. The error fields $e_{00}^{(n)}(i,j), e_{10}^{(n)}(i,j), e_{11}^{(n)}(i,j)$ and $e_{01}^{(n)}(i,j)$ correspond, respectively, to the first, the second, the third and the fourth quarter plane prediction error fields at the output of the *n*-th lattice stage. The initialization is as follows:

$$e_{00}^{(0)}(i,j) = e_{10}^{(0)}(i,j) = e_{11}^{(0)}(i,j) = e_{01}^{(0)}(i,j) = u(i,j) \tag{3.2}$$

Here $u(i,j)$ represents the 2-D input data or image data. Compactly, the input-output relation of a 2-D lattice filter is given as a linear combination of input prediction error fields as follows:

$$\mathbf{e}^{(n)} = \mathbf{K}^{(n)} \tilde{\mathbf{e}}^{(n-1)} \tag{3.3}$$

where $\mathbf{e}^{(n)}$ and $\tilde{\mathbf{e}}^{(n-1)}$ are, respectively, the output and the delayed input vectors containing forward and backward prediction error fields associated with stage *n*. $\mathbf{K}^{(n)}$ is the matrix of reflection coefficients associated with stage *n.* In vector-matrix notations, the pixel indices will not be written explicitly for clarity. The twelve-parameter lattice filter is illustrated in Fig. 1.

Each row of the matrix in Eq. (3.1) defines the parameters of the relevant prediction error filter. In the basic three-parameter lattice filter, all rows are the permutations of the first row thus it is sufficient to design only one of the filters i.e. it is sufficient to solve one set of normal equations and the other prediction error filters can easily be obtained by simple row, column and row, and column reversals [1]. This is the result of imposing four-quadrant symmetry to the power spectral density. For the twelve-parameter filter, on the other hand, four sets of normal equations have to be solved each

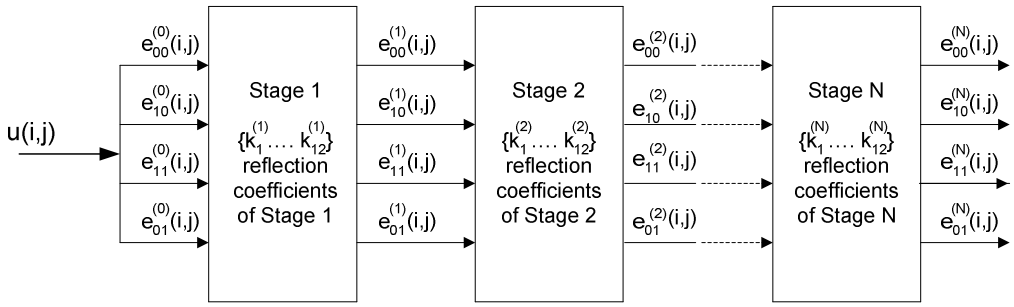

Figure 1a Block Diagram of a 2-D Lattice Filter Structure

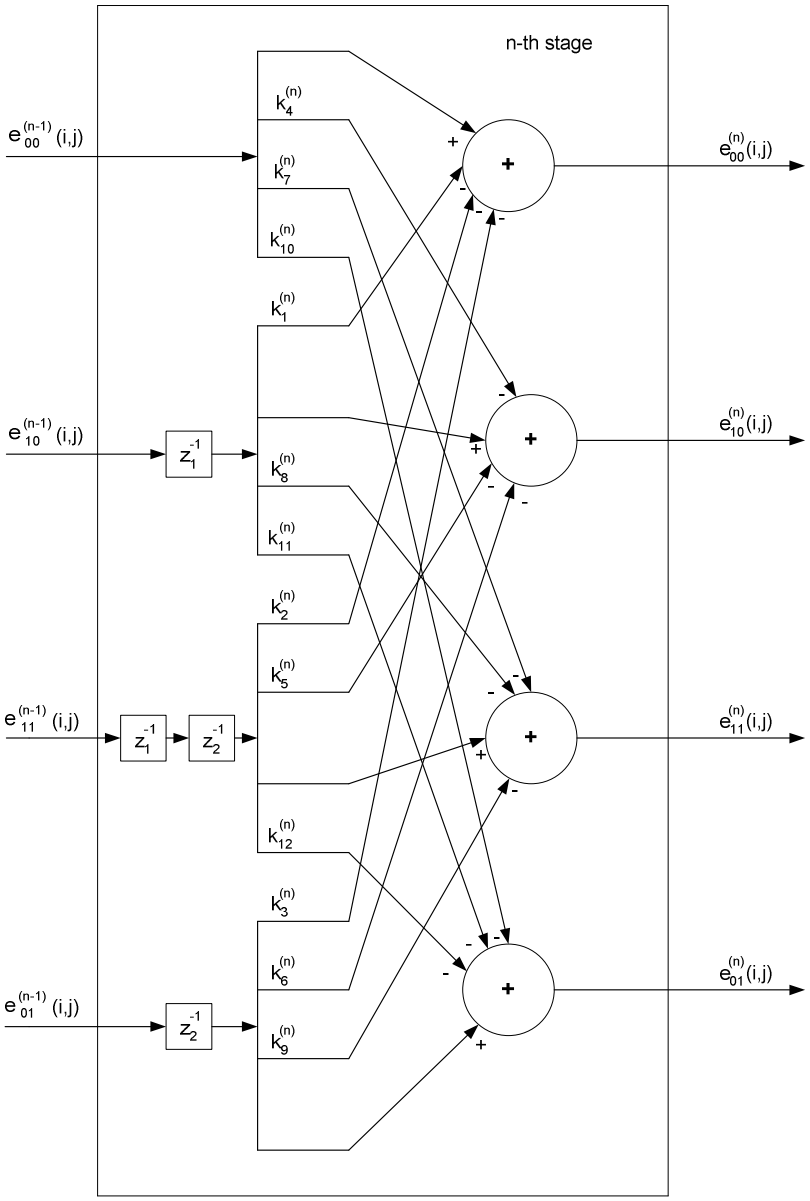

Figure 1b Linear Combination of the prediction error fields
in the *n* –th stage of the twelve-parameter lattice filter

corresponding to one of the quarter-plane prediction error filters. The mean squared error, $Q^{(n)}$, is the cost function for calculating the reflection coefficients, for the *n*-th stage of lattice model is defined as:

$$Q^{(n)} = E[\mathbf{e}^{(n)}(i,j)^T \Lambda \mathbf{e}^{(n)}(i,j)] \tag{3.4}$$

where E[ ] is the expectation operator and T denotes vector transposition. Λ is a diagonal matrix whose elements are equal to either 0 or 1; location of digit 1 shows that the optimization is done in the field relevant to that position [1]. By minimizing $Q^{(n)}$ with respect to the reflection coefficients of the corresponding stage, the following normal equations (*i.e.* the least squares solution) can be obtained for each of the quarter-plane filters:

$$\begin{bmatrix} e_{10}^{(n-1)}(i-1,j) \\ e_{11}^{(n-1)}(i-1,j-1) \\ e_{01}^{(n-1)}(i,j-1) \end{bmatrix} \begin{bmatrix} e_{10}^{(n-1)}(i-1,j) & e_{11}^{(n-1)}(i-1,j-1) & e_{01}^{(n-1)}(i,j-1) \end{bmatrix} \begin{bmatrix} k_1^{(n)} \\ k_2^{(n)} \\ k_3^{(n)} \end{bmatrix} = e_{00}^{(n-1)}(i,j) \begin{bmatrix} e_{10}^{(n-1)}(i-1,j) \\ e_{11}^{(n-1)}(i-1,j-1) \\ e_{01}^{(n-1)}(i,j-1) \end{bmatrix} \tag{3.5a}$$

$$\begin{bmatrix} e_{00}^{(n-1)}(i,j) \\ e_{11}^{(n-1)}(i-1,j-1) \\ e_{01}^{(n-1)}(i,j-1) \end{bmatrix} \begin{bmatrix} e_{00}^{(n-1)}(i,j) & e_{00}^{(n-1)}(i-1,j-1) & e_{01}^{(n-1)}(i,j-1) \end{bmatrix} \begin{bmatrix} k_4^{(n)} \\ k_5^{(n)} \\ k_6^{(n)} \end{bmatrix} = e_{01}^{(n-1)}(i-1,j) \begin{bmatrix} e_{00}^{(n-1)}(i,j) \\ e_{11}^{(n-1)}(i-1,j-1) \\ e_{01}^{(n-1)}(i,j-1) \end{bmatrix} \tag{3.5b}$$

$$\begin{bmatrix} e_{01}^{(n-1)}(i,j-1) \\ e_{00}^{(n-1)}(i,j) \\ e_{10}^{(n-1)}(i-1,j) \end{bmatrix} \begin{bmatrix} e_{01}^{(n-1)}(i,j-1) & e_{00}^{(n-1)}(i,j) & e_{10}^{(n-1)}(i-1,j) \end{bmatrix} \begin{bmatrix} k_9^{(n)} \\ k_7^{(n)} \\ k_8^{(n)} \end{bmatrix} = e_{11}^{(n-1)}(i-1,j-1) \begin{bmatrix} e_{01}^{(n-1)}(i,j-1) \\ e_{00}^{(n-1)}(i,j) \\ e_{10}^{(n-1)}(i-1,j) \end{bmatrix} \tag{3.5c}$$

$$\begin{bmatrix} e_{11}^{(n-1)}(i-1,j-1) \\ e_{00}^{(n-1)}(i,j) \\ e_{10}^{(n-1)}(i-1,j) \end{bmatrix} \begin{bmatrix} e_{11}^{(n-1)}(i-1,j-1) & e_{00}^{(n-1)}(i,j) & e_{10}^{(n-1)}(i-1,j) \end{bmatrix} \begin{bmatrix} k_{12}^{(n)} \\ k_{10}^{(n)} \\ k_{11}^{(n)} \end{bmatrix} = e_{01}^{(n-1)}(i,j-1) \begin{bmatrix} e_{11}^{(n-1)}(i-1,j-1) \\ e_{00}^{(n-1)}(i,j) \\ e_{10}^{(n-1)}(i-1,j) \end{bmatrix} \tag{3.5d}$$

The above equations can also be expressed in terms of the autocorrelation matrix $\mathbf{R}_m^{(n-1)}$ and the cross-correlation vector $\mathbf{r}_m^{(n-1)}$ of the input data and the reflection coefficient vector $\mathbf{k}_m^{(n)}$ of stage *n* for the corresponding prediction error filter.

$$\mathbf{R}_m^{(n-1)} \mathbf{k}_m^{(n)} = \mathbf{r}_m^{(n-1)} \qquad (m = 1,2,3,4) \tag{3.6}$$

where m designates the quadrant of the prediction error filters. The vector and matrix valued variables in Eq. (3.6) will be defined subsequently. The autocorrelation matrices are defined as follows:

$$\mathbf{R}_1^{(n-1)} = \begin{bmatrix} \Phi^{(n-1)}_{e_{10}e_{10}} & \Phi^{(n-1)}_{e_{10}e_{11}} & \Phi^{(n-1)}_{e_{10}e_{01}} \\ \Phi^{(n-1)}_{e_{11}e_{10}} & \Phi^{(n-1)}_{e_{11}e_{11}} & \Phi^{(n-1)}_{e_{11}e_{01}} \\ \Phi^{(n-1)}_{e_{01}e_{10}} & \Phi^{(n-1)}_{e_{01}e_{11}} & \Phi^{(n-1)}_{e_{01}e_{01}} \end{bmatrix} \quad (3.7a) \qquad \mathbf{R}_2^{(n-1)} = \begin{bmatrix} \Phi^{(n-1)}_{e_{00}e_{00}} & \Phi^{(n-1)}_{e_{00}e_{11}} & \Phi^{(n-1)}_{e_{00}e_{01}} \\ \Phi^{(n-1)}_{e_{11}e_{00}} & \Phi^{(n-1)}_{e_{11}e_{11}} & \Phi^{(n-1)}_{e_{11}e_{01}} \\ \Phi^{(n-1)}_{e_{01}e_{00}} & \Phi^{(n-1)}_{e_{01}e_{11}} & \Phi^{(n-1)}_{e_{01}e_{01}} \end{bmatrix} \quad (3.7b)$$

$$\mathbf{R}_3^{(n-1)} = \begin{bmatrix} \Phi^{(n-1)}_{e_{01}e_{01}} & \Phi^{(n-1)}_{e_{01}e_{00}} & \Phi^{(n-1)}_{e_{01}e_{10}} \\ \Phi^{(n-1)}_{e_{00}e_{01}} & \Phi^{(n-1)}_{e_{00}e_{00}} & \Phi^{(n-1)}_{e_{00}e_{10}} \\ \Phi^{(n-1)}_{e_{10}e_{01}} & \Phi^{(n-1)}_{e_{10}e_{00}} & \Phi^{(n-1)}_{e_{10}e_{10}} \end{bmatrix} \quad (3.7c) \qquad \mathbf{R}_4^{(n-1)} = \begin{bmatrix} \Phi^{(n-1)}_{e_{11}e_{11}} & \Phi^{(n-1)}_{e_{11}e_{00}} & \Phi^{(n-1)}_{e_{11}e_{10}} \\ \Phi^{(n-1)}_{e_{00}e_{11}} & \Phi^{(n-1)}_{e_{00}e_{00}} & \Phi^{(n-1)}_{e_{00}e_{10}} \\ \Phi^{(n-1)}_{e_{10}e_{11}} & \Phi^{(n-1)}_{e_{10}e_{00}} & \Phi^{(n-1)}_{e_{10}e_{10}} \end{bmatrix} \quad (3.6d)$$

The crosscorrelation vectors are defined as follows:

$$\mathbf{r}_1^{(n-1)} = \begin{bmatrix} \Phi^{(n-1)}_{e_{00}e_{10}} & \Phi^{(n-1)}_{e_{00}e_{11}} & \Phi^{(n-1)}_{e_{00}e_{01}} \end{bmatrix}^T \quad (3.8a) \qquad \mathbf{r}_2^{(n-1)} = \begin{bmatrix} \Phi^{(n-1)}_{e_{10}e_{00}} & \Phi^{(n-1)}_{e_{10}e_{11}} & \Phi^{(n-1)}_{e_{10}e_{01}} \end{bmatrix}^T \quad (3.8b)$$

$$\mathbf{r}_3^{(n-1)} = \begin{bmatrix} \Phi^{(n-1)}_{e_{11}e_{01}} & \Phi^{(n-1)}_{e_{11}e_{00}} & \Phi^{(n-1)}_{e_{11}e_{10}} \end{bmatrix}^T \quad (3.8c) \qquad \mathbf{r}_4^{(n-1)} = \begin{bmatrix} \Phi^{(n-1)}_{e_{01}e_{11}} & \Phi^{(n-1)}_{e_{01}e_{00}} & \Phi^{(n-1)}_{e_{01}e_{10}} \end{bmatrix}^T \quad (3.8d)$$

where $\Phi^{(n-1)}_{e_{xy}e_{pq}}$'s are the correlations between the prediction errors $e^{(n-1)}_{xy}$ and $e^{(n-1)}_{pq}$; $(x,y,p,q) \in (0,1)$ given as:

$$\Phi^{(n-1)}_{e_{xy}e_{pq}} = E[e^{(n-1)}_{xy}(i-x, j-y)e^{(n-1)}_{pq}(i-p, j-q)] \qquad (x, y, p, q) \in (0,1) \qquad (3.9)$$

In Eqs. (3.7)-(3.9), the pixel indices *(i,j)*'s are left out for clarity. The reflection coefficient vectors $\mathbf{k}^{(n)}_m$'s are defined as follows:

$$\mathbf{k}_1^{(n)} = [k_1^{(n)}\ k_2^{(n)}\ k_3^{(n)}]^T \quad (3.10a) \qquad \mathbf{k}_2^{(n)} = [k_4^{(n)}\ k_5^{(n)}\ k_6^{(n)}]^T \quad (3.10b)$$

$$\mathbf{k}_3^{(n)} = [k_9^{(n)}\ k_7^{(n)}\ k_8^{(n)}]^T \quad (3.10c) \qquad \mathbf{k}_4^{(n)} = [k_{12}^{(n)}\ k_{10}^{(n)}\ k_{11}^{(n)}]^T \quad (3.10d)$$

Calculation of the lattice filter coefficients, which is a major task in lattice filters, involves the solution of the so-called normal equations, namely Eq. (3.6). Alternatively, the reflection coefficients can be calculated adaptively [4-8]. There have been a number of adaptive algorithms developed to update the reflection coefficients of the 2-D lattice filter. A new 2-D adaptive lattice algorithm based on the RLSL concepts and robust regression will be elaborated in the next section.

IV. 2-D ITERATIVELY REWEIGTED LEAST SQUARES LATTICE ALGORITHM

The attractive features of RLS algorithms have initiated the need of obtaining RLSL algorithms in 1-D [9] and 2-D [7]. RLS algorithms, whether in 1-D or 2-D, are based on the recursive

methods for finding the inverse of the autocorrelation matrix **R** [9]. However, in the RLSL algorithm derived by Ffrench *et.al* [7], the correlation values, namely $\Phi^{(n-1)}_{e_{xy}e_{pq}}$, are calculated recursively and the inverses of the autocorrelation matrices are calculated directly. In the derivation of the proposed 2-D IRLSL algorithm, the robust estimation concepts will be incorporated into the RLSL algorithm of [7].

4.1 Background on the 2-D Recursive Least Squares Lattice (RLSL) Algorithm

In the RLSL algorithm of [7], the cost function is the mean squared value of the total prediction error power at the output of stage *n* as

$$Q^{(n)}(m_1,m_2) = \sum_{i=0}^{m_1}\sum_{j=0}^{m_2}\left[\mathbf{e}^{(n)}(i,j)^T\mathbf{e}^{(n)}(i,j)\right]\lambda^{(m_1-i)}\ \lambda^{(m_2-j)} \tag{4.1}$$

here m_1 and m_2 are the pixel indices and λ is the forgetting term, a constant in the interval (0,1), which allows the algorithm to converge to new image statistics or new image features in the least squares sense for non-stationary data. Eq. (4.1) where forgetting terms are included is the 2-D lattice counterpart of Eq. (2.3). It is desirable to calculate the correlation values recursively. In other words the correlation at each pixel *(i,j)* is calculated based on previous pixels *(i-1,j)* and *(i,j-1).* In order to process an image by scanning it in the horizontal direction (i.e. in the m_2 direction), first define a sum of vertical correlation components, namely $\varphi^{(n-1)}_{e_{xy}e_{pq}}(m_1,j)$, and then a recursive horizontal sum of the sum of vertical correlation values. The vertical sum, $\varphi^{(n-1)}_{e_{xy}e_{pq}}(m_1,j)$, is defined as follows:

$$\varphi^{(n-1)}_{e_{xy}e_{pq}}(m_1,j) = \sum_{i=0}^{m_1} e^{(n-1)}_{xy}(i,j)e^{(n-1)}_{pq}(i,j)\lambda^{(m_1-i)} \quad ;\ (x,y,p,q)\in(0,1) \tag{4.2}$$

This sum can be updated recursively as

$$\varphi^{(n-1)}_{e_{xy}e_{pq}}(m_1,j) = \lambda\,\varphi^{(n-1)}_{e_{xy}e_{pq}}(m_1\text{-}1,j) + e^{(n-1)}_{xy}(m_1-x,j-y)e^{(n-1)}_{pq}(m_1-p,j-q)\ ;\ (x,y,p,q)\in(0,1) \tag{4.3}$$

The autocorrelation and cross-correlation values, $\Phi^{(n-1)}_{e_{xy}e_{pq}}(m_1,m_2)$, are then defined in terms of the vertical correlations as follows:

$$\Phi^{(n-1)}_{e_{xy}e_{pq}}(m_1,m_2) = \sum_{j=0}^{m_2}\varphi^{(n-1)}_{e_{xy}e_{pq}}(m_1,j)\ \lambda^{(m_2-j)};\ \ (x,y,p,q)\in(0,1) \tag{4.4}$$

These can be recursively calculated as follows:

$$\Phi^{(n-1)}_{e_{xy}e_{pq}}(m_1,m_2) = \lambda\, \Phi^{(n-1)}_{e_{xy}e_{pq}}(m_1,m_2\text{-}1) + \varphi^{(n-1)}_{e_{xy}e_{pq}}(m_1,m_2); \quad (x,y,p,q) \in (0,1) \tag{4.5}$$

In Eq. (4.3), the vertical correlations are calculated and these are used to calculate the true correlations in (4.5). Thus the true correlations, used for defining the autocorrelation matrices and the cross correlation vectors defined in Section III, are *totally independent* of the scanning scheme used. In this algorithm [7], the correlation values are calculated recursively and since the sizes of the autocorrelation matrices are small, their inverses are taken directly. In this respect the RLSL algorithm of [7] is different than the 1-D RLS algorithm where the inverse of the autocorrelation matrix is calculated recursively [9].

4.2. 2-D Iteratively Reweighted Least Squares Lattice (IRLSL) Algorithm

IRLSL algorithm is a novel approach that extends the idea of using weights in an iterative manner from the 1-D theory of *robust regression* [10,11] to 2-D lattice filters. With this approach, it is intended to ensure that, whatever probability distribution the prediction errors may have, small weights are assigned to the outliers so that the least squares algorithm will be less sensitive to the outliers and improved false alarm rate will be achieved.

Similar to the 1-D case, the 2-D robust estimation provides a method to detect outliers and reduce their effect. When the error distribution is not close to the normal distribution, the cost function to be minimized with respect to the unknown parameters will be given as follows:

$$Q^{(n)} = \sum_{i,j} \rho\left(\frac{\mathbf{e}^{(n)}(i,j)}{d}\right) \tag{4.6}$$

where $\rho(.)$ is an appropriately chosen objective function. This equation is the 2-D counter part of Eq. (2.4) where error is a vector-valued quantity represented by $\mathbf{e}^{(n)}(i,j)$. Different types of ρ functions can be used to reduce the effects of outliers. Some examples of well known $\rho(.)$ functions and the weight functions associated with those objective functions are illustrated in Fig. (2). Weight functions are designed to make sure that smaller weights are given to outliers. For any given objective function $\rho(s)$, there corresponds a weight function related to the first derivative of $\rho(s)$. Thus ρ(s) gives an idea on the general behavior of the weight function in comparison to the mean-squared error. The weight function that corresponds to the squared error is constant 1. Certainly, for distributions other than the normal distribution, the maximum likelihood estimator will be different than the least squares estimator.

The solution of the 2-D least squares lattice estimator is given by Eq. (3.5). On the other hand, the solution to the IRLSL predictor will be modified similar to the algorithm given in Appendix and the solution for the first quadrant (*i.e .m= 1*) prediction error filter is now [13]:

$$\begin{bmatrix} e_{10}^{(n-1)}(i-1,j) \\ e_{11}^{(n-1)}(i-1,j-1) \\ e_{01}^{(n-1)}(i,j-1) \end{bmatrix} \left[e_{10}^{(n-1)}(i-1,j) \;\; e_{11}^{(n-1)}(i-1,j-1) \;\; e_{01}^{(n-1)}(i,j-1) \right] \mathbf{W} \begin{bmatrix} k_1^{(n)} \\ k_2^{(n)} \\ k_3^{(n)} \end{bmatrix} = e_{00}^{(n-1)}(i,j) \begin{bmatrix} e_{10}^{(n-1)}(i-1,j) \\ e_{11}^{(n-1)}(i-1,j-1) \\ e_{01}^{(n-1)}(i,j-1) \end{bmatrix} \mathbf{W} \tag{4.7}$$

where **W** is a 3-by-3 diagonal weight matrix whose elements are closely related to the first derivative of the objective function ρ with respect to the lattice parameters of the first quadrant filter. The diagonal elements of the weight matrix are designed to ensure that smaller weights are given to outliers. The RLSL algorithm of [7] can thus be viewed as a special form of the proposed IRLSL algorithm where all the weight values are equal to 1. For a general objective function ρ, the weight values are defined as follows:

$$w_i = \frac{\dfrac{\partial \rho \left(\dfrac{\mathbf{K}^{(n)} \tilde{\mathbf{e}}^{(n-1)}}{d} \right)}{\partial k_i^{(n)}}}{\left(\dfrac{2e_{00}^{(n)}}{d} \right)} \qquad ; \quad i = 1,2,3 \tag{4.8}$$

Under the recursive estimation approach, the minimization of Eq. (4.6) or the solution given by Eq. (4.7) is equivalent to redefining the calculation of the vertical sum given by Eq. (4.2) in the following manner:

$$\varphi_{e_{xy}e_{pq}}^{(n-1)}(m_1,j) = \sum_{i=0}^{m_1} e_{xy}^{(n-1)}(i,j) w_a e_{pq}^{(n-1)}(i,j) \lambda^{(m_1-i)} \quad ; \qquad (x,y,p,q) \in (0,1) \tag{4.9}$$

where w_a is one of the weights defined in Eq. (4.8) depending on the values of *x, y, p* and *q*.

The recursive update equation for the vertical sum thus is modified as follows:

$$\varphi_{e_{xy}e_{pq}}^{(n-1)}(m_1,j) = \lambda \, \varphi_{e_{xy}e_{pq}}^{(n-1)}(m_1\text{-}1,j) + e_{xy}^{(n-1)}(m_1-x,j-y) w_a e_{pq}^{(n-1)}(m_1-p,j-q) \; ; (x,y,p,q) \in (0,1) \tag{4.10}$$

Eqs. (4.4) and (4.5) need no modifications.

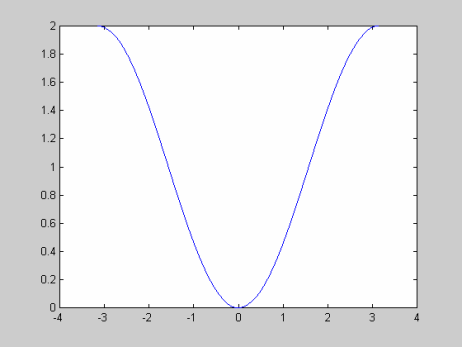

Figure 2a: Objective function 1- cos(s)

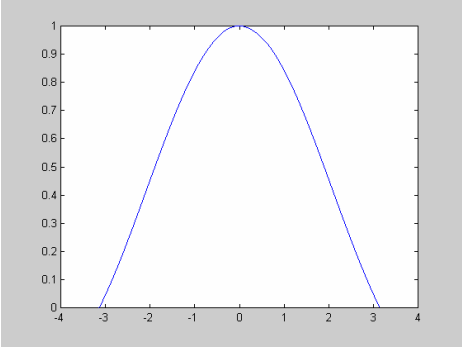

Figure 2b: Weight function (1/s) sin(s) associated with Fig. 2a

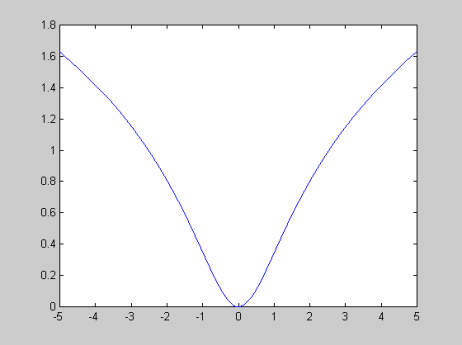

Figure 2c: Objective function (1/2) $\log(1+s^2)$

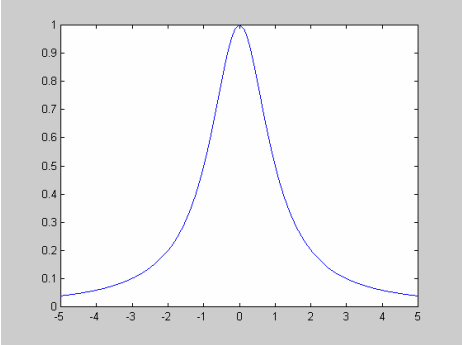

Figure 2d: Weight function $(1+s^2)^{-1}$ associated with Fig. 2c

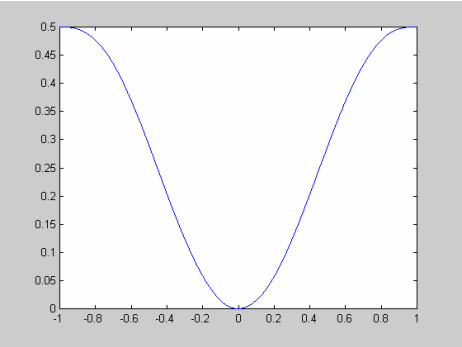

Figure 2e: Objective function (1/2) $(1-(1-s^2)^3)$

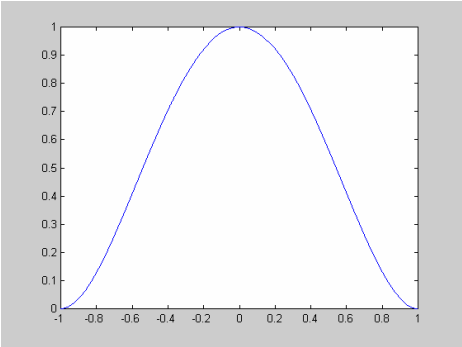

Figure 2f: Weight function $(1-s^2)^2$ associated with Fig. 2e

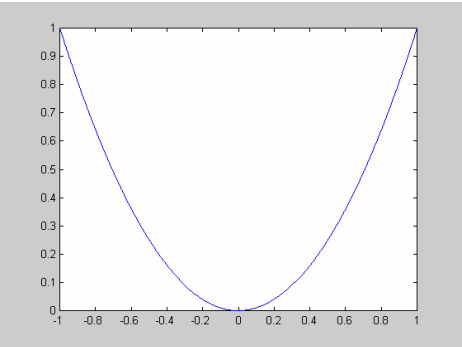

Figure 2g: Objective function s^2

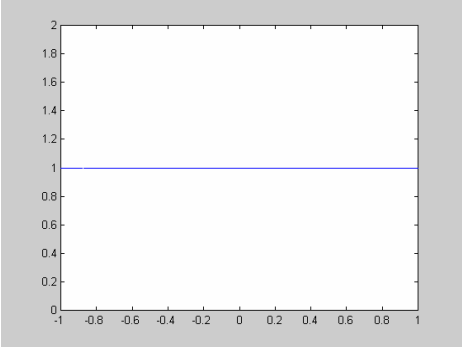

Figure 2h: Weight function 1 associated with Fig. 2g

The equations for the second, the third and the fourth quadrant prediction error filters can be modified similarly.

The proposed IRLSL algorithm [13] is summarized in Table I. It should be noted that in the calculation of parameter *d* in Eq. (4.8) mean values are used rather than the median as in Eq. (A.7) since the prediction error field values are close to each other.

TABLE I. 2-D Iteratively Reweighted Least Squares Lattice (IRLSL) Algorithm

1. Define an objective function in the form of Eq. (4.6). Choose function ρ.
2. *m* denotes the quadrant number *m* = 1,2,3,4
3. Choose forgetting factor λ to be a positive constant less than 1.
4. Determine a small positive constant δ and final stage number N
5. Define the initial values for Φ and ϕ
5. Set
   $e_{pq}^{(0)}(i,j) = I(i,j); \quad ((p,q)\in (0,1))$ where *I (i,j)* is the input texture image data
6. Initialize
   stage number *n=1*
7. Initialize
   iteration *t=1*
8. Set initial values for the reflection coefficients to zero as
   $\mathbf{k}_m^{(n)}(t) = \mathbf{0} \qquad m=1,2,3,4$
9. Set w_{mi}=1 (this corresponds to no weight) for *i=1,2,3,....12* for *m=1,2,3,4.*
10. Compute the correlation values using Eq (4.10) and (4.5) and construct $\mathbf{R}_m^{(n-1)}$ and $\mathbf{r}_m^{(n-1)}$ *m=1,2,3,4* using Eqs. (3.7) and (3.8)
11. Compute $\mathbf{k}_m^{(n)}(t+1) = \left(\mathbf{R}_m^{(n-1)}\right)^{-1} \mathbf{r}_m^{(n-1)} \qquad m=1,2,3,4$ using Eq. (3.6)
12. Compute the prediction errors using Eq. (3.1)
13. If $\left\|\hat{\mathbf{k}}_{\mathbf{m}}^{(n)}(t+1) - \mathbf{k}_{\mathbf{m}}^{(n)}(t)\right\| \le \delta$ where δ is a predetermined tolerance, go to 17
14. Define a distance measure in terms of the prediction errors
    $d = \text{mean}\,|e_{pq}^{(n)}(i,j) - \text{mean}(e_{pq}^{(n)}(i,j))| \quad (p,q)\in (0,1)$
15. Define the new weight values w_{mi} using Eqs. (4.8)
16. Increase iteration as *t=t+1*, go to 10
17. Estimate $\mathbf{k}_m^{(n)}$ by $\hat{\mathbf{k}}_m^{(n)}(t+1)$ for *m=1,2,3,4.*
18. Increase stage number as *n= n+1* until the desired lattice order N is reached and go to Step 7

V. APPLICATION TO TEXTILE DEFECT DETECTION PROBLEM

To illustrate some practical issues of the IRLSL algorithm presented in this paper, it is applied to texture defect detection, more specifically to the textile fabric inspection which is a topical issue in manufacturing.

The automation and the integration of quality control clearly have vital implications for industry. Quality control is designed to ensure that defective products are not allowed to reach the customer.

Texture defect detection is one possible application domain for the proposed IRLSL algorithm; it can be applied to any image processing problem where the undesirable effects of outliers should be alleviated.

In the defect detection applications, AR modeling of the textile images is performed by the 2-D lattice filters. The 2-D lattice filter rather than directly computing the AR model parameters, calculates the reflection coefficients and models the 2-D data in terms of these coefficients. There is a one-to-one correspondence between the reflection coefficients and the AR parameters through Levinson-Durbin type of recursions [1]. In order not to increase the computational complexity, the textures at the input of the lattice filters are modeled in terms of the reflection coefficients rather than the AR parameters. Hence, the reflection coefficient vector will be used as the feature vector to represent the input texture. The reflection coefficients are real numbers in the range of -1 and 1.

5.1. Texture Defect Detection

Texture defect detection can be defined as the process of determining the location and/or the extent of a collection of pixels in a textured image with remarkable deviation in their intensity values or spatial arrangement with respect to the background texture.

The defect detection system used in the experiments consists of two stages: (i) The feature extraction part utilizes prediction error filtering of the textured images and calculates the reflection coefficients of the twelve parameter lattice filter using the proposed algorithm. (ii) The detection part is a Mahalanobis distance classifier being trained by defect-free samples.

The algorithms for each block are provided below:

(i) Feature Extraction:

a) Divide each N x N image into non-overlapping sub-windows S_i of size p x p.

b) Process each sub-window using the twelve-parameter lattice filter and adaptively calculate the reflection coefficients associated for each sub-window

c) Construct the feature vector in terms of the reflection coefficients of the lattice filter for the i-th sub-window S_i

$$\mathbf{s}_i = [k_1^{(1)} k_2^{(1)} k_3^{(1)} \cdots k_{12}^{(1)} \; k_1^{(2)} \; k_2^{(2)} \cdots k_1^{(M)} \cdots k_{12}^{(M)}]^T$$

where $k_{(i)}^{(j)}$ is the i-th reflection coefficient of the j-th stage.

(ii) Detection:

The detection part of the system consists of a learning phase and a classification phase.

1. Learning phase
    a) Given L defect-free N x N fabric images, calculate the feature vectors for each sub-window of the image using the feature extraction scheme given above. Consider these vectors as the true feature vectors and name them as $\mathbf{t}_i$
    b) Compute the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ for the feature vectors $\mathbf{t}_i$.

2. Classification phase
    a) Given a test image of size N x N, calculate the feature vectors $\mathbf{s}_i$'s for each sub-window $\mathbf{S}_i$ using the feature extraction scheme given above.
    b) Compute the Mahalanobis distance h_i between each feature vector $\mathbf{s}_i$ and the mean vector $\boldsymbol{\mu}$

$$h_i = (\mathbf{s}_i - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{s}_i - \boldsymbol{\mu}) \tag{5.1}$$

    where $\boldsymbol{\mu}$ is the mean vector and $\boldsymbol{\Sigma}$ is the covariance matrix; both are determined in the learning phase.
    c) Classify a subwindow S_i for which h_i exceeds a threshold value α as defective; else identify it as nondefective. i.e.

$$S_i = \begin{cases} \text{defective} & \text{if} \quad h_i > \alpha \\ \text{nondefective} & \text{otherwise} \end{cases}$$

The detection part of the system consists of a learning phase and a classification phase. In the learning phase, feature vectors for each sub-window of L defect-free images are calculated. Their mean is found and the mean vector is considered as the true feature vector. In the classification phase, feature vectors corresponding to the sub-windows of a test image are calculated and the Mahalanobis distance h_i between feature vector of each sub-window and the true vector is computed. Each sub-window S_i for which h_i exceeds a threshold value α is classified as defective; else identify it is identified as non-defective. The threshold value is determined by the formula

$$\alpha = D_m + \eta(D_q - D_m) \tag{5.2}$$

D_m and D_q are, respectively, the sample median and the upper quartile of the order statistics D_i (distances h_i arranged in ascending order). For an image divided into M subwindows $D_m = (D_{M/2} + D_{M/2+1})/2$ ve $D_q = (D_{M-M/4} + D_{M-M/4+1})/2$. The second term of summation in Eq. (5.2) is the

confidence interval and parameter η is a constant determined experimentally or automatically by methods like cross-validation. η is normally a critical parameter and its choice is a compromize between the number of false alarms and the number of misses [15]. Simulations with FFT- based methods, Markov Random Fields, Principle Component Analysis and Co-occurrence matrices in the context of defect detection have shown that they are sensitive to the choice of η [14], on the other hand it is observed that the proposed IRLSL algorithm is quite insensitive [13]. This may be accounted for the ability of the proposed algorithm in reducing the effects of the outliers since the choice of the η, is a compromise between the false alarm rate and the detection rate in other detect detection methods mentioned above.

Intuitively, the classifier labels the sub-windows with considerable difference from the rest as defective. In calculating the threshold, for an image, the median of the distances of sub-windows from the learned sample in place of mean is used as the mean will not be a reliable measure if there are defective sub-windows.

5.2. Implementation and Experimental Results

For the experimental justification of the algorithm, real fabric images acquired by a CCD camera in a laboratory environment are used [13]. The database consists of 256x256 sized 8-bit long gray level images. Front lighting has been used during the acquisition of the images, that is the camera and the light source are placed on the same side of the fabrics. Texture images we used were taken in a real environment with light and intensity variations, hence they all include realistic noise. The performance of the algorithms are tested under the realistic noise conditions. Each of the acquired images corresponds to 8.53 cm x 8.53 cm fabric with resolution of 3.33 pixels/mm, which is the same resolution required in the factory environment. Effort has been made to include various textures and different types of defects. The defective images used in the experiments may be observed in Fig.3.

Defective and non-defective images are subdivided to non-overlapping sub-windows of size 32 x 32 and the model parameters, namely the reflection coefficients are calculated using the RLSL [7], FLRLS [8] and the proposed IRLSL algorithms. Window size chosen, in scanning the images depends both on the resolution of the camera used for image acquisition and the textural properties of the fabrics as well as how localized the defects are. In the experiments, the highest performance is obtained by using 32x32 sized non-overlapping sub-windows for the original image [13]. In the experiments, parameter η in Eq. (5.2) is chosen as 3. With this value of η, all adaptive lattice filters performed with moderate rate of false alarms and acceptable defect detection capability. In the other cases, either the defect has not been detected or there has been a big false alarm rate.

The IRLSL algorithm is employed using various objective functions ρ in Eq. (4.6). Three different cases are (a) $\rho(s) = 1 - \cos(s)$; (b) $\rho(s) = (1/2)\log(1 + s^2)$ and

(c) $\rho(s) = (1/2)(1-(1-s^2)^3)$; the plots corresponding to those functions are shown in Fig. (2). The parameter λ used in Eqs. (4.9) and (4.10) is chosen to be 0.99 in the experiments. The forgetting factor and the initial value of the covariance matrix for the FLRLS [8] algorithm are chosen as 0.9998 and 0.0003, respectively. The IRLSL algorithms converged in 7 iterations.

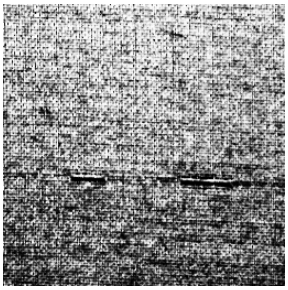

Figure 3a Defect 1

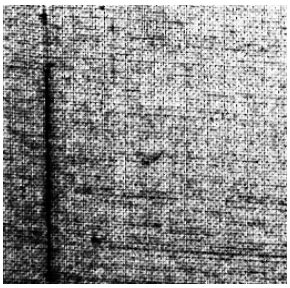

Figure 3b Defect 2

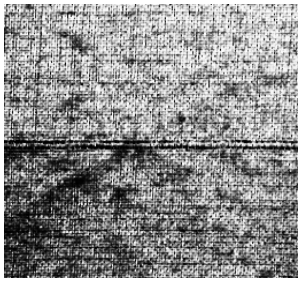

Figure 3c Defect 3

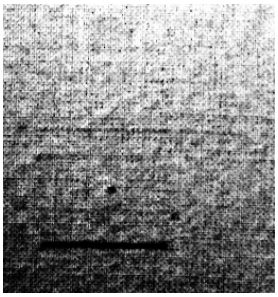

Figure 3d Defect 4

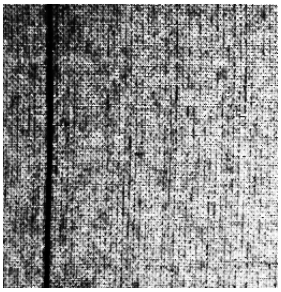

Figure 3 e Defect 5

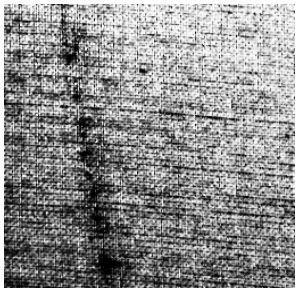

Figure 3f Defect 6

The results are presented in Table II for the RLSL [7], the proposed IRLSL using different objective functions and the FRLRS [8] algorithms, respectively. The correctly labeled defective blocks sum up to the number defined as PP (actually **present** and labeled as **present**). The number of false alarms sum up to the number AP (actually **absent** but labeled as **present**). The undetected defective blocks sum up to PA (actually **present** but labeled as **absent**). This is the number of missed blocks. And finally the number AA represents the number of correctly classified non-defective blocks (actually **absent** and labeled as **absent**).

TABLE II. Results of simulations for different algorithms
For IRLSL-a $\rho(s) = 1-\cos(s)$ **; For IRLSL-b** $\rho(s) = (1/2)\log(1+s^2)$ **;**

for IRLSL-c $\rho(s) = (1/2)(1-(1-s^2)^3)$.

IMAGE	METHOD USED	PP	AP	PA	AA	defective blocks	non-defective blocks	detection ratio
defect 1	RLSL	4	4	3	53	7	57	0.89
	IRLSL-a	2	1	5	56	7	57	0.90
	IRLSL-b	3	0	4	57	7	57	0.93
	IRLSL-c	2	0	5	57	7	57	0.92
	FLRLS	3	8	3	50	7	57	0.83
defect 2	RLSL	0	2	8	54	8	56	0.84
	IRLSL-a	1	4	7	52	8	56	0.82
	IRLSL-b	0	2	8	54	8	56	0.84
	IRLSL-c	2	3	6	53	8	56	0.85
	FLRLS	0	5	6	51	8	56	0.84
defect 3	RLSL	4	1	4	55	8	56	0.92
	IRLSL-a	6	3	2	53	8	56	0.92
	IRLSL-b	6	4	2	52	8	56	0.90
	IRLSL-c	6	0	2	56	8	56	0.96
	FLRLS	4	4	4	52	8	56	0.88
defect 4	RLSL	4	2	0	58	4	60	0.96
	IRLSL-a	4	2	0	58	4	60	0.96
	IRLSL-b	6	4	2	52	4	60	0.90
	IRLSL-c	4	0	0	60	4	60	1.00
	FLRLS	3	5	1	55	4	60	0.91
defect 5	RLSL	8	4	0	52	8	56	0.93
	IRLSL-a	8	4	0	52	8	56	0.93
	IRLSL-b	7	0	1	56	8	56	0.98
	IRLSL-c	8	3	0	53	8	56	0.95
	FLRLS	8	2	0	54	8	56	0.97
defect 6	RLSL	2	0	7	55	9	55	0.89
	IRLSL-a	1	1	8	54	9	55	0.85
	IRLSL-b	2	1	7	54	9	55	0.87
	IRLSL-c	3	0	6	55	9	55	0.90
	FLRLS	2	5	6	51	9	55	0.83

It can be observed from Table II that the IRLSL algorithms in general give better results compared to the RLSL algorithm and the FRLRS algorithm. Among the IRLSL algorithms, the best performance is obtained when an appropriate weight function is used. In our case IRLSL-c gives the best results where the weights corresponding to the objective function $\rho(s) = (1/2)(1-(1-s^2)^3)$ is used. The performance is evaluated in terms of the false alarm rate (the AP column). When the number of false alarms is compared for the RLSL (where no weights are used) and the IRLSL-c, it is observed that the IRLSL-c achieves better or comparable performance with that of RLSL where no weights are used. For comparison purposes, the detection ratio is calculated as the ratio of the truly identified defective and non-defective blocks to the total number of blocks, numerically being equal to (PP+AA)/(defective + non-defective). The experiments on the actual defective images reveal that the

best performance among all the algorithms is given by IRLSL algorithm when weights corresponding to an objective function $\rho(s) = (1/2)(1-(1-s^2)^3)$ are used. With this choice, all the defects are successfully detected with the least number of false alarms.

There is a decrease in the false alarm rate when weights are used. However, the computational complexity is multiplied by a factor of 7 compared with RLSL. This drawback can be eliminated using VLSI technology.

VI. CONCLUSIONS

In this work, a 2-D IRLSL algorithm, which is robust to outliers, is introduced to handle the adaptive defect detection problem in textured images. The proposed algorithm is a novel method that combines concepts from robust regression theory and 2-D recursive least squares lattice algorithm. The algorithm is developed for the twelve-parameter 2-D lattice filter structure that is the most general structure in the sense that no spectral symmetry assumptions are imposed on the input data. However with small modifications, this algorithm can easily be applied to various 2-D lattice structures. Success of the algorithm is verified by computer examples employing images acquired from real textile products containing various defects. Satisfactory results, in terms of defect detection ratio, are obtained with the proposed IRLSL algorithm. The IRLSL algorithm reduced the false alarm rate, considerably which demonstrates the importance of the proposed algorithm in reducing the effect of the outliers that generally correspond to false alarms in classification of textures as defective or nondefective.

Through this work, it has been demonstrated that adaptive 2-D lattice filters can be used in the context of texture analysis in a supervised manner. Since this technique is computationally intensive, use of special hardware might be necessary for real time application of the technique.

APPENDIX

Robust Regression

The classical method of least-squares is known to be unreliable when there are outliers in the observations (these may result from non-normal errors) and a way of making the solution robust is by minimizing a sum of less rapidly increasing function of the residuals:

$$J = \sum_{i=1}^{t} \rho(\varepsilon_i / d) \qquad \text{(A.1)}$$

where d is a robust estimator of scale and ε_i's are the errors defined as

$$\varepsilon_i = y_i - (\hat{c}_0 + \sum_{j=1}^{p} x_{ij} c_j) \qquad i = 1,\ldots,n \qquad \text{(A.2)}$$

where y_i's are the element of n –by- 1 vector of responses , namely **y**, x_{ij} 's are the elements of n-by-p matrix of known predictor or independent variables, namely **X,** ε_i's are the elements of n-by-1 vector of errors **ε** and he c_i's are the element of p –by- 1 unknown parameter vector **c**. In order to solve for the unknown parameters, the first partial derivatives of Eq. (A.1) with respect to the elements of vector **c** will be equated to zero, this is equivalent to finding the solution to *p* equations

$$\sum_{i=1}^{n} \psi \left(\frac{y_i - (\hat{c}_0 + \sum_{j=1}^{p} x_{ij} c_j)}{d} \right) x_{ik} = 0 \qquad j = 1, \ldots p \tag{A.3}$$

where $\psi = \rho'$., i.e. the first partial derivative of ρ with respect to elements of vector **c.**

If w_i's are defined as follows.

$$w_i = \psi(\varepsilon_i/d)/(\varepsilon_i/d) \tag{A.4}$$

then this is clearly the set of linear equations that must be solved for robust least squares whose solution is given by

$$\hat{\mathbf{c}} = (\mathbf{X^T W X})^{-1} \mathbf{X^T W y} \tag{A.5}$$

In the above equation **W** is the diagonal weight matrix with elements w_i.

An iterative algorithm for solving Eq. (A.3) is given as follows [10]:

1. t:=0; t is the number of iterations, the notation ":=" is used for "is defined to be".
2. $w_i = 1, \qquad i=1,\ldots,n$
3. Define a diagonal matrix $\mathbf{W}^{(t)}$ with $w_i^{(t)}$ as its i-th diagonal element.
4. Solve the following equation for $\hat{\mathbf{c}}^{(t+1)}$:

$$\left(\mathbf{X^T W}^{(t)} \mathbf{X}\right) \hat{\mathbf{c}}^{(t+1)} = \mathbf{X^T W}^{(t)} \mathbf{y}$$

5. $$\varepsilon_i^{(t+1)} := y_i - (\hat{c}_0^{(t+1)} + \sum_{j=1}^{p} \hat{c}_j^{(t+1)} x_{ij}) \qquad i = 1,\ldots,n \tag{A.6}$$

where ε_i's are the residuals based on the least squares estimates.

6. $$d := median\left|\varepsilon_i^{(t+1)} - median(\varepsilon_i^{(t+1)})\right| / 0.6745 \tag{A.7}$$

where the numerator *d* is called the median of the absolute deviations. The number 0.6745 is chosen because then $d \approx \sigma$ if data length n is large and if the sample arises from a normal distribution.

7. $$w_i^{(t+1)} := \psi\left(\frac{\varepsilon_i^{(t+1)}}{d}\right) \Big/ \left(\frac{2\varepsilon_i^{(t+1)}}{d}\right) \quad \text{if} \quad \varepsilon_i^{(t+1)} \neq 0 \qquad i = 1,\ldots,n, \tag{A.8}$$

$$w_i^{(t+1)} := \rho'' / 2 \qquad \text{otherwise} \qquad i=1,\ldots,n; \tag{A.9}$$

8. If $\left\|\hat{\mathbf{c}}^{(t+1)} - \hat{\mathbf{c}}^{(t)}\right\| \le \delta$ where δ is a predetermined tolerance, go to 11.
9. t:=t+1, go to 3.
10. Estimate $\mathbf{c}$ by $\hat{\mathbf{c}}^{(t+1)}$.

The process continues until successive iterates on both **c** and d have converged to desired accuracy. Huber and Dutter have shown that this algorithm always converges due to the special nature of the objective function defined by Eq. (A.1) for certain ρ functions[16].

ACKNOWLEDGEMENTS

We would like to thank Altınyıldız A.Ş. for providing the defective fabrics.

REFERENCES

[1] Parker, S. R. and A.H.Kayran, "Lattice Parameter Autoregressive Modeling of Two-Dimensional Fields Part I: One quarter-plane case", *IEEE Trans. Acoust., Speech and Sig. Proc.,* vol. 32, pp. 872-885, Feb. 1984.

[2] Ertüzün, A., A. H. Kayran and E. Panayırcı, "An Improved 2-D Lattice Filter and Its Entropy Relations", *Signal Processing*, vol. 28, pp. 1-24, July 1992.

[3] Ertüzün, A., A. H. Kayran and E. Panayırcı, "Further Improved 2-D Lattice Filter Structure Employing Missing Reflection Coefficients", *Circuits, Systems and Signal Process.,* vol.14, pp. 473-494, 1995.

[4] Moro, H., T. Watanabe, A. Taguchi and N. Hamada, "On the Adaptive Algorithm and its Convergence Rate Improvement of 2-D Lattice Filter", *Proc. IEEE Int. Symp. Circuits and Systems,* pp. 430-434, 1988.

[5] Youlal, H., M. Janati-I and M. Najim, "Two-Dimensional Joint Process Lattice for Adaptive Restoration of Images", *IEEE Trans. Image Proc.* vol.1, pp.366-378, July 1992.

[6] Meylani, R., S. Sezen, A. Ertüzün, and Y. Istefanopulos, "LMS and Gradient Based Adaptation Algorithms for the Eight-Parameter Two-Dimensional Lattice Filter", *Proc. European Conf. Circuit Theory and Design,* İstanbul, Turkey, Aug. 1995, pp.741-744,

[7] Ffrench, P. A., Z. A. Zeidler, and W. H. Ku, "Enhanced Detectability of Small Objects in Correlated Clutter Using an Improved 2-D Adaptive Lattice Algorithm", *IEEE Trans. Image Proc.*, vol. 6, March 1997, pp.383-397.

[8] Liu, X. and M. Najim, "A Two-Dimensional Fast Lattice Recursive Least Squares Algorithm", *IEEE Trans. on Signal Proc.,* Vol. 44, No. 10, p. 2557-2567, October 1996.

[9] Haykin, S., *Adaptive Filter Theory,* Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[10] Erçil, A., *Robust Ridge Regression*, Research Publication, General Motors Research Laboratories, GMR-5349, 1986.

[11] Huber, P. J.,*Robust Statistics,* John Wiley, 1981.

[12] Green, J., *Iteratively Reweighted Least Squares for Maximum Likelihood Estimation, and some Robust and Resistant Alternatives*, J.R. Statist.Soc. B, Vol. 46, No. 2, P 149-192, 1984.

[13] Meylani, R., *Texture Analysis Using Adaptive Two-Dimensiona l Lattice Filters*, M.S. Thesis, Boğaziçi University, İstanbul, Turkey, 1997.

[14] Özdemir, S., A. Baykut, R. Meylani, A. Erçil and A. Ertüzün, "Comparative Evaluation of Texture Analysis Algorithms for Defect Inspection of Textile Products", in *Proc. Int. Conf. on Computer Vision and Pattern Recognition*, Aug. 14-17, 1998, Brisbane, Australia, pp.1738-1740.

[15] Latif-Amet, A., A. Ertüzün and A. Erçil, ‘‘An Efficient Method for Texture Defect Detection: Subband Domain Co-Occurrence Matrices”, *Image and Vision Computing*, vol. 18, 2000, pp.543-553.

[16] Huber, P.J. and R. Dutter, *Numerical Solution of robust regression problems*, COMPSTAT 1974 Proc. Symposium on Computational Statistics, G. Brushmann, ed., Physike Verlag, Berlin, pp. 165-172.