Real-time portable system for fabric defect detection using an ARM processor

J.A. Fernandez-Gallego[a], J.P. Yañez-Puentes[a], B. Ortiz-Jaramillo[a], J. Alvarez[a], S.A. Orjuela-Vargas[a,b] and W. Philips[b]

[a]Universidad Antonio Nariño, Cra 10 N 17 - 35, Ibagué, Colombia;
[b]Gent University, TELIN-IPI-IBBT, St-Pietersnieuwstraat 41, B-9000 Gent, Belgium;

ABSTRACT

Modern textile industry seeks to produce textiles as little defective as possible since the presence of defects can decrease the final price of products from 45% to 65%. Automated visual inspection (AVI) systems, based on image analysis, have become an important alternative for replacing traditional inspections methods that involve human tasks. An AVI system gives the advantage of repeatability when implemented within defined constrains, offering more objective and reliable results for particular tasks than human inspection.

Costs of automated inspection systems development can be reduced using modular solutions with embedded systems, in which an important advantage is the low energy consumption. Among the possibilities for developing embedded systems, the ARM processor has been explored for acquisition, monitoring and simple signal processing tasks. In a recent approach we have explored the use of the ARM processor for defects detection by implementing the wavelet transform. However, the computation speed of the preprocessing was not yet sufficient for real time applications.

In this approach we significantly improve the preprocessing speed of the algorithm, by optimizing matrix operations, such that it is adequate for a real time application. The system was tested for defect detection using different defect types. The paper is focused in giving a detailed description of the basis of the algorithm implementation, such that other algorithms may use of the ARM operations for fast implementations.

Keywords: `ARMv4`, defect detection, embedded system, floating point, texture analysis, wavelet transform

1. INTRODUCTION

The modern textile industry aims to produce textiles as little defective as possible since the presence of defects can decrease the final price of products from 45% to 65%.[1] High levels of production require acceptance sampling plans consisting in assessing the quality of fibers or groups checking a certain portion of samples. Detection of defects requires to localize the defective regions, which are categorized into degrees of importance such as major, minor and critical flaws.[2] Usually, these defect detection tasks are performed by visual human inspection. Automated visual inspection (AVI) systems, based on image analysis, have become an important alternative for replacing traditional inspections methods that involve human tasks. An AVI system gives the advantage of repeatability when implemented within defined constrains, offering more objective and reliable results for particular tasks than human inspection.

There exist AVI systems to detect defects in cloths and fabrics.[3] Companies report more effectiveness and objectiveness using AVI than human systems. Reports show that AVI systems increase from 70% to 80% in terms of defect detection averages.[4] Reproducibility and inspection speed are also reported to be higher using AVI systems. Automation of in-process inspections is the most difficult due to the complexity of the weaving

Further author information: (Send correspondence to Jose Fernandez)
Jose Armando Fernandez Gallego: E-mail: fernandezgallego@gmail.com, Telephone: +57 3112104189

process.[5] AVI system tasks include gauging; location and measurement of known defects; detection of unknown defects; warning functions and mechanisms for stopping processes; defect libraries and quality reports.

The most important part in an AVI system is the core of the central precess unit, i.e., the processor. Currently, the processor is integrated in embedded architectures for reducing the power consumption and reducing the size of the final system. Also, Embedded systems are available as an alternative for applications of low power consumption, portability and reduced size devices.[6] However, the embedded systems have limited resources compared with conventional computing systems, e.g., limited RAM, speed of the processor, cache memory, number of registers, among others. Despite this, embedded systems are very popular in a wide range of applications such as telecommunications, process control, smart systems, entertainment, among others, mostly because their portability capabilities.[7]

Particularly, the ARM processors are very popular in embedded systems because they provide low power consumption, low cost, portability and high computation speed that is required into specific application devices. Also, this popularity is due to the fact that every device with ARM processors is compatible in binary code with any processor of that family.[8] On the other hand, the low power consumption and portability are achieved by using less transistors than similar technologies.[9] Furthermore, the RISC design strategy provides higher computation speed because it is based on the fact that a simplified instruction set can perform faster executions of the process.[7] Also, that computation speed is supported by using pipeline co-processors architecture.[10]

We present an improvement of the computation of wavelet transform for defect detection in the `Zeus Epic` board into `Windows CE 5.0` operating system. This improvement was achieved by optimizing matrix operations, such that it is adequate for a real time application. Also, as the ARM included into the `Zeus Epic` board has not an efficient matrix manipulation module, we perform a mathematical modification of the matrices for an efficient implementation of the algorithm.

This paper is focused in giving a detailed description of the basis of the algorithm implementation, such that other algorithms may use of ARM operations for fast implementations. The results shows that classification rates are similar to developments in conventional computers with an acceptable computational time for real time applications.

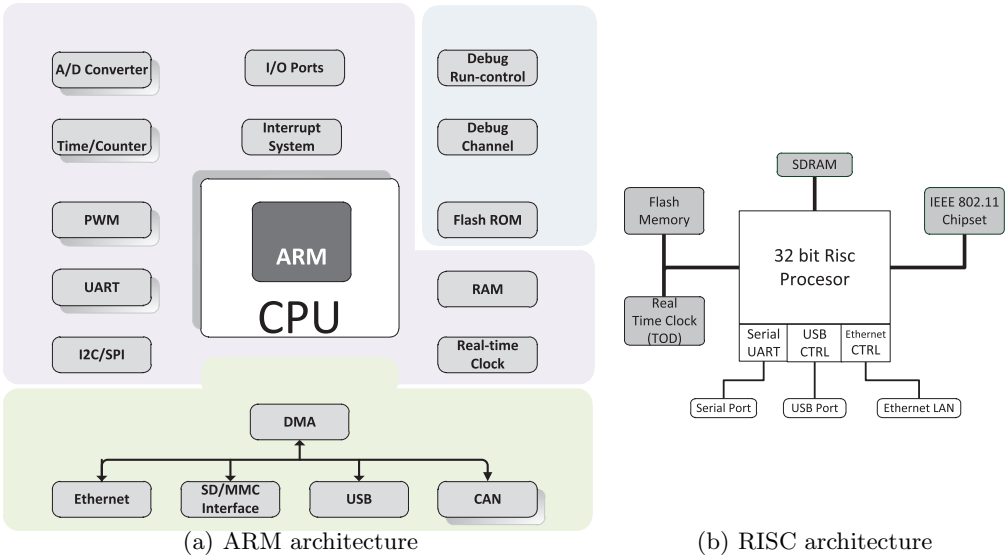
The paper is organized as follows. In Section 2 we discuss the architecture of ARM devices. Also, in the same Section we explain how discrete wavelet transform is optimized by means of matrix operations. Furthermore, the proposed algorithm is also provided in the same Section. In Section 3 we discuss the experimental setup for defect detection by using Haralick descriptors and the algorithm implemented into the ARM device. In Section 4 we report the results and discuss the findings. Finally, in Sections 5 conclusions are drawn and future work is proposed.

2. MATERIALS AND METHODS

2.1 ARM devices

The ARM (Acorn RISC Machine) is a general purpose processor used in a wide range of applications in embedded systems such as cellular telephones, pagers, VCRs, camcorders, thermostats, automated supermarket stockers, computerized inventory control devices, printers, portable video games, TV set-top boxes, among others.[11] These kind of processors have the characteristic that all posses the same basic core for all families. This characteristic provides the advantage of using the same instruction set for all the ARM devices. Currently, these devices are widely included in general purpose systems (covering at least the 75% of the daily used devices).

The `ARMv4` posses a type SoC (System on Chip) 32 bit processor with a RISC (Reduced Instruction Set) architecture from strong ARM family (see Figure 1). Also, the `ARMv4` has several I/O chipsets such as video output, PS/2, USB controller, Ethernet, SD DIO disk, among others. These I/O chipsets gives the possibility of integrating all the devices in only one main board. Furthermore, the `ARMv4` possesses the following characteristics:



(a) ARM architecture (b) RISC architecture

Figure 1. ARM and RISC architectures.

a) a five stage pipeline design for sequential instructions, b) 31 registers with 32 bits each, c) clock speeds between 166 MHz and 520 MHz, d) 16KB cache memory.

One of the main disadvantages of the ARM, v6 or lower, is that it has no optimized floating point accelerator (FPA). Because of that, the processor cannot operate with floating point numbers limiting the accuracy of the processing results. To cope with this shortcoming, an internal emulator capable to support floating point operations is commonly created. However, this kind of operations are very inefficient because one floating point operation takes ten times longer than an integer operation. On the other hand, the ARM v4 or higher possesses a vector floating point (VFP) unit which is capable of performing soft floating operations, i.e., it can perform operations with a five digits precision according with the standard `IEEE-754`. Usually, the operations in this unit are faster than the operations of the FPA unit.

The ARM processor can be included in a wide range of boards such as `Zeus Epic`, Colibri 600, NEONPBX-A9, among others. Particularly, the `Zeus Epic` board uses a `Windows CE 5.0` operating system. This operating system allows the use of libraries for data processing. Specifically, the image processing library used in `Zeus Epic` with `Windows CE 5.0` is the Imaging Factory library. This library is very inefficient for performing pixel to pixel operations because the API instruction set of the `Windows CE 5.0` needs to be interpreted by a high level compiler before of executing the instruction in the processor. Then, it is preferred to perform the image processing methods without using that library, at least when only an ARM v6 or lower is available.

2.2 Discrete wavelet transform

The wavelet transform can be defined as the inner product of a signal with a set of functions generated from a prototype function by means of translations and scalings of itself.[12] Usually, the discrete wavelet transform (DWT) is defined as a number of convolutions and downsamplings of the analyzed signal. Figure 2 shows the two dimensional DWT, which is a one dimensional DWT across rows and columns. Here, X^0 is a $M \times N$ image, J indicates the level of decomposition, and h and g respectively are a low and a high pass filters. In this decomposition, every level is represented by three images (except for the $J-1$th level of decomposition which have an extra image termed approximation image), i.e., $\left\{X_{LH}^1, X_{HL}^1, X_{HH}^1, \ldots, X_{App}^{J-1}, X_{LH}^{J-1}, X_{HL}^{J-1}, X_{HH}^{J-1}\right\}$, representing details at jth level of decomposition along vertical, horizontal and diagonal direction, respectively.[13]

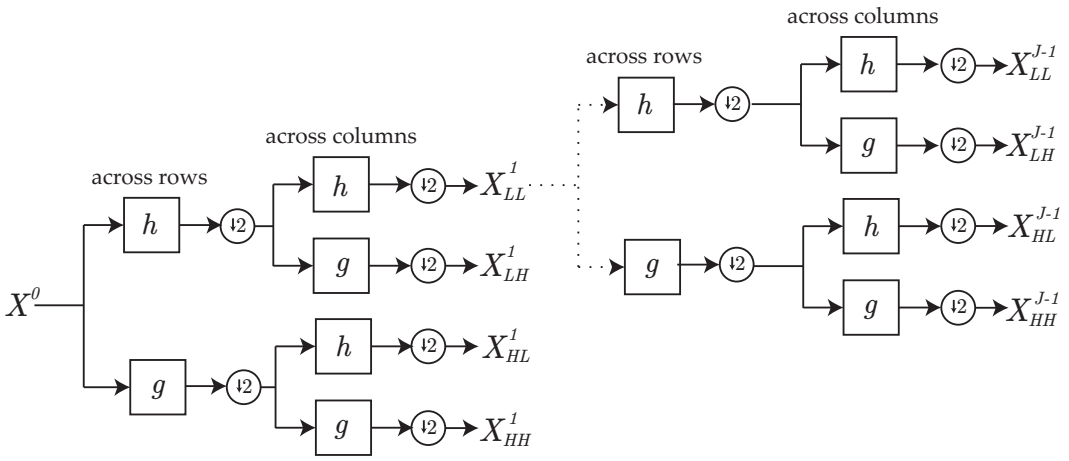


Figure 2. Two-dimensional discrete wavelet transform of J decomposition levels.

It is possible to perform this kind of transformation by matrix multiplication, i.e., the inputs of the filter can be placed into a Toeplitz matrix (Equation (1)).[14]

$$\overline{H}_1 = \begin{bmatrix} h_k & \dots & h_1 & h_0 & 0 & \dots & 0 & 0 \\ 0 & h_k & \dots & h_1 & h_0 & 0 & \dots & 0 \\ 0 & 0 & h_k & \dots & h_1 & h_0 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & h_k \end{bmatrix}, \tag{1}$$

where, $\overline{H}_1$ is the low pass convolution matrix of size $M \times M$. After every convolution it is necessary to perform a downsampling procedure across rows, then the following matrix can be defined as a matrix of size $M/2 \times M$ as

$$H_1 = \begin{bmatrix} h_k & \dots & h_1 & h_0 & 0 & \dots & 0 & 0 \\ 0 & 0 & h_k & \dots & h_1 & h_0 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

similarly it is possible to define a matrix H_2 with the same nature of the transpose matrix of H_1, i.e.,

$$H_2 = \begin{bmatrix} h_k & \dots & h_1 & h_0 & 0 & \dots & 0 & 0 \\ 0 & 0 & h_k & \dots & h_1 & h_0 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}^T,$$

where the size of H_2 is $N \times N/2$. Analogous it is possible to obtain matrices G_1 and G_2 by using the high pass filter g. Then, by using those matrices it is possible to compute directly the coefficients of the wavelet transform as:

$$\begin{aligned} X^{j+1}_{LL} &= H_1 X^j_{LL} H_2 \\ X^{j+1}_{HL} &= G_1 X^j_{LL} H_2 \\ X^{j+1}_{LH} &= H_1 X^j_{LL} G_2 \\ X^{j+1}_{HH} &= G_1 X^j_{LL} G_2 \end{aligned}$$

With this previous operation, an optimized version, available in numerical recipes, of the discrete wavelet transform is obtained. This optimized version is based on processing the convolution and downsampling processes in the same matrix product as shown in Figure 2.

2.3 Proposed algorithm

We explain in the following the algorithm used for implementing the wavelet transform on the ARM device. First at all, the images to be processed are read from a SD memory card. Those images were previously stored. Giving that libraries for reading images using `Windows CE` are computationally expensive, we used a direct method for reading the images using an interpretation of the headers from the file. Thus, we created a reading binary buffer for the reconstruction of the image using the data extracted from the file.

We used the matrix optimization of wavelet transform explained above to perform the processing of the data. Due to the fact that the `ARMv4` has not a matrix manipulation module, we used the bilinear forms of those matrices. For instance, the bilinear form of the Haar wavelet, used in this work, is written as follows:

$$
\begin{aligned}
X_{LL}^{j+1}(m,n) &= \frac{1}{2}\left(X_{LL}^{j+1}(2m-1,2n-1)+X_{LL}^{j+1}(2m-1,2n)+X_{LL}^{j+1}(2m,2n-1)+X_{LL}^{j+1}(2m,2n)\right)\\
X_{HL}^{j+1}(m,n) &= \frac{1}{2}\left(-X_{LL}^{j+1}(2m-1,2n-1)-X_{LL}^{j+1}(2m-1,2n)+X_{LL}^{j+1}(2m,2n-1)+X_{LL}^{j+1}(2m,2n)\right)\\
X_{LH}^{j+1}(m,n) &= \frac{1}{2}\left(-X_{LL}^{j+1}(2m-1,2n-1)+X_{LL}^{j+1}(2m-1,2n)-X_{LL}^{j+1}(2m,2n-1)+X_{LL}^{j+1}(2m,2n)\right)\\
X_{HH}^{j+1}(m,n) &= \frac{1}{2}\left(X_{LL}^{j+1}(2m-1,2n-1)-X_{LL}^{j+1}(2m-1,2n)-X_{LL}^{j+1}(2m,2n-1)+X_{LL}^{j+1}(2m,2n)\right),
\end{aligned}
$$

where, $m = 1,\ldots,M/2$ and $n = 1,\ldots,N/2$. Here, the bilinear form was used with purpose of reducing the computation time.

Noteworthy is that the result of previous transform has floating point operations. In the `ARMv4` there are two ways of performing the floating point operations, one is by using the VFP unit and the other is by using the FPA unit. on the one hand, the VFP unit was optimized in the `ARMv4` for soft floating point operations according with the standard `IEEE-754`. On the other hand, the FPA unit was not optimized in the `ARMv4` for floating point operations. It means that it is necessary to perform the floating point operations by using the VFP unit for saving computational time. To cope with this shortcoming, we implemented an hybrid algorithm to perform operations with integers representing floating point numbers.

Particularly, the proposed hybrid algorithm takes advantage of the `ARMv4` integer operations with the purpose of using the VFP unit.

Figure 3 the flow chart of the proposed algorithm for performing the wavelet transform. In the flow chart the hybrid algorithm consist in using the VFP unit as data processing unit. For doing that we propose to perform a representation change from floating point to integer data by using the following instruction `P=float(int(Q*100.0+K/10))/100.0`. With previous instruction the number termed `Q` (floating point number) is rounded in an automatic way using the `int` function. Afterwards, that number is converted again to a floating point number using the `float` function. Here, we multiply and add two constants (`100.0` and constant `K/10`), this operation is performed because by using this combination the processor, internally, activate the fast math floating point flag which is exclusive of the VFP unit. Thus, the use of the assembler language for this purpose is avoided. This fact can be seen as a BUG of the compiler used for expressing in C this instruction. Noteworthy is that for `K=0,...,5` the resulting number is represented with `K` precision digits. Here, the desire precision is achieved by using $K = 5$ which is enough for obtaining the same result of `Matlab`®. Finally, to preserve the format and the scale of the original variable `Q`, it is necessary to divide by `100.0`.

3. EXPERIMENTAL SETUP

The methodology for defect detection comprises the following stages: a) preprocessing or global image transformation, b) feature extraction, c) feature selection and d) classification stage. Usually, the feature selection and the training of the classifier are performed offline, i.e., the computational load in this stage is carried out by a conventional computer and not by the embedded system. Also, the feature extraction and classification stages

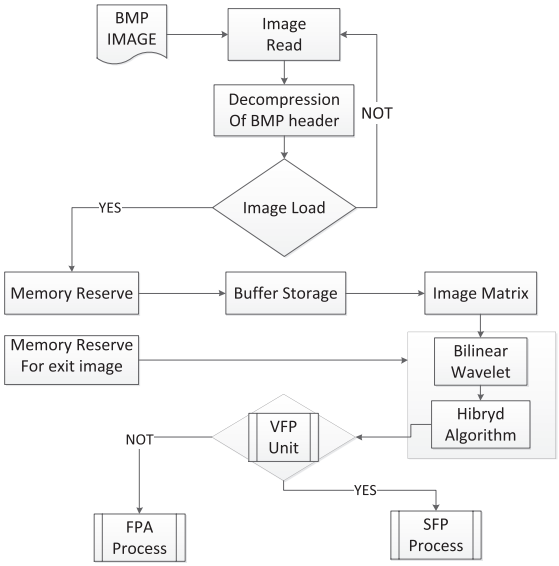
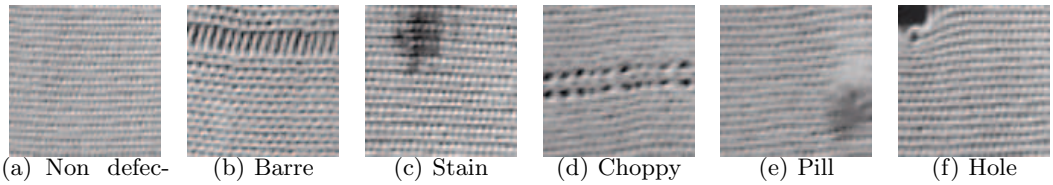


Figure 3. Proposed Algorithm flow chart.

are not critical for the computation speed of the embedded system. Therefore, we only show in this work the time consumption of the preprocessing or global image transformation, which depends on the incoming samples.

3.1 Database and preprocessing

We use 50 images of class C_1 and 10 images for each of the other classes to train the classifier (see Figure 4). The images have been randomly chosen from a set of 1000 images of non defective samples and 2000 images of defective samples, all images with size of 100×100 pixels. We complement the images before applying the Haar wavelet decomposition. This is to obtain features with values around zero as an attempt to reduce the use of resources of the ARM processor by using small numbers in the majority of the calculations. Figure 4 shows examples of a non defective fabric sample together with samples of the five types of defects considered in this work.

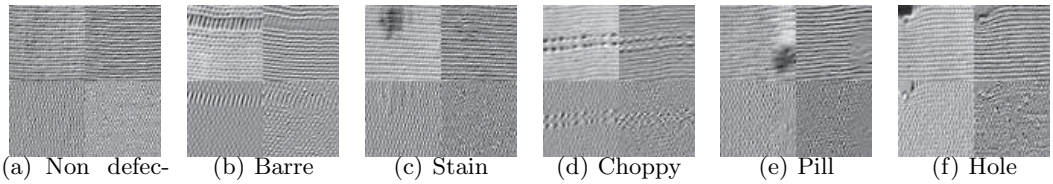


(a) Non defective fabric (b) Barre (c) Stain (d) Choppy (e) Pill (f) Hole

Figure 4. Types of images used in the database.

The preprocessing stage was performed by using the one level wavelet transform with the Haar filter bank as suggested in.[15] Also, this wavelet transform was implemented into the ARM by using the algorithm proposed

in Section 2. Figure 5 shows results of applying the Haar wavelet transform, implemented into the ARM, on examples of a non defective and defective samples. Here the images were normalized for visualization purpose.



(a) Non defective fabric (b) Barre (c) Stain (d) Choppy (e) Pill (f) Hole

Figure 5. Results of applying the Haar wavelet transform on types of images used in the database.

For obtaining the same wavelet coefficients presented on `Matlab`® the algorithm was tuned by using 0.5 as decision rounding point, e.g., if the output value in wavelet transform is 128.5 the final value is 129.

3.2 Feature extraction and classifier

To recognize defective fabrics, we characterize the texture on the decomposed images computing features by applying the cooccurrence matrix technique. Formally, Haralick et al.[16] define a matrix of relative frequencies $P(g_1, g_2|(i,j))$, where g_1 and g_2 are gray level values of two disjoint pixels separate by a displacement (i,j). The number of occurrences of g_1 and g_2, separated by the vector (i,j), contributes to the (g_1,g_2)th entry in the matrix of relative frequencies. Then, a set of features can be extracted from every matrix P with a displacement (i,j). The most common features are energy, entropy, contrast, homogeneity and correlation.[17] In our experiment we used as the displacement vector the set $(i,j) \in \{(0,1),(-1,1),(-1,0),(-1,-1)\}$ as is suggested in.[15,17]

The classes are differentiated using the Mahalanobis distance, which uses the sample mean and the dispersion of the classes.[18] The classifier uses the Mahalanobis distance given by: $d(i,j) = (V_i - m_j)^T S_j^{-1}(V_i - m_j)$, where m_j denotes the sample mean of class j, V_i the input sample to the classifier, S_j^{-1} the covariance matrix of the class j and $d(i,j)$ is the distance between sample i and class j. The sample is associated to the class with minimal distance.

4. RESULTS AND DISCUSSION

The time performance of our algorithm implemented in the `ARMv4` was compared with the wavelet transform implemented on `Matlab`® and `C++` languages. A computer with an `Intel`® `Atom N450` dual core processor of 1.67 GHz and 2 GB of RAM was used for measuring the times into both languages. Also, the algorithms were implemented, in both languages, by using the bilinear form of the Haar transform with the purpose of conserving the same algorithm in the two systems, i.e., the `ARMv4` and the computer.

Table 1. Processing time comparison

Time in seconds			
`Matlab`®	`C++`	`ARMv4` (FPA)	`ARMv4` (VFP)
0.0342	0.0119	0.4625	0.0191

Table 1 shows the time spent by using our algorithm implemented in the `ARMv4` using the FPA and VFP units. Also, the Table shows results for the wavelet transform implemented on `Matlab`® and `C++`. This values are the time spent for processing one sample. As was stated before, we only measure the time consumed by the acquisition and the wavelet transform.

The results shows that time consumed by the `ARMv4` with VFP unit is at least 20 times lower than the time consumed by the `ARMv4` with FPA. This may due to the incorporation of the fast math floating point flag used in our proposed algorithm. Also, the Table shows that the time consumed by our algorithm is not the lower value.

However, the time consumed by our proposed algorithm still enough for online applications.

We validate the classifier using 100 images. The results are shown in Table 2 in terms of correct classification using a confusion matrix. The (i, j)th element of that matrix is the number of patterns of class j that are classified as class i by the classifier. This measure is very useful for identifying how the error rate is decomposed. Therefore, the sum of the columns in the table must be 100% but the sum of the rows may differ.

Table 2. Classification results within a confusion matrix.

		Predicted class					
		No defect	Barre	Stain	Choppy	Pill	Hole
True class	No defect	98%	2%	0%	0%	0%	0%
	Barre	0%	96.7%	0%	0%	0%	3.3%
	Stain	0%	0%	85%	5%	5%	5%
	Choppy	0%	14.3%	0%	71.4%	14.3%	0%
	Pill	0%	0%	0%	0%	100%	0%
	Hole	0%	0%	0%	0%	0%	100%

Table 2 shows the confusion matrix obtained after applying a classifier in the validation set. The results show that only 2% of non defective validated samples where incorrectly classified. The type of defect that was more incorrectly was the defect type choppy. We believe this is because the size of defect change from sample to sample and therefore we may include another constrain, such as size invariance, for its correct recognition.

5. CONCLUSIONS

In this work we use a standard optimized implementation of the wavelet transform, available at numerical recipes, in an ARM processor by using matrix operations. We found an algorithm capable of performing floating point operations in the `ARMv4` in an efficient way. Then by combining both results we found an optimization of the time consumed by the wavelet transform in low cost devices. This results may help in the reduction of costs of automated inspection systems using modular solutions with embedded system, in which an important advantage is the low energy consumption. Among the possibilities for developing embedded systems, we have explored the use of an ARM processor for defects detection by implementing the wavelet transform. The results suggest that accurate systems for specific applications can be developed with the ARM processors. Also, the results shows that the proposed hybrid algorithm provides an improvement of the computational time which is enough for online applications.

REFERENCES

[1] R. Stojanovic, P. Mitropulosa, C. Koulamas, Y. Karayiannis, S. Koubias, and G. Papadopoulos, "Real-time vision-based system for textile fabric inspection," *Real-Time Imaging* **7**, pp. 507–518, 2001.

[2] T. Atilgan, "Acceptable quality levels in the textile sector and their effect on the level of competition," *Fibres & Textiles in Eastern Europe* **15**, pp. 16–23, 2007.

[3] A. Dockery, "Automatic fabric inspection: Assessing the current state of the art." online, 2001.

[4] A. Islam, S. Akhter, and T. Mursalin, "Automated textile defect recognition system using computer vision and artificial neural networks," *Transactions on Engineering, Computing and Technology* **13**, pp. 1–6, 2006.

[5] A. Kumar, "Computer-vision-based fabric defect detection: A survey," *IEEE Transactions on Industrial Electronics* **55**, pp. 348–363, 2008.

[6] Z. Pang, H. Yang, and G. Gao, "An embedded image processing system based on wince," in *Second International Symposium on Computational Intelligence and Design*, pp. 355–358, 2009.

[7] C. Hallinan, *Embedded Linux Primer: A Practical Real-World Approach*, Prentice hall, Boston, United States, 2006 (2st edition).

[8] ARM, "Arm product overview." online, 2000.

[9] S. Roy, N. Ranganathan, and S. Katkoori, "Compiler-directed leakage reduction in embedded microprocessors," in *Proceedings of the IEEE international conference on Computer design*, pp. 35–40, 2009.

[10] C. Arandilla, J. Constantino, A. Glova, A. Ballesil-Alvarez, and J. Reyes, "High-level implementation of the 5-stage pipelined arm9tdm core," in *IEEE Region 10 Conference TENCON*, pp. 1696–1700, 2010.

[11] R. Wang and S. Yang, "The design of a rapid prototype platform for arm based embedded system," *IEEE Transactions on Consumer Electronics* **50**, pp. 746–751, 2004.

[12] P. Van-Fleet, *Discrete Wavelet Transformation an Elementary Approach with Applications*, Wiley-Interscience, Canada, 2008 (1st edition).

[13] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*, Academic Press, Burlington, MA 01803, United States, 2009 (3rd edition).

[14] M. Frazier, *An Introduction to Wavelets Through Linear Algebra*, Springer-Verlag, New York, United States, 2001 (2nd edition).

[15] J. Fernandez, S. Orjuela, J. Alvarez, and W. Philips, "Fabric defect detection using the wavelet transform in an arm processor," in *Proceedings of the SPIE on Electronic Imaging, Image Processing: Machine Vision Applications V*, pp. 83000N–1–83000N–8, 2012.

[16] R. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics* **3**, pp. 610–621, 1973.

[17] T. Randen and J. Husy, "Filtering for texture classification: A comparative study," *IEEE Transactions On Pattern Analysis And Machine Intelligence* **21**, pp. 291–310, 1999.

[18] P. Mahalanobis, "Normalization of statistical variates and the use of rectangular co-ordinates in the theory of sampling distributions," *Sankhay* **3**, pp. 1–40, 1937.