Practical considerations for real-time super-resolution implementation techniques over video coding platforms

Gustavo M. Callicó[*a], Sebastián López[a], Rafael Peset Llopis[b], Ramanathan Sethuraman[c], Antonio Núñez[a], José Fco. López[a], Margarita Marrero[a], Roberto Sarmiento[a]
[a]Applied Microelectronics Research Institute, Department of Electronic and Automatic Engineering, University of Las Palmas de G.C., Canary Islands, Spain;
[b]Philips Consumer Electronics, Royal Philips Eindhoven, The Netherlands;
[c]Philips Research Laboratories Eindhoven, Royal Philips Eindhoven, The Netherlands

ABSTRACT

This paper addresses practical considerations for the implementation of algorithms developed to increase the image resolution from a video sequence by using techniques known in the specialized literature as super-resolution (SR). In order to achieve a low-cost implementation, the algorithms have been mapped onto a previously developed video encoder architecture. By re-using such architecture and performing only slight modifications on it, the need for specific, and usually high-cost, SR hardware is avoided. This modified encoder can be used either in native compression mode or in SR mode, where SR can be used to increase the image resolution over the sensor limits or as a smart way to perform electronic zoom, avoiding the use of high-power demanding mechanical parts. Two SR algorithms are presented and compared in terms of execution time, memory usage and quality. These algorithms features are analyzed from a real-time implementation perspective. The first algorithm follows an iterative scheme while the second one is a modified version where the iterative behavioural has been broken. The video encoder together with the new SR features constitutes an IP block inside Philips Research, upon which several System on Chip (SoC) platforms are being developed.

Keywords: Super-Resolution, Multimedia Architectures, Image Reconstruction, Hardware/Software Co-design, Video Compressors.

1. INTRODUCTION

There are multiple scenarios where it is important to increase the resolution of the pictures acquired by an imaging system: from medicine to astronomy, going through domestic pictures or surveillance systems among others. There are two straightforward ways to increase sensor resolution. The first one is based on increasing the number of light-sensors and therefore the area of the overall sensor, but that results in an important cost increase. The second one is focused on preserving the overall sensor area by decreasing the size of the light-sensors. Although this size reduction increases the number of light-sensors, the size of the active pixel area where the light integration is performed becomes decreased. As fewer amounts of light reach the sensor it will be more sensitive to the shot noise. However, it has been estimated that the minimum photo-sensors size is around 50μm2 [1], a limit that has already been reached by the CCD technology. A smart solution to this problem is to increase the resolution using algorithms such as the super-resolution (SR) ones, wherein high-resolution images are obtained using low-resolution sensors at lower costs. Super-resolution can be defined as a technique that estimates a high-resolution sequence by using multiple low-resolution observations of the scene. In order to obtain significant improvements in the resulting SR image, some amount of aliasing in the input low-resolution images must be provided. In other words, if all the high frequency information has been removed from the input images (for instance by using lenses with optical low-pass filter effect), it will be impossible to recover the edge details contained in the high frequencies. Some of the most important applications of SR are:

(i) Still image improvement [1, 2], where several images from the same scene are obtained and used to form a higher resolution image.
(ii) Analogical video frame improvement [3, 4]. Due to the low quality of analogical video frames, they are not normally suitable for performing directly a printed copy as in the digital photography case. The quality of

[*] gustavo@iuma.ulpgc.es; phone +34 952845-1271; fax +34 92845-1283;iuma.ulpgc.es

VLSI Circuits and Systems II, edited by José Fco. López, Francisco V. Fernández, José María López-Villegas, José M. de la Rosa, Proceedings of SPIE Vol. 5837 (SPIE, Bellingham, WA, 2005) 0277-786X/05/$15 · doi: 10.1117/12.608322

the image is increased using several consecutive frames combined in a higher resolution image by using SR algorithms.

(iii) Surveillance systems, where SR is used to increase the quality in video surveillance systems, making it possible to use such recorded sequences as forensic digital video, and even to be admitted as evidence in the courts of law. SR can also improve night vision systems when images have been acquired with infrared sensors [5] and can help in the face recognition process for security purposes [6].

(iv) Text extraction process from image sequences [7] is highly improved if the Regions Of Interest (ROI) containing the text are first super-resolved.

(v) Medical image acquisition [8]. Many medical types of equipment as the Computer Aided Tomography (CAT), the Magnetic Resonance Images (MRI) or the echography or ultra-sound images, allow the acquisition of several images, which can be combined, thus obtaining a higher resolution image.

(vi) Improvement of images from compressed video [9, 10]. For example, in [11] the image high frequency information recovery, lost in the compression process, is addressed. These missing data is incorporated from transform-domain quantization information obtained from the compressed video bit stream. An excellent survey of SR algorithms from compressed video can be found in [12].

(vii) Improvement of radar images [13, 14]. In this case SR allows a clearer observation of details sometimes critical for air or maritime security [15] or for land observations [16, 17].

(viii) Quality improvement of images obtained from the outer space. An example is exposed in [2] with images taken by the Viking satellite.

(ix) Image Based Rendering (IBR) of 3D objects use cameras to obtain rich models directly from the real-world data [18]. SR is used to produce high-resolution scene texture from an omnidirectional image sequence [19].

This paper addresses low-cost solutions for the implementation of SR algorithms on SOC (System-On-Chip) platforms in order to achieve high-quality image improvements. Low-cost constrains are accomplished in the sense that SR is performed without developing a specific hardware, but re-using a video encoder. This encoder can be used either in compression mode or in SR mode as an added value to the encoder. Due to this reason, SR is used in the video encoder as a smart way to perform image zooming of regions of interest (ROI) without using mechanical parts to move the lenses, thus saving power consumption. It is important to remark that although the SR algorithms presented in this paper have been implemented on an encoder architecture developed by Philips Research, the same SR algorithms can be easily adapted to other hybrid video encoder platforms.

The SR approaches that will be depicted consists of gathering information from a set of images in the spatial-temporal domain in order to integrate all the information (when possible) in a new quality-improved super-resolved image. This set is composed of several images, where small spatial shifts have been applied from one image to the other. This can be achieved by recording a video sequence at high frame rates with a hand-held camera.

The rest of the paper is organized as follows. Firstly, the most relevant publications directly related with this work are reviewed, followed by a brief description of the hybrid video compression architecture where the developed SR algorithms have been mapped on. In the second section the ISR algorithms is described while in Section 3 the experimental setup to evaluate the quality of the iterative and non-iterative algorithms is presented, and based on it, a set of experiments are developed in Section 4 in order to assess the correct behavioural of the ISR algorithm, showing as a result an important increase in the super-resolved output images. As far as an iterative behavioural seriously jeopardize a real-time implementation, in Section 5 a novel SR algorithm is described, where the previous iterative feature has been removed. In this same section the adjustments carried out in the architecture in order to obtain a feasible implementation are explained, while Section 6 shows the results achieved with this non-iterative algorithm. In Section 7 are compared the advantages and drawbacks for the described ISR and NISR algorithms. Finally, in Section 8 the most remarkable results of this work are discussed.

1.1. Super-Resolution Algorithms

The possibility of reconstructing a super-resolved image from a set of images was initially proposed by Huang and Tsay in 1984 [20], although the general sampling theorems previously formulated by Yen in 1956 [21] and Papoulis in 1977 [22] showed exactly the same concept (from a theoretical point of view). When Huang and Tsay originally proposed the idea of the SR reconstruction, they faced the problem from the frequency domain to demonstrate the possibility of reconstructing an image with improved resolution from several low-resolution undersampled images without noise and from the same scene, based on the spatial aliasing effect. They assume a purely translational model and solve the dual problem of registration and restoration (the registration implies estimating the relative shifts among the observations and

the restoration implies the estimation of samples on a uniform grid with a higher sampling rate). The restoration stage is actually an interpolation problem dealing with non-uniform sampling. From the Huang and Tsay proposal until the present days, several research groups have developed different algorithms for this task of reconstruction, obtained from different strategies or analysis of the problem.

The great advances experimented by computer technology in the last years has led onto a renewed and growing interest in the theory of image restoration. The main approaches are based on non-traditional treatment of the classical restoration problem, oriented to new restoration problems of second generation, and the use of algorithms that are more complex and exhibit a higher computational cost. Based on the resulting image, these new second-generation algorithms can be classified in: problems of an image restoration [20, 23, 24], restoration of an image sequence [25, 26], and reconstruction of an imaged improved with SR [27, 28]. This paper is positioned on the last mentioned approach, both for the reconstruction of static image as for the reconstruction of images sequences with SR improvements.

The classical theory of image restoration from blurred images and with noise has caught the attention of many researches over the last three decades. In the scientific literature, several algorithms have been proposed for this classical problem and to the problems related with it, contributing to the construction of a unified theory that comprises many of the existing restorations methods [29]. In the image restoration theory, mainly three different approaches exist that are widely used in order to obtain reliable restoration algorithms: Maximum Likelihood Estimators (MLE) [50, 51, 52], Maximum A-Posteriori (MAP) probability, [29, 30, 31] and the Projection Onto Convex Sets (POCS) [30].

An alternative classification [32] based on the processing approach can be made, where the work on SR can be divided into two main categories: reconstruction-based methods [27, 33] and learning-based methods [34, 35]. The theoretical foundations for reconstruction methods are non-uniform sampling theorems, while learning-based methods employ generative models that are learned from samples. The goal of the former is to reconstruct the original (super-sampled) signal while that of the latter is to create the signal based on learned generative models. In contrast with reconstruction methods, learning-based SR methods assume that corresponding low-resolution and high-resolution training image pairs are available. The majority of SR algorithms belong to the signal reconstruction paradigm that formulates the problem as a signal reconstruction problem from multiple samples. Among this category are frequency-based methods, Bayesian methods, Back-Projection (BP) methods, Projection Onto Convex Set (POCS) methods, and hybrid methods. From this second classification, this paper is positioned in the reconstruction-based methods, as it seeks to reconstruct the original image without making any assumption about the generative models and assuming that only the low-resolution images are available.

The problem of a specific image reconstruction from a set of lower quality images with some relative movement among them is known as the static SR problem. On the other side, it is the dynamic SR problem, where the objective is to obtain a higher quality sequence from another lower resolution sequence, seeking that both sequences have the same length. These two problems also can be denominated as the SR problem for static images and the SR problem for video, respectively [36]. The work presented in this paper only deals with static SR as the output sequences do not have the same length than the input low-resolution sequences.

Most of the proposed methods mentioned above lack feasible implementations, leaving aside the more suitable process architectures and the required performances in terms of speed, precision or costs. Moreover, all the previous SR approaches demand a huge amount of computation, and for this reason, in general they are not suitable for real time applications. Until now, none of them have been implemented over a feasible hardware architecture. This paper addressed this fact and offers a low-cost solution. The ISR algorithm exposed in this paper is a modified version of [37], adapted to be executed inside a real video encoder, i.e. restricting the operators needed to those that can be found in such kind of platforms. New operator blocks to perform the SR process have being implemented inside the existing co-processors in order to minimize the impact on the overall architecture, as it will be demonstrated in next sections.

1.2. The hybrid video encoder platform

All the algorithms described in this paper have been implemented in an architecture developed by Philips Research. This architecture is shown in (Fig. 1). The software tasks are executed on an ARM processor and the hardware tasks are executed on the Very Long Instruction Word (VLIW) processors (namely, pixel processor, motion estimator processor, texture processor and stream processor). The pixel processor (PP) communicates with the pixel-domain (image sensor or display) and performs input lines to macro-block (MB) conversions. The motion estimator processor (MEP) evaluates a set of candidate vectors received from software and selects the best vector for full, half and quarter pixel refinements. The output of the MEP consists of motion vectors, sum-of-absolute-difference (SAD) values, and texture metrics. This information is processed by the general purpose embedded microprocessor ARM to determine the encoding approach for the current MB.

The texture processor (TP) performs the MBs encoding and stores the decoded MBs in the loop memory. The output of the TP consists of variable length encode (VLE) codes for the discrete cosine transform (DCT) coefficients of the current MB. Finally, the stream processor (SP) packs the VLE coded coefficients and headers generated by the TP and the ARM processor respectively.
Communications among modules are performed through two buses, a control bus and a data bus, each of them controlled by a Bus Control Unit (BCU). Both buses can be directly communicated through a bridge. Images that will be processed by the ISR and the NISR algorithms come from the data bus.

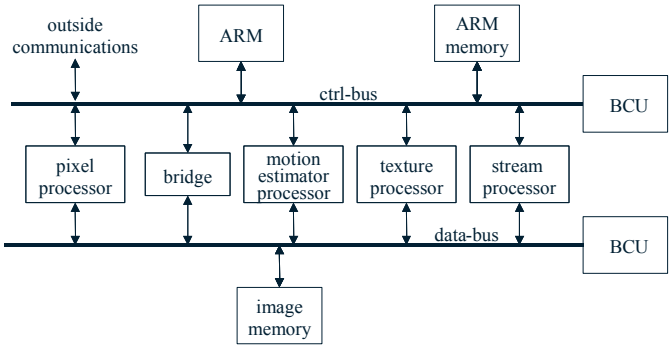

Figure 1: Architecture for the Multi-Standard Video/Image Codec developed by Philips Research.

2. ITERATIVE SUPER-RESOLUTION ALGORITHM

The implemented ISR algorithm is shown in (Fig.2). All memories are eight bits wide, except HR_A, which must be nine bits wide. This memory must be wider because it must store arithmetical results that can overflow 8 bits, especially in the beginning of the iterations. LR_I[] are the low-resolution input frames, HR_B is the SR image result, LR_B is the low-resolution version of HR_B; HR_T is a temporal high-resolution image to avoid overlapping while performing the motion compensation and due to the pipeline of the video-encoder [39]; HR_A accumulates the average error that will be used as an update for the SR image; HR_S stores the error between the SR image (HR_B) shifted to the input frame position and the upsample input image, and finally, MV_ref2fr[] and MV_fr2ref[] are the motion vectors memories to store the motion between the reference and the input frames and vice versa. The number of frames to be combined to create a higher resolution image is 'nr_frames' while 'nr_iterations' stands for the maximum number of pre-established iterations. The algorithm can be split up in the following main steps [38]:

1. Initially, the first low-resolution image is stored in LR_B, used as the low-resolution version of the super-resolved image that will be stored in HR_B. The super-resolved image HR_B is initialized with an upsample version of the first low-resolution image.
2. The iterative process starts obtaining LR_B as a downsampled version of the super-resolved image in HR_B, except for the first iteration, where this assignation has been already made.
3. The motion vectors from the frame being processed to the reference frame are set to zero for frame zero as the frame zero is now the reference.
4. The remainder motion vectors are computed between the other low-resolution input frames and the low-resolution version of the super-resolved image: LR_B (the reference). Instead of computing again the inverse motion, i.e. the motion between the reference and every low-resolution frame, the approximation of considering this motion as the inverse of the previous computed motion is made. Firstly, a great amount of computation is save due to that approximation and secondly, as far as the motion is computed as a set of translational motion vectors in horizontal and vertical directions, the model is mathematically consistent.
5. As the motion vectors are computed in the low-resolution grid, they must be properly scaled to be used in the high-resolution grid. In this case only single shift of one bit to the left is needed.
6. The accumulative image HR_A is set to zero prior to the summation of the average shifted errors. This average error will be the update to the super-resolved image through the iterative process.
7. Now the super-resolved image HR_B is shifted to the position of every frame, using the motion vectors previously computed for every frame.
8. In such position the error between the current frame and the super-resolved frame in the frame position is computed.

9. The error image is shifted back again to the super-resolved image position, using the motion vectors previously computed and these errors are averaged in HR_A.
10. The super-resolved image is improved using the average of all the errors between the previous super-resolved and the low-resolution frames, computed in the frame position and shifted to the super-resolved image position, as an update to the super-resolved image.
11. If the variance of the update is below a certain threshold, then very few changes will be made in the super-resolved image. In this case, continuing the iterative process makes no sense and therefore it is preferable to abort the process.
12. Anyhow, the iterative process will stop when the maximum number of pre-established iterations is reached.

The previous steps numbers have been introduced between parentheses as labels in the beginning of the appropriate lines for clearness. The memory HR_A is in boldface to remark the different bit wide when compared to the other image memories. (Fig. 3) shows the ISR algorithm data flow, using the memories and the resources available in the hybrid video encoder platform.

```
(1) LR_B = LR_I[0]
(1) HR_B = Upsample(LR_I[0])

FOR it = 0 .. nr_iterations-1
    (2) IF (it ≠ 0)   LR_B = Downsample(HR_B)
     (3) MV_fr2ref[0] = 0
     (3) MV_ref2fr[0] = 0
    FOR fr = 1 .. nr_frames-1
       (4) MV_fr2ref[fr] = Calc_Mot_Estimation (LR_I[fr], LR_B)
       (4) MV_ref2fr[fr] = −MV_fr2ref[fr]
       (5) MV_fr2ref[fr] = 2 × MV_fr2ref[fr]
       (5) MV_ref2fr[fr] = 2 × MV_ref2fr[fr]
    END FOR
    (6) HR_A = 0
    FOR fr = 0 .. nr_frames-1
       (7) HR_S = Motion_Compensation (HR_B, MV_ref2fr[fr])
       (8) HR_S = Upsample(LR_I[fr]) – HR_S
       (9) HR_T = Motion_Compensation (HR_S, MV_fr2ref[fr])
       (9) HR_A = HR_A + HR_T/nr_frames
    END FOR
    (10) HR_B = HR_B + HR_A
    (11) variance = variance(HR_A)
    (11) If (variance < variance_threshold) Then break
END FOR
```

Figure 2: Pseudo-code of the ISR algorithm implemented on the video encoder.

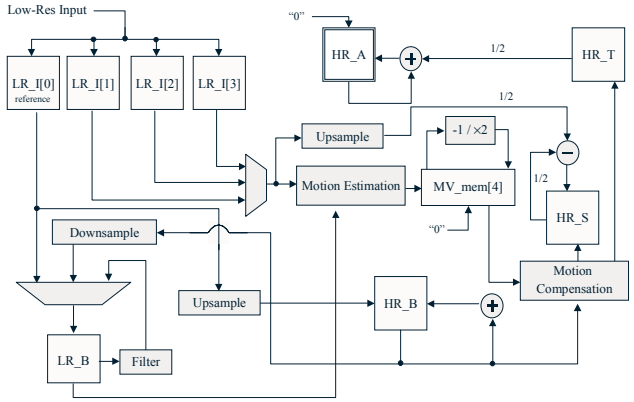

Figure 3: ISR algorithm data flow.

2.1. Implementation issues

Table 1 summarizes the memory requirements that the implementation of the ISR algorithm demands for nr_frames=4 as a function of the input MBs. The number of MBs in the horizontal direction (columns) has been labelled as MB_x, and the number of MBs in the vertical direction (rows) has been labelled as MB_y. For instance, the HR_A memory would have a number of macro-blocks equal to (2·MB_x)×(2·MB_y). Because it is a high-resolution image, its size is double in both directions. As every macro-block has 16×16 luminance pixels and 8×8 chrominance pixels and, furthermore, there exists two chrominance components, the blue and the red ones, this supposes that the overall pixel number is (2·MB_x·2·MB_y·16·16) for the luminance and (2·MB_x·2·MB_y·8·8·2) for the chrominance components. Nevertheless, it must be taken into account that the HR_A memory is 9 bits wide, and for this reason, to obtain the total number of bits it is necessary to multiply by 9 bits each pixel. The remaining memories will be multiplied by 8 bits per pixel. These requirements include four input memories as the number of frames to be combined has been settled on four. Also a buffer of three rows of macro-blocks for reading the input images, as part of the encoder memory requirements, has been included [41]. These memory requirements also take into account the chrominance and the extra bit of HR_A. The total memory requirements, as a function of the number of MBs is MB_y·(6724·MB_x+4608) expressed in bytes.
To perform the up-sample and down-sample operations it is necessary to include in hardware an up-sampling and down-sampling blocks, in charged of performing these operations on a MB basis. A hardware implementation it is desirable as the upsample/downsample processes are very intensive computational tasks in the sense that they are performed over all

the image MBs. A software implementation of these blocks could compromise the real time performance, and for this reason these two tasks have been included in the texture processor. Up-sampling is performed by nearest neighbour replication from an 8×8 pixels block to a 16×16 pixels MB. Down-sampling is achieved by picking one pixel from every set of four neighbour pixels, obtaining an 8×8 block from a 16×16 MB.
The motion estimation and motion compensation tasks are performed using the motion estimator and the motion compensator co-processors. These co-processors have been modified to work in quarter pixel precision because, as it was previously established, the accuracy of the computed displacements is a critical aspect in the ISR algorithm. The arithmetic operations such as additions, subtractions and arithmetic shifts are implemented on the texture processor. Finally, the overall control of the ISR algorithm is performed by the ARM processor (Fig. 1).

Label	ISR Algorithm Memory		
	Luminance (bits)	Chrominance (bits)	Total (bits)
HR_A	*(2·MB_x·2·MB_y·16·16·9)*	*(2·MB_x·2·MB_y·8·8·2·9)*	*13,824·MB_x·MB_y*
HR_B	*(2·MB_x·2·MB_y·16·16·8)*	*(2·MB_x·2·MB_y·8·8·2·8)*	12,288·*MB_x·MB_y*
HR_S	*(2·MB_x·2·MB_y·16·16·8)*	*(2·MB_x·2·MB_y·8·8·2·8)*	12,288·*MB_x·MB_y*
3 HR Stripes	*(2·3·2·MB_y·16·16·8)*	*(2·3·2·MB_y·8·8·2·8)*	36,864·*MB_y*
LR_B	*(MB_x·MB_y·16·16·8)*	*(MB_x·MB_y·8·8·2·8)*	*3,072·MB_x·MB_y*
LR_I[0]	*(MB_x·MB_y·16·16·8)*	*(MB_x·MB_y·8·8·2·8)*	*3,072·MB_x·MB_y*
LR_I[1]	*(MB_x·MB_y·16·16·8)*	*(MB_x·MB_y·8·8·2·8)*	*3,072·MB_x·MB_y*
LR_I[2]	*(MB_x·MB_y·16·16·8)*	*(MB_x·MB_y·8·8·2·8)*	*3,072·MB_x·MB_y*
LR_I[3]	*(MB_x·MB_y·16·16·8)*	*(MB_x·MB_y·8·8·2·8)*	*3,072·MB_x·MB_y*
MV_mem[0]	(MB_x·MB_y·8)	0	8· *MB_x·MB_y*
MV_mem[1]	(MB_x·MB_y·8)	0	8· *MB_x·MB_y*
MV_mem[2]	(MB_x·MB_y·8)	0	8· *MB_x·MB_y*
MV_mem[3]	(MB_x·MB_y·8)	0	8· *MB_x·MB_y*
Total (bits)	MB_y·(35,872·MB_x + 24,576)	MB_y·(17,920· MB_x + 12288)	MB_y ·(53,792 MB_x + 36,864)

Table 1: Memory requirements of the ISR as a function of the number of the input image macro-blocks.

3. EXPERIMENTAL SETUP

A large set of synthetic sequences have been generated with the objective of assessing the algorithm itself, independently of the image peculiarities, and to enable the measure of reliable metrics. These sequences share the following characteristics: firstly, in order to isolate the metrics from the image peculiarities, the same frame has been replicated all over the sequence. Thus, any change in the quality will only be due to the algorithm processing and not to the image entropy. Secondly, the displacements have been randomly generated, except for the first image of the low-resolution input set, used as the reference for the motion computation, where a null displacement is considered. This frame is used as the reference in the peak signal to noise ratio (PSNR) computation. (Fig. 4) depicts the experimental setup to generate the test sequences [42].
The displacements introduced in the VGA images in pixel units are reflected in the low-resolution input pictures divided by four, i.e. in quarter pixel units. As this is the precision of the motion estimator, the real (artificially introduced) displacements and the ones delivered by the motion estimator are compared, in order to assess the goodness of the motion estimator used to compute the shifts among images. Several sets of input frames from random motion vectors have been generated. These synthetic sequences are used as the input for the SR process. (Fig. 6)-(a) shows the reference picture KRANT together with the sub-sampled sequences that constitute the input low-resolution sequence (b) and the nearest neighbour (c) and bilinear interpolations (d) obtained from the first low-resolution frame (frame with zero motion vector).
The pictures obtained with the SR algorithms are always compared to the ones obtained with the bilinear and nearest neighbour replication interpolations in terms of PSNR. In this work, the quality of the SR algorithms are compared with the bilinear and nearest neighbour interpolation algorithms as they suppose an alternative way to increase the image resolution without the complexity that SR implies. The main difference between interpolation and SR is that the later adds new information from other images while the former only uses information from the same picture. The PSNR

obtained with interpolation methods represents a lower bound in the sense that a PSNR above the interpolation level implies SR improvements.

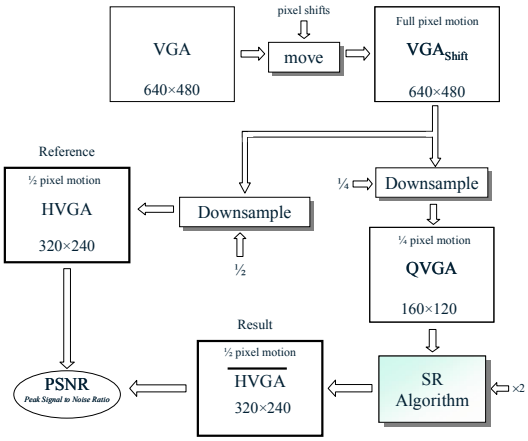

Figure 4: Experimental setup for the test sequences generation.

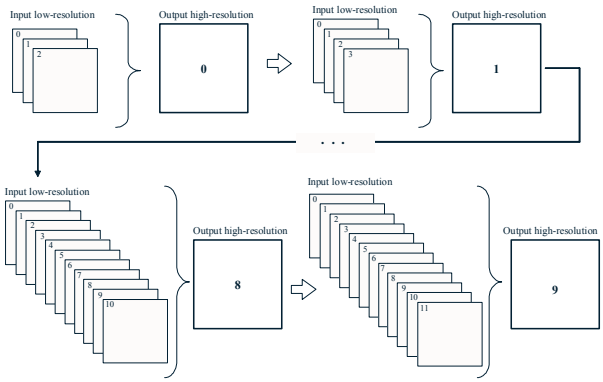

Figure 5: Incremental test for assessing the SR algorithms.

In order to demonstrate the quality increase in the SR image when combining several low resolution images, the experiment denoted in (Fig. 5) has been designed. In this experiment, referred as the incremental test, a set of 12 displacement vectors have been generated, wherein the first is the zero vector and the remaining are eleven random vectors. The first displacement vector is (0,0) to assure that the resulting image will remain with zero displacement with respect to the reference, enabling reliable quality measurements. From this vector set, the first three vectors are applied to the first frame of the KRANT sequence in order to generate the low-resolution input image set, from which the super-resolved image zero is obtained. After that, a new vector is added to the previous set and these four vectors are applied again to the frame 0 of KRANT to generate the super-resolved image one, based on four input low-resolution images. This process is repeated until a super-resolved image based on 12 low-resolution input frames is generated. In total, a number of 3+4+5+6+7+8+9+10+11+12=75 low-resolution frames have been used as inputs to the SR algorithms in order to generate 10 output frames.

(a) (b) (c) (d)

Figure 6: Reference KRANT picture (a), the low-resolution input sequence derived from it (b) and the nearest neighbour (c) and bilinear interpolations (d).

4. ISR ALGORITHM RESULTS

In this section the test procedures exposed in the previous section have been applied to the ISR algorithm. The incremental test described in (Fig. 5) was applied to the ISR algorithm using the KRANT sequence. Initially, an amount of 80 iterations were settled for every output frame, obtaining the luminance PSNR shown in (Fig. 7). From this chart, it is noticeable that for certain frames (combinations of more than 8 frames) the quality rises up to a maximum value as the number of iterations increases, while for the other frames, the quality starts to rise and after a few iterations it drastically drops. The reason of this unexpected result is that the displacements were randomly generated and so, the samples presented in each frame are randomly distributed. If the samples contain all the original information (fragmented over the input frames) then the SR process will be able to properly reconstruct the image. If some data is missing in the sampling

task, then the SR process will try to adapt the SR image to the available input set, including the missing data that has been set to zero values, producing undesirable artefacts when there is a lack of information. As the motion among frames has been randomly generated, the probability of having the entire original input data distributed among the low-resolution frames available to the SR algorithm increases with the number of incoming frames. In this case, after the combination of 8 low-resolutions frames, the ISR algorithm is able to increase the quality with the iterations number.

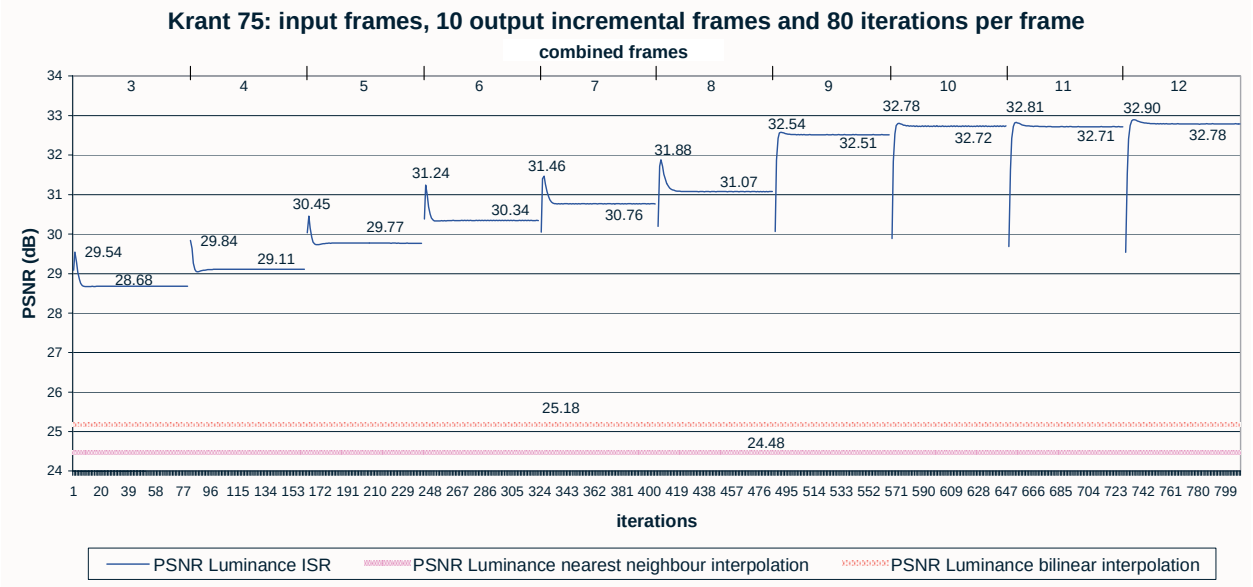

Figure 7: PSNR of the KRANT sequence with 10 incremental output frames using the ISR algorithm with 80 iterations.

Three enlarged details of the pencils of the KRANT frame are shown in (Fig. 8). Image (a) is the nearest neighbour interpolation, (b) is the SR image and (c) is the bilinear interpolation of the input low-resolution sequence.

(a) (b) (c)

Figure 8: Enlarged detail of the nearest neighbour interpolation of the input image (a), the ISR image combining 12 low-resolution input frames (b) and the bilinear interpolation of the input image (c) for the KRANT sequence.

5. NON-ITERATIVE SUPER-RESOLUTION ALGORITHM

Although the iterative version previously described offers very good image quality when mapped onto a hybrid video encoder, the challenge is to create a new type of algorithm that, using the same resources, could operate in a single step, i.e. a non iterative algorithm suitable for real time applications. The underlying idea is based on the following considerations:

- Every new image adds new information that must be combined into a new high-resolution grid.
- It is impossible to know 'a priori' (for the SR algorithm scope) the position of the new data and whether or not they contribute with new information

Based on the previous considerations, a novel non-iterative super-resolution (NISR) algorithm has been developed. This algorithm performs its operations by considering the following steps:

1. Initially, the first low-resolution image is translated onto a high-resolution grid, leaving the unmatched pixels to a zero value. The size increases in a factor of two, both in the horizontal and vertical directions.
2. Next, the contributions of the pixels are generated. These contributions represent the amount of information that each low-resolution pixel provides to its corresponding neighbours in the high-resolution grid. As several low-resolution images are initially combined in a grid 2-by-2 times bigger, an initial contribution of 4, for ½ pixel precision in low-resolution will be enough in order to keep contributions as integer values. If the resolution of the motion estimator is increased or the motion-estimation is performed in high-resolution, higher values are necessary. These contributions are expressed over the high-resolution grid.
3. The relative displacements between the next input image and the first image, that it is considered as the reference one, are estimated. These displacements are stored in memory, as they will be used later on.
4. Steps 1 and 2 are applied to the new input image, i.e. it is adapted to the high-resolution grid, leaving the missing pixels to zero and generating the initial contributions.
5. In this step, both the new image over the high-resolution grid and its associated contributions are motion compensated towards the reference image. The real contributions of the new pixels to the high-resolution reference image will be reflected in the compensated contributions.
6. The arithmetical addition between the initial image and the compensated images is performed. The same process is completed with initial and compensated contributions. This summation assures further noise reduction in the resulting image.
7. Steps 3 to 6 are applied to the next incoming images.
8. Once step 7 is finished, a high-resolution image with the summation of all the compensated pixels (stored in HR_A) and a memory with the summation of all the compensated contributions (HR_Cont) are obtained. Then the high-resolution image is adjusted depending on the contributions, as it is indicated in equation (1), where 'N' stands for the number of frames to be combined, '*SR(i, j)'* is the SR image, '*LR_I'* is the low-resolution input image and '*contributions*' represents the contributions memory. Assigning to the accumulative memory HR_A a length of 12 bits, 16 frames can be safely stored in it. In the worst case, an accumulation of a value of 255 will be performed, which multiplied by 16 gives 4080, that fits in 12 bits.
9. $$SR(i,j) = N \cdot \frac{\sum_{fr=1}^{N} Motion\_Compensate(Upsample\_Holes(LR\_I[fr]))}{\sum_{fr=1}^{N} Motion\_Compensate(contributions[fr])} \qquad (1)$$
10. After the adjustment, it is possible that some pixel positions remain empty, because certain positions from the input image set did not add new information. This case will be denoted with a zero value, both in the high-resolution image position and in the contributions. The only solution to this problem is to interpolate the zeroes with the surrounding information.
11. As HR_B will store the final super-resolved image, its values cannot be neither below zero nor above 255. Therefore, a clip of the final pixel values between 0 and 255 is made.

In (Fig. 9) can be seen the NISR algorithm, where the previous steps numbers have been introduced between parenthesis as labels in the appropriate lines for clearness. In (Fig. 10) it is shown the NISR algorithm data flow, using the memories and the resources of the hybrid video encoder. Once again memory HR_A is in boldface to remark the different bit wide when compared to the other memories. The block-diagram has been divided in two: on the left side is the zone dedicated to the image processing, which makes use of memories HR_A, HR_T, HR_S, LR_I_0 and LR_I, besides of storing the motion vectors in MV_ref2fr. On the right side is the zone dedicated to the contributions processing, which makes use of memories HR_S2, HR_T2 and HR_Cont. In order to clarify the relations among them, the image data flow has been drawn in solid lines, the contribution data flow in dotted-lines and the motion vector flow in dashed-lines. Moreover, the functions '*upsample*' and '*motion compensation*' have been superscripted with and asterisk to point out their different behavioural when they are executing the in SR mode.

```
(1) HR_A.lum  = Upsample_Holes(LR_I_0.lum)
(1) HR_A.chrom = Upsample_Neighbours(LR_I_0.chrom)
(2) HR_Cont  = Create_image_contributions
(7) FOR fr = 1 .. nr_frames-1
      (3) MV = Calc_Motion_Estimation ( LR_I, LR_I_0)
      (3) MV = 2 .* MV
      (4) HR_S.lum     = Upsample_Holes(LR_I.lum)
      (4) HR_S.chrom = Upsample_ Neighbours(LR_I.chrom)
      (4) HR_S2 = Create_image_contributions
      (5) HR_T   = Motion_Compensation(HR_S, MV)
      (5) HR_T2 = Motion_Compensation(HR_S2, MV)
      (6) HR_A  = HR_A + HR_T
      (6) HR_Cont = HR_Cont + HR_T2
(7) END FOR
(8) HR_A = 4*HR_A /HR_Cont
(9) If  (HR_Cont(i,j)==0) THEN HR_B = Interpolate(HR_A(i,j))
(9) ELSE   HR_B = HR_A
(10)Clip(HR_B, 0, 255) // result image in HR_B
```

Figure 9: Pseudo-code of the NISR algorithm implemented on the video encoder.

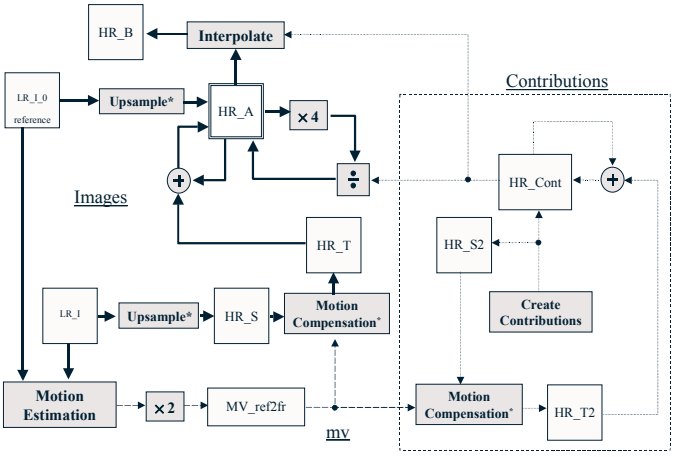

Figure 10: NISR algorithm data flow.

5.1. Implementation issues

In order to fit the NISR algorithm in the video encoder originally developed by Philips Research, it is necessary to perform additional changes in the architecture, namely in the motion compensator and in the chrominance treatment, thereby creating a more flexible SOC platform.

Table 2 summarizes the memory requirements that the NISR algorithm requests. Compared with the ISR it can be appreciated that now there are five high-resolution memories instead of three, although the low-resolution and motion estimation memories have been reduced from four to two and from four to one respectively.

Label	NISR Algorithm Memory		
	Luminance (bits)	Chrominance (bits)	Total (bits)
HR_A	*(2·MB_x·2·MB_y·16·16·12)*	*(2·MB_x·2·MB_y·8·8·2·12)*	*18,432·MB_x·MB_y*
HR_B	*(2·MB_x·2·MB_y·16·16·8)*	*(2·MB_x·2·MB_y·8·8·2·8)*	12,288·*MB_x·MB_y*
HR_S	*(2·MB_x·2·MB_y·16·16·8)*	*(2·MB_x·2·MB_y·8·8·2·8)*	12,288·*MB_x·MB_y*
HR_S2	*(2·MB_x·2·MB_y·16·16·8)*	*(2·MB_x·2·MB_y·8·8·2·8)*	12,288·*MB_x·MB_y*
HR_Cont	*(2·MB_x·2·MB_y·16·16·8)*	*(2·MB_x·2·MB_y·8·8·2·8)*	12,288·*MB_x·MB_y*
3 Stripes HR	*(2·3·2·MB_y·16·16·8)*	*(2·3·2·MB_y·8·8·2·8)*	36,864·*MB_y*
LR_I[0]	*(MB_x·MB_y·16·16·8)*	*(MB_x·MB_y·8·8·2·8)*	*3,072·MB_x·MB_y*
LR_I[1]	*(MB_x·MB_y·16·16·8)*	*(MB_x·MB_y·8·8·2·8)*	*3,072·MB_x·MB_y*
MV_mem[0]	(MB_x·MB_y·8)	0	8· *MB_x·MB_y*
Total (bits)	MB_y·(49,160·MB_x + 24,576)	MB_y·(24,576· MB_x + 12288)	MB_y ·(73,736 MB_x + 36,864)

Table 2: Memory requirements of the NISR as a function of the number of the input image macro-blocks.

5.2. Adjustments in the motion compensator

The motion compensator implemented in the existing video encoder was designed to avoid visual distortions in the resulting images when decompressing them, and in that sense, when an image is shifted out of the physical boundaries, it fills the empty zone by replicating the borders. As the motion vectors are usually small compared with the image size and due to the lower attention of the human eye to the borders when compared with the centre of the images, this effect is negligible. However, to obtain SR improvements the artificial introduction of non-existing data results in quality degradation in the borders. The solution is to modify the motion compensator to fill the empty values with zeroes, so that the NISR algorithm would have an opportunity to fill the holes with valid values coming from other images.

6. NISR ALGORITHM RESULTS

The qualities resulting from applying the incremental test to the KRANT sequence are shown in (Fig. 11). As it was expected, as the number of input frames to be combined increases, the PSNR increases until it reaches a maximum of 38.45 dB when combining 12 low-resolution input frames. The perceptual quality exhibits few variations after combining 6 input frames, i.e. when 34.57 dB are reached. In addition, it can be seen that the greatest increment in the quality (the greatest PSNR slope) takes place in the first four output frames. All the established facts lead to the conclusion that a NISR system can be limited to combine 5 or 6 input frames, depending on the available resources and the desired output quality.
In (Fig. 12) is shown an enlarged detail of the frame obtained as the gathering of 12 frames by the NISR algorithm. In (a) it is shown the nearest neighbour interpolation, in (b) the SR image and in (c) the bilinear interpolation. The quality improvement of the image is clearly manifested, especially in the case of the letters above the newspaper headlines that become legible in the super-resolved image.

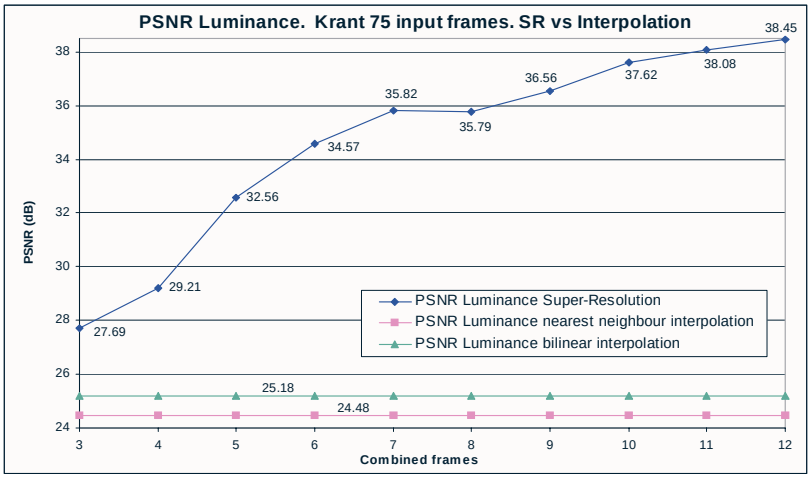

Figure 11: Incremental test of the KRANT sequence with 10 output frames.

(a) (b) (c)

Figure 12: Enlarged detail of the nearest neighbor interpolation of the input image (a), the NISR image combining 12 low-resolution input frames (b) and the bilinear interpolation of the input image (c) for the KRANT sequence.

7. ALGORITHMS COMPARISON

The two developed SR algorithms have been compared attending to three main features: the quality reached (measured in PSNR dBs), the simulation time (measured in milliseconds) and the memory requirements (measured in Mbytes).
In (Fig. 13) it is exposed a comparison between the quality obtained by the NISR algorithm and the best results of the ISR algorithm (maximum values for 80 iterations). It is clear how the NISR algorithm outperforms the ISR algorithm

when the number of low-resolution frames is above 4 frames. The larger PSNR difference occurs when 12 low-resolution frames are combined, getting a gain of 5.54 dBs the NISR over the ISR algorithm. The average difference is of 3.08 dB from the combination of 3 to 12 low-resolution frames.
In (Fig. 14) is shown the ISR frame number 9, as a result of the combination of 12 low-resolution frames. Image (a.1) is the SR frame in the spatial domain, and (a.2) is the error image when compared with the original one. Major errors are located in the edges zones, i.e. in the high-frequencies. The bi-dimensional Fourier transform in magnitude is shown in (b.1) and the error image is in (b.2). As expected, the central zone of the magnitude, corresponding to the lower spatial frequencies, exhibit the lower errors. (Fig. 15) shows the same results but for the NISR algorithm. Although the perceptual qualities seem to be very similar, the NISR achieve lower errors in the edges in the spatial domain and a wider area of low error in the frequency domain, especially in the central low-frequencies zone.
(Fig. 16) shows the simulation time for the two described algorithms applied the test video input sequences KRANT. In both cases 10 output frames have been generated. The results of the SR algorithms come from the incremental combination of three to twelve low-resolution frames, but the ISR algorithm performs 8 iterations over the low-resolution frames to be combined. The total amount of time inverted in the execution of the NISR algorithm is five times lower than the ISR algorithm, placing the former in a best situation for a real-time execution scenario.

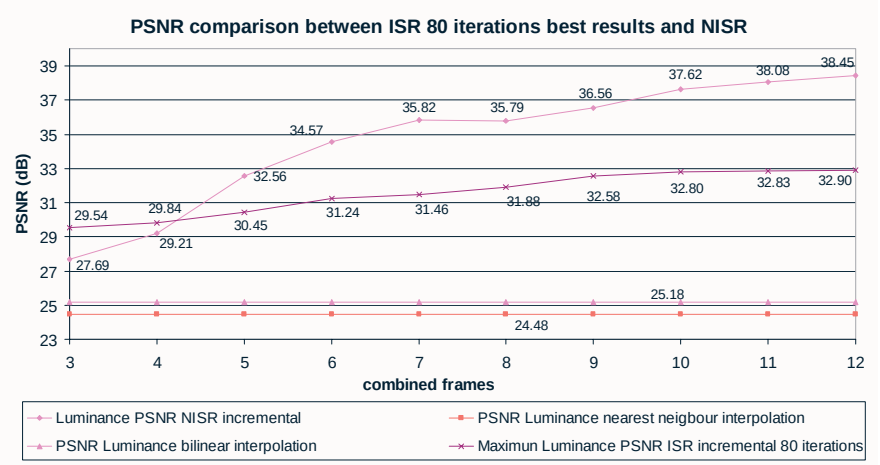

Figure 13: PSNR of the KRANT sequence with 10 incremental output frames using the NISR and the ISR algorithm with 80 iterations.

One of the most restrictive hardware resources in the encoder platform is the memory, as it plays a crucial role in the final integrated circuit cost and in the power consumption. Based on tables 1 and table 2, table 3 and table 4 summarize the memory requirements of the ISR and the NISR algorithms for the most common input sizes. (Fig. 17) shows a comparison of the total memory requirements of the two algorithms. Although until now the NISR algorithm has exhibited better time responses and better qualities than the ISR algorithm, this fact comes at the cost of higher memory requirements. The memory needed by the NISR algorithm is 1.36 times above the ISR memory. This is mainly due to the introduction of the high-resolution memories HR_Cont and HR_S2.

Size	MB_x	MB_y	Memory (Kbytes)
SQCIF (128×96)	8	6	342.1875
QCIF (176×144)	11	9	690.5742
CIF (352×288)	22	18	2681.2968
VGA (640×480)	40	30	8014.6875
4CIF (704×576)	44	36	10563.1875

Table 3: Memory requirements of the ISR as a function of the number of the input image macro-blocks.

Size	MB_x	MB_y	Memory (Kbytes)
SQCIF (128×96)	8	6	459.0468
QCIF (176×144)	11	9	931.5966
CIF (352×288)	22	18	3645.3867
VGA (640×480)	40	30	10936.1718
4CIF (704×576)	44	36	14419.5468

Table 4: memory requirements of the NISR algorithm for different sizes of the input image.

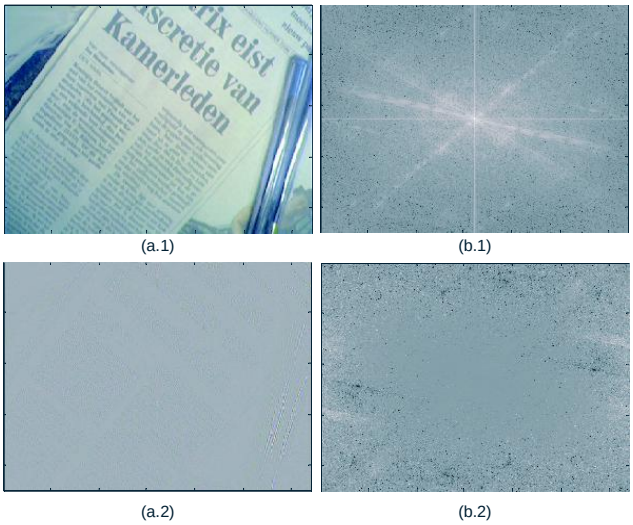

Figure 14: ISR frame after combining 12 low-res frames (a.1) and the magnitude in the frequency domain (b.1) together with their associated errors (a.2 and b.2).

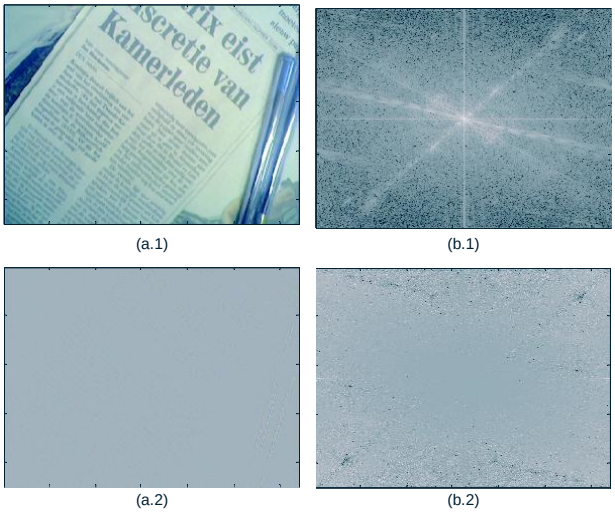

Figure 15: NISR frame after combining 12 low-res frames (a.1) and the magnitude in the frequency domain (b.1) together with their associated errors (a.2 and b.2).

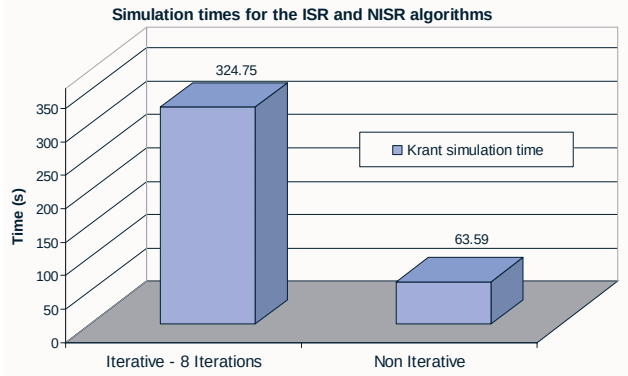

Figure 16: Execution time comparison between the ISR and NISR algorithms presented in this paper

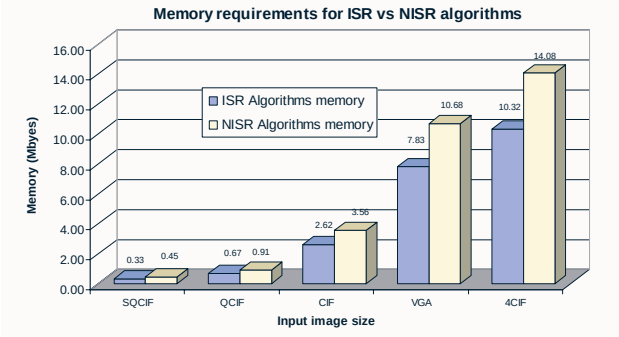

Figure 17: Memory requirements for the ISR and NISR algorithms.

8. CONCLUSIONS

High resolution images are often required in nowadays multimedia services and applications. In this way, the achievement of high quality images by using low-cost SR techniques mapped onto a generic hybrid video encoder has been presented in this paper. Low-cost constraints are accomplished by performing only minimal changes on the architecture, such as using sub-pixel motion estimation, enough loop memory and additional block-level operator and accumulators. In particular, two different SR algorithms are proposed in this work. The fist one, named iterative super-resolution (ISR) algorithm, presents a much higher performance than classical image interpolation methods. However, it is not always possible to achieve SR improvements by using this approach, because if all the sampled data is not present (this can happen in non-type 'a' frames), the quality decreases with the number of iterations due to the lack of additional 'a priori' information included in the algorithm. In that sense, it is preferable to limit the number of iterations, as it has been stated in this paper. Although the ISR algorithm exhibits a good behaviour and robustness in presence of noise and/or inaccurate motion knowledge as well as low memory utilization, real-time conditions are not guaranteed due to its iterative nature. In order to solve this drawback, a non-iterative super-resolution (NISR) algorithm has been developed. The experiments carried out reveal a clear quality increase of the super-resolved image as the number of low-resolution frames to be combined also increases. The introduction of the contribution concept allows the algorithm to be independent from the problems of the borders and, at the same time, supposes adaptive weights for every pixel, depending on the motion and therefore on the new incoming information. The NISR algorithm allows obtaining higher

image qualities than the ones obtained by the ISR algorithms in a single step, but at the expenses of using higher amounts of memory.
The SR algorithms developed have been successfully implemented onto a HW/SW platform by reusing a generic hybrid video encoder instead of developing a specific SR system. The most remarkable results obtained in the implementation process in terms of simulation time, memory usage and image quality for both cases have been also presented in this paper, establishing a detailed comparison among the two algorithms. The followed methodology assures that the platform can be used both in coding mode and/or in SR mode, opening new avenues in the field of high performance multiprocessor system on chip (MSoC) video platforms.

REFERENCES

1. T. Komatsu, T. Igarashi, K. Aizawa, T. Saito, "Very high resolution imaging scheme with multiple different-aperture cameras," Signal Processing: Image Communication vol. 5, pp. 511-526, Dec. 1993.
2. P. Cheeseman, B. Kanefsky, R. Kraft, J Stutz and R. Hanson, "Super-resolved surface reconstruction from multiple images," technical report FIA-94-12, NASA Ames Research Center, Moffet Field, CA, December 1994.
3. V. Avrin and I. Dinstein, "Restoration and resolution enhancement of video sequences," ICASSP IV, pp. 549-553, 1997.
4. Peter Eggleston, "Quality Photos from Digital Video: Salient Stills' Algorithms Provide a Missing Link," Advanced Imaging, pp. 39-41, May 2000.
5. D. Sale, R. R. Schultz and R.J. Szczerba, "Super-resolution enhancement of night vision image sequences, " IEEE International Conference on Systems, Man, and Cybernetics, vol. 3 , pp. 1633 -1638, 2000.
6. B. K. Gunturk, A.U. Batur, Y. Altunbasak, M.H. Hayes and R.M. Mersereau, "Eigenface-based super-resolution for face recognition," Proceedings of the International Conference on Image Processing, vol. 2, pp. 845 -848, 2002.
7. D. Capel and A. Zisserman, "Super-resolution enhancement of text image sequences," Proceedings of the 15th International Conference on Pattern Recognition, vol 1 , pp. 600-605, 2000.
8. J. A. Goyette, G. D. Lapin, Moon Gi Kang and A. K. Katsaggelos , "Improving autoradiograph resolution using image restoration techniques," IEEE Engineering in Medicine and Biology Magazine, vol. 13, no. 3, pp. 571-574, Sept. 1994.
9. K.J. Erickson and R.R. Schultz, "MPEG-1 super-resolution decoding for the analysis of video still images," Proceedings of the 4th IEEE Southwest Symposium on Image Analysis and Interpretation, pp. 13 -17, 2000.
10. B. Martins and S. Forchhammer, "A unified approach to restoration, deinterlacing and resolution enhancement in decoding MPEG-2 video, " IEEE Transactions on Circuits and Systems for Video Technology, vol.12, issue 9, pp. 803 -811, 2002.
11. Y. Altunbasak, A.J. Patti and R.M. Mersereau, "Super-resolution still and video reconstruction from MPEG-coded video," IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, issue 4, pp. 217-226, April 2002.
12. C.A. Segall, R. Molina and A. Katsaggelos," High Resolution Images from a Sequence of Low Resolution and Compressed Observations: A Review," IEEE Signal Processing Magazine, vol. 20, no. 3, pp. 37-48, May 2003.
13. Frank M. Candocia, "A Unified Superresolution Approach for optical and synthetic Aperture Radar Images," Ph.D. dissertation, University of Florida, 1998.
14. Y. Cheng, Y. Lu and Z. Lin, "A super resolution SAR imaging method based on CSA," IEEE International Geoscience and Re-mote Sensing Symposium IGARSS '02, vol. 6, pp. 3671 -3673, 2002.
15. D. Pastina, P. Lombardo, A. Farina and P. Daddi, "Super-resolution of polarimetric SAR images of a ship," IEEE 2001 International Geoscience and Remote Sensing Symposium, IGARSS '01, vol. 5, pp. 2343-2345, 2001.
16. H. Yamada, Y. Yamaguchi, E. Rodriguez, Y. Kim and W.M. Boerner, "Polarimetric SAR interferometry for forest canopy analysis by using the super-resolution method," IEEE 2001 International Geoscience and Remote Sensing Symposium, IGARSS '01, vol. 3 , pp. 1101 -1103, 2001.
17. A. J. Tatem, H.G. Lewis, P.M. Atkinson and M.S. Nixon, "Super-resolution mapping of multiple-scale land cover features using a Hopfield neural network," IEEE International Geoscience and Remote Sensing Symposium, IGARSS '01, vol. 7, pp. 3200 -3202, 2001.
18. W.T. Freeman, T.R. Jones and E.C. Pasztor, "Example-based super-resolution," IEEE Computer Graphics and Applications, vol. 22, issue 2, pp. 56 -65, March/April 2002.
19. H. Nagahara, Y. Yagi and M. Yachida, "Resolution improving method for a 3D environment modeling using omnidirectional image sensor," Pro-ceedings of the IEEE International Conference on Robotics and Automation (ICRA '02), vol. 1, pp. 900 -907, 2002.

20. T. S. Huang and R. Y. Tsay, “Multiple Frame Image Restoration and Registration,” Advances In Computer Vision and Image Processing (Ed. -T. S. Huang), vol. 1, JAI Press Inc., Greenwich, CT, 1984, pp. 317-339.
21. L. J. Yen, “On Nonuniform Sampling of Bandwidth Limited Signals,” IRE Trans. Circuits Theory, vol. 3, pp. 251-257, April 1956.
22. A. Papoulis, “Generalized Sampling Theorem,” WEE Trans. Circuits and Systems, vol. 24, pp. 652-654, November 1977.
23. M. B. Zervakis, “Optimal Restoration of Multichannel Images based on Constrained Mean-Square Estimation,” Journal of Visual Communication and Image Representation, vol. 3 pp. 392-411, December 1992.
24. A. K. Katsaggelos, “A Multiple Input Image Restoration Approach,” Journal of Visual Communication and Image Representation, vol. 1, pp. 93-103, September 1990.
25. A. K. Katsaggelos, R. P. Kleihorst, S. N. Efstratiadis and R. L. Lagendijk, “Adaptive Image Sequence Noise Filtering Methods,” in Proc. SPIE - The international Society for optical Engineering, vol. 1606, pp. 716-727, 1991.
26. A. J. Patti, A. M. Tekalp and M. T Sezan, “Image Sequence Restoration and De-Interlacing by Motion-Compensated Kalman Filtering,” SPIE, vol. 1903, 1993.
27. M. Elad and A. Feuer, “Restoration of Single Super-Resolution Image From Several Blurred, Noisy and Down-Sampled Measured Images,” IEEE Trans. on Image Processing, vol. 6, no. 12, pp.1646-1658, 1997.
28. J. Park, S. Rhee and Moon Gi Kang, “Multichannel dealiasing technique for superresolution,” ITC98, 1998.
29. R. L. Lagendijk and J. Biemond, Iterative Identification And Restoration Of Images, Kluwer Academic Publishing, Boston, 1991.
30. A. Zomet, A. Rav-Acha and S. Peleg, “Robust super-resolution,” Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), vol. 1, pp. I-645 -I-650, 2001.
31. D. C. Youla, “Generalized Image Restoration by the Method of Alternating orthogonal Projections,” IEEE Transactions on Circuits & Systems, vol. 25, pp. 694-702, 1978.
32. W. Zhao, H. Sawhney, M. Hansen and S. Samarasekera, “Super-fusion: a super-resolution method based on fusion,” Proceedings of the 16th International Conference on Pattern Recognition, vol. 2, pp. 269 -272, 2002.
33. M. Irani and S. Peleg, “Motion Analysis for Image Enhancement: Resolution, Occlusion, and Transparency,” Journal of Visual Communication and Image Representation (VCIR), vol. 4, pp. 324-335, December 1993.
34. D. Capel and A. Zisserman, “Super-resolution from multiple views using learnt image models,” Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), vol. 2, pp. II-627-II-634, 2001.
35. S. Baker and T. Kanade, “Limits on super-resolution and how to break them,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, issue 9, pp. 1167 -1183, Sep 2002.
36. Subhasis Chaudhuri editor, Super-resolution Imaging, Kluwer Academic Publishers, 2001.
37. Marc J. Op De Beeck and Richard P. Kleihorst, “Super-Resolution of Regions of Interest in a Hybrid Video Encoder,” Philips conference on DSP, 1999.
38. Gustavo M. Callicó, Rafael P. Llopis, Antonio Núñez, Ramanathan Sethuraman, “Mapping of Real-Time and Low-Cost Super-Resolution Algorithms on a Hybrid Video Encoder”, SPIE, vol. 5117, pp. 42-52, Maspalomas, Spain, May 2003.
39. R. Peset Llopis, M. Oosterhuis, S. Ramanathan, P. Lippens, R. Kleihorst, R. van der Vleuten and J. Lin, "A Low-Cost Low-Power H.263 Video Encoder for Mobile Applications," Second International Symposium on Mobile Multimedia Systems & Applications, Delf, The Netherlands, Nov. 2000.
40. Gustavo M. Callicó, Rafael P. Llopis, Antonio Núñez, Ramanathan Sethuraman, Marc Op de Beeck. “A Low-Cost Implementation of Super-Resolution based on a Video Encoder,” IEEE IECON, vol. 2, pp. 1439-1444, Seville, Spain, Nov. 2002.
41. R. Peset Llopis, M. Oosterhuis, S. Ramanathan, P. Lippens, A. van der Werf, S. Maul and J. Lin, “HW-SW Codesign and Verification of a Multi-Standard Video and Image Codec,” IEEE ISQED, San Jose, California, March 2001, pp. 393-398.
42. Gustavo M. Callicó, Rafael P. Llopis, Antonio Núñez, Ramanathan Sethuraman. “Low-Cost and Real-Time Super-Resolution over a Video Encoder IP,” IEEE ISQED, pp. 79-84, San Jose, California, USA, March 2003.