International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:4, No:3, 2010

Single Frame Supercompression of Still Images, Video, High Definition TV and Digital Cinema

Mario Mastriani

***Abstract*—**Super-resolution is nowadays used for a high-resolution image produced from several low-resolution noisy frames. In this work, we consider the problem of high-quality interpolation of a single noise-free image. Such images may come from different sources, i.e., they may be frames of videos, individual pictures, etc. On the other hand, in the encoder we apply a downsampling via bidimen-sional interpolation of each frame, and in the decoder we apply a upsampling by which we restore the original size of the image. If the compression ratio is very high, then we use a convolutive mask that restores the edges, eliminating the blur. Finally, both, the encoder and the complete decoder are implemented on General-Purpose computation on Graphics Processing Units (GPGPU) cards. In fact, the mentioned mask is coded inside texture memory of a GPGPU.

***Keywords*—**General-Purpose computation on Graphics Processing Units, Image Compression, Interpolation, Super-resolution.

I. Introduction

DIGITAL image capture produces discrete representations of continuous scenes. This discretisation in both space and intensity is a sampling process that creates aliasing, and infor-mation at frequencies above the Nyquist rate is lost. It is common to wish to construct a higher resolution image from a template image or a set of images, but the aliasing and loss of frequency information makes this an ill-posed (inverse) problem.

The typical solution to this problem (known in the literature as image super-resolution reconstruction, or simply super-resolution) is to use an ensemble of related lower-resolution images. As each of these images has aliased the higher frequency information slightly differently, under certain conditions it is possible to "unwrap" some of the aliasing and reconstruct the lost higher frequencies.

There are numerous methods [1-5] of performing super-resolution. Many of them are computationally expensive in nature, but allow for complicated motion models, significant noise and image degradation, and other aspects that are not considered in this work. Given assumptions of global translational motion, low noise and linear space and time invariant blur due to the imaging sensor point spread function (PSF), image super-resolution reconstruction can be split into there distinct steps:

- Registration
- Reconstruction/Interpolation
- Deblurring

Image registration is a technique that can be used to determine the relative translations between the input images. Generally, the desire is to do this from the contents of the images alone, without any prior knowledge. There are many different methods for performing registration [1]; however, in the context of image super-resolution, image registration is required to determine the offsets between the images with accuracy down to a small fraction of a pixel [1]**.**

Once the images have been registered, all the pixels from the ensemble can be combined to form a composite image. The resultant image is no longer sampled on a uniform rectangular grid, but due to global translational motion, it has a semi-uniform structure, as can be seen in Figure 1. Reconstructing the image data at all points on a high resolution grid requires that the semi-regular data is interpolated and resampled. It is this interpolation problem that is addressed in this paper.

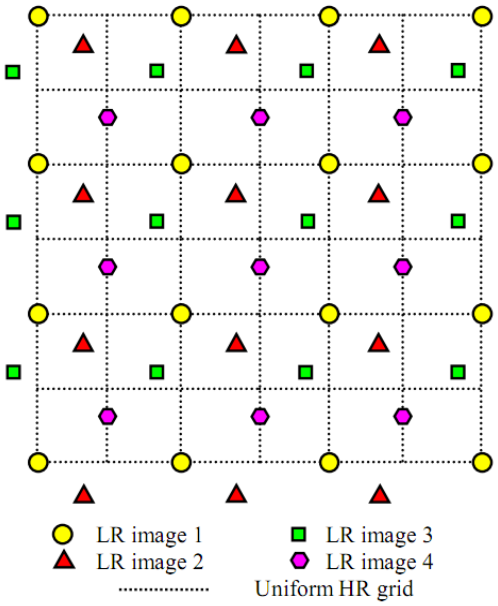

Fig.1: Composite image exhibits a semi-uniform structure.

Mario Mastriani is with the Grupo de Investigación sobre Procesamiento de Señales e Imágenes (GIPSI), Univ. Nac. de Tres de Febrero (UNTreF), 910 Florida St., Floor 6th, Room B, (C1005AAT), CABA, Argentina. phone: +54-11-4015-2295; fax: +54-11-4893-2204; e-mail: mmastriani@untref.edu.ar.

For the full super-resolution approach, a deblurring procedure can now be applied that restores the high frequencies that have been suppressed by the low-resolution imaging process. In this paper we perform this deblurring after the interpolated process via a convolution between the up-sampled image and a convolutive mask deduced specifically.

However, the main goal of Super-Resolution (SR) methods –in practice– is to recover a high-resolution image from one or more low-resolution input images. Methods for SR can be broadly classified into two families of methods:

1. The classical multi-image super-resolution, and
2. Example-Based super-resolution.

In the classical multi-image SR (e.g., [2] to name just a few) asset of low-resolution images of the same scene are taken (at subpixel misalignments). Each low-resolution image imposes asset of linear constraints on the unknown high-resolution intensity values. If enough low-resolution images are available (at subpixel shifts), then the set of equations becomes determined and can be solved to recover the high-resolution image. Practically, however, this approach is numerically limited only to small increases in resolution [2] (by factors smaller than 2).

These limitations have lead to the development of "Example-Based Super-Resolution" also termed "image hallucination" (see references in [2]). In example-based SR, correspondences between low and high-resolution image patches are learned from a data base of low and high-resolution image pairs (usually with a relative scale factor of 2), and then applied to a new low-resolution image to recover its most likely high-resolution version. Higher SR factors have often been obtained by repeated applications of this process. Example-based SR has been shown to exceed the limits of classical SR. However, unlike classical SR, the high resolution details reconstructed ("hallucinated") by example-based SR are not guaranteed to provide the true (unknown) high-resolu-tion details.

Sophisticated methods for image up-scaling based on learning edge models have also been proposed (e.g., [3-5]). The goal of these methods is to magnify (up-scale) an image while maintaining the sharpness of the edges and the details in the image. In contrast, in SR (example-based as well as classical) the goal is to recover new *missing high-resolution details* that are not explicitly found in any individual low-resolution image (details beyond the Nyquist frequency of the low-resolution image). In the classical SR, this high-frequency information is assumed to be split across multiple low-resolution images, implicitly found there in aliased form. In example-based SR, this missing high-resolution information is assumed to be available in the high-resolution data base patches, and learned
from the low-res/high-res pairs of examples in the database.

However beyond what we have said above, in this paper, we consider the interpolation of a single image.
On the other hand, the rendering of lower resolution image data on higher resolution displays has become a very common task, in particular because of the increasing popularity of webcams, camera phones, and low-bandwidth video streaming. Thus, there is a strong demand for real-time, high-quality image magnification. In this work, we suggest to exploit the high performance of general-purpose computation on program-mable graphics processing units (GPGPUs) for an original image magnification method. To this end, we propose a GPGPU-friendly algorithm for image up-sampling with edge restoration image interpolation, which avoids ringing artifacts, excessive blurring, and stair-casing of oblique edges. At the same time it features gray-scale invariance, is applicable to color images, and allows for real-time processing of full-screen images on today's GPGPUs [6-9].

The Bidimensional Interpolation is outlined in Section II, where we discuss the problem of interpolating visually acceptable images at a higher resolution. We first present the interpolation problem and why linear interpolation filters are inade-quate for image data. To represent the major mathematical approaches to image processing, we discuss and evaluate five different image interpolation methods. Super-resolution sche-me for compression including linear interpolation are outlined in Section III. Metrics are outlined in Section IV. Simulations are outline in Section V. Finally, Section VI provides a conclu-sion of the paper.

II. Bidimensional Interpolation

A digital image is not an exact snapshot of reality, it is only a discrete approximation. This fact should be apparent to the average web surfer, as images commonly become blocky or jagged after being resized to fit the browser. The goal of image
Interpolation is to produce acceptable images at different resolutions from a single low-resolution image. The actual resolution of an image is defined as the number of pixels, but the effective resolution is a much harder quantity to define as it depends on subjective human judgment and perception. The goal of this section is to explore different mathematical formulations of this essentially aesthetic quantity.

The image interpolation problem goes by many names, depending on the application: image resizing, image up-sampling/down-sampling, digital zooming, image magnification, resolution enhancement, etc. The term super-resolution is sometimes used, although in the literature this generally refers to producing a high-resolution image from multiple images such as a video sequence. If we define interpolation as "filling in the pixels in between," the image interpolation problem can be viewed as a subset of the inpainting problem (see Figure 2).

The applications of image interpolation range from the common place viewing of online images to the more sophisticated magnification of satellite images. With the rise of consumer-based digital photography, users expect to have a greater control over their digital images. Digital zooming has a role in picking up clues and details in surveillance images and video. As high-definition television (HDTV) technology enters the marketplace, engineers are interested in fast interpolation algorithms for viewing traditional low-definition programs on HDTV. Astronomical images from rovers and probes are received at an extremely low transmission rate

(about 40 bytes per second), making the transmission of high-resolution data infeasible [10]. In medical imaging, neurologists would like to

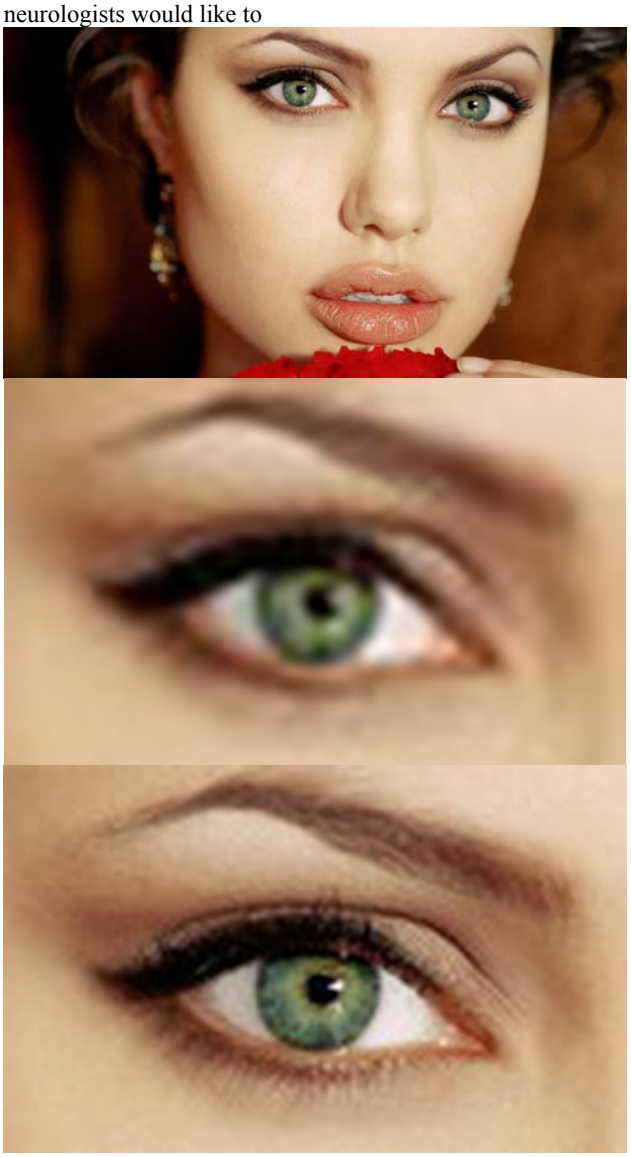

Fig.2: Image interpolation using linear method of interp2 built-in MATLAB® function. Top: original image. Medium: close-up of eye in image. Down: interpolated image.

have the ability to zoom in on specific parts of brain tomography images. This is just a short list of applications, but the wide variety cautions us that our desired interpolation result could vary depending on the application and user.

A. The Image Interpolation Problem

In this section, we will establish the notation for image interpolation used throughout the paper. Suppose our image is defined over some rectangle $\Omega \subset \Re^2$. Let the function $f : \Omega \to \Re$ be our ideal continuous image. In an abstract sense, we can think of *f* as being "reality" and Ω as our "viewing window". Our observed image u_0 is a discrete sampling of *f* at equally spaced points in the plane. If we suppose the resolution of u_0 is $\delta x \times \delta y$, we can express u_0 by

$$u_0(x,y) = C_{\delta x,\delta y}(x,y) f(x,y), \qquad (x,y) \in \Omega \tag{1}$$

where *C* denotes the Dirac comb:

$$C_{\delta x,\delta y}(x,y) = \sum_{k,l \in Z} \delta(k\delta x, l\delta y), \qquad (x,y) \in \Re^2. \tag{2}$$

The goal of image interpolation is to produce an image *u* at a different resolution $\delta x' \times \delta y'$. For simplicity, we will assume that the Euclidean coordinates are scaled by the same factor *K*:

$$u(x,y) = C_{\frac{\delta x}{K},\frac{\delta y}{K}}(x,y) f(x,y), \qquad (x,y) \in \Omega. \tag{3}$$

Given only the image u_0, we will have to devise some reconstruction of *f* at the pixel values specified by this new resolution. We will refer to *K* as our zoom or magnification factor. Obviously, if *K* = 1 we trivially recover u_0 .The image u_0 is upsampled if $K \uparrow 1$ and downsampled if $K \downarrow 1$. In this paper, we will focus on the upsampling case when $K \uparrow 1$ is an integer.

Let $\Omega_K \subset \Omega$ denote the lattice induced by (3) for a fixed zoom *K*. Note that the lattice of the original image u_0 in (2) is Ω_1. Also note that for infinite magnification we obtain $\Omega_K \subset \Omega$ as $K \to \infty$. For computation purposes, we can shift the lattices to the positive integers. So if the observed image u_0 is an *m* x *n* image,

$$\Omega_K = [1,2,\ldots,Km] \times [1,2,\ldots,Kn], \qquad K \in Z_+. \tag{4}$$

Many interpolation techniques impose the constraint $\Omega_1 \subseteq \Omega_K$. In this case, only a subset of the pixels in Ω_K needs to be determined and the interpolation problem becomes a version of the inpainting problem.

Given the notation above, we can state the image interpolation problem succinctly: Given a low-resolution image $u_0 : \Omega_1 \to \Re$ and a zoom $K \uparrow 1$, find a high-resolution image $u : \Omega_K \to \Re$. Obviously, this is an ill-posed problem. We need to impose assumptions on the reconstruction of *f* in equation (3). The choice of interpolation technique depends on the choice of assumptions. In other words, we need a mathematical understanding of what constitutes our perception of "reality" *f*.

Interpolation methods differ in their mathematical description of a "good" interpolated image. Although it is difficult to compare methods and judge their output, [10] proposes 9

basic criteria for a good interpolation method. The first 8 are visual properties of the interpolated image, the last is a computational property of the interpolation method.

1) *Geometric Invariance*: The interpolation method should preserve the geometry and relative sizes of objects in an image. That is, the subject matter should not change under interpolation.
2) *Contrast Invariance*: The method should preserve the luminance values of objects in an image and the overall contrast of the image.
3) *Noise*: The method should not add noise or other artifacts to the image, such as ringing artifacts near the boundaries.
4) *Edge Preservation*: The method should preserve edges and boundaries, sharpening them where possible.
5) *Aliasing*: The method should not produce jagged or "staircase" edges.
6) *Texture Preservation*: The method should not blur or smooth textured regions.
7) Over-smoothing: The method should not produce undesirable piecewise constant or blocky regions.
8) *Application Awareness*: The method should produce results appropriate to the type of image and order of resolution. For example, the interpolated results should appear realistic for photographic images, but for medical images the results should have crisp edges and high contrast. If the interpolation is for general images, the method should be independent of the type of image.
9) *Sensitivity to Parameters*: The method should not be too sensitive to internal parameters that may vary from image to image.

Of course, these are qualitative and somewhat subjective criteria. We unlike [10], do not hope to develop a mathematical model of image interpolation and error analysis, but simply apply the most efficient method for our development. In a sense, the method employed in this paper presents a mathema-tical model of these visual criteria.

B. Linear Interpolation Filters

The simplest approach is to assume that f in equation (3) is reconstructed by a convolution kernel $\varphi : \Re^2 \to \Re$ where $\int \varphi(x,y)\,dy\,dx = 1$. Then we can approximate f by

$$f \approx u_0 * \varphi . \tag{5}$$

Substituting this into (3) gives rise to a general linear interpolation filter

$$u(x,y) = C_{\frac{\delta x}{K},\frac{\delta y}{K}}(x,y).(u_0 * \varphi)(x,y), \qquad (x,y) \in \Omega. \tag{6}$$

The simplest linear filters are the bilinear and bicubic interpolation, which assume the pixel values can be fit locally to linear and cubic functions, respectively [10]. Along with simple nearest neighbor interpolation, these two filters are the most common interpolation schemes in commercial software. These methods are easy to code as matrix multiplications of u_0. However, an image contains edges and texture, in other words

Fig.3: Part of Lena image down-sampled and then up-sampled by

factor K = 16. Top: Original, Second: Nearest Neighbor, Third: Bilinear, and Down: Bicubic.

discontinuities. So the assumptions that pixel values locally fit a polynomial function will produce undesirable results. The bilinear and bicubic interpolation methods [10] may introduced blurring, create ringing artifacts, and produce a jagged aliasing effect along edges (see Fig.3). The blurring effects arise from the fact that the methods compute a weighted average of nearby pixels, just as in Gaussian blurring. The aliasing effects arise because the linear filters do not take into conside-ration the presence of edges or how to reconstruct them.

Other linear interpolation filters include quadratic zoom, the B-spline method, and zero-padding. But these schemes produce the same undesirable effects as the bilinear and bicubic methods, as mentioned in [10]. Linear filters differ in the choice of φ, which essentially determine show to compute the weighted average of nearby pixels. While this is a natural interpolation scheme for general data sets, this is not necessarily appropriate for *visual* data. In order to improve upon these linear filters, we need to consider interpolation methods that some how quantify and preserve visual information.

C. Which Methods to Consider?

Generally speaking, mathematical approaches to image processing can be divided into five categories:

1. Partial-Differential Equation (PDE)-Based Methods (e.g. heat diffusion, Perona-Malik, Navier-Stokes, and mean curvature).
2. Variations of Energy (e.g. Total Variation, Mumford-Shah, active contours)
3. Multiscale Analysis (e.g. wavelets, Fourier analysis, Gabor analysis, Laplacian pyramids)
4. Machine Learning (e.g. unsupervised learning, data mining, Markov networks)
5. Statistical / Probabilistic Methods (e.g. Bayesian inference, Natural Scene Statistics, pattern theory)

We are trying to describe the field in broad terms, but not to rank or pigeonhole work in computer vision. Indeed, many techniques such as TV-wavelets inpainting certainly do not fit into one category. Also, these methods differ at the mathematical level, but not necessarily at the conceptual level. For example, some versions of the TV energy can be minimized by solving a PDE or by optimizing a variation of energy.

In our attempt to survey recent work in image interpolation and also display the variety of mathematics used, we will highlight one method from each of the five categories [10]

1. A PDE-Based Approach: anisotropic heat diffusion
2. A Variation of Energy Approach: Mumford-Shah inpainting
3. A Multiscale Approach: wavelet-based interpolation
4. A Machine Learning Approach: LLE-based neighbor embeddings
5. A Statistical Approach: NL-means interpolation

These methods are, in some sense, representative of the mathematical approaches to the image interpolation problem and, in a larger sense, to the field of image processing. For example, the heat equation is the most studied PDE in image processing and the Mumford-Shah energy has generated dozens, if not hundreds, of research papers. However, these methods have a number of disadvantages to its implementation, reason, we choose a configuration based on linear interpolation. The main disadvantages are:

1. Their hard coding
2. They depend heavily on initial conditions
3. Their high computational complexity
4. Their visual quality is not superior to linear interpolation, except for high levels of downsampling/upsampling, with automatically means a high rate of compression/decompression.

In the latter case, we use a convolutive mask [11-14] to enhance the edges, as discussed in the next section.

III. Super-resolution Scheme For Compression

This section is organized into four parts, for a better understanding of the concepts:

A. Super-resolution vs Deblurring,
B. Compression vs Super-compression,
C. Deduction of the mask
D. Applications

A. Super-resolution vs Deblurring:

As we saw in Section I, there is much confusion between the concepts of super-resolution and deblurring in Digital Image Processing [15, 16]. We are going to establish here two rigo-rous definitions for the purpose of eliminating this confusion.

We say that a process is super-resolution if it restores the sharpness of an image involving an increase in the resolution of the same [1-5, 17-19], see Appendix.

We say that a process is deblurring if it restores the sharpness of an image not involving an increase in the resolution of the same. This process is applied when the image sharpness suffers an aberration called blur [15, 16]*, which comes from a high relative speed of the object in focus in relation to the camera, fast opening and closing the shutter, etc.*

We consider important to mention that both processes can involve each other as part of the process of improving the sharpness of the image. In fact, we can understand the super-resolution as a process of increasing the resolution followed by a restoration of the edges by a deblurring process. On the other hand, previously established definitions are fundamental to understanding what follows.

B. Compression vs Super-compression:

We define compression as the process reduces the average number of bit-per-pixel (bpp) of an image. In Fig. 4, we represent the set of bit-planes in which decomposes a gray or color image. As seen in Fig. 4, the compression process does not alter the image size [15, 16].

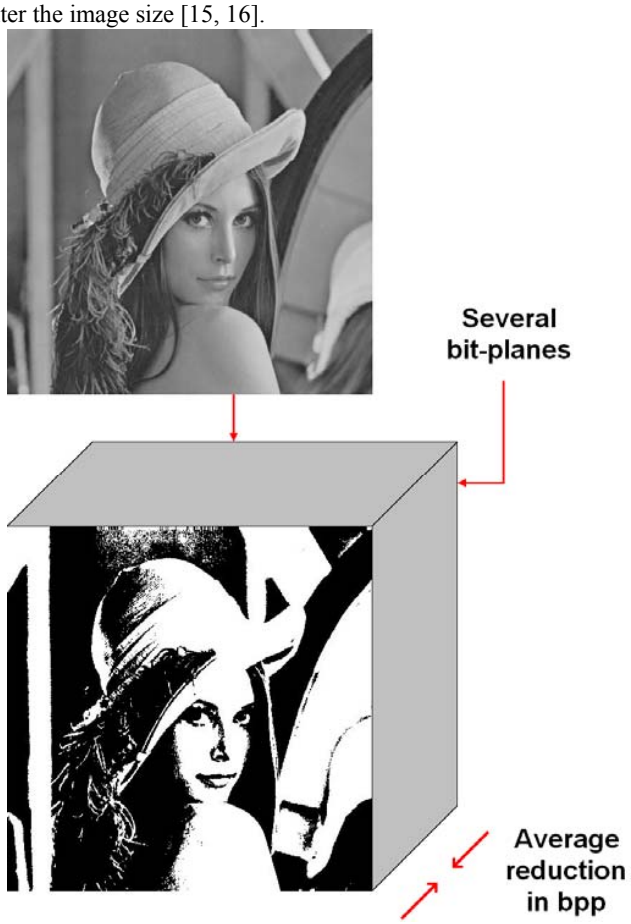

Fig.4: Compression.

Instead, we define supercompression as the process reduces the average number of bit-per-pixel (bpp) of an image after downsizing. The size reduction process is performed by down-sampling, which takes shrinkage in rows and columns, without obligation to respect the aspect ratio (16:9). In fact, for ISDB-Tb (Integrated Services Digital Broadcasting) Brazilian Digital TV System we use 5:1 as compression rate over the original compression of the system, which uses H.264 as video compression standard [20]. When we say, *we increase the standard compression 5 times*, this means that we move from a resolution of 1920x1080 (Full-High Definition: Full-HD) to another 5 times lower of 720x576 (Standard Definition: SD). The standard video com-pression H.264 is not affected by the supercompression. As discussed in Sub-Section D, supercompression requires mini-mal equipment at the transmitter and the reverse procedure to supercompression in the receiver (set-top-box) [21]. However, the unavailability of the latter, the system is compatible, since the receiver will send the SD signal to the Liquid Display Crystal (LCD) TV, which naturally made upsampling obvious-ly changing the aspect ratio, as when a Full-HD LCD TV receive a SD signal. In Fig. 5, we represent the set of bit-pla-nes in which decomposes a gray or color image.

As discussed in Sub-Section D, our supercompression procedure consists in two parts spread in transmitter and receiver.

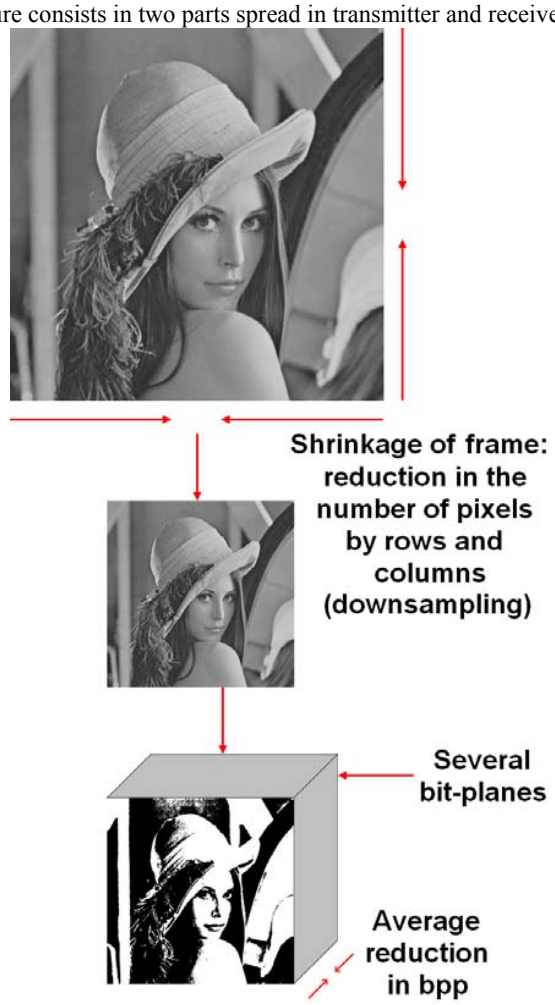

Fig.5: Supercompression.

In transmitter we have three steps:
1. Video slicing: frame-by-frame
2. Downsampling
3. Video reassembling

and in receiver inside set-top-box we have four steps:
1. Receiver of streaming/H.264
2. H.264^{-1}
3. Upsampling
4. Deblurring

In our case, the downsampling and upsampling is done with bilinear interpolation, while the deblurring is done by a bidi-mensional convolutive mask of NxN pixels, which makes a

rafter over the upsampled (blurred) image. The parameters of this squared mask (where N is odd) are criticals, therefore, the such parameters must be calculated and adjusted with total accuracy.

In the next section, we will proceed to deduct the mask and set the optimal relationship between its parameters. Later we will proceed to adjust them via a Genetic Algorithm [22].

C. Deduction of the mask:

Based on the last section, the single frame is recovered after suffering a pair of processes: downsampling and upsampling, see left side of Fig.6. In this figure:

$\mathbf{X_t}$ means original single frame.
$\mathbf{Y_t}$ means recovered (blurred) single frame.
$\mathbf{M_b}$ means square mask of NxN pixels (where N is odd). This mask is known as a blurred mask, smoothing operator or Point Spread Function (PSF) [1].
Sub-index ***t*** means *t-iteration*.
$\downarrow$ means downsampling.
$\uparrow$ means upsampling.

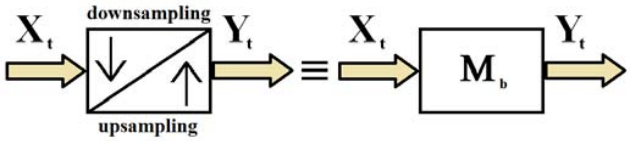

Fig.6: Downsampling/upsampling as a blurred mask.

In these processes ($\downarrow$ and $\uparrow$), the single frame is affected by a space/time invariant blur, we which interpret as the result of the action of a bidimensional convolution between the origi-nal single frame and a mask known in Digital Image Pro-cessing as a mask of mean filtering. The idea of mean filtering is simply to replace each pixel value in an image with the mean (`average') value of its neighbors, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Mean filtering is usually thought of as a convolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighbor-hood to be sampled when calculating the mean. Often a 3×3 square kernel is used, although larger kernels (e.g. 5×5 squa-res) can be used for more severe smoothing. (Note that a small kernel can be applied more than once in order to produce a similar but not identical effect as a single pass with a large kernel). In Fig.7, we consider the most general case, for NxN kernel, always with odd N, where:

$$\varphi = \frac{1}{N \times N} \tag{7}$$

Computing the straightforward convolution of an image with this kernel carries out the mean filtering process.

On the basis of the above, we need an estimator to recover the single frame of the processes affecting it. Then, for an image affected by a bidimensional convolution with a mask as Fig.7, we deduce that the best estimator is a Constant Model Kalman's filter [23].

The set of equations reflecting the above model can be divided into two stages: the model and the estimator [23].

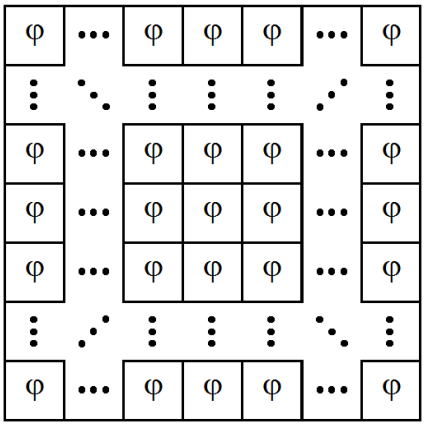

Fig.7: N×N averaging kernel often used in mean filtering.

Based on Fig.8, where Δ means unitary delay for each element of single frame $\boldsymbol{X}_{t+1}$, we have:

Model:

$$X_{t+1} = X_t \tag{8}$$

$$Y_t = M_b \otimes X_t \tag{9}$$

Where $\otimes$ means bidimensional convolution.

Estimator:

$$\widehat{X}_{t+1} = \widehat{X}_t + K \times \varepsilon_t \tag{10}$$

$$K = k \times I \text{ (Kalman's gain)} \tag{11}$$

$$\varepsilon_t = Y_t - \widehat{Y}_t \tag{12}$$

$$\widehat{Y}_t = M_b \otimes \widehat{X}_t \tag{13}$$

$$P_t = (I - k \times I) \times P_t^- = (1-k) \times P_t^- \tag{14}$$

Where ***I*** means identity matrix, and $0<k<2$ is a constant parameter to adjust. Therefore,

$$\lim_{t \to \infty} P_t = 0 \tag{15}$$

Being ***O*** the null matrix (all its elements equal to zero), and originally,

$$P_t = E\left\{ \varepsilon_t \times \varepsilon_t^T \right\} \tag{16}$$

Where $E\{\bullet\}$ represents the mathematical expectation of "•".

On the other hand, the computational implementation of the above set of equations involves the use of four nested *for*'s plus a strict control of the stability of the Eq.14 from restricting the possible values of k, i.e., only it is possible to use $0<k<2$.

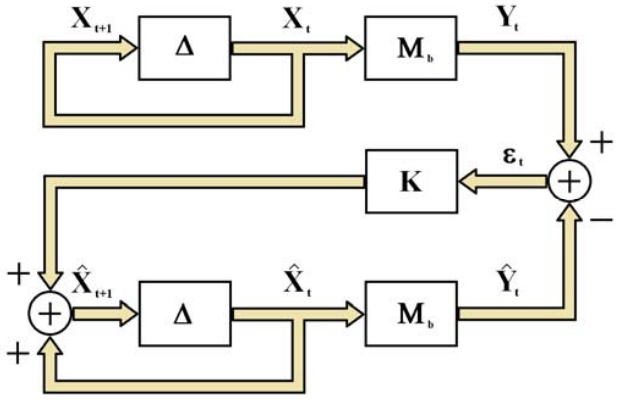

Fig.8: Constant Model Kalman's Filter.

Therefore, it is much more efficient to implement such filtering through a simple bidimensional mask convolution, eliminating the predictor form of Eq.10, which allows much more efficient implementations using - for example - a convolution through the Fast Fourier Transform (FFT) [15, 16]. In conse-quence, we need deduce such mask. If we replace Eq.13 in Eq.12, we have,

$$\varepsilon_t = Y_t - M_b \otimes \hat{X}_t \tag{17}$$

Now, we replace Eqs. 11 and 17 inside Eq.10, obtaining,

$$\hat{X}_{t+1} = \hat{X}_t + k \times I \times \left(Y_t - M_b \otimes \hat{X}_t\right) \tag{18}$$

Reagrouping terms of Eq.18, and remembering a model of low noise and linear space and time invariant blur, we have,

$$\hat{X}_{t+1} = M_d \otimes Y_t \tag{19}$$

Where $\boldsymbol{M_d}$ is a mask as shown in Fig.9, and the following relationships to consider are very important,

$$(N^2 - 1) \times \alpha + \beta = 1, \text{ (for deblurring)} \tag{20}$$

$$(N^2 - 1) \times \alpha + \beta = 0, \text{ (for edge detection)} \tag{21}$$

Thus, a new and simplified model of deblurring appears on the scene, see Fig.10, where $\alpha < 0$ and $\beta > 1$. We need to establish precisely both parameters, then, there are two possible ways forward:

1. Choose N (integer, positive, odd and small), and $\beta > 1$ (and arbitrarily less than 2), then α is derived from Eq.20.
2. Start with arbitrary values of α and β (about certain recommendations, e.g., $-0.1 < \alpha < 0$ and $1 < \beta \leq 2$) and generating a random population of the pair $[\alpha, \beta]$, and deducting $\boldsymbol{N}$ from Eq.20.

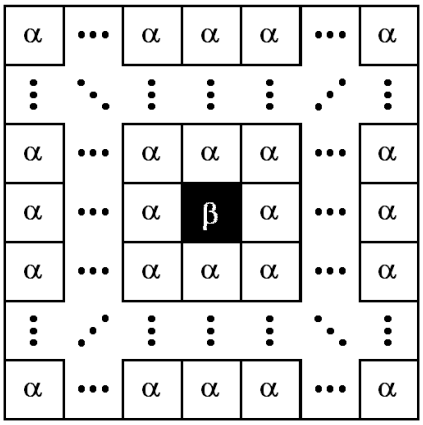

Fig.9: Deblurring mask $\mathbf{M_d}$.

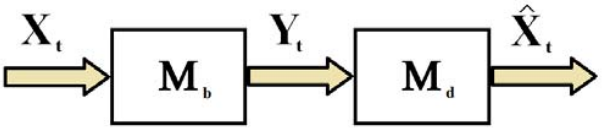

Fig.10: New and simplified model of deblurring.

The mentioned pair serves of initial population for the Genetic Algorithm [22] of Fig.11, where the pair is called chromosome, and α and β are called genes. The metric for the adjustment is the Mean Squared Error (MSE), which is defined in the next section [15, 16].

On the other hand, the employed Genetic Algorithm is composed of three big modules:

a) Scoring,
b) Crossover Operator, and
c) Mutation Operator

The first consists of the following submodules:

a.1) Set-point where the error pixel-by-pixel arises
a.2) The MSE calculation with the error pixel-by-pixel
a.3) Sorting from minimum to maximum MSE
a.4) Genocide Operator eliminates the chromosomes with biggest MSE, i.e., there are a fixed number of chromosomes that survive per cycle, the fittest. Such fixed number is a design parameter of the Genetic Algorithm.

The Crossover Operator (or Mating Operator) crosses the parent chromosomes (selected randomly) generating new son chromosomes, which will be better and/or worse than their parents [22].

The Mutation Operator must have a low frequency of action for the purpose of not disturbing the nature of the species, i.e., skip to solve another problem [22].

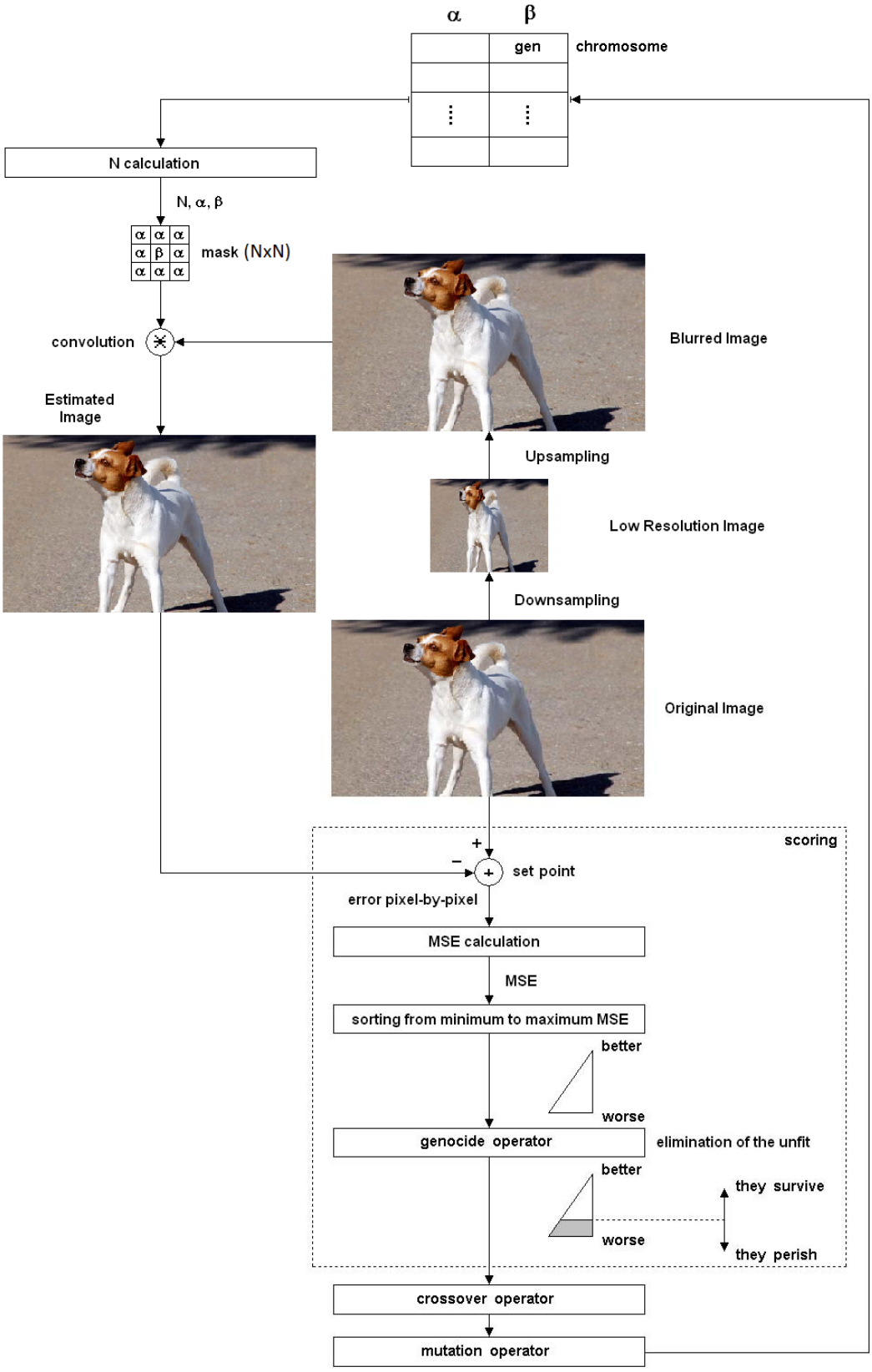

Fig.11: Genetic Algorithm for calculating the parameters of the mask.

D. Applications:

We present two main applications of video compression in real time for Digital TV, according to standard ISDB-Tb [24].

In the first, we move from a resolution of 1920x1080 Full-HD to another 5 times lower of 720x576 SD. As we have said before, the standard video compression H.264 is not affected by the supercompression.

The Fig.12 shows a diagram of the encoder with three modules embedded into GPGPU cards.

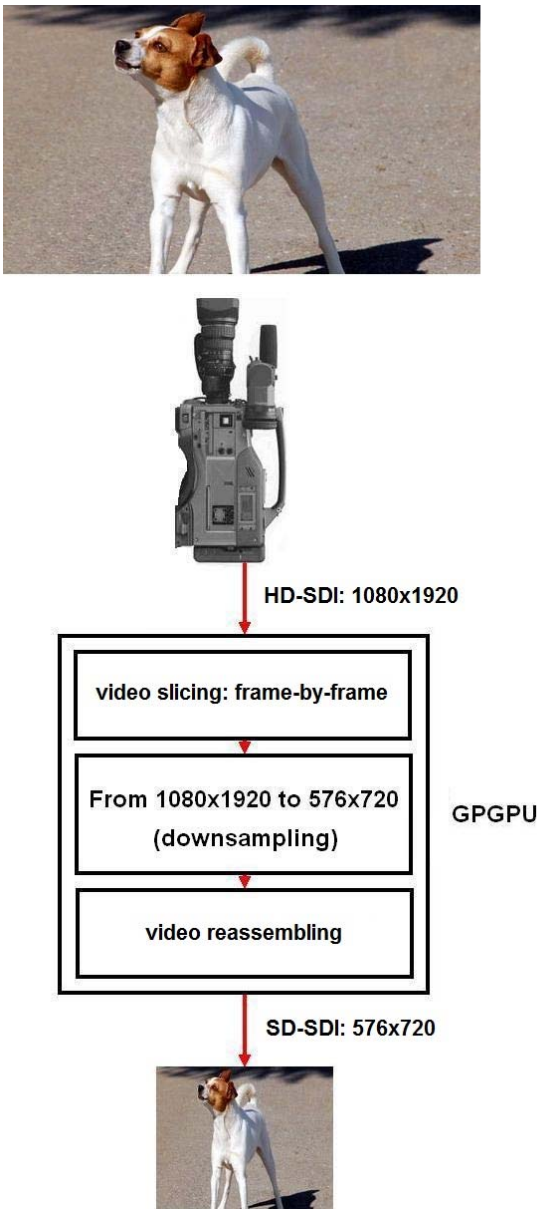

Fig.12: Encoder.

Fig.13 shows in detail the employed technology for the real implementation of Fig.12, which consists in two Quadro GPUs [25] the first for video slicing. frame-by-frame, and the second for video reassembling, respectively.

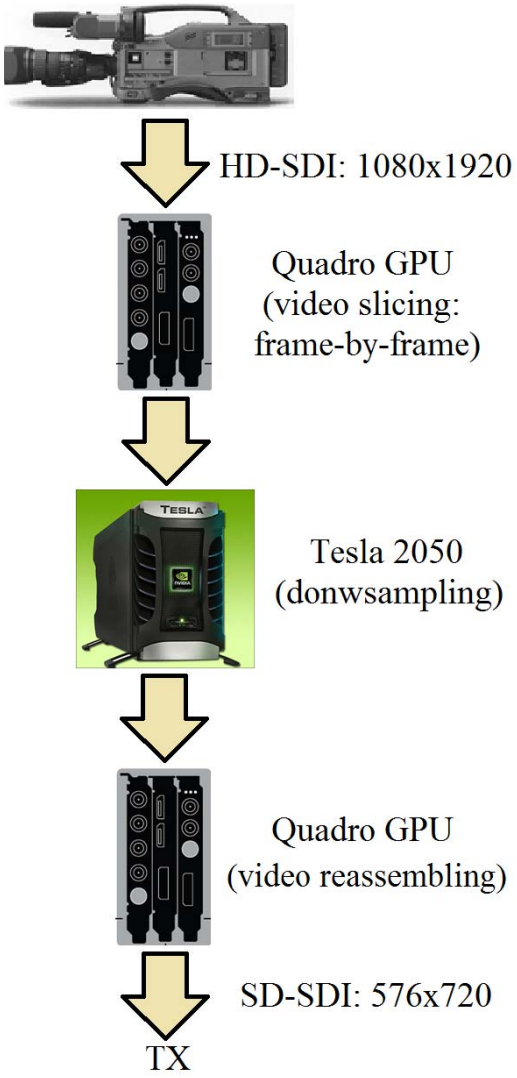

Fig.13: Encoder implementation with GPGPUs.

The downsampling is im-plemented on a Tesla 250 [25]. Moreover, in Fig.13:

TX means transmitter
AAC "Advanced Audio Coding", and it is the compression audio format employed for ISDB-Tb [24].

On the other hand, Fig.14 shows a diagram of the decoder implemented inside a set-top-box (STB). So that, if the STB has the superdecompression and depending on the resolution of the LCD TV, we obtain resolutions of High Definition (HD) 720x1280 or Full-HD 1080x1920. However, if the STB hasn't the superdecompression, the system must be compatible, there-fore we obtain only SD 576x720.

The Fig.15 shows the Super-Resolution Module (SRM) used inside STB of Fig.14, which includes upsampling and deblurring, thus restoring the original resolution.

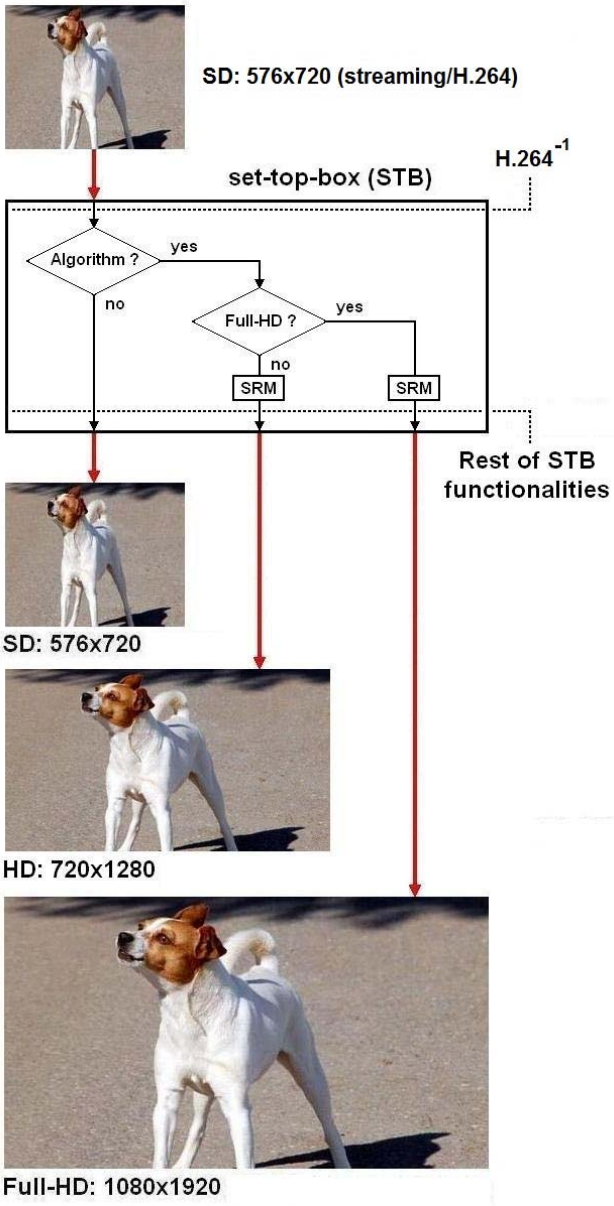

Fig.14: Decoder.

Fig.16 represents the real implementation of Fig.14, in which, we can see, the set-top-box used by UNTreF, i.e., the Univ. Nac. de Tres de Febrero [21]. This STB works equally with Terrestrial Digital TV, IPTV, WebTV, 3DTV and Digital Cinema. Besides, this STB has camera and motion sensors, which can be used as interactive gaming platform.

Actually, we are working on an integrated circuit (chip) [26] to replace the current GPGPU inside the STB, minimizing the power consumption and the size of this [21].

Finally, the second application of this technology presented here is shows in Fig.17, where we use a mobile phone with High-Definition Multimedia Interface (HDMI) video out as a receptor.

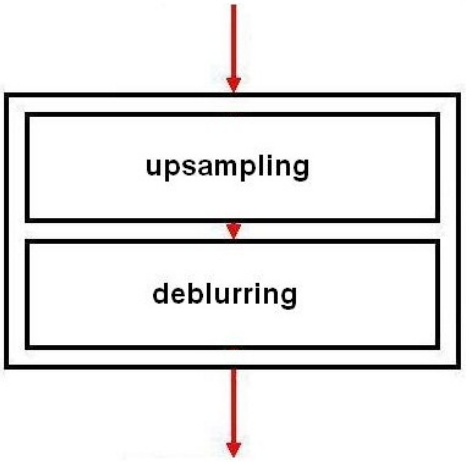

Fig.15: Super-resolution Module (SRM).

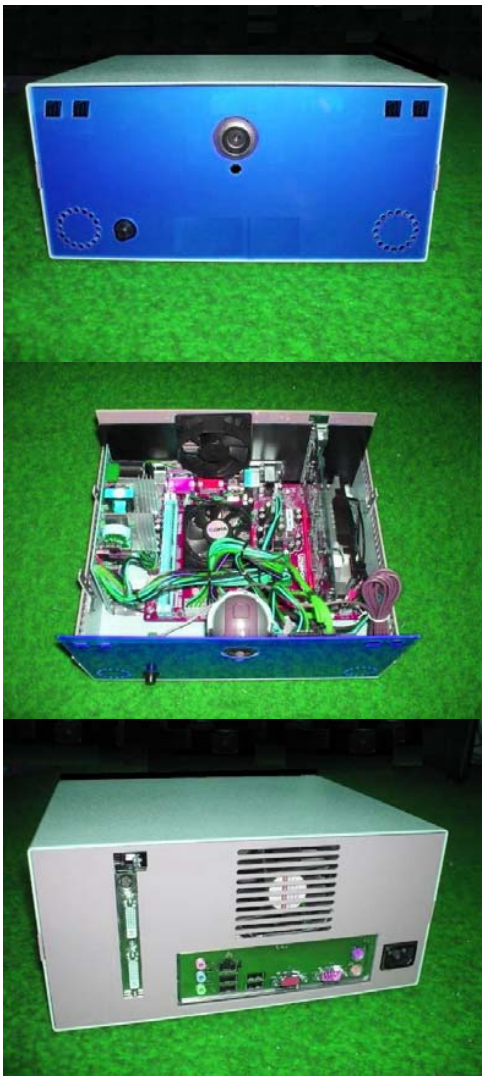

Fig.16: Set-top-box used by UNTreF.

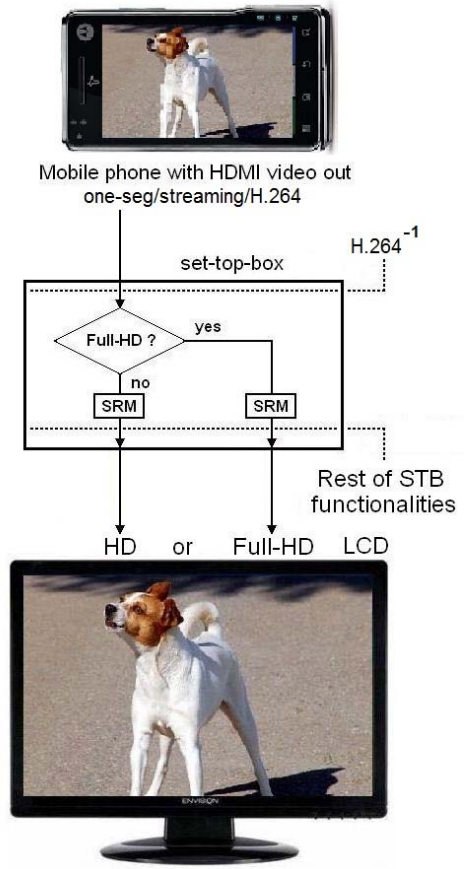

Fig.17: Mobile phone as HD or Full-HD receptor.

As shows in Fig.17, we take the HDMI video out, and we introduce it in the STB. Depending on the resolution of the LCD TV we obtain HD o Full-HD resolutions.

The original resolution of the mobile phone employed is Low Definition (LD) 320x240 *One-Seg* (one of 13 segments that form the ISDB-T norm, see Fig.18). In this case, the additional compression rate of STB on H.264 is 27:1 [21].

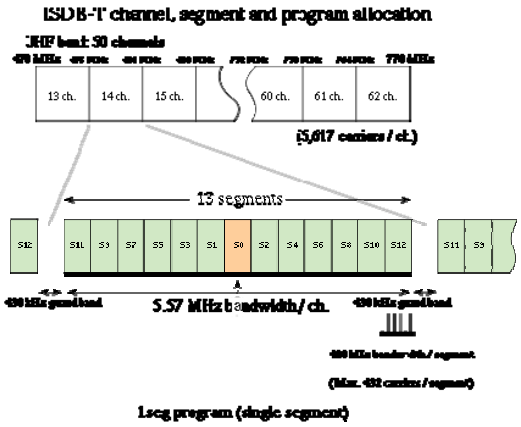

Fig.18: Detail of 13 segments inside ISDB-T channel.

IV. Metrics

A. Data Compression Ratio (CR)

Data compression ratio, also known as compression power, is a computer-science term used to quantify the reduction in data-representation size produced by a data compression algorithm. The data compression ratio is analogous to the physical compression ratio used to measure physical compression of substances, and is defined in the same way, as the ratio between the *uncompressed size* and the *compressed size* [15, 16]:

$$CR = \frac{Uncompressed\ Size}{Compressed\ Size} \tag{22}$$

Thus a representation that compresses a 10MB file to 2MB has a compression ratio of 10/2 = 5, often notated as an explicit ratio, 5:1 (read "five to one"), or as an implicit ratio, 5X. Note that this formulation applies equally for compression, where the uncompressed size is that of the original; and for decompression, where the uncompressed size is that of the reproduction.

B. Bit-per-pixel (bpp)

The "bits per pixel" refers to the sum of the bits in all three color channels and represents the sum colors available at each pixel before compression (bpp_{bc}). However, as a compression metric, the bits-per-pixel refers to the average of the bits in all three color channels, after of compression process (bpp_{ac}).

$$bpp_{ac} = \frac{Compressed\ Size}{Uncompressed\ Size} \times bpp_{bc} = \frac{bpp_{bc}}{CR} \tag{23}$$

Besides, bpp is also defined as

$$bpp_{ac} = \frac{Number\ of\ coded\ bits}{Number\ of\ pixels} \tag{24}$$

C. Mean Absolute Error (MAE)

The mean absolute error is a quantity used to measure how close forecasts or predictions are to the eventual outcomes. The mean absolute error (MAE) is given by

$$MAE = \frac{1}{NRxNC} \sum_{nr=0}^{NR-1} \sum_{nc=0}^{NC-1} \left\| X(nr,nc) - \hat{X}(nr,nc) \right\| \tag{25}$$

which for two $NR \times NC$ (rows-by-columns) monochrome images $\boldsymbol{X}$ and $\hat{X}$, where the second one of the images is considered a decompressed approximation of the other of the first one.

D. Mean Squared Error (MSE)

The mean square error or MSE in Image Compression is one of many ways to quantify the difference between an origi-

nal image and the true value of the quantity being decompressed image, which for two $NR \times NC$ (rows-by-columns) monochrome images $\boldsymbol{X}$ and $\widehat{X}$, where the second one of the images is considered a decompressed approximation of the other is defined as:

$$MSE = \frac{1}{NRxNC} \sum_{nr=0}^{NR-1} \sum_{nc=0}^{NC-1} \left\| X(nr,nc) - \widehat{X}(nr,nc) \right\|^2 \qquad (26)$$

E. Peak Signal-To-Noise Ratio (PSNR)

The phrase peak signal-to-noise ratio, often abbreviated PSNR, is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale. The PSNR is most commonly used as a measure of quality of reconstruction in image compression, etc [15]. It is most easily defined via the mean squared error (MSE), so, the PSNR is defined as [16]:

$$PSNR = 10 \log_{10}(\frac{MAX_X^2}{MSE}) = 20 \log_{10}(\frac{MAX_X}{\sqrt{MSE}}) \qquad (27)$$

Here, MAX_X is the maximum pixel value of the image. When the pixels are represented using 8 bits per sample, this is 256. More generally, when samples are represented using linear pulse code modulation (PCM) with B bits per sample (bps), maximum possible value of MAX_X is 2^B-1.

For color images with three red-green-blue (RGB) values per pixel, the definition of PSNR is the same except the MSE is the sum over all squared value differences divided by image size and by three [15, 16].

Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, where higher is better.

V. Simulations

The simulations are organized in four experiments, separated in two groups: still images (for obvious reasons, however, identical results were achieved in video, HDTV and Digital Cinema) by color and gray. All experiments include calculations of MAE, MSE, PSNR, bpp and CR.

All these experiments involve the comparison between the use of JPEG vs SC (JPEG+SR), and JPEG2000 vs SC (JPEG2000+SR) for still color and gray images, in both cases over a BMP file (which doesn't have compression, to raw data mode), where the used acronym means:

BMP: BitMap file format [27]
JPEG: Joint Picture Group [27]
JPEG2000: JPEG with wavelets [28]
SC: Super-compression
SR: Super-resolution

A. Group 1: Main characteristics of employed image:
File = angelina.bmp
Color = yes
Size = 1920-by-1080 pixels
Original bpp = 24

Experiment 1: JPEG vs SC (JPEG+SR)

JPEG: See Table I, column JPEG, and Fig.19 (2nd from top).
Encoder:
1. From BMP (24 bpp, 1920x1080)
2. To JPEG (0.6853 bpp, 1920x1080)

Channel/storage
Decoder:
1. From JPEG (0.6853 bpp, 1920x1080)
2. To BMP(24 bpp, 1920x1080)

SC (JPEG+SR): See Table I, column SC (JPEG+SR), and Fig.19 (3rd from top).
Encoder:
1. BMP (24 bpp, 1920x1080)
2. Downsampling (24 bpp, 720x576)
3. JPEG (0.1445 bpp, 720x576)

Channel/storage
Decoder:
1. JPEG (0.1445 bpp, 720x576)
2. Upsampling (0.4323 bpp, 1920x1080)
3. Deblurring (0.5004 bpp, 1920x1080)
4. BMP (24 bpp, 1920x1080)

Experiment 2: JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: See Table II, column JPEG2000, and Fig.19 (4th from top).
Encoder:
1. From BMP (24 bpp, 1920x1080)
2. To JPEG2000 (2.6285 bpp, 1920x1080)

Channel/storage
Decoder:
1. From JPEG2000 (2.6285 bpp, 1920x1080)
2. To BMP (24 bpp, 1920x1080)

SC (JPEG2000+SR): See Table II, column SC (JPEG2000+SR), and Fig.19 (down).
Encoder:
1. BMP (24 bpp, 1920x1080)
2. Downsampling (24 bpp, 720x576)
3. JPEG2000 (0.8148 bpp, 720x576)

Channel/storage
Decoder:
1. JPEG2000 (0.8148 bpp, 720x576)
2. Upsampling (1.3903 bpp, 1920x1080)
3. Deblurring (2.2397 bpp, 1920x1080)
4. BMP (24 bpp, 1920x1080)

The following tables show the metrics vs the Algorithms for both cases, i.e., JPEG and JPEG2000 vs Supercompression.

TABLE I
ANGELINA (COLOR, 24 BPP, 1920X1080): JPEG VS SC (JPEG+SR)

Metrics	JPEG	SC (JPEG+SR)
MAE	0.5333	1.0009
MSE	2.3137	7.6264
PSNR	43.6693	38.2393
bpp	0.6853	0.1445
CR	35.0210	166.1154

TABLE II
ANGELINA (COLOR, 24 BPP, 1920X1080): JPEG2000 VS SC (JPEG2000+SR)

Metrics	JPEG2000	SC (JPEG2000+SR)
MAE	0.0446	0.2961
MSE	0.0472	1.1385
PSNR	61.3884	47.5673
bpp	2.6285	0.8148
CR	9.1307	29.4538

B. Group 2: Main characteristics of employed image:
File = lena.bmp
Color = gray
Size = 512-by-512 pixels
Original bpp = 8

Experiment 3: JPEG vs SC (JPEG+SR)

JPEG: See Table III, column JPEG, and Fig.20 (2nd from top).

Encoder:
1. From BMP (8 bpp, 512x512)
2. To JPEG (0.8953 bpp, 512x512)

Channel/storage

Decoder:
1. From JPEG (0.8953 bpp, 512x512)
2. To BMP(24 bpp, 512x512)

SC (JPEG+SR): See Table III, column SC (JPEG+SR), and Fig.20 (3rd from top).

Encoder:
1. BMP (8 bpp, 512x512)
2. Downsampling (8 bpp, 256x256)
3. JPEG (0.2957 bpp, 256x256)

Channel/storage

Decoder:
1. JPEG (0.2957 bpp, 256x256)
2. Upsampling (0.6502 bpp, 512x512)
3. Deblurring (0.7727 bpp, 512x512)
4. BMP (8 bpp, 512x512)

Experiment 4: JPEG2000 vs SC (JPEG2000+SR)

JPEG2000: See Table IV, column JPEG2000, and Fig.20 (4th from top).

Encoder:
1. From BMP (8 bpp, 512x512)
2. To JPEG2000 (3.7242 bpp, 512x512)

Channel/storage

Decoder:
1. From JPEG2000 (3.7242 bpp, 512x512)
2. To BMP (8 bpp, 512x512)

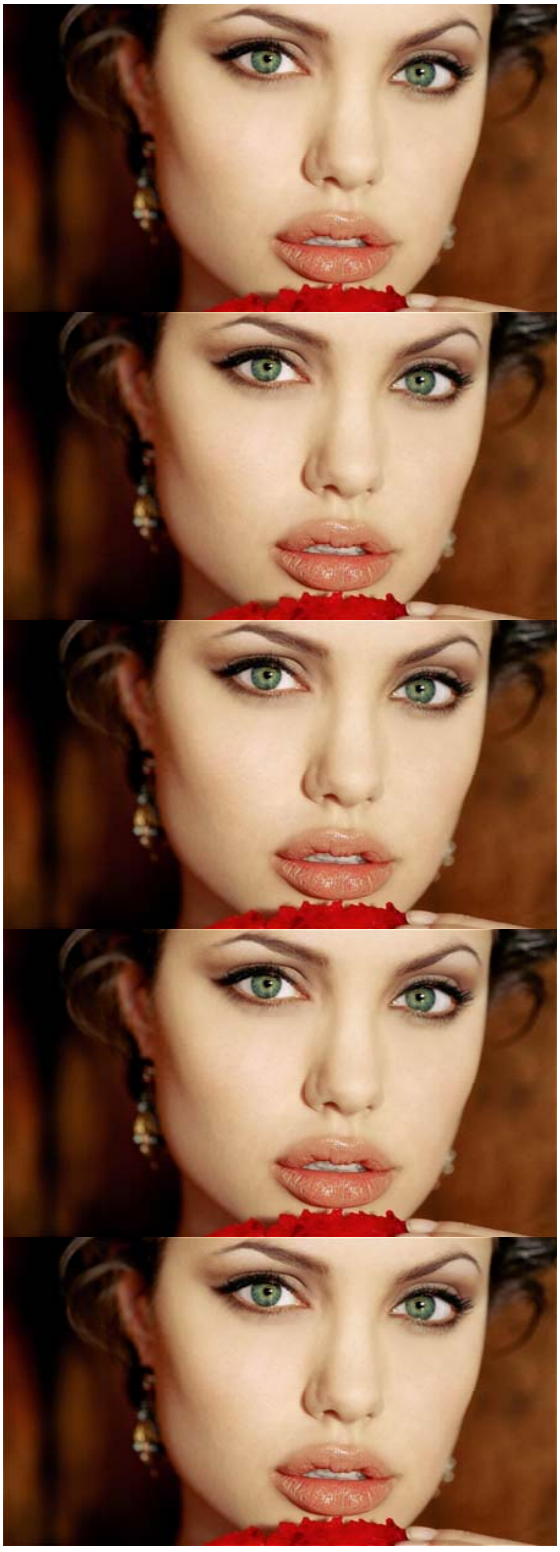

Fig.19: First (top) original image, second (coded and decoded with JPEG), third (coded and decoded with JPEG+Supercompression), fourth (coded and decoded with JPEG2000), fifth (down, coded and decoded with JPEG2000+Supercompression).

SC (JPEG2000+SR): See Table IV, column SC (JPEG2000+ SR), and Fig.20 (down).

Encoder:

1. BMP (8 bpp, 512x512)
2. Downsampling (8 bpp, 256x256)
3. JPEG2000 (1.0066 bpp, 256x256)

Channel/storage

Decoder:

1. JPEG2000 (1.0066 bpp, 256x256)
2. Upsampling (1.6421 bpp, 512x512)
3. Deblurring (2.4230 bpp, 512x512)
4. BMP (8 bpp, 512x512)

The following tables show the metrics vs the Algorithms for both cases, i.e., JPEG and JPEG2000 vs Supercompression.

TABLE III
LENA (GRAY, 8 BPP, 512X512): JPEG VS SC (JPEG+SR)

Metrics	JPEG	SC (JPEG+SR)
MAE	1.0785	2.0243
MSE	4.4363	14.6230
PSNR	41.6606	36.4804
bpp	0.8953	0.2957
CR	8.9358	27.0526

TABLE IV
LENA (GRAY, 8 BPP, 512X512): JPEG2000 VS SC (JPEG2000+SR)

Metrics	JPEG2000	SC (JPEG2000+SR)
MAE	0.0902	1.5312
MSE	0.0905	9.2596
PSNR	58.5647	38.4649
bpp	3.7242	1.0066
CR	2.1481	7.9475

Finally, all techniques were previously implemented in MATLAB® R2010b (Mathworks, Natick, MA) [29] on a Notebook with an Intel® Core(TM) i5 CPU M 520 @ 2.40 GHz processors and 2 GB RAM on Microsoft® Windows 7© 32 bits, and then in OpenCL and CUDA© of NVIDIA® [25] on NVIDIA® Quadro 6000 + Tesla 2050 + Quadro 6000 GPUs for encoder, and NVIDIA® GTX285 GPU inside STB used by UNTreF [21] for decoder, as shown in Fig.16.

VI. CONCLUSION

A. Group 1:

Experiment 1: JPEG vs SC (JPEG+SR)

In this experiment SC (JPEG+SR) has MAE, MSE and PSNR with practically the same order of magnitude than JPEG alone, however, bpp is five times lower, at the same time, CR is five times higher, see Table I.

As shown in Fig.19, the second (coded and decoded with JPEG) and the third (coded and decoded with JPEG+Super-compression) from the top, have the same look-and-feel and image quality than the top, i.e., original image of Angelina.

Experiment 2: JPEG2000 vs SC (JPEG2000+SR)

We make similar considerations for this experiment, regar-

Fig.20: First (top) original image, second (coded and decoded with JPEG), third (coded and decoded with JPEG+Supercompression), fourth (coded and decoded with JPEG2000), fifth (down, coded and decoded with JPEG2000+Supercompression).

ding to the last experiment, see Table II and Fig.19 (fourth coded and decoded with JPEG2000 alone, and fifth coded and decoded with JPEG2000+Supercompression), however, there is a big difference between JPEG and JPEG-2000 to compress this type of image (compare bpp and CR of Table I and II).

B. Group 2:

Experiment 3: JPEG vs SC (JPEG+SR)

In this experiment SC (JPEG+SR) has MAE, MSE and PSNR with practically the same order of magnitude than JPEG alone, however, bpp is five times lower, at the same time, CR is five times higher, see Table III, idem Experiment 1.

As shown in Fig.20, the second (coded and decoded with JPEG) and the third (coded and decoded with JPEG+Supercompression) from the top, have the same look-and-feel and image quality than the top, i.e., original image of Lena.

Experiment 4: JPEG2000 vs SC (JPEG2000+SR)

Identical considerations than Experiment 2 are necessary, see Table IV and Fig.20, with the same conclusions about the difference between JPEG and JPEG-2000 to compress this type of image (compare bpp and CR of Table III and IV).

C. For both groups:

We used Texture Memory inside STB [21] GPGPU to a computational efficient implementation of the bidimensional convolutive mask of deblurring module, allowing us to reach TV times, i.e., a frame every 40 milliseconds.

APPENDIX

Super-resolution method is typically used to restore a high-resolution image from several low-resolution noisy observations [3]. In this paper, we consider the interpolation of a single image. So, we will formulate the problem as

$$A\,x = y \tag{28}$$

where x is the unknown high-resolution image (represented as a vector of pixel values), y is the known low-resolution image, and A is the downscaling operator typically consisting of decimation D following a low-pass filtering H:

$$A = D\,H \tag{29}$$

The choice of the low-pass filtering operator depends on a point spread function of the imaging system that produced the low resolution image. If the imaging system is unknown we will assume that operator H is a simple box filter.

A. Regularization

The Eq.28 is generally ill-posed and a small change of the input vector y can cause a huge change of the resulting vector x. For the Eq.28, the regularized solution is found as:

$$x = \arg\min \|Ax - y\|_n^p + \alpha\, F(x) \tag{30}$$

where the first term is called as *"discrepancy"*, $F(x)$ is the *stabilizer* and $\alpha > 0$ is the *coefficient of regularization* [3].

The most popular and universal stabilizer is the Tikhonov functional. It is calculated as a grid approximation of the functional:

$$F(x) = \|\Delta x\|_2^2 \tag{31}$$

and $n = 2$, $p = 2$. For each $\alpha > 0$ the solution x is correct: it is unique, defined for every y and continuously depends on y. We can write the Euler equation for this case:

$$(A^T A + \Delta^2)x = A^T y \tag{32}$$

But in this case the algorithm becomes linear because x is the solution of the system of linear equations. So, this method inherits drawbacks of linear interpolation algorithms and we need to find more adaptive stabilizer for image resampling.

We will consider Total Variation (TV) and Bilateral TV (BTV) stabilizers [3], which are working in l_1 norm ($n = 1$, $p = 1$):

$$TV(x) = \|\nabla x\|_1 \tag{33}$$

where ∇x is the gradient operator (its modulus),

$$BTV(x) = \sum_{s,t=-p}^{s,t=p} \gamma^{|s|+|t|} \left\| x - S_x^s S_y^t x \right\|_1 \tag{34}$$

where S_x^s and S_y^t are shift operators along x and y axes by s and t pixels respectively, $\gamma = 0.8$.

B. Inverse iterations

To solve the equation (30) with a stabilizer (34) the iterative steepest-descent method can be used:

$$x_{n+1} = x_n - \beta \Big\{ H^T D^T sign(DHx - y) + \alpha \sum_{s,t=-p}^{s,t=p} \gamma^{|s|+|t|} (I - S_x^{-s} S_y^{-t})\, sign(S_x^s S_y^t x) \Big\} \tag{35}$$

$z = sign\ x$ is a vector with per-element applied sign function; D^T is an up-scaling operation. If D in (29) is the simplest decimation operator that takes every *k-th* pixel, D^T is the up-scaling operator by zero insertion. If H in (29) is a symmetric filtering, then H^T is equal to H. x_0 is the initial approximation of the high resolution image.

ACKNOWLEDGMENT

M. Mastriani thanks Prof. Martin Kaufmann, vice chancellor of Universidad Nacional de Tres de Febrero, for his tremendous help and support.

REFERENCES

[1] A. Gilman, D. G. Bailey, S. R. Marsland, "Interpolation Models for Image Super-resolution," in *Proc. 4th IEEE International Symposium on Electronic Design, Test & Applications*, DELTA 2008, Hong Kong, 2008, pp.55-60.
[2] D. Glassner, S. Bagon, M. Irani. Super-Resolution from a Single Image. Available: http://www.wisdom.weizmann.ac.il/~vision/single_image_SR/files/single_image_SR.pdf
[3] A. Lukin, A. S. Krylov, A. Nasonov. Image Interpolation by Super-Resolution. Available: http://graphicon.ru/oldgr/en/publications/text/LukinKrylovNasonov.pdf
[4] Y. Huang, "Wavelet-based image interpolation using multilayer perceptrons," *Neural Comput. & Applic.*, vol.14, pp.1-10, 2005.
[5] N. Mueller, Y. Lu, and M. N. Do. Image interpolation using multiscale geometric representations. Avalilable: http://lcav.epfl.ch/~lu/papers/interp_contourlet.pdf
[6] M. Kraus, M. Eissele, and M. Strengert. GPU-Based Edge-Directed Image Interpolation. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.69.5655
[7] -. NVIDIA CUDA: Best Practices Guide, version 3.0, 2/4/2010. Available: http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_BestPracticesGuide.pdf
[8] V. Podlozhnyuk. Image Convolution with CUDA, June 2007. Available: http://developer.download.nvidia.com/compute/cuda/1_1/Website/projects/convolutionSeparable/doc/convolutionSeparable.pdf
[9] V. Simek, and R. Rakesh, "GPU Acceleration of 2D-DWT Image Compression in MATLAB with CUDA," in *Proc. Second UKSIM European Symposium on Computer Modeling and Simulations*, Liverpool, UK, 2008, pp.274-277.
[10] T. C. Wittman, "Variational Approaches to Digital Zooming," Ph.D. dissertation, Dept. Math, Univ. of Minnesota, Saint Paul, MN, 2006.
[11] M. Mastriani, and A. E. Giraldez, "Microarrays denoising via smoothing of coefficients in wavelet domain," *International Journal of Biological and Life Sciences*, vol. 1:1, pp.7-14, 2005.
[12] M. Mastriani, and A. E. Giraldez, "Enhanced directional smoothing algorithm for edge-preserving smoothing of synthetic-aperture radar images," *Journal of Measurement Science Review*, vol.4:3, pp.1-11, 2004.
[13] M. Mastriani, and A. E. Giraldez, "Smoothing of coefficients in wavelet domain for speckle reduction in synthetic-aperture radar images," *ICGST-GVIP Journal*, vol.5:6, pp.1-8, 2005.
[14] M. Mastriani. Directional smoothing for speckle reduction in synthetic-aperture radar imagery. Available: http://fundesco.eurofull.com/img/paper2.pdf
[15] R.C. Gonzalez, R.E. Woods, Digital Image Processing, 2nd Edition, Prentice- Hall, Jan. 2002, pp.675-683.
[16] A.K. Jain, *Fundamentals of Digital Image Processing,* Englewood Cliffs, New Jersey, 1989.
[17] M. Mastriani, "Denoising and compression in wavelet domain via projection onto approximation coefficients," *International Journal of Information and Communication Engineering*, vol. 5:1, pp. 20-30, 2009.
[18] C. Kim, K. Choi, K. Hwang, and J. Beom Ra. Learning-based super-resolution using a multi-resolution wavelet approach. Available: http://www-isl.kaist.ac.kr/Papers/IC/ic123.pdf
[19] C. S. Boon, O. G. Guleryuz, T. Kawahara and, Y. Suzuki, "Sparse super-resolution reconstructions of video from mobile devices in digital TV broadcast applications," in *Proc. SPIE Conf. on Applications of Digital Image Processing XXIX, in Algorithms, Architectures, and Devices*, San Diego, CA, Aug. 2006.
[20] I. E. Richardson, H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia, Ed. Wiley, N.Y., 2003.
[21] http://www.untref.edu.ar/carreras_de_grado/ing_computacion.htm
[22] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Ed. Pearson, N.Y., 1989.
[23] **.-**, Kalman Filtering: Theory and Application, Sorenson H. W. ed., IEEE Press, 1985, New York.
[24] http://www.forumsbtvd.org.br/
[25] NVIDIA® (NVIDIA Corporation, Santa Clara, CA).
[26] P. S. Mandolesi, P. Julian, A. G. Andreou, "A scalable and program-mable simplicial CNN digital pixel processor architecture," IEEE Trans. Circuits and Systems – I: Regular Papers, Vol. 51, No. 5, pp. 988- 996, May 2004.
[27] J. Miano, Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP; Ed. Addison-Wesley, N.Y., 1999.
[28] T. Acharya, and P-S Tsai, JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures, Ed. Wiley, N.Y., 2005.
[29] MATLAB® R2010b (Mathworks, Natick, MA).

Mario Mastriani was born in Buenos Aires, Argentina on February 1, 1962. He received the B.Eng. degree in 1989 and the Ph.D. degree in 2006, both in electrical engineering. Besides, he received the second Ph.D. degree in Computer Science in 2009. All degrees from Buenos Aires University. He is Professor of Digital Signal and Image Processing of the Engineering College, at Buenos Aires University (UBA). Professor Mastriani is the Coordinator of Technological Innovation (CIT) of ANSES, and the Computer Engineering Department of the National University of Tres de Febrero, at Buenos Aires, Argentina. He published 50 papers. He is a currently reviewer of IEEE Transactions on Neural Networks, Signal Processing Letters, Transactions on Image Processing, Transactions on Signal Processing, Communications Letters, Transactions on Geoscience and Remote Sensing, Transactions on Medical Imaging, Transactions on Biomedical Engineering, Transactions on Fuzzy Systems, Transactions on Multimedia; Springer-Verlag Journal of Digital Imaging, SPIE Optical Engineering Journal; and Taylor & Francis International Journal of Remote Sensing.
He (M'05) became a member (M) of WASET in 2004.
His areas of interest include Digital Signal Processing, Digital Image Processing, Compression and Super-resolution.