Selective Image Compression Using MSIC Algorithm

Enrique Pelayo, David Buldain and Carlos Orrite

Abstract This paper presents a new algorithm, Magnitude Sensitive Image Compression (MSIC), as a reliable and efficient approach for selective image compression. The algorithm uses MSCL neural networks (in direct and masked versions). These kind of neural networks tend to focus the learning process in data space zones with high values of a user-defined magnitude function. This property can be used for image compression to divide the image in irregular blocks, with higher resolution in areas of interest. These blocks are compressed by Vector Quantization in a later step, giving as a result that different areas of the image receive distinct compression ratios. Results in several examples demonstrate the better performance of MSIC compared to JPEG or other SOM based image compression algorithms.

Keywords Image compression · Competitive learning · Neural networks · Saliency · Self organizing maps · JPEG · DCT · MSCL

1 Introduction

In the human vision system the attention is attracted to visually salient stimuli, and therefore only scene locations sufficiently different from their surroundings are processed in detail. This provides the necessary motivation to devise a novel image compression method capable of applying distinct compression ratios to different zones of the image according to their saliency.

In this paper we make use of the Magnitude Sensitive Competitive Learning Algorithm (MSCL) [1]. MSCL is a Vector Quantization method based on competitive learning, where units compete not only by distance but also by a user defined magnitude. Using saliency as the magnitude, units tends to model more accurately

This work is partially supported by Spanish Grant TIN2010-20177 (MICINN) and FEDER and by the regional government DGA-FSE.

E. Pelayo (✉) · D. Buldain · C. Orrite
Aragon Institute for Engineering Research, University of Zaragoza, Zaragoza, Spain
e-mail: epelayoc@gmail.com
url: http://i3a.unizar.es/en

K. Madani et al. (eds.), *Computational Intelligence*,
Studies in Computational Intelligence 613,
DOI 10.1007/978-3-319-23392-5_23

the salient areas of the images, and therefore the neural network behaviour imitates the human vision system.

Vector quantization (VQ) is a classical quantization method. In the context of image processing, basic vector quantization consists in dividing the input image into regular blocks of pixels of a pre-defined size, where each block is considered as a *D-dimensional* vector. Each of these input vectors from the original image is replaced by the index of its nearest codeword, so only this index is transmitted through the media. The whole codebook serve as a database known on the reconstruction site. This scheme reduces the transmission rate while maintaining a good visual quality. Figure 1a shows this scheme.

In VQ, compression level depends on two factors, the number of blocks and the level of compression of each block. Both factors are related in an inverse way. Lower number of blocks means that they are higher in size, and therefore higher is the bit depth necessary to codify each block for a similar quality.

Some authors [2–5] have already used some VQ variants, such as Kohonen neural network [6] for image compression. These algorithms use a fixed block size and concentrate in several ways to get a smaller codification of each block or to improve the quality of the codification. Laha [2] uses surface fitting of data assigned to each codeword instead of the codeword itself, which improves the visual quality of the results. [3–5] apply DCT filtering [7] to each block previous to the quantization step to lower the dimension of the input data. On the other hand, [3] takes advantage of the topological ordering property of the SOM neural network to codify indexes with a few bytes.

In this paper blocks may have different size, chosen according to its relevance (which is selected following the image saliency). Blocks located in areas of high image saliency are smaller than those assigned with low saliency. As bit depth used in the quantization step is the same for all blocks, quantization error increases directly with the block size in areas of low image saliency. Therefore, a lower number of blocks are used to represent the whole image increasing the overall image compression and preserving at the same time the quality of most relevant areas.

Another important difference with the above mentioned methods is that, in our approach, block shapes are, in general, irregular, i.e., neither rectangular nor squared. Therefore, quantization has to take into account samples that may have invalid components. Figure 1b shows the basic idea of the proposed algorithm for grayscale images. It requires to transmit the block centers and index. At the receiver, it is possible to regenerate the shape and mask of each block and locate it with it center and magnitude. Then, with its index, the block image is regenerated and summed up to form the whole image. In Sect. 4 we present the complete algorithm, more complex to reduce the amount of data to be transmitted.

The remainder of this paper is organized as follows. Section 2 describes the MSCL algorithm. Section 4 shows its use to achieve selective image compression focused on the most salient regions of an image with the method that we call Magnitude Sensitive Image Compression (MSIC). A comparative between MSCL and classical JPEG and SOM based VQ algorithms for a high compression ratio task is carried out in Sect. 5. Finally, Sect. 6 concludes with a discussion and ideas for future work.

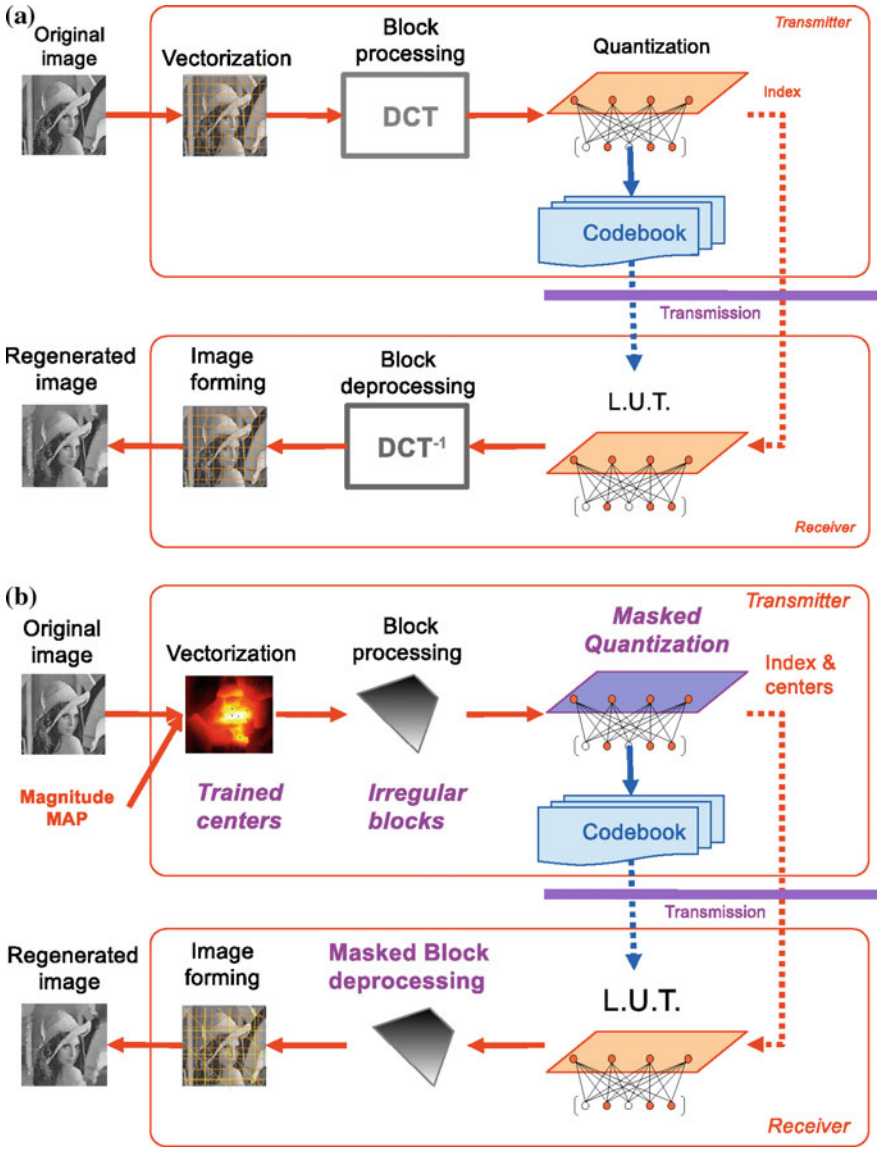

Fig. 1 Basic idea of Competitive Learning algorithms in the task of image compression for grayscale images. *Top* Common CL algorithm. *Bottom* MSIC algorithm. Differences with other CL algortihms are the use of a MSCL to get block centers (centers are trained weights of MSCL units), the use of irregular blocks and the masked quantization/deprocessing

2 The MSCL Algorithm

MSCL is a type of artificial neural network that is trained using unsupervised learning to produce a representation of the input space of the training samples depending on a magnitude. Codebook $\mathcal{M}$ is formed by M weight vectors. Prototype of unit $m \in \mathcal{M}$ is described by a vector of weights $\mathbf{w}_m(t) \in \mathbb{R}^D$ and the magnitude value mu_m. This value is calculated with the function $MF(i,t)$, that is a measure of any feature or property of the data inside the Voronoi region of unit m, or a function of its parameters.

The idea behind the use of this magnitude term is that, during competition between two units situated at similar distance from the input sample, the winner will be the unit with the lowest magnitude value. As a result of the training process units will be forced to move from the data regions with low $MF(i,t)$ values to regions where this magnitude function is higher. MSCL follows next steps, that are repeated until a termination condition is achieved:

Global Unit Competition. At this point, we form the local winner set $\mathcal{S}$, ($\mathcal{S} \subset \mathcal{M}$) with the M_{local} units closest to the input sample as: $\mathcal{S} = \{s_1, s_2, ..., s_{M_{local}}\}$.

$$\|\mathbf{x}(t) - \mathbf{w}_s(t)\| \leq \|\mathbf{x}(t) - \mathbf{w}_m(t)\| \; \forall m \notin \mathcal{S} \wedge s \in \mathcal{S} \tag{1}$$

Local Unit Competition. Winner unit j is selected from units belonging to $\mathcal{S}$, as the one that minimizes the product of its magnitude value with the distance of its weights to the input data vector, following this equation:

$$j = \underset{s \in \mathcal{S}}{\operatorname{argmin}}(mu_s(t)^{\gamma} \cdot \|\mathbf{x}(t) - \mathbf{w}_s(t)\|) \tag{2}$$

Winner and Magnitude Updating. For all units in the map, weights and magnitude are adjusted iteratively for each training sample, following:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \alpha(t)\left(\mathbf{x}(t) - \mathbf{w}_j(t)\right) \tag{3}$$

$$mu_j(t+1) = MF(j,t) \tag{4}$$

In the above equations, γ defines the strength of the magnitude during the competition. $\alpha(t)$ is the learning factor, calculated as $\alpha(t) = (1/h_j(t))^{\beta}$, where $h_j(t)$ stands for the number of input signals for which unit j has been winner so far, and β is a scalar value between 0 and 1. The winner j is also called the best matching unit (BMU).

3 The Masked MSCL Algorithm

The new proposed image compression algorithm will require the capability of dealing with incomplete data, as blocks to be compressed are irregular (in shape and size). Here we present a masked version of MSCL that is able to deal with data samples of different size (we speak of 'masked' data). To use this algorithm we will consider that each data sample consists in two vectors, $\mathbf{x} = (x_1, \ldots, x_D) \in \mathbb{R}^D$ the data vector itself (with the maximal possible dimension of a data sample D), and its corresponding mask $\mathbf{msk} = (msk_1, \ldots, msk_D) \in \mathbb{R}^D$. The mask is a vector with ones in the the valid components of $\mathbf{x}$ and zeros in the other components.

The algorithm follows the same steps than MSCL, but both competition and updates are slightly more complex as it has to be considered the mask. Changes are the following:

1. Only valid components (corresponding mask is one) are considered for global and local competition:

$$\|\mathbf{msk}(t) \circ (\mathbf{x}(t) - \mathbf{w}_s(t))\| \leq \|\mathbf{msk}(t) \circ (\mathbf{x}(t) - \mathbf{w}_m(t))\| \; \forall m \notin \mathcal{S}, s \in \mathcal{S} \quad (5)$$

$$j = \underset{s \in \mathcal{S}}{\operatorname{argmin}}(mu_s(t)^{\gamma} \cdot \|\mathbf{msk}(t) \circ (\mathbf{x}(t) - \mathbf{w}_s(t))\|) \quad (6)$$

2. Instead of scalar α, algorithm uses vector $\mathbf{alpha} = (\alpha_1, \ldots, \alpha_D) \in \mathbb{R}^D$ as the learning factor, where α_d is:

$$\alpha_d = \begin{cases} 1/h_{jd} & \text{if } msk_d = 1 \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

   Here h_{jd} is the number of times up to the moment that component d has taken a valid value when unit j was a winner.
3. Only valid components (those with $msk_d = 1$) of winner weights are updated:

$$w_{jd}(t+1) = w_{jd}(t) + \alpha_d \left(x_d(t) - w_{jd}(t)\right) \quad (8)$$

4 Magnitude Sensitive Image Compression

Figure 2 shows the whole MSIC algorithm applied to grayscale images, where image compression, in the transmitter, is represented on the top and the image restoration process at the receiver is depicted on the bottom. Image is compressed with different quality according to a selected user magnitude. Section 4.6 explain how to extent this methodology to color images.

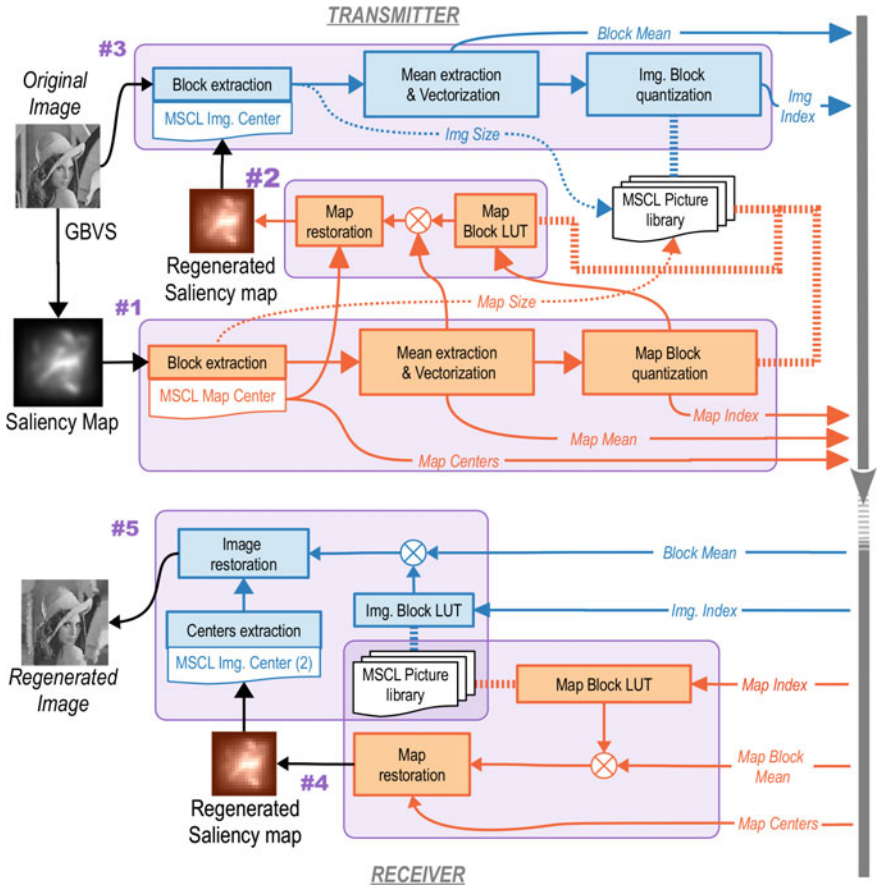

Fig. 2 Global algorithm for *grayscale* images. *Marked* with **#n** the corresponding subsection with the detailed explanation and, also showing the order of processing steps in the transmitter and receiver

In this work we use as magnitude the saliency map, with the same size as the processed image, provided by a user function. Section 5 explain these functions.

The results of the compression are a group of image blocks encoded by indexes. Unlike other image compression methods, our algorithm uses blocks of different sizes, which are located at any position of the image. Therefore, this implies that block centers and sizes has to be sent to the receiver, apart from the corresponding index. As this approach would mean the transmission of huge quantity of information, we have adopted an alternative solution.

We use the saliency map to train a MSCL network, using as inputs the coordinates (x_1, x_2) of each pixel and the saliency as magnitude. After training, the weights of its units (codewords) are the block centers $(bc(k), k = 1...N_{bc})$. The surrounding assigned to the Voronoi region of each block-center configure the corresponding

blocks. The image is so fragmented in so many blocks as units in this network (N_{bc}). In Sect. 4.1 we will show how to determine the block sizes (and block limits) for each codeword or unit. This process encodes the saliency map with low quality, and both the encoded image and the encoded map are transmitted.

At the receiver first the saliency map is regenerated, and with it, the image block limits and centers can be calculated. They are used with the image indexes to restore the image.

It is worth noting that it is necessary an additional step at the transmitter. Instead of using directly the saliency map to extract the image blocks, we first decode a saliency map from the encoded map that has to be transmitted. Then we calculate the image centers and limits of image blocks using this Regenerated Saliency Map that will be also regenerated by the receiver.

Summarizing the MSIC algorithm steps are:

1. Map quantization (at transmitter).
2. Map restoration (at transmitter).
3. Image quantization (at transmitter).
4. Map restoration (at receiver).
5. Image restoration (at receiver).

MSIC algorithm uses several MSCL networks: $MSCL_{MC}$ (map center) to extract map blocks, $MSCL_{IC}$ (image center) to extract image blocks, and a pool of MSCLs that he call MSCL picture library ($MSCL_{PIC}$) to generate indexes that encode each block pixels, and act as Look-Up-Table to decode the block shapes with these indexes. This library is calculated using the masked version of MSCL (see Sect. 3) as blocks may have irregular shapes. The first and second neural networks are trained online during map and image quantization. Their codewords are the block centers. However $MSCL_{PICT}$ form a codebook database that is trained offline. It is known by the transmitter and the receiver as a library of the method. Finally receiver uses another MSCL ($MSCL_{IC2}$), that becomes identical to $MSCL_{IC}$ when trained at receiver. Following sections explain the process in detail.

4.1 Saliency Map Quantization

The idea is to consider the saliency map as an image and apply the same compression steps that will be applied to the image.

First step corresponds to the block extraction from the saliency map according to the saliency values. We train a MSCL network ($MSCL_{MC}$) using the 2D coordinates of each pixel(x) as inputs and the following magnitude function:

$$MF(i,t) = \frac{\sum_{Vi} saliency(\mathbf{x}_{Vi})}{V_i(t)} \tag{9}$$

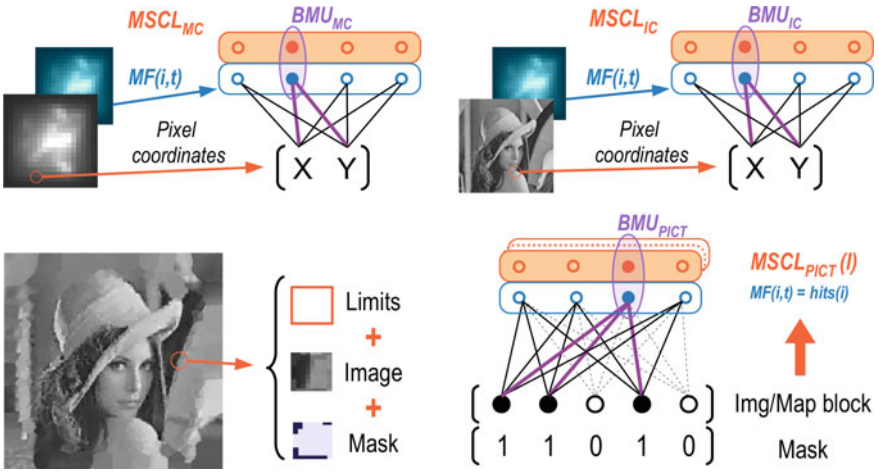

Fig. 3 Neural networks used in the MSIC algorithm: *Top* BMU_{MC} and BMU_{IC}. It is important to mention that this last MSCL is used also in receiver (BMU_{IC2}). *Bottom* Block extraction phase. Each block delivers the block limits, the image and a binary mask. $MSCL_{PICT}(l)$ neural network, where a input sample (vectorized block from the extraction phase) has several masked components

where $\mathbf{x}_{Vi}$ are the data samples belonging to the Voronoi region of unit i at time t, $V_i(t)$ is the number of samples in the Voronoi region, and $saliency(\mathbf{x})$ is the pixel saliency of the corresponding sample. Trained unit weights correspond to the coordinates of the unit in the image, and the magnitude value is the mean of the saliency inside its Voronoi region. Once trained, it is possible to find the best matching unit (BMU_{MC}) assigned to every pixel (using magnitude during competition). The block assigned to each unit is the rectangle wrapping its Voronoi region. A block mask of equal size than the block is also provided in order to mark the pixels belonging to that irregular Voronoi region, see Fig. 3. We used 40 units for $MSCL_{MC}$ in our experiment. With this small number of units a coarse saliency map is obtained, but it is enough to define areas with high saliency.

To codify each of the blocks by VQ, we first resize the block to a squared shape with side value as the maximum between its horizontal and vertical block sizes. The block and the mask are inserted in the squared image filling with zeros the void rows or columns. After that, both are resized to a vector form. We use mean-removed vectors to have a better quantification. Mean value of saliency in each block of pixels, that we call mean block-saliency ($m_b(x, y)$), is sent encoded by 7 bits.

The resulting vector is separated according to its size and dispatched for training or testing to the MSCL picture library ($MSCL_{PICT}(l)$). This pool of codebooks are trained separately only once and become a lookup table in the algorithm. In order to avoid the transmission of the whole codebook pool it is known by both the transmitter and the receiver.

Each codebook of the pool, with 256 codewords, is dedicated to a precise input-vector length. This election of the same number of codewords for different block

sizes forces that larger blocks present less detail in pictorial content than smaller blocks. We have chosen a limited group of sizes that model several size possibilities (the value of l is the length of the square edge to which we have resized the block): $l = [4, 6, 7, 8, 10, 15, 29]$.

This pool of codebooks can be specialized in the type of images considered in the transmission task, or can be generated using an universal library of training images. The images for training are processed following previous described steps, but the magnitude function chosen for these $MSCL_{PICT}(l)$ networks is the hit frequency of each unit, that is:

$$MF(i,t) = hits(i,t) \tag{10}$$

During competition the BMU_{PICT} is calculated using the masked version of MSCL in order to avoid the zero-padding mentioned before. Each time a sample is presented to each neural network of the pool, the corresponding mask is also presented, and only masked weight components are used to compete (see Fig. 3, Right). Each sample might have different masked components. In this way, only pixels corresponding to the Voronoi region of a block are used to find its BMU_{PICT}.

At the end of this step, the magnitude map has been divided in 40 blocks. We have to send to the receiver the following information of each block: Map indexes (BMU_{PICT}) (1 byte), Map mean (7 bits) and Map Centers (2 bytes). Size of each block is not necessary because it is calculated with the block centers.

4.2 Map Restoration at Transmitter

Map representing the saliency of the image is also restored at transmitter with the information generated at the previous step. This is because the restored map will be used at both transmitter and receiver to define the block centers of the image, so results are the same in both sides. Map restoration is accomplished following the previous step in inverse order. First we calculate Voronoi regions assigned to each of the Map Centers by searching the BMU_{MC} of each pixel in $MSCL_{MC}$. The codewords of this neural network are the Map Centers. Additionally, we calculate block limits and mask wrapping by a rectangle the area corresponding to the Voronoi region of each center.

With the i index of the new block, it is converted again into an image block by the look-up table created with $MSCL_{PICT}(l)$. The codeword of the BMU_{PICT} consists of the pictorical content of the block image, but needs to be displaced with the mean block-saliency value of the corresponding block. After summing the mean, it is masked by the binary mask and added to the regenerated saliency map. Repeating the process for all the blocks we obtain the regenerated saliency map, that will represent the saliency values of pixels for the reconstructed image.

4.3 Image Quantization

A similar strategy to the previously described step is followed for image quantization. Blocks are extracted training the $MSCL_{IC}$ (with the coordinates at each pixel) to get the image block centers according to the Regenerated Saliency Map at the transmitter. Then the Voronoi regions of each of these centers are calculated. Blocks are extracted and vectorized. After removing the mean, each image block is processed using the masked version of MSCL with the $MSCL_{PICT}(l)$ (once again using the masked version of MSCL) that corresponds its size, in order to use the most similar pictorial content of the library that will be included in the reconstructed image. It is only necessary to send the corresponding block mean and index from the $MSCL_{PICT}(l)$ for each block.

4.4 Map Restoration at Receiver

Map restoration at receiver is accomplished following exactly the same process than map restoration at transmitter. To do it, the receiver uses for each block its Map index, mean block-saliency, block-center and the same offline $MSCL_{PICT}(l)$ picture library. As operations are the same and they are applied to the same data, the Regenerated Saliency Map at receiver is exactly the same than the one at the transmitter.

4.5 Image Restoration

Last step in the whole process is image restoration, using the received means of block-saliency, the pixel indexes and the regenerated saliency map. This step is similar to the previous described Map restoration with small changes.

The main difference is that the image block centers are not available (they have not been transmitted). They are calculated training a new MSCL ($MSCL_{IC2}$), with the coordinates of each pixel, and the magnitude values in the Regenerated Saliency Map (magnitude that was calculated with (9) at the emitter). This neural network becomes identical to $MSCL_{IC}$. The weights of $MSCL_{IC2}$ are the centers of the image blocks, and their Voronoi regions define the masks and limits.

Once again, image indexes are presented to the look-up table created with $MSCL_{PICT}(l)$ (according to the block size) that returns the block shape. Final image is regenerated by adding means of block-saliency, masking each block and positioning it in the image (adding it to the regenerated image as we had done before with the saliency map).

4.6 Extension to Color Images

Figure 4 defines the flowchart to use MSCL in the case of color images. The process is similar to the used in the case of grayscale images, but applied to each of the color components of the image.

First, we calculate the saliency map from the color image. With this saliency map we extract and quantify blocks as described in Sect. 4.1, blocks which are restored at transmitter as mentioned in Sect. 4.2. As a result of this step we get the map block-centers, block-means and indexes. Encoding is made with the previously trained $MSCL_{PICT}(l)$ picture library.

Then, original RGB image is transformed to the L-a-b color space. The reason of selecting this color codification is that it has been demonstrated its suitability for interpreting the real world [8].

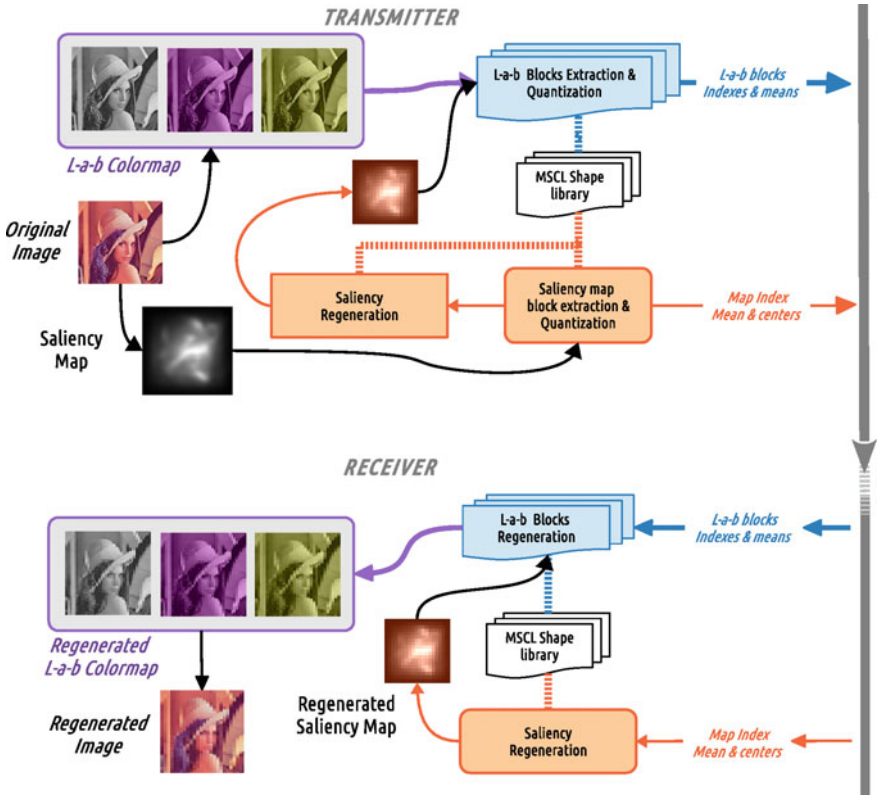

Fig. 4 Global algorithm for color images. Each color component is processed separately as in the *grayscale* method. However this process is exemplified with a different magnitude definition for the saliency map, oriented to preserve the detail of the image for certain colors selected by the user

Now with these L-a-b color components of the image, we follow the process indicated in Sect. 4.3. Each of them will be trained with a MSCL neural network ($MSCL_{IC-L}$, $MSCL_{IC-a}$, $MSCL_{IC-b}$,) and it will return the block sizes and indexes for each component. The indexes of the blocks are also encoded with $MSCL_{PICT}(l)$.

Once at receiver saliency map is restored (see Sect. 4.4). Then, we follow the image restoration step, applied to each L-a-b component. Its centers are calculated training three MSCL networks ($MSCL_{IC2-L}$, $MSCL_{IC2-a}$, $MSCL_{IC2-b}$,), with the coordinates of each pixel, and the regenerated saliency map. These neural networks becomes identical to those at the transmitter.

To get the final image, we transform the restored L-a-b image to RGB.

5 Experimental Results

5.1 Grayscale Images

Simulations were conducted on four 256×256 gray scaled images (65536 bytes), all of them are typical in image compression benchmarking tasks.

We applied the MSIC algorithm, with the following MSCL training parameters: 15 cycles and learning factor varying along the training process from 0.9 to 0.05. We used Graph-Based Visual Saliency $GBVS(\mathbf{x})$ ([9]) as the pixel saliency of the corresponding sample. However, it is possible to use other kind of magnitudes to define which areas of the image are compressed more or less deeply.

JPEG was applied with the standard Matlab implementation and a compression Quality of Q = 3 or Q = 5 (i.e., with a high compression ratio).

We also compare with the algorithm described in [5], whose main steps are followed for all the mentioned SOM based algorithms: The original image is divided into small blocks (we select a size of 8×8 to achieve a similar compression ratio to JPEG or MSCL). Then, 2-D DCT is first performed on each block. The DC term is directly send for reconstruction, and the AC terms after low-pass filtering (we only consider 8 AC coefficients) is fed to a SOM network for training or testing. All experiments were carried out with the following parameters: 256 units, 5 training cycles and β calculated so the learning factor decreases from 0.9 to 0.05.

The number of bytes used to compress each image was the same for MSCL and JPEG (see Table 1) and fixed to 2048 for SOM.

For evaluation purpose, we use the mean squared error (MSE) as an objective measurement for the performance. Table 1 shows the resulting mean of the MSE in 10 tests using our algorithm compared to JPEG and SOM applied to 4 test images. We present a second column showing the value of MSE but only calculated in those pixels which saliency is over 50 %. Standard deviation is also shown (in brackets).

To obtain the generic pictorial library $MSCL_{PICT}(l)$ we used three additional images different to the images used in testing from [10] with the same training

Table 1 Mean MSE for the whole image as well as for areas with saliency over 50 % (grayscale example)

Image	*Q/Bytes*	JPEG(Tot/50 %)	SOM(Tot/50 %)	MSIC(Tot/50 %)
Lena	*Q5/2010*	212.3/340.4	**205.4**/374.0	501.1(18.2)/**211.0(6.1)**
Street	*Q5/2127*	**302.3**/369.0	322.1/465.3	466.2(7.8)/**210.6(4.2)**
Boat	*Q5/1988*	**263.9**/383.7	280.4/486.6	436.4(12.3)/**282.0(5.6)**
Fish	*Q3/2090*	485.7/597.7	**466.3**/904.3	895.8(15.8)/**254.2(9.6)**

Standard deviation is also shown (in brackets)

parameters. This number is quite low, but enough to show the good performance of our proposal. However, in a real scenario it would be necessary to use a higher number of images to get a suitable pictorial library. Moreover, we have not used any entropic coding applied to indexes which would have result in a further compression.

As expected, the MSE value calculated for the whole image area given by JPEG is lower than the one provided by MSIC, because prototypes tend to focus on zones with high saliency while other areas in the image are under-represented.

However, when MSE was calculated taking into account only those pixels with high saliency, MSIC obtained better results than JPEG or SOM. This effect can be clearly appreciated by visual inspection of the images represented in Fig. 5. They show how MSIC achieves a higher detail level at image areas of high saliency. In the case of JPEG, it tends to fill up big portions of the image with plain blocks, being unable to obtain a good detail at any part of the image. On the other hand, SOM produces slightly blurred images due to the low frequency filtering.

The new algorithm could also be used in compression applications with other magnitude functions instead of saliency. Figure 6 shows the compressed results of applying MSIC using different Magnitude Functions to the street image. From left to right, first image is the original one, second image is MSIC using the same Magnitude Functions than the one used in (9). The Magnitude function in third image is the same of equation (9), but using $1 - GBVS(x)$ instead of the pixel saliency of the corresponding sample. The fourth image uses the value of the vertical coordinate (normalized to one) and finally the fifth one uses the value of the vertical coordinate (normalized to one) minus one. It can be clearly seen that depending on the defined Magnitude Function, certain areas are compressed in with quality (foreground, background, top or bottom of the image).

This toy example was only presented to show the possibilities of achieving selective compression in different areas of the image just by varying the Magnitude Function.

MSIC algorithm is much more slower than JPEG. In a serial execution on single core computer, JPEG processing takes only 0.11 % of the total processing time of MSIC (that in our tests it take 6.8 s for compressing each of the grayscale test images). Most of the time (91.6 %) is spent on block extraction (34 % of which is used in extracting blocks from the saliency map and 66 % in extracting blocks from the

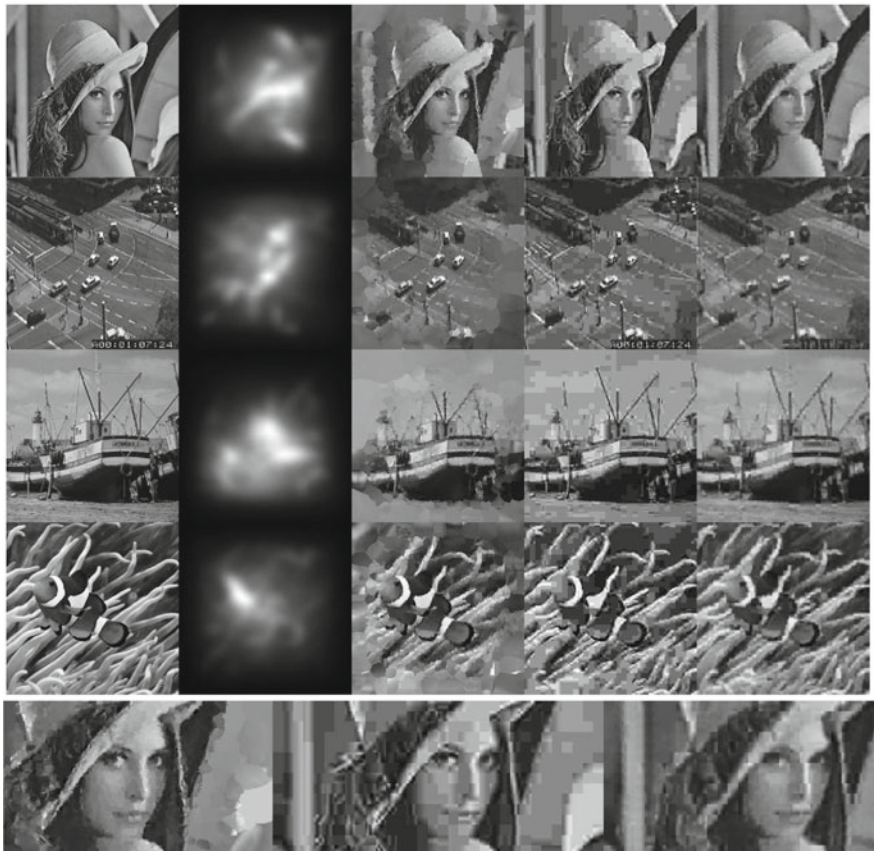

Fig. 5 *Top in columns* Original image, saliency map, MSIC, JPEG and SOM compression for the test images. *Bottom* Lena detail in the three methods. It can be clearly seen that the Lena face, compressed with MSIC shows a more natural view (almost like painted with Pointillism technique) than the other methods that have *square block borders*

Fig. 6 Original 'Street' image and the compressed images using MSIC with four different magnitude functions

image). Block encoding and decoding takes 6.7 % of the time, and 1.7 % the rest of the algorithm.

However processing time can be reduced using parallel processing and compiled libraries (now simulated in Matlab). The slowest task is finding the best matching unit for both, defining the Voronoi region to extract a block, and for encoding-decoding. This task represents the 68 % of the block extraction time, and the 51 % of the encoding-decoding time. It is a slow process because in our sequential implementation we must, for each sample, calculate the distances from sample to each of the units. In a parallel implementation, this processing could be applied simultaneously for all units. Then using for instance 1000 units, block extraction time could be only 29.3 % of initial total time. Using similar approach for encoding-decoding the final processing time can be reduced to be 2.3 s (34.3 % of the original processing time).

5.2 Color Images

In the color experiments, it is applied the same method explained in Sect. 4.6, with the same parameters used in the grayscale case.

We use a different saliency definition focused in those image zones with colors selected by the user. This type of compression, preserving with more detail image zones with certain color selection, may have different applications. For instance, in medical images, the specialist may define the colors of those areas that has to be well preserved. Other application is in video transmission limited by narrow bandwidths, as in underwater image transmission. In that case it is possible to work with a highly compressed global image, and if the user wants a higher definition in areas of a specific color, MSIC could get to a better definition of those areas, obviously degrading others to keep the limited bandwidth.

To calculate the saliency map with the magnitude values for the pixels, we first calculate the saliency map for each color in the set of colors. The saliency map of a selected color is obtained by binarizing the image, based on thresholding the distance of the pixel color and the selected color. Then we apply a border detection algorithm to get the edges of the image zones painted in that color.

The saliency map of the image is obtained as the maximum of the filtered edge images for all the set of colors. Using this value of magnitude, we get more units in the interesting regions whose colors are similar to the defined set. JPEG was implemented using Matlab and different compression qualities.

The experiments use the four test images depicted in the first column of Fig. 7. The second column shows the resulting saliency maps for the images. To maintain the details of the fish in the first image, it is used as color set: orange and white. The flower image uses dark and clear pink, the boat image uses only brown and the parachute image uses pink and black from the parachutist.

Table 2 shows in the first column the resulting mean of the MSE in 10 tests using MSCL compared to JPEG. Second column shows the value MSE calculated in

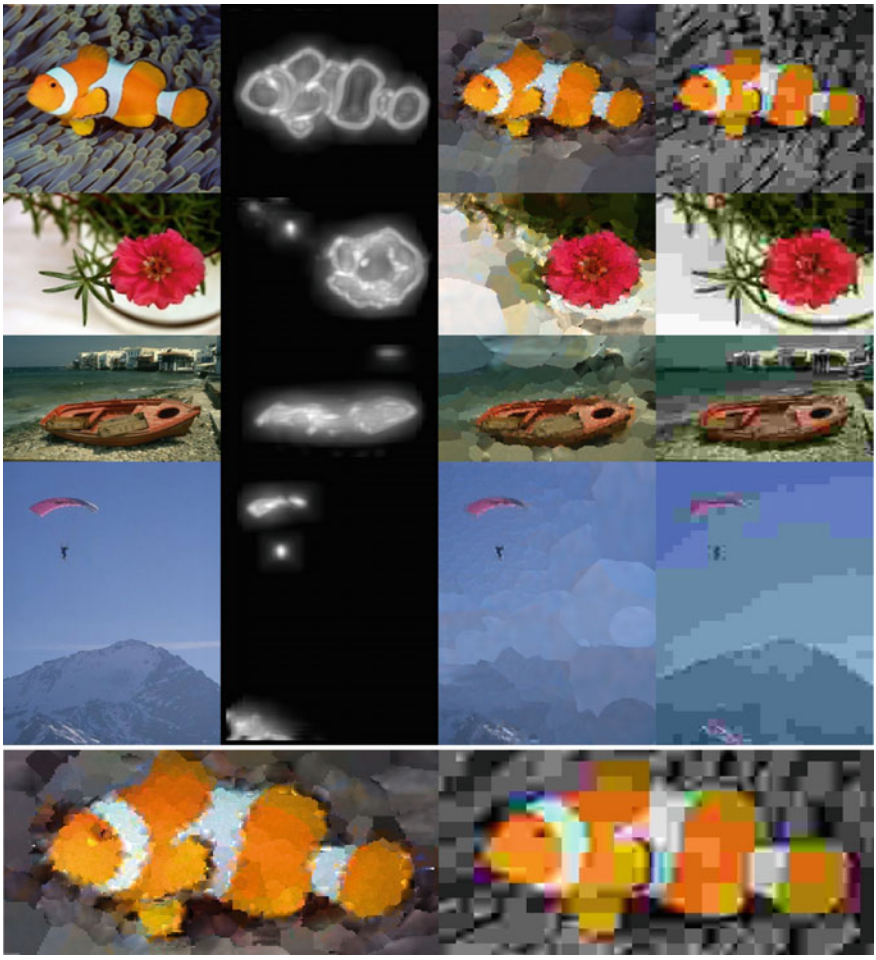

Fig. 7 *Top in columns* Original color image, saliency map generated for a one or two-color selection (fish with *orange* and *white*; flower with *dark* and *clear pink*; boat with *brown*; parachute with *pink* and *black*), MSIC and JPEG compression for the test images. *Bottom* Fish image detail in both compression methods

those pixels with saliency over 50 %. Standard deviation is also shown (in brackets). Number of bytes and quality are also shown.

As expected, the MSE value calculated for the whole image area is lower using JPEG than the one provided by MSIC. However, when MSE was calculated taking into account only those pixels exhibiting a high saliency, MSIC obtained the best results.

Table 2 Mean MSE for the whole image as well as for areas with saliency over 50 % (color example)

Image	*Q/Bytes*	JPEG(Tot/50 %)	MSIC(Tot/50 %)
Fish	*Q3/1702*	**1328**/2695	2193(20.7)/**1789(40.3)**
Flower	*Q5/1722*	**862**/1299	3540(227.1) /**1167(49.4)**
Boat	*Q6/1720*	**1303**/1570	2366(87.4)/**1190(25.3)**
Sky	*Q5/1706*	967/2312	**240(58.2) /468(19.7)**

Standard deviation is also shown (in brackets)

6 Conclusions

In this paper we have shown how grayscale and color images compressed with MSIC exhibit a higher quality in relevant areas of the image when compared to other compression methods such as JPEG or SOM based algorithms.

MSIC has been proved to be a reliable and efficient approach to achieve selective Vector Quantization. This selectivity can be used in image compression to set the block centers focused on certain areas of the image to be compressed in a further step by Vector Quantization. The novelty of the algorithm is that areas of interest, which can be defined by a magnitude function, would receive lower compression than the rest of the image. Another novelty of the algorithm is that the image composition uses irregular blocks of pixels that tend to be smaller in zones of high interest and broader in zones of low interest.

These properties of the algorithm may be modulated for different applications by choosing the adequate magnitude function according to the desired task. For instance, it could be a good choice to use the Viola-Jones algorithm instead of GBSV to highlight some particular areas when dealing with facial areas in images with people. Another potential application is the compression of satellite and aerial imagery of the Earth. In that case, Automatic Building Extraction from Satellite Imagery algorithms may be used to define the areas of interest. Then, MSIC may compress the images keeping higher detail in the built areas. In a similar way, medical image storage tools might use MSIC to save images compressed with higher detail in certain biological tissues or anatomical structures.

Several applications that require image transmission with low bandwidth may use the algorithm, as in underwater image transmission, where there are low data rates compared to terrestrial communication. Another example of magnitude would be simply the predicted position of the user's fovea on the image in the next frame. This magnitude is useful for application in virtual reality glasses, where the image zone, that is predicted the user is going to focus his fovea, will present the highest detail, while surrounding zones can be more compressed.

Future work comprises several research lines such as the use of entropy coding for the information of each compressed image block, filtering each image with DCTs, and comparison against other compression algorithms. Another point to be analysed is the kind of images used to generate the generic pictorial codebooks used for

compression and restoration, as the library of training images can be selected for the chosen task. The test of the algorithm in different tasks as mentioned in the previous paragraph is another research line left for future.

References

1. Pelayo, E., Buldain, D., Orrite, C.: Magnitude sensitive competitive learning. Neurocomputing **112**, 4–18 (2013)
2. Laha, A., Pal, N., Chanda, B.: Design of vector quantizer for image compression using self-organizing feature map and surface fitting. IEEE Trans. Image Process. **13**(10), 1291–1303 (2004)
3. Amerijckx, C., Legat, J.D., Verleysen, M.: Image compression using self-organizing maps. Syst. Anal. Modell. Simul. **43**(11), 1529–1543 (2003)
4. Harandi, M., Gharavi-Alkhansari, M.: Low bitrate image compression using self-organized kohonen maps. In: Proceedings 2003 International Conference on Image Processing, ICIP'03, vol. 3, pp. 267–270 (2003)
5. Liou, R.J., Wu, J.: Image compression using sub-band DCT features for self-organizing map system. J. Comput. Sci. Appl. **3**(2) (2007)
6. Kohonen, T.: The self-organizing map. Neurocomputing **21**(1), 1–6 (1998)
7. Ahmed, N., Natarajan, T., Rao, K.R.: Discrete cosine transform. IEEE Trans. Comput. **100**(1), 90–93 (1974)
8. Cheung, Y.: On rival penalization controlled competitive learning for clustering with automatic cluster number selection. IEEE Trans. Knowl. Data Eng. **17**, 1583–1588 (2005)
9. Harel, J., Koch, C., Perona, P.: Graph-based visual saliency. In: NIPS'06, pp. 545–552 (2006)
10. Computer Vision Group, U.o.G.: Dataset of standard 512×512 grayscale test images. http://decsai.ugr.es/cvg/CG/base.htm (2002)