nal 1D DWT and 1D IDWT which benefit from the relation between filter coefficients together with the decimation or interpolation process involved, thus halving the number of LUTs required per octave. Figs. 1 and 2 illustrate the proposed RNS-based 1D DWT and 1D IDWT architectures for eight-tap wavelet filters. These consist of a set of registers and multiplexers, N LUTs and two modular adder trees built with $N-1$ modulo adders or subtractors each. Thus, the N LUTs required by the lowpass filter path are shared to produce both the lowpass and the highpass filter products in even and odd cycles, respectively. In accordance with the decimation or interpolation process involved in the 1D DWT or 1D IDWT, these products are added by two synchronous modular adder trees clocked at half of the sampling rate by the out-of-phase clocks CLK1 and CLK2, so that the two half the sampling rate approximation and detail sequences, $a_n^{(i)}$ and $d_n^{(i)}$, or double the input rate approximation sequence $\hat{a}_n^{(i-1)}$ can be obtained.

Simulation and results: Two's complement arithmetic and RNS versions of the 1D DWT and 1D IDWT architectures proposed were implemented over FPL (field-programmable logic) devices to assess the hardware complexity and performance. Altera FLEX10K [5] devices were used. Modern FPL device families offer a synergy with RNS-based architectures since modular adders can benefit from short carry chains and pipelining while modulo multipliers, based on small synchronous LUTs, provide a considerable throughput increase over binary multipliers. One- and two-octave RNS and binary 2's complement architectures were synthesised using VHDL to compare parameters such as the area and performance. Eight-bit input samples and ten-bit filter coefficients were assumed, so that one and two octaves required 21 and 34 bit dynamic ranges, respectively. Table 1 shows the results obtained for six-, seven- and eight-bit RNS channels and for binary 2's complement implementations over grade-4 speed FLEX10K devices. The hardware requirements were assessed in terms of the number of LEs (logic elements) and EABs (embedded array blocks) while the performance was evaluated in terms of the register-to-register maximum delay path. The performance advantage of the RNS structures when compared to equally pipelined traditional binary 2's complement systems was up to 23.45 and 96.58% for the proposed one- and two-octave schemes, respectively, assuming 6 bit RNS channels. It is immediately apparent that the advantage of the proposed RNS architectures over the corresponding binary systems increases with the dynamic range, since it is only necessary to add more channels to support the extended word width. This does not affect the overall throughput, which is only limited by LUT latency.

Conclusion: RNS-FPL architectures for the orthogonal 1D DWT and 1D IDWT have been proposed which implement the analysis and synthesis filter banks using the relation between the filter coefficients and the decimation or interpolation process involved to halve the number of LUTs required. They show 23.45 and 96.58% performance improvements for one- and two-octave implementations, respectively, when compared to binary 2's complement architectures.

Acknowledgment: The authors were supported by the Dirección General de Enseñanza Superior (Spain) under project PP98-1354. The CAD tools and supporting material were provided by Altera Corp. under the Altera University Program.

© IEE 2000 *3 April 2000*
Electronics Letters Online No: 20000896
DOI: 10.1049/el:20000896

J. Ramírez, A. García, L. Parrilla and A. Lloris (*Departmento de Electrónica y Tecnología de Computadores, Campus Universitario Fuentenueva, 18071 Granada, Spain*)

E-mail: jramirez@ditec.ugr.es

P.G. Fernández (*Departamento de Ingeniería Eléctrica, Escuela Politécnica Superior, 23071 Jaén, Spain*)

References

1 MEYER-BAESE, U., BUROS, J., TRAUTMANN, W., and TAYLOR, F.: 'Fast implementation of orthogonal wavelet filterbanks using field-programmable logic'. 1999 Int. Conf. Acoustics, Speech and Signal Processing, Mar. 1999
2 VISHWANATH, M., OWENS, M., and IRWIN, M.J.: 'VLSI architectures for the discrete wavelet transform', *IEEE Trans.*, 1995, **CAS-II-42**, (5), pp. 305–316
3 PARHI, K.K., and NISHITANI, T.: 'VLSI architectures for discrete wavelet transforms', *IEEE Trans.*, 1993, **VLSI-1**, pp. 191–202
4 RAMÍREZ, J., GARCÍA, A., FERNÁNDEZ, P.G., and LLORIS, A.: 'An efficient RNS architecture for the computation of discrete wavelet transforms on programmable devices'. X European Signal Processing Conf. EUSIPCO 2000, Sept. 2000
5 Altera Corporation: '1998 Data Book'. 1998

Segmentation approach using local image statistics

A.P. Mendonça and E.A.B. da Silva

An efficient segmentation technique is described that uses local application of the K-means method with feature vectors based on image statistics. It requires little computational effort and works well for different classes of images, comparing favourably to others in the literature.

Introduction: The importance of image segmentation techniques cannot be overemphasised. Image segmentation applications range from computer vision to image compression. The MPEG-4 standard, allowing the division of an image into video object planes, has increased the need for the development of image and video segmentation techniques [1].

In this Letter, we propose a simple algorithm that processes a natural image and generates a reduced set of regions, based on the local analysis of the mean and variance in a pixel's neighbourhood. The image is scanned sequentially and, by the end of the scanning process, an initial segmentation is obtained. This is then refined by carrying out region analysis, where regions with large areas are identified.

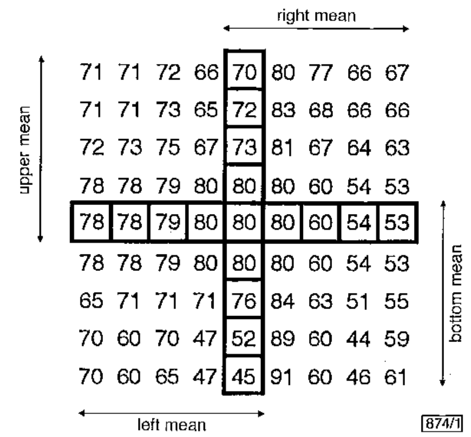

Fig. 1 *Example of calculation of CDM*
Numbers represent pixel values and central point is target pixel
Left mean is 79, right mean is 65.4, upper mean is 75 and bottom mean is 66.6. CDM is therefore 79

Local feature choice: The choice of image features heavily influences the performance of a segmentation technique. In this Letter, we investigate the use, for each pixel, of a feature vector composed of two features that we refer to as the closest directional mean (CDM) and smallest directional variance (SDV). An example of how the CDM is calculated is shown in Fig. 1, where the CDM associated with the central point is 79, relating to the mean in the left direction, because it is the directional mean nearest to 80, the current pixel value. Simulations confirm that the CDM is effective and accurate in determining the boundaries of a smooth region. We also need the SDV in order to segment regions with textures or edges. As with the CDM, the SDV is computed using the pixels shown in Fig. 1.

The use of the CDM and SDV can be justified by considering a pixel close to a region boundary. If the region is reasonably smooth, in general at least one of the four sets of pixels indicated in Fig. 1 will belong to the same region as the central pixel. Therefore, its mean tends to be close to the central pixel value and the CDM characterises the pixel belonging to this region. Similarly, the SDV effectively characterises regions with textures or large intensity variations. Experiments have indicated that using more directions in the computation of the CDM and SDV does not provide significant gains.

Image scanning: To find regions with similar features, an image of dimensions $K_1Np \times K_2N_p$ was divided into $K_1 \times K_2$ blocks of $Np \times N_p$ pixels. We denote each $2N_p \times 2N_p$ set of four $N_p \times N_p$ blocks a superblock. For each superblock, the K-means algorithm [2] (three iterations) was applied using at most four initial centres. Each resulting region was represented by the pixel the feature vector of which had the smallest Euclidean distance to the mean feature vector in that region. If the Euclidean distance of the feature vectors of two regions was smaller than a threshold '*LimCen*', these two regions were merged.

The $N_p \times N_p$ blocks were scanned as follows:

(i) The image was scanned by rows of superblocks, from left to right, starting from the bottom. The initial centres of the first superblock were chosen to be the feature vectors of its four corners.

(ii) For the first row, each new superblock was scanned with an overlap of two $N_p \times N_p$ blocks from the previously scanned superblock. For these superblocks, the K-means algorithm was applied with the following restrictions: two initial centres were chosen as the centres of the two most populated regions that crossed the right boundary of the two $N_p \times Np$ overlapped blocks. Another centre was chosen as the pixel having, in feature space, the largest harmonic mean of the Euclidean distances to the two previous centres. The last centre should have the largest harmonic mean of the distances in feature space to the other three centres. It is important to note that the only pixels from the two overlapped $N_p \times N_p$ blocks that should be included in the K-means algorithm are those corresponding to the first two initial centres.

(iii) Each new row of superblocks had a vertical overlap of two $N_p \times N_p$ blocks with the previous superblock. For the first superblock, the only pixels from the two overlapped blocks that were included in the K-means algorithm were those belonging to the two most populated regions of the superblock from the previous row that crossed the upper boundary of the overlapped block. The remainder of the process is as in step (ii).

(iv) For the subsequent superblocks, there was an overlap of 3 $N_p \times N_p$ blocks with the previously scanned ones. The initial centres were the centres of the three most populated regions that crossed the boundaries of the overlapped blocks that touched the left and bottom boundaries of the non-overlapped block. The fourth initial centre was chosen as the pixel with the largest harmonic mean, in feature space, of the Euclidean distances to the other three.

(v) By scanning the whole image, segmentation was achieved.

Segmentation refinement: After image scanning, the mean feature vector of each region was evaluated. We denoted this '*CodeBook*[α]', where α represents a region. All image pixels X_{ij} were then re-scanned in order to refine the segmentation. Let α and β be distinct regions. We define their boundary by the equation

$$X_{ij} \in \alpha \quad X_{mn} \in \beta \qquad |i-m| \le 1 \quad |j-n| \le 1 \qquad (1)$$

Considering $v(X_{ij})$ the feature vector of a pixel X_{ij}, if the conditions

$$\|v(X_{ij}) - v(X_{mn})\| \le LimCen \qquad (2)$$

$$\|CodeBook[\alpha] - CodeBook[\beta]\| \le LimCen \qquad (3)$$

are satisfied for any pixel of the boundary (eqn. 1), then the regions α and β are merged.

It is possible that spatially distant regions separated by a slow intensity variation may be segmented as the same region. To avoid this type of problem, each region was submitted to an analysis process. In this process, we began with the two most distant pixels in feature space, X_{max} and X_{min}. If $\|v(X_{max}) - v(X_{min})\| > T$, the two new regions associated with X_{max} and X_{min} are defined using the following equations, where T is a tolerance factor:

$$\|v(X_{ij}) - v(X_{min})\| \le T \qquad (4)$$

$$\|v(X_{ij}) - v(X_{max})\| \le T \qquad (5)$$

Pixels X_{ij} that satisfy either eqn. 4 or eqn. 5 and are connected will define new regions. If $\|v(X_{min}) - v(X_{max})\| \le 2T$, the value of T in eqns. 4 and 5 is substituted by $1/2\ \|v(X_{min}) - v(X_{max})\|$. This procedure is repeated until a condition of stability is achieved in the region map.

The above procedures were applied with the restriction that regions with less than P_{min} pixels were not allowed to be segmented. If, at the end of the process, regions with less than P_{min} pixels existed, they were eliminated and each pixel within each of these regions was assigned to the nearest neighbouring region.

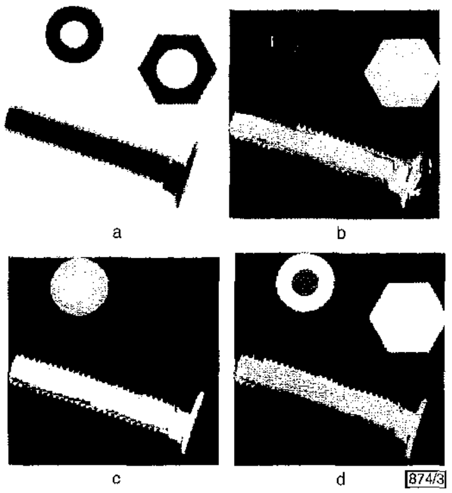

Fig. 2 *Simulation with 'Hardware' image*

a Original image
b Segmented image with 44 regions
c Second segmented image with 22 regions
d Image with added zero mean white uniform noise of variance 8.3 segmented in 21 regions using same parameters as in *c*

Simulation results and conclusions: The proposed segmentation algorithm was evaluated using the grey scale image 'Hardware' (Fig. 2). The dynamic ranges of the CDM and SDV were normalised to 1 and '*LimVar*', respectively. We used the following set of parameters: $N_p = 8$, $LimCen = 0.07$, $LimSat = 3000$, $LimVar = 0.14$, $P_{min} = 20$, $T = 0.25$. The 'Hardware' image was segmented into 44 regions. Making '*LimCen*' = 0.14 and $T = 0.4$, a new segmentation resulted with 22 regions. Fig. 2*d* shows the segmentation of 'Hardware' with added noise. Note that the algorithm is quite robust to noise addition.

The computational cost of the proposed algorithm is very low and it can be easily implemented in a digital signal processor for real-time applications. The only procedure that involves a high computational cost is the separation of regions joined by progressive intensity variations. However, this does not cause a significant increase in processing time.

We have proposed an image segmentation technique using the K-means method in a set of vectors based on two features, the closest directional mean (CDM) and the smallest directional variance (SDV). This technique is efficient, and has a low computational cost and low noise sensitivity. Another advantage of this technique is that it does not require knowledge of either texture [3] or background [4] statistics.

© IEE 2000 *10 May 2000*
Electronics Letters Online No: 20000869
DOI: 10.1049/el:20000869

A.P. Mendonça (*IME/COPPE, DE/3, Pç Gen. Tibúrcio, 80, CEP 22.290-270, Rio de Janeiro, Brazil*)

E-mail: alexmend@aquarius.ime.eb.br

E.A.B. da Silva (*PEE/COPPE/DEL/EE/UFRJ, Cx.P. 68504, CEP. 21945-970, Rio de Janeiro, Brazil*)

References

1 PAPPAS, T.N.: 'An adaptative clustering algorithm for image segmentation', *IEEE Trans. Signal Process.*, 1992, **SP-40**, (4), pp. 901–914
2 TOU, J.T., and GONZALEZ, R.C.: 'Pattern recognition principles (Addison Wesley, 1974)
3 BOUKERROUI, D., BASSET, O., and BASKURT, A.: 'Multiresolution adaptative image segmentation based on global and local statistics'. IEEE Int. Conf. Image Processing, 1999, pp. 358–361
4 LIM, D.K., and HO, Y.S.: 'Image segmentation using hierarchical meshes'. IEEE Int. Conf. Image Processing, 1999

VQ-based digital image watermarking method

Zhe-ming Lu, Jeng-shyang Pan and Sheng-he Sun

A vector-quantisation (VQ)-based watermarking method is presented which utilises the codebook expansion technique. This method is efficient, provides enhanced security and the watermarked image is robust against the effects of VQ compression. Moreover, the watermark extraction can be performed without the original image. Experimental results are presented which demonstrate the effectiveness of this algorithm.

Introduction: Digital images and video sequences are now widely used and distributed on the Internet and via CD-ROM, which means that the protection and enforcement of intellectual property rights assumes an even greater importance. Consequently, digital image watermarking has become an area of increased research activity.

Digital watermarking is a method of embedding secret or confidential information directly into a digital medium to establish ownership or identity a purchaser. Most transform-domain watermarking techniques are based on the discrete cosine transform (DCT) [1], discrete Fourier transform (DFT) [2], discrete wavelet transform (DWT) [3] and the chirp-Z transform [4]. Recently, we introduced a novel watermarking technique based on vector quantisation (VQ) [5] by embedding the watermark information in codeword indices. In this Letter we present a more efficient VQ-based image watermarking technique, in which the watermark extraction can be performed without the original image.

Proposed VQ-based digital image watermarking: The key requirement in VQ-based digital watermarking [5] is to obtain a partition S (the secret key for watermarking) from the codebook $C = \{\mathbf{c}_0, \mathbf{c}_1, ..., \mathbf{c}_{N-1}\}$. The partition $S = \{S_1, S_2, ..., S_M\}$ of the codebook C for a certain threshold $D > 0$ should satisfy

$$S = \bigcup_{i=1}^{M} S_i \tag{1}$$

$$S_i \cap S_j = \Phi \quad (\forall i \neq j) \tag{2}$$

$$d(\mathbf{c}_l, \mathbf{c}_p) \leq D \quad (\forall \mathbf{c}_l, \mathbf{c}_p \in S_i) \tag{3}$$

$$\|S_i\| = 2^{n(i)} \tag{4}$$

where $1 \leq i, j \leq M$, $0 \leq l, p \leq N-1$, $d(\mathbf{c}_l, \mathbf{c}_p)$ denotes the squared Euclidean distortion between $\mathbf{c}_l$ and $\mathbf{c}_p$, $\|S_i\|$, denotes the number of codewords in S_i and $n(i)$ is a natural number.

In the original VQ-based watermarking algorithm [5], the partition S is generated by the tabu search algorithm [6]. We present a simple codebook expansion technique which is used to produce the partition S in order to meet the requirements of watermark extraction without the original image. First, a basic codebook $C^b = \{\mathbf{c}_0^b, \mathbf{c}_1^b, ..., \mathbf{c}_{M-1}^b\}$ is generated by the well-known Linde-Buzo-Gray (LBG) algorithm [7], where M is the basic codebook size. Secondly, for each basic codeword, $K-1$ extended codewords are randomly produced in its neighbouring region, each of which has less distortion from the basic codeword than D, where K is a natural number and $K > 1$, thus an extended codebook C^e of size $(K-1) \cdot M$ is generated. Finally, the basic codebook is combined with the extended codebook and the codewords scrambled to obtain the final user codebook $C = \{\mathbf{c}_0, \mathbf{c}_1, ..., \mathbf{c}_{N-1}\}$, where $N = K \cdot M$. Thus, each subset in the partition S has K codewords, including one basic codeword from C^b and $K-1$ neighbouring codewords from C^e.

Fig. 1 *Original Lena image*

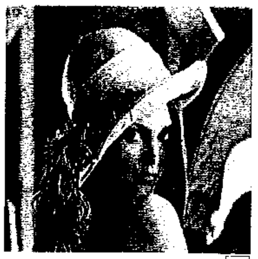

Fig. 2 *VQ compressed Lena image (PSNR = 30.78dB)*

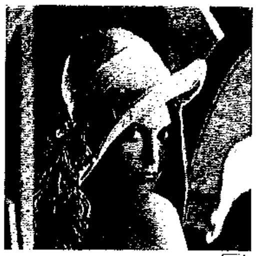

Fig. 3 *Watermarked Lena image (PSNR = 30.59dB)*

Assume that the original image X is segmented into T blocks, i.e. $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$. To describe the algorithm, we set $K = 4$, thus each image block can be embedded with two bits of watermark information. The embedding process is performed block by block. For each input block $\mathbf{x}_t$, $1 \leq t \leq T$, the embedding process can be expressed as follows:

(i) Search the nearest codeword $\mathbf{c}_i^b$ for $\mathbf{x}_t$ in the basic codebook.
(ii) Translate the corresponding two bits of watermark information into an integer g; for example, if the two bits of the watermark are '10', then $g = 2$.
(iii) If $g = 0$, find the corresponding index n of $\mathbf{c}_i^b$ in the user codebook and transmit it to the receiver through the channel (or save it in a file); otherwise, assume that the three extended codewords of $\mathbf{c}_i^b$ in the extended codebook are $\{\mathbf{c}_j^e, \mathbf{c}_{j+1}^e, \mathbf{c}_{j+2}^e\}$, where $j = 3 \cdot i$, then find the corresponding index n of $\mathbf{c}_{j+1-g}^e$ in the user codebook and transmit it to the receiver through the channel (or save it in a file).
(iv) Receive the index n (or read the index n from the file) and use the codeword $\mathbf{c}_n$ in the user codebook to reconstruct the input image block.