Abstract: - In this paper, we address the problem of design a wavelet lossy image coder that allows preservation of image features such as edges using a modified version of SPIHT algorithm. The modification is made by changing the concept of pixel significance, a coefficient is selected not only by magnitude but as part of a Feature Of Interest (FOI) map obtained with SUSAN edge detector and mapping the correspondent positions to the wavelet domain. The bit rate spends more bits in those important zones, and finally the bit stream is entropy coded. We show experiments in which is demonstrated the edge preserving by using the decompressed images as inputs to quality automatic inspection process using the commercial software “Vision Builder”.

Key-Words: - Image coding, edge preserving, wavelet, SPIHT, parent-child relationship, automatic inspection.

1 Introduction
There has been a lot of research to design algorithms to compress images which can be mainly classified in lossy and lossless methods. The main interest of the researchers is to build image coders obtaining a big compression factor and good quality reconstructed images [1]. One of the main reasons to design this coders is due to the great growth of the use of Internet and mobile communications devices in which the information delivery needs to be efficient in the sense of use few inform to avoid bandwidth saturation and to perceive and recognize images as early as possible.

An important branch of image compression are the progressive coders which seeks to achieve the highest image quality across an entire range of bit rates in contrast to non progressive coders that seeks the highest image quality at a single target bit rate [2]. An ideal progressive coder is that one reconstructs the image showing first the preserved (important) features needed for a particular application such as image understanding or inspection. Some of the most important features needed for image understanding are edges and are the key for the success of artificial vision tasks.

The preservation of important features becomes imperative in areas such as medical and textile industry in which there are laws and restrictions about the original images uses and the information loss is not recommended [3]. Feature preserving means that the location, strength and shape of features are unchanged after the application of a general filter, of course, natural differences can occur due to resolution changes [4].

In the literature there are a few works which address the problem of feature preserving image compression. The firsts works, present schemes defining Regions Of Interest (ROI), this are coders in which the bit budget is more spend in ROI areas and other image parts are represented with lower quality [5]. A Region Based Discrete Wavelet Transform (RBDWT) coder is presented in [6], first an image is segmented into regions described by it contours and textures and then each part is coded separately. The work presented in [2] shows two progressive image coders designed to improving the visual clarity of image edges in progressive code stream. The locations of edges are captured with an edge detection step, and then are encoded and transmitted to the decoder as a part of the image header. Finally, in [3] a methodology to preserve edges using a vector with features of interest and the use EZW for the quantization stage is proposed. The scheme combines the representation of the wavelet coefficients based in trees and the implicit data transmission over the image features that need to be accentuated or preserved; this work is the base to build the methodology presented in this work.

In this paper we present a lossy image coder based in wavelets which allows edge information preserving by the use of a proposed modification of SPIHT algorithm.

2 Proposed Model
An image contains several features such as edges, textures, details associated to edges, etc., that exhibit the ability through which the image can offer...
information to people about the objects presented in image. In order to design a lossy image coder with feature preserving it is important first, identify the features needed to preserve and second, spend more bits to represent important information and sacrifice fidelity or quality in other image regions [7].

As we explain in section one, in order to solve the problem of feature preserving typical coders use two main approaches: a) send side information about the features, and b) use different approaches to code different features.

The model proposed in this paper differs with those approaches, because we do not send side information and we only use one coder to compress images. The stages of the model are: a) Features Of Interest (FOI) extraction with SUSAN, b) Domain transformation with reversible wavelet, c) Mapping FOI to the wavelet domain, d) Image coding with modified SPIHT and e) Image decoding. Each stage is briefly explained in the next subsections.

### 2.1 Features Of Interest (FOI) Extraction with SUSAN

First, we need to select the images to compress and compute the FOI map (edge positions) for this, we use the Smallest Unvalue Segment Assimilating Nucleus (SUSAN) edge detector which is a more robust and effective method than Canby [8] in the sense that it provides much better edge localization and connectivity. SUSAN use a predetermined window centered on each image pixel, applying a locally acting set of rules to give an edge response. This response is then processed to give as output a set of edges [9]. In summary SUSAN performs the following three steps at each image pixel:

1. Place a circular mask around the pixel in question (the nucleus).
2. Calculate the number of pixels within the circular mask which have similar brightness to the nucleus.
3. Subtract the USAN size from the geometric threshold to produce an edge strength image.

Figure 1 shows the FOI edge map obtained for each image used for the tests. The threshold used to obtain the FOI map in both images is 50.

![Fig. 1. FOI maps. a) Battery clamp and b) Spray bottle.](image)

### 2.2 Domain Transformation with Reversible Wavelets

After FOI extraction a reversible wavelet transform is applied to the original images finding two goals: a) reduce the correlation among the transform coefficients, and b) taking advantage of the energy compaction property to code only a fraction of the transform coefficients without producing serious distortion [10].

The Discrete Wavelet Transform (DWT) allows decomposing hierarchically an input signal into referential signal series of low resolution and its associated detail signals [11]. The DWT is obtained by convolving the image columns and rows with a low pass filter (scalling function $\Phi$ wavelet father) and a high pass filter (wavelet function $\Psi$ wavelet mother). For this paper the images are decomposed $\log_2($image size$) - 1$ levels as it is used in [12] and the wavelet filter used is the Biorthogonal 2.2. In figure 2 an example of a three level wavelet transform over two images is shown.

![Fig. 2. Three level Discrete Wavelet Transform. a) Battery clamp and b) Spray bottle.](image)

### 2.3 Mapping FOI to the Wavelet Domain

The FOI edge maps obtained with SUSAN are used in order to map the coordinate points to the wavelet domain. In the wavelet decomposition a coefficient of scale $i$ affect an area of $2^i \times 2^i$ positions of the original domain, there are a hierarchic relation between the coefficients which allows defining a structure known as spatial orientation tree.

A pixel in the lower wavelet subband is father of three coefficients at the same position in high frequency subbands at same scale. Any other coefficient not pertaining to lower band have four childs which can be obtained by equation 1.

$$\text{childs} (x, y), s \neq 1 = \begin{cases} (2x - 1, 2y - 1) \\ (2x - 1, 2y) \\ (2x, 2y - 1) \\ (2x, 2y) \end{cases}$$  

(1)
In resume, the mapping process is: a) Use the edge map obtained with SUSAN, b) Doubled the position of a pixel pertaining to FOI (2 x 2 as in wavelet transform), c) Downsample the image, d) Repeat b and c an n number of levels, and e) Mark the descendent coefficients as a part of FOI until the original size is reach.

2.4 Image Coding with Modified SPIHT

In 1996 Said and Pearlman present an improved version of the Embedded Zerotree Wavelet (EZW) coder in which propose a different tree structure, the algorithm were called Set Partitioning In Hierarchical Trees (SPIHT) [13]. The principle behind SPIHT is to define significance of a pixel if its value is larger or equal to given threshold and then the coefficient can be coded.

The SPIHT modification proposed here is that the significance of a pixel is defined both by its significance and the pixel position corresponding to FOI in the wavelet domain. A similar idea was proposed in [3] and the main differences are: we use SPIHT instead of EZW, in our model we can preserve two or more features at same time, and we use a different wavelet filter. In order to illustrate how SPIHT modification works we use a structure similar to that used in [13], we use tables both for SPIHT and for the proposed modification.

We use the figure 3 in order to present an example. In figure 3a a portion of a 4 x 4 image is shown and in figure 3b the FOI areas are marked with ones.

First O(i, j) is defined as the set of offspring (direct descendants) of a tree node defined by pixel location (i, j).

D(i, j) is the set of descendants of node defined by pixel location (i, j).

L(i, j) is the set defined by L(i, j) = D(i, j) – O(i, j).

The following explanation refer to the respective numbered entries in table 1 for proposed modified SPIHT and compared to original SPIHT shown in table 2.

1. Initial SPIHT settings. The initial threshold is set to 32, and then LIS (List of Insignificant Sets), LIP (List of Insignificant Pixels) and LSP (List of Significant Pixels) are initialized.

2. SPIHT begins coding the significance of the pixels in LIP. The position (0, 0) is insignificant and the position (0, 1) is significant because is larger than the threshold and pertain to the FOI map. Different to original (table 2) in which the two coordinates (0, 0) and (0, 1) are significant. For both schemes the positions (1, 0) and (1, 1) are insignificant.

3. After testing pixels, SPIHT begins to test sets following the entries in LIS. D(0, 1) is the set of four coefficients {(0, 2), (0, 3), (1, 2), (1, 3)}. Because D(0, 1) is significant SPIHT test the significance of the four offsprings. Finally (0, 1) is removed from LIS.

4. Same procedure to the comment 3 is applied with D(1, 0), since is insignificant no action need to be taken, and check for the next element of LIS.

5. D(1, 1) is insignificant no action need to be taken, the first pass ends and the refinement pass starts and is made equals to original SPIHT.

At the final of the first pass the modified SPIHT spend 13 bits compared with the 14 bits spend by original SPIHT, the LSP obtained is: LSP = {{0, 1), (0, 2)}, and for the original SPIHT is: LSP = {{0, 0), (0, 1), (0, 2)}.

If we use a bit rate of 5 for figure 3a, then the final LSP with original SPIHT is: LSP = {{0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (1, 3), (2, 0), (2, 1), (3, 0), (2, 3), (3, 2), (3, 3), (3, 1)} and the bits spend are 55.

With modified SPIHT LSP = {{0, 1), (0, 2), (0, 3), (1, 2), (1, 3)}, which corresponds to the positions that we need to preserve and the bits spend are 56.

Table 1. Image coding with modified SPIHT.

<table>
<thead>
<tr>
<th>Comment</th>
<th>Pixel or set treated</th>
<th>Output bit</th>
<th>Action</th>
<th>Control List</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(0, 0)</td>
<td>0</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td>(0, 1)</td>
<td>1</td>
<td>to LSP</td>
<td>LSP = (0, 0)</td>
</tr>
<tr>
<td>(1)</td>
<td>(0, 2)</td>
<td>0</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td>D(0, 1)</td>
<td>1</td>
<td>Test offsprings</td>
<td>LSP = (0, 1)</td>
</tr>
<tr>
<td>(0, 1)</td>
<td>(0, 2)</td>
<td>0</td>
<td>to LSP</td>
<td>LSP = (0, 1)</td>
</tr>
<tr>
<td>(1, 2)</td>
<td>(0, 3)</td>
<td>0</td>
<td>to LIP</td>
<td>LIP = (0, 0, 1, 1, 0)</td>
</tr>
<tr>
<td>(1, 3)</td>
<td>(0, 2)</td>
<td>0</td>
<td>to LIP</td>
<td>LIP = (0, 0, 1, 1, 0, 1)</td>
</tr>
<tr>
<td>(2)</td>
<td>(0, 1)</td>
<td>0</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td>(0, 1)</td>
<td>0</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>(4)</td>
<td>D(0, 1)</td>
<td>0</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>(5)</td>
<td>D(0, 1)</td>
<td>0</td>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>
Finally the bit stream obtained from modified SPIHT is entropy coded to obtain the final compressed file, to the stream we add information about image size, wavelet level and filter used.

2.5 Image Decoding

At this stage entropy decoding is made follow by the SPIHT decoding and Inverse Discrete Wavelet Transform (IDWT) is computed to obtain decompressed images.

The decompressed images obtained from figure 3a are shown in figure 4a for original SPIHT and in figure 4b for modified SPIHT. The Mean Square Error (MSE) of original SPIHT is 8.3125 and the Peak Signal to Noise Ratio (PSNR) is 38.9335 db. The MSE for modified SPIHT is 403.0625 and the PSNR is 22.0771 db.

![Fig. 4. Reconstructed images. a) Original SPIHT and b) Proposed SPIHT.](image)

With the results showed in table 3 we can give several comments: first, the process of inspection is successful performed with the original and the decompressed images except for the image in which we add a lot of noise.

Second, we can observe that in the case of fail (last row of table 3) the fail is in the last stage where the distance between the calipers (gap between two branches) is very short. Third, as we can see in figure 5d due to the noise Vision Builder measure the gap distance (noise) as a part of the edge even with the good edge reconstruction.

### Table 3. Results for battery clamp verification.

<table>
<thead>
<tr>
<th>Image</th>
<th>Match pattern</th>
<th>Set coordinate system</th>
<th>Find circular edge</th>
<th>Detect object</th>
<th>Detect object</th>
<th>Caliper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery</td>
<td>#Matches 1 PASS</td>
<td>PASS</td>
<td>Edmis: 59.32 PASS</td>
<td>#Objects 1 PASS</td>
<td>#Objects 2 PASS</td>
<td>Distance: 28.32 pc</td>
</tr>
<tr>
<td>Battery0.5</td>
<td>#Matches 1 PASS</td>
<td>PASS</td>
<td>Edmis: 59.18 PASS</td>
<td>#Objects 1 PASS</td>
<td>#Objects 2 PASS</td>
<td>Distance: 25.27 pc</td>
</tr>
<tr>
<td>Battery0.1</td>
<td>#Matches 1 PASS</td>
<td>PASS</td>
<td>Edmis: 59.47 PASS</td>
<td>#Objects 1 PASS</td>
<td>#Objects 2 PASS</td>
<td>Distance: 29.41 pc</td>
</tr>
<tr>
<td>Batterynoisy</td>
<td>#Matches 1 PASS</td>
<td>PASS</td>
<td>Edmis: 56.87 PASS</td>
<td>#Objects 1 PASS</td>
<td>#Objects 2 PASS</td>
<td>Distance: 18.24 pc fail distance to small</td>
</tr>
</tbody>
</table>

### Table 2. Image coding with original SPIHT.

<table>
<thead>
<tr>
<th>Comment</th>
<th>Pixel set tested</th>
<th>Output bit</th>
<th>Action</th>
<th>Control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td>(0,0)</td>
<td>1+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0,1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1,0)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1,1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td>D(0,1)</td>
<td>1</td>
<td>Test offset</td>
<td></td>
</tr>
<tr>
<td>(0,2)</td>
<td></td>
<td>1+</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0,3)</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1,2)</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1,3)</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4)</td>
<td>D(0,0)</td>
<td>0</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>(5)</td>
<td>D(0,1)</td>
<td>0</td>
<td>none</td>
<td></td>
</tr>
</tbody>
</table>

Vision Builder for Automated Inspection (AI) is a configurable machine vision development environment that requires no programming [14]. The software can be used to visually test when a product is manufactured and assembled correctly. For tests we use two Vision Builder modules of inspection and classification of industrial pieces that verifies quality criteria of the parts to classify the pieces in good or bad quality.

We made a compression/decompression process with original images and a bit rate of 0.1 and 0.5. The first test is the quality verification of a battery clamp. First, Vision Builder detects the battery clamp (locate the part) in the image, and then a coordinate system based on the part localization is determined. After that the circular hole is found and measured and the radio of the clamp is verified. Then check for the presence of the two fixation holes. Finally the gap between the two branches (caliper) is computed.

In table 3 the results obtained for battery clamp inspection are shown. The first row shows the results for original image, the second row the results for image compressed with 0.5 bit rate, the third row the results for image compressed with 0.1 bit rate and finally a special case in which we add a lot of noise to the decompressed image obtained with a bit rate of 0.5. In figure 5 the reconstructed images are shown.
Finally, the process is made successfully with the decompressed images; with this the edge preservation is demonstrated.

The second test, performs a visual inspection to verify the assembly of plastic spray bottles. First, the bottle is located in the image, left edge of the bottle is localized and a coordinate system is created. Then, verifies if the cap has been screwed down properly on the bottle (on both sides). After that the left distance is measured and the right edges with it respective distance is computed. Finally the presence of spray nozzle and cap is verified.

In table 4 the results obtained for spray bottle inspection are shown. The first row shows the results for original image, the second row the result for image compressed with 0.5 bit rate, the third row the results for image compressed with 0.1 bit rate and finally a special case in which we add a lot of noise to the decompressed image obtained with a bit rate of 0.5. In figure 5 the reconstructed images are shown.

For the spray verification tests we obtain again good verification results except for the noisy image. Here the inspection process fails in the first stage by that all the process fails too.

In order to compare the coding performance of the feature preserving lossy image coder we consider the following five objective measures: Compression Factor \((CF)\), Mean Square Error \((MSE)\) equation 2), Peak Signal to Noise Ratio \((PSNR)\) equation 3), Frobenius norm \((F)\) equation 4) and Norm2 \((N_2)\) equation 4).

\[
MSE = \frac{1}{MN} \sum_{y=1}^{M} \sum_{x=1}^{N} [I(x,y) - I'(x,y)]^2 \quad (2)
\]

\[
PSNR = 10 \log_{10} \left( \frac{255}{\sqrt{MSE}} \right) \quad (3)
\]

\[
N_2, F = \frac{\|I - I'|_{2,F}}{\|I\|^2_{2,F}} \quad (4)
\]

In table 5 we show the error measures obtained for all the images for the automatic inspection test cases. As it is shown in table 5 the error measures can be bad, but as it is demonstrated with the test the inspection process is made satisfactory. The compression factors are good to compare with other similar algorithms adding the advantage that we not send any type of side information.

With this is time to remark that the importance of the proposed method is no to improve the error measures but to perform the inspection process in a good way.

**Table 4.** Results for spray bottle verification.

<table>
<thead>
<tr>
<th>Image</th>
<th>Locate bottle left edge</th>
<th>Bottle referenced</th>
<th>Left edges</th>
<th>Left distance</th>
<th>Right edges</th>
<th>Right distance</th>
<th>Spray nozzle</th>
<th>Presence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spray0.2</td>
<td>120.20</td>
<td>PASS</td>
<td>2</td>
<td>10.63</td>
<td>PASS</td>
<td>2</td>
<td>6.56</td>
<td>PASS</td>
</tr>
<tr>
<td>Spray0.5</td>
<td>142.20</td>
<td>PASS</td>
<td>2</td>
<td>10.32</td>
<td>PASS</td>
<td>2</td>
<td>7.10</td>
<td>PASS</td>
</tr>
<tr>
<td>Spray0.1</td>
<td>130.66</td>
<td>PASS</td>
<td>2</td>
<td>11.82</td>
<td>PASS</td>
<td>2</td>
<td>7.56</td>
<td>PASS</td>
</tr>
</tbody>
</table>

**Spraynoisy**

- Two bottle edges fail
- Coordinate system not available
- Fail
- Coordinate system not available
- Fail
- Coordinate system not available
- Fail
- Coordinate system not available
- Fail
- Coordinate system not available
- Fail

**Fig. 5.** Original and reconstructed images. a) Original battery clamp, b) Battery0.5, c) Battery0.1, d) Batterynoisy, e) Original spray bottle, f) Spray0.5, g) Spray0.1 and f) Spraynoisy.
4 Conclusions and Further Works

We have presented a wavelet based lossy image coder which allows edge preserving; this coder can be used if an early recognition of images is needed or in environments such as medical in which there are laws for the use of images and feature preserving is imperative.

To design the coder we first obtain the image edge map using SUSAN, then a domain transformation is computed and a mapping process of the points in the original domain to the wavelet domain is made. Finally, the images are coded and decoded using the modified version of SPIHT algorithm.

With the tests and results presented we demonstrate the ability of the coder to reconstruct images spending more bits and given more quality to that coordinates defined by a FOI map. Objectives measures can not give information of the method goodnes, and remember we propose image feature preserving even at very low bit rates and not to improve the error measures. Even with that, the error measures obtained are close to those methods like SPIHT and EZW.

In the future we hope to work in finding a method to reduce the ringing that appears in images and testing another transform such as contourlet transform in order to solve the problem of edge directionality and image geometry.

References:

Table 5. Error Measures for battery clamp and spray bottle.

<table>
<thead>
<tr>
<th>Image</th>
<th>C.F.</th>
<th>MSE</th>
<th>PSNR</th>
<th>F</th>
<th>N2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery0.5</td>
<td>18.179:1</td>
<td>615.235</td>
<td>20.2404</td>
<td>0.1055</td>
<td>0.0914</td>
</tr>
<tr>
<td>Battery0.1</td>
<td>85.780:1</td>
<td>740.257</td>
<td>19.437</td>
<td>0.1157</td>
<td>0.0923</td>
</tr>
<tr>
<td>BatteryNoisy</td>
<td>18.719:1</td>
<td>1041.962</td>
<td>17.9523</td>
<td>0.1373</td>
<td>0.09558</td>
</tr>
<tr>
<td>Spray0.5</td>
<td>17.296:1</td>
<td>353.8215</td>
<td>22.64</td>
<td>0.0845</td>
<td>0.0697</td>
</tr>
<tr>
<td>Spray0.1</td>
<td>86.459:1</td>
<td>467.2724</td>
<td>21.4351</td>
<td>0.0971</td>
<td>0.0724</td>
</tr>
<tr>
<td>SprayNoisy</td>
<td>19.793:1</td>
<td>1091.2189</td>
<td>17.7517</td>
<td>0.1485</td>
<td>0.0980</td>
</tr>
</tbody>
</table>