

Low-Cost Super-Resolution Algorithms Implementation over a HW/SW Video Compression Platform

Gustavo M. Callicó,¹ Rafael Peset Llopis,² Sebastian López,¹ José Fco. López,¹ Antonio Núñez,¹ Ramanathan Sethuraman,³ and Roberto Sarmiento¹

¹ *The University of Las Palmas de Gran Canaria, Institute for Applied Microelectronics (IUMA), Tafira Baja, 35017, Spain*

² *Philips Consumer Electronics, SFJ-6, P.O. Box 80002, 5600 JB, The Netherlands*

³ *Philips Research Laboratories, WDC 3.33, Professor Holstlaan 4, 5656 AA Eindhoven, The Netherlands*

Received 1 December 2004; Revised 5 July 2005; Accepted 8 July 2005

Two approaches are presented in this paper to improve the quality of digital images over the sensor resolution using super-resolution techniques: iterative super-resolution (ISR) and noniterative super-resolution (NISR) algorithms. The results show important improvements in the image quality, assuming that sufficient sample data and a reasonable amount of aliasing are available at the input images. These super-resolution algorithms have been implemented over a codesign video compression platform developed by Philips Research, performing minimal changes on the overall hardware architecture. In this way, a novel and feasible low-cost implementation has been obtained by using the resources encountered in a generic hybrid video encoder. Although a specific video codec platform has been used, the methodology presented in this paper is easily extendable to any other video encoder architectures. Finally a comparison in terms of memory, computational load, and image quality for both algorithms, as well as some general statements about the final impact of the sampling process on the quality of the super-resolved (SR) image, are also presented.

Copyright © 2006 Hindawi Publishing Corporation. All rights reserved.

1. INTRODUCTION

Here 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, resulting 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 decreases. 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 $50 \mu\text{m}^2$ [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 observations of the scene using lower-resolution sequences. 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 as follows.

- (i) Still-image improvement [1–4], where several images from the same scene are obtained and used to construct a higher-resolution image.
- (ii) Analog video frame improvement [5, 6]. Due to the low quality of analog video frames, they are not normally suitable to directly perform a printed-copy digital photography. The quality of the image is increased using several consecutive frames combined in a higher-resolution image by using SR algorithms.
- (iii) Surveillance systems [7], where SR is used to increase the quality in video surveillance systems, using such recorded sequences as forensic digital video, and even to be admitted as evidence in the courts of law. SR improves night vision systems when images have been acquired with infrared sensors [8] and helps in the face recognition process for security purposes [9].
- (iv) Text extraction process from image sequences [10] is highly improved if the regions of interest (ROI) containing the text are first super-resolved.

- (v) Medical image acquisition [11]. Many medical types of equipment as the computer-aided tomography (CAT), the magnetic resonance images (MRI), or the echography or ultrasound images allow the acquisition of several images, which can be combined in order to obtain a higher-resolution image.
- (vi) Improvement of images from compressed video [12–15]. For example, in [16] the images high-frequency information recovery, lost in the compression process, is addressed. The missing data are 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 [17].
- (vii) Improvement of radar images [18, 19]. In this case SR allows a clearer observation of details sometimes critical for air or maritime security [20] or even for land observations [21–24].
- (viii) Quality improvement of images obtained from the outer space. An example is exposed in [4] with images taken by the Viking satellite.
- (ix) Image-based rendering (IBR) of 3D objects uses cameras to obtain rich models directly from the real-world data [26]. SR is used to produce high-resolution scene texture from an omnidirectional image sequence [26, 27].

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 constraints are accomplished by reusing a video encoder, rather than developing a specific hardware. This encoder can be used either in the compression mode or in the 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 dissipation. 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 consist 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 is achieved by recording a video sequence at high frame rates with a hand-held camera.

The reconstruction problem using SR can be defined as the objective of reconstructing an image or video sequence with a higher quality or resolution from a finite set of lower-resolution images taken from the same scene [28, 29], as shown in Figure 1. This set of low-resolution images must be obtained under different capturing conditions of the image, from different spatial positions, and/or from different cameras. This reconstruction problem is an aspect of the most general problem of sensor fusion.

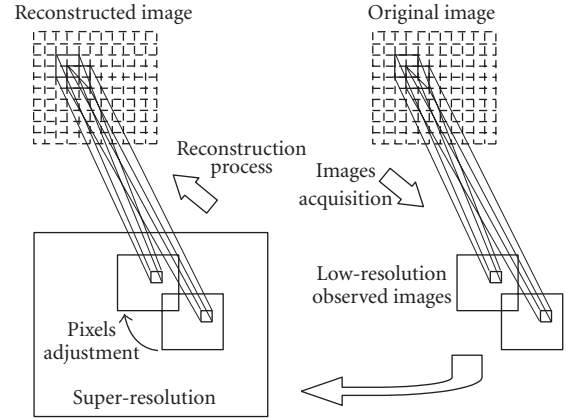


FIGURE 1: Model of the reconstruction process using super-resolution.

The rest of the paper is organized as follows. Firstly, the most important publications directly related to this work are reviewed, followed by a brief description of the hybrid video compression architecture where the developed SR algorithms have been mapped. In the second section the bases of the ISR algorithms are established while in Section 3 the modifications needed to be implemented onto the video encoder are described. In Section 4 the experimental setup to evaluate the quality of the iterative and noniterative algorithms is presented, and based on it, a set of experiments is developed in Section 5 in order to assess the correct behavior of the ISR algorithm, showing as a result an important increase in the super-resolved output images. As far as an iterative behavior seriously jeopardizes a real-time implementation, in Section 6 a novel SR algorithm is described, where the previous iterative feature has been removed. In the same section, the adjustments carried out in the architecture in order to obtain a feasible implementation are explained, while Section 7 shows the results achieved with this noniterative algorithm. In Section 8 the advantages and drawbacks of the described ISR and NISR algorithms are compared and finally, in Section 9, the most remarkable results of this work are presented.

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 [30], although the general sampling theorems previously formulated by Yen in [31] and Papoulis in [32] 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, with respect to the frequency domain, of demonstrating 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 nonuniform 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 analyses of the problem.

The great advances experimented by computer technology in the last years have led to a renewed and growing interest in the theory of image restoration. The main approaches are based on nontraditional treatment of the classical restoration problem, oriented towards 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 into problems of an image restoration [30, 33–36], restoration of an image sequence [37–40], and reconstruction of an image improved with SR [41–47]. This paper is based on the last mentioned approach, both for the reconstruction of static image as for the reconstruction of image sequences with SR improvements.

The classical theory of image restoration from blurred images and with noise has caught the attention of many researchers over the last three decades. In the scientific literature, several algorithms have been proposed for this classical problem and for the problems related to it, contributing to the construction of a unified theory that comprises many of the existing restoration methods [48]. 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) [48–50], maximum a posteriori (MAP) probability [48–51], and the projection onto convex sets (POCS) [52].

An alternative classification [53] based on the processing approach can be made, where the work on SR can be divided into two main categories: reconstruction-based methods [46, 54] and learning-based methods [55–57]. The theoretical foundations for reconstruction methods are nonuniform sampling theorems, while learning-based methods employ generative models that are learned from samples. The goal of the former is to reconstruct the original (supersampled) 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 based on 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 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 [58]. The work presented in this paper only deals with static SR as the output sequences do not have the same length of 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. Although some important optimization effort has been done [59], most of 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 [60], adapted to be executed inside a real video encoder, that is, restricting the operators needed to those that can be found in such kind of platforms. New operator blocks to perform the SR process have been implemented inside the existing coprocessors in order to minimize the impact on the overall architecture, as will be demonstrated in the 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 Figure 2. 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 macroblock (MB) conversions. The motion estimator processor (MEP) evaluates a set of candidate vectors received from the software part 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 MB 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 by two buses, a control bus and a data bus, each of them controlled by a bus control unit (BCU), with both buses communicating

through a bridge. Images that will be processed by the ISR and NISR algorithms come from the data bus.

2. ITERATIVE SUPER-RESOLUTION ALGORITHMS

In this section the bases for the formation of super-resolved images starting from lower-resolution images are exposed. For this purpose, if $f(\tilde{x}, \tilde{y}, t)$ represents the low-resolution input image, and it is assumed that all the input subsystem effects (lenses filtering, chromatic irregularities, sample distortions, information loss due to format conversions, system blur, etc.) are included in $h(x, y)$, the input to the iterative algorithm is obtained by the two-dimensional convolution expressed as

$$g(x, y, t) = f(\tilde{x}, \tilde{y}, t) * h(x, y), \quad (1)$$

where a lineal behavior for all the distortion effects has been supposed. Denoting $SR(\tilde{x}, \tilde{y})$ as the SR algorithm, the image obtained $S(\tilde{x}, \tilde{y}, t)$ after applying this algorithm is as follows:

$$S(\tilde{x}, \tilde{y}, t) = g(x, y, t) * SR(\tilde{x}, \tilde{y}), \quad (2)$$

where (x, y) are the spatial coordinates in the low-resolution grid, (\tilde{x}, \tilde{y}) are the spatial coordinates in the SR grid, and “ t ” represents the time when the image was acquired. These relationships are summarized in Figure 3(a) concerning the real system and are simplified in Figure 3(b).

The algorithm characterized by $SR(\tilde{x}, \tilde{y})$ starts supposing that a number of “ p ” low-resolution images of size $N \times M$ pixels are available as $g(x, y, t_i)$, where “ t_i ” denotes the sampling time of the image. The possibility of increasing the size of the output image in every direction on a predefined amount, called scale factor (SF), has been considered. Therefore, the output image has a size of $SF \cdot N \times SF \cdot M$. As the algorithm refers to only the last “ p ” images, from now on the index “ l ”, defined as $l = i \bmod p$, will be used to refer to the images inside the algorithm’s temporal window (Figure 4). Thus, the memory image $g'_l(x, y)$ is linked to $g(x, y, t_i)$ as follows:

$$g'_l(x, y) = \frac{g(x, y, t_i)}{l} = i \bmod p. \quad (3)$$

In this way, $\bar{g}'_l(x, y)$ represents the average input image, as given in (4), which is used as the first reference in the following steps:

$$\bar{g}'(x, y) = \frac{1}{p} \sum_{l=0}^{p-1} \left(\frac{1}{N \cdot M} \cdot \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} g'_l(i, j) \right), \quad \forall x, y. \quad (4)$$

The average error for the first iteration is then obtained by computing the differences between this average image and each of the input images, as shown in (5), where the superscript denotes the iteration number (first iteration in this case):

$$e_l(x, y)^{(1)} = g'_l(x, y) - \bar{g}'(x, y), \quad l = 0, \dots, (p-1). \quad (5)$$

This error must be transformed to high-resolution coordinates by means of a nearest-neighbor replication interpolator (6) of size SF. It is essential to use this type of interpolator as it will preserve the necessary aliasing required for the SR process. In Section 5, the undesirable effect of using a bilinear interpolator will be shown:

$$e_l(\tilde{x}, \tilde{y})^{(1)} = \text{upsample}(e_l(x, y)^{(1)}, SF), \quad l = 0, \dots, (p-1). \quad (6)$$

Once the upsample process has been completed, the error must be adjusted to the reference frame by shifting the error image $\Delta\delta_l(x, y)_{(fr2ref)}^{(1)}$ and $\Delta\lambda_l(x, y)_{(fr2ref)}^{(1)}$ amounts in the horizontal and vertical coordinates, respectively, where $(fr2ref)$ means that the displacement is computed from every frame to the reference and $(ref2fr)$ means that the displacement is computed from the reference to every frame. In principle, these displacements are applied to every pixel individually, depending upon the employed motion estimation technique. As far as these displacements will be used in high resolution, they must be properly scaled by SF as shown:

$$\begin{aligned} \Delta\delta_l(\tilde{x}, \tilde{y}) &= SF \cdot \Delta\delta_l(x, y), \\ \Delta\lambda_l(\tilde{x}, \tilde{y}) &= SF \cdot \Delta\lambda_l(x, y). \end{aligned} \quad (7)$$

When all the errors have been adjusted to the reference, they are averaged, taking this average as the first update of the SR image, as shown:

$$\begin{aligned} S_0(\tilde{x}, \tilde{y})^{(1)} &= \frac{1}{p} \cdot \sum_{l=0}^{p-1} e_l(\tilde{x} + \Delta\delta_l(\tilde{x}, \tilde{y})_{(fr2ref)}^{(1)}, \tilde{y} + \Delta\lambda_l(\tilde{x}, \tilde{y})_{(fr2ref)}^{(1)})^{(1)}. \end{aligned} \quad (8)$$

Equation (8) reflects the result of the first iteration, where $S_0(\tilde{x}, \tilde{y})^{(1)}$ is the first version of the SR image, corresponding to $t = t_0$, being upgraded with each iteration. The n th iteration begins obtaining a low-resolution version of this image by decimation, followed by the computation of the displacements between every one of these inputs images and this decimated image and vice versa, that is, between the decimated image and the input images. In this way, the displacements of the n th iteration will be available: $\Delta\delta_l(x, y)_{(fr2ref)}^{(n)}$, $\Delta\lambda_l(x, y)_{(fr2ref)}^{(n)}$, $\Delta\delta_l(x, y)_{(ref2fr)}^{(n)}$, and $\Delta\lambda_l(x, y)_{(ref2fr)}^{(n)}$. The low-resolution version of the image obtained in high resolution is given by (9).

$$S_0(x, y)^{(n)} = \text{downsample}(S_0(\tilde{x}, \tilde{y})^{(n-1)}, SF). \quad (9)$$

The downsample operation is defined in the following where only pixels in certain coordinates given by SF are kept:

$$\begin{aligned} g(x, y) &= \text{downsample}(f(\tilde{x}, \tilde{y}), SF), \\ g(x, y) &= f(SF \cdot x, SF \cdot y). \end{aligned} \quad (10)$$

The next step is to compensate for the motion of the high-resolution image towards the input frames using the

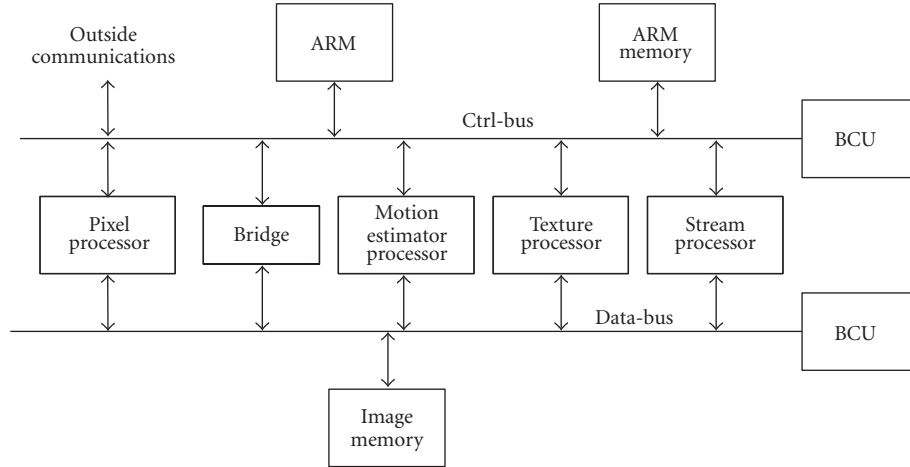


FIGURE 2: Architecture for the multistandard video/image codec developed in Philips Research.

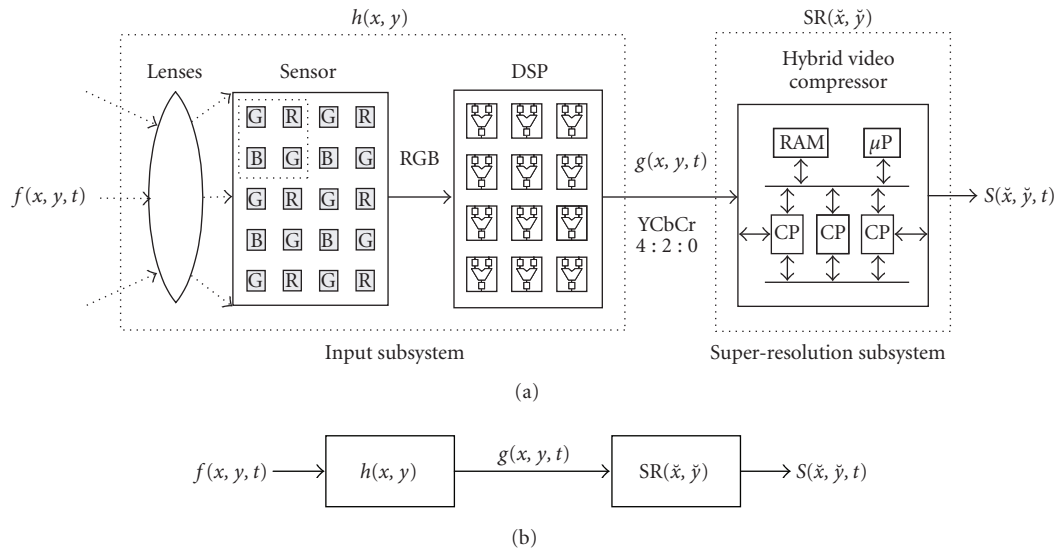


FIGURE 3: (a) Scheme of the real overall system and (b) the simplified model, together with the used terminology.

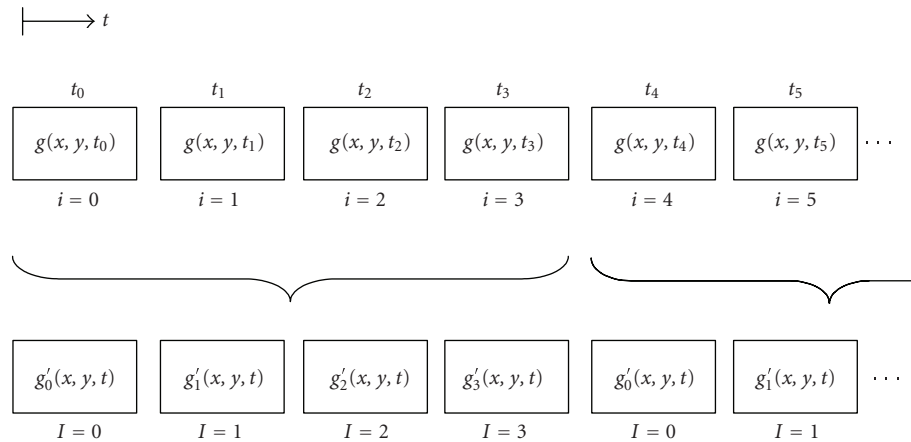


FIGURE 4: Nomenclature and numeration used to refer to the input images.

displacements $\Delta\delta_l(x, y)_{(\text{ref2fr})}^{(n)}$ and $\Delta\lambda_l(x, y)_{(\text{ref2fr})}^{(n)}$, converting them to low-resolution and getting the error with respect to every input image, as shown:

$$e_l(x, y)^{(n)} = g'_l(x, y) - S_0\left(x + \Delta\delta_l(x, y)_{(\text{ref2fr})}^{(n)}, y + \Delta\lambda_l(x, y)_{(\text{ref2fr})}^{(n)}\right)^{(n)}, \quad l = 0, \dots, (p-1). \quad (11)$$

This low-resolution error image must be transformed again into a high-resolution one through interpolation and compensate for its motion again towards the reference. The average of these “ p ” errors constitutes the n th incremental update of the high-resolution image, as shown:

$$S_0(\check{x}, \check{y})^{(n)} = S_0(\check{x}, \check{y})^{(n-1)} + \frac{1}{p} \cdot \sum_{l=0}^{p-1} e_l\left(\check{x} + \Delta\delta_l(\check{x}, \check{y})_{(\text{fr2ref})}^{(n)}, \check{y} + \Delta\lambda_l(\check{x}, \check{y})_{(\text{fr2ref})}^{(n)}\right)^{(n)}. \quad (12)$$

The convergence is reached when the changes in the average error are negligible, that is, when the variance of the average error is below a certain threshold determined in an empirical procedure.

Once the SR image is obtained for time t_0 with the first “ p ” images, the process must be repeated with the next “ p ” images to obtain the next SR image using a previously established number of iterations or iterating until convergence is reached. To obtain an SR image implies the use of “ p ” low-resolution images, hence, at instant t_i , the SR image $k = \text{integer}(i/p)$ is generated. In such case, (11) must be generalized to

$$S_k(\check{x}, \check{y})^{(n)} = S_k(\check{x}, \check{y})^{(n-1)} + \frac{1}{p} \cdot \sum_{l=0}^{p-1} e_l\left(\check{x} + \Delta\delta_l(\check{x}, \check{y})_{(\text{fr2ref})}^{(n)}, \check{y} + \Delta\lambda_l(\check{x}, \check{y})_{(\text{fr2ref})}^{(n)}\right)^{(n)}. \quad (13)$$

This equation shows the SR image at instant k as a combination of “ p ” low-resolution images after “ n ” iterations.

3. MODIFICATIONS FOR THE IMPLEMENTATION ON A VIDEO ENCODER

3.1. Algorithm modifications

The modifications to the ISR algorithm previously presented are intended to adapt the algorithm in terms of basic actions that are easily implemented on a video encoder, as will be detailed in this section. First of all, instead of starting with an average image as indicated in (4) several experiments carried out have demonstrated that it is faster and easier to start with an upsampled version of the first low-resolution input image. Therefore, the final SR image will be aligned with the first image, whose motion is well known.

The straightforward way in a video encoder to determine the displacements between pictures is by using the motion estimator, which is normally used to code interpictures of type P or B in many video coding standards such as MPEG or H.26x. Furthermore, as the displacements computation is one of the most sensitive steps in the ISR algorithm, as well as in all the SR algorithms found in the literature, it has been decided to use a motion estimator with a quarter-pixel precision for this task. Consequently, the motion compensator must be also prepared to work with the same precision in order to displace a picture. The main drawback is that the ISR algorithm presented is intended to work on a pixel basis, while the motion estimator and compensator of the compressor work on a block basis. This mismatch produces quality degradation when the motion does not match the block sizes, that is, when the object is smaller than the block size or when more than one moving object exist inside the block.

Another problem found in order to map the ISR algorithm into the video encoder architecture is derived from the fact that the addition of two N -bit numbers produces an $N+1$ bit number. Every pixel inside the encoder architecture is represented as an 8-bit number. Inside the co-processors, the pixel values are processed performing several additions, and for this reason the precision of the adders has been increased. On the other hand the results must be stored in an 8-bit image memory. For video compression, this precision loss is not a significant problem, but when reusing the same architecture for general image processing, the limitation of storing the intermediate results in 8-bit memories becomes an important issue. Due to that, the following solutions have been adopted [61]:

- (i) implement as many arithmetic operations as possible inside the coprocessor, increasing the precision;
- (ii) rearrange the arithmetic operations in such a way that, when storing the intermediate results, these are bounded, as close as possible, to 8-bit numbers.

The implemented algorithm shown in Algorithm 1 includes these two modifications for $SF = 2$. All memories are 8-bit wide, except for HR_A , which must be 9-bit wide. This memory must be wider because it must store arithmetic results able to overflow 8-bits, especially in the beginning of the iterations. $LR_I[\cdot]$ 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 [62]; 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 upsampled input image; and finally, $MV_ref2fr[\cdot]$ and $MV_fr2ref[\cdot]$ are the motion-vector memories storing 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 preestablished iterations. The algorithm is split up in the following main steps [63].

```

(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

```

ALGORITHM 1: Pseudocode of the ISR algorithm implemented on the video encoder.

(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 upsampled 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 remaining motion vectors are computed between the other low-resolution input frames and the low-resolution version of the super-resolved image, named LR_B (the reference). Instead of computing again the inverse motion, that is, 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 saved due to the mentioned 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.

(6) The accumulative image HR_A is set to zero prior to the summation of the average shifted errors. These average errors 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 a 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 preestablished iterations is reached. Figure 5 shows the ISR algorithm data flow, using the memories and the resources available in the hybrid video encoder platform. The previous step numbers have been introduced between parentheses as labels at the beginning of the appropriate lines for clearness. The memory **HR_A** is in boldface to remark the different bit width when compared to the other image memories.

As the motion estimation is the most expensive operation in terms of time and power consumption, it has been assumed that the motion between the reference and the frame

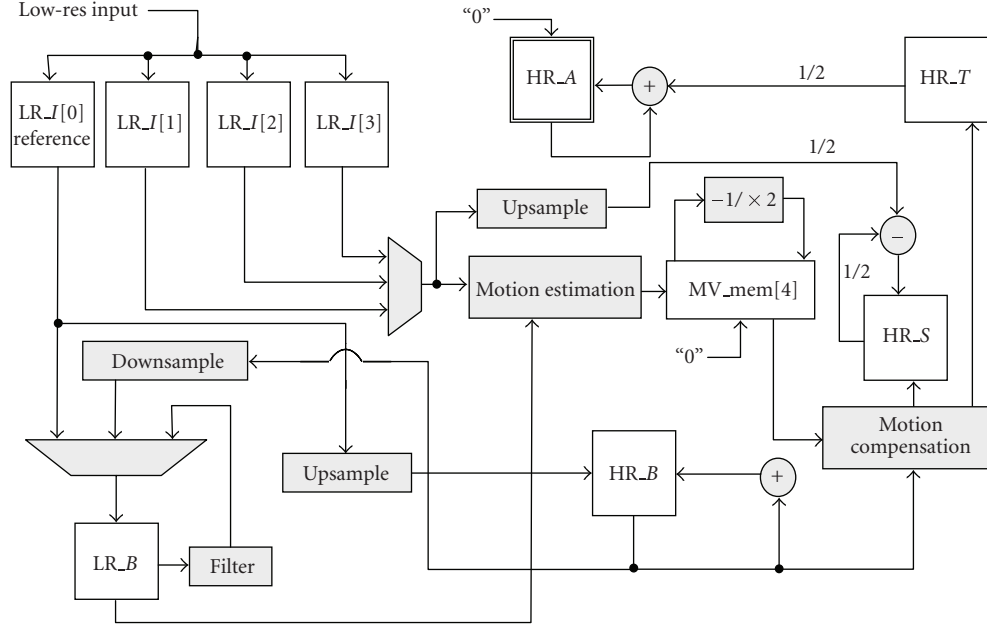


FIGURE 5: ISR algorithm data flow.

TABLE 1: Memory requirements of the ISR algorithm as a function of the number of the input image macroblocks.

Label	ISR algorithm memory		
	Luminance (bits)	Chrominance (bits)	Total (bits)
HR_A	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 16 \cdot 16 \cdot 9)$	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 9)$	$13,824 \cdot MB_x \cdot MB_y$
HR_B	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$12,288 \cdot MB_x \cdot MB_y$
HR_S	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$12,288 \cdot MB_x \cdot MB_y$
3-stripe HR	$(2 \cdot 3 \cdot 2 \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(2 \cdot 3 \cdot 2 \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$36,864 \cdot MB_y$
LR_B	$(MB_x \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(MB_x \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$3,072 \cdot MB_x \cdot MB_y$
LR_I[0]	$(MB_x \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(MB_x \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$3,072 \cdot MB_x \cdot MB_y$
LR_I[1]	$(MB_x \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(MB_x \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$3,072 \cdot MB_x \cdot MB_y$
LR_I[2]	$(MB_x \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(MB_x \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$3,072 \cdot MB_x \cdot MB_y$
LR_I[3]	$(MB_x \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(MB_x \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$3,072 \cdot MB_x \cdot MB_y$
MV_mem[0]	$(MB_x \cdot MB_y \cdot 8)$	0	$8 \cdot MB_x \cdot MB_y$
MV_mem[1]	$(MB_x \cdot MB_y \cdot 8)$	0	$8 \cdot MB_x \cdot MB_y$
MV_mem[2]	$(MB_x \cdot MB_y \cdot 8)$	0	$8 \cdot MB_x \cdot MB_y$
MV_mem[3]	$(MB_x \cdot MB_y \cdot 8)$	0	$8 \cdot MB_x \cdot MB_y$
Total (bits)	$MB_y \cdot (35,872 \cdot MB_x + 24,576)$	$MB_y \cdot (17,920 \cdot MB_x + 12288)$	$MB_y \cdot (53,792 MB_x + 36,864)$

is the inverse of the motion between the frame and the reference, increasing in this way the real motion consistency. It is interesting to highlight that the presence of aliasing in the low-resolution input images largely decreases the accuracy of the motion vectors. Due to this reason a spatial lowpass filter of order three has been applied to the input images prior to performing the motion estimation.

3.2. Implementation issues

Table 1 summarizes the memory requirements that the implementation of the ISR algorithm demands for $nr.frames = 4$ and $SF = 2$ as a function of the input MBs. The number

of MBs in columns has been labeled as MB_x , and the number of MBs in rows has been labeled as MB_y . For instance, the HR_A memory has a number of macroblocks equal to $(2 \cdot MB_x) \times (2 \cdot MB_y)$. Because it is a high-resolution image, its size is doubled in both directions. As every macroblock has 16×16 luminance pixels and 8×8 chrominance pixels and, furthermore, there exist two chrominance components, the blue and the red ones, the overall pixel number is $(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 16 \cdot 16)$ for the luminance and $(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 8 \cdot 8 \cdot 2)$ for the chrominance components. Nevertheless, it must be taken into account that the HR_A memory is 9-bit wide, and for this reason, it is necessary to multiply each pixel by 9-bit in order to obtain

TABLE 2: Memory requirements of the ISR algorithm for different sizes of the input image.

Size	MB _x	MB _y	Memory (KB)
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	10 563.1875

the total number of bits. The remaining memories will be multiplied by 8-bit per pixel. These requirements include four input memories as the number of frames to be combined has been settled upon as four. Also, a buffer containing three rows of macroblocks for reading the input images, as part of the encoder memory requirements, has been included [64]. These memory requirements also take into account the chrominance and the additional bit of HR_A. The total memory requirements, as a function of the number of MBs, is $MB_y \cdot (6724 \cdot MB_x + 4608)$, expressed in bytes. Table 2 summarizes the memory requirements for the ISR algorithm with the most common input sizes. It must be mentioned that the size of the output images will be doubled in every direction, thus having a super-resolved image four times larger.

To perform the upsample and downsample operations, it is necessary to include upsampling and downsampling blocks in hardware, being in charge of performing these operations on an MB basis. A hardware implementation is desirable as the upsample/downsample processes are very intensive computational tasks in the sense that they are performed on the entire 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. Upsampling is performed by nearest-neighbor replication from an (8×8) -pixel block to a (16×16) -pixel MB. Downsampling is achieved by picking one pixel from every set of four neighbor 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 coprocessors. These coprocessors 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 which was shown in Figure 2.

4. EXPERIMENTAL SETUP

A large set of synthetic sequences have been generated with the objective of assessing the algorithm itself, independently of the image characteristics, and to enable the measurement 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. Finally, in order to avoid the border effects when shifting the frame, large image formats together with a later removal of the borders have been used in order to compute reliable quality metrics. Figure 6 depicts the experimental setup to generate the test sequences [65].

The displacements introduced in the VGA images in pixel units are reflected in the low-resolution input pictures divided by four, that is, 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 40 input frames from 40 random motion vectors have been generated. These synthetic sequences are used as the input for the SR process. The ISR algorithm initially performs 80 iterations over every four-input-frame set. The result is a ten-frame sequence, where each SR output frame is obtained as the combination of four low-resolution input frames.

Figure 7(a) shows the reference picture Kantoor together with the subsampled sequences that constitute the input low-resolution sequence (b), and the nearest-neighbor (c) and bilinear interpolation images (d) obtained from the first low-resolution frame (frame with zero motion vector). Figure 8(a) shows the reference picture Krant together with the subsampled sequences that constitute the input low-resolution sequence (b) and the nearest-neighbor (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-neighbor replication interpolations in terms of PSNR. In this work, the quality of the SR algorithms are compared with the bilinear and nearest-neighbor 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.

In order to demonstrate the quality increase in the SR image when combining several low-resolution images, the experiment denoted in Figure 9 has been designed. In this experiment, referred to 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 are applied to

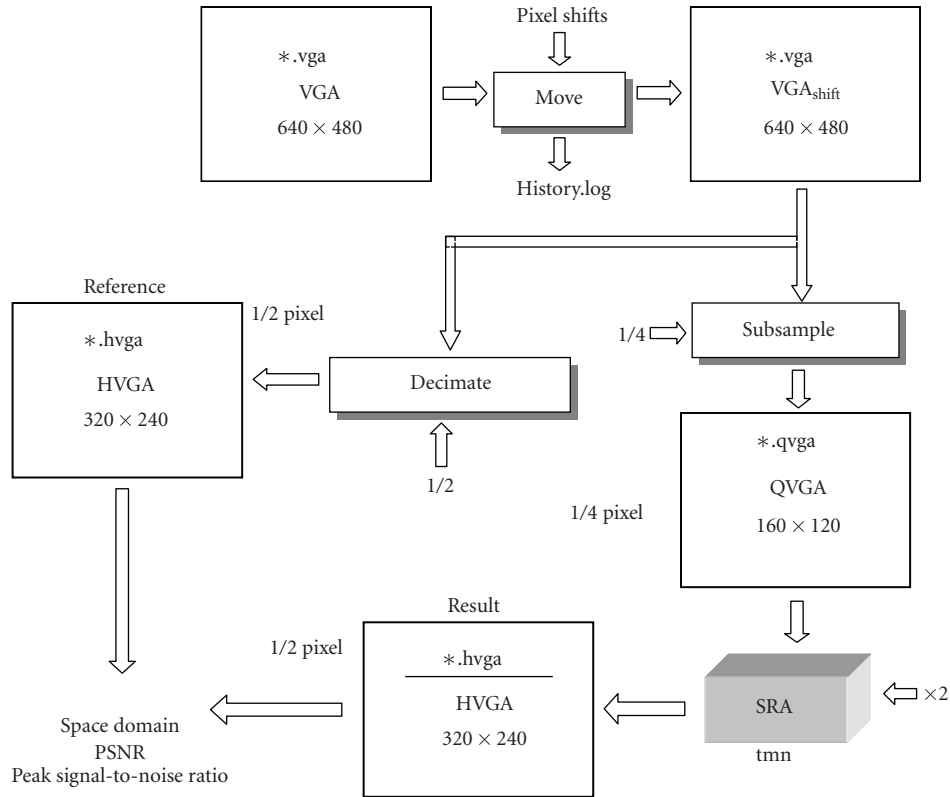


FIGURE 6: Experimental setup for the test sequence generation.

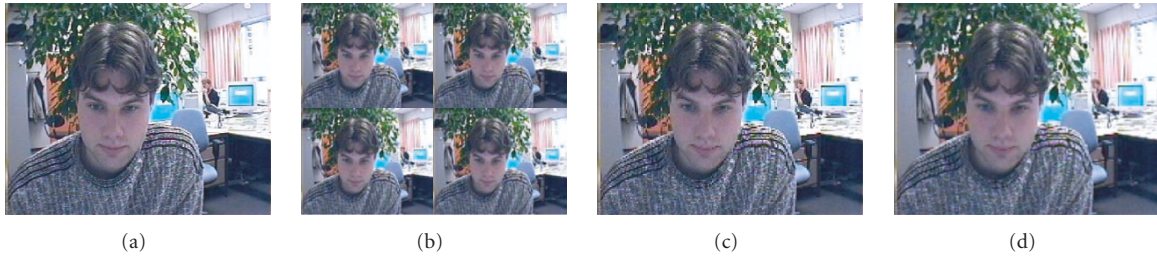


FIGURE 7: (a) Reference Kantoor picture, (b) the low-resolution input sequence derived from it, (c) the nearest-neighbor, and (d) bilinear interpolations.

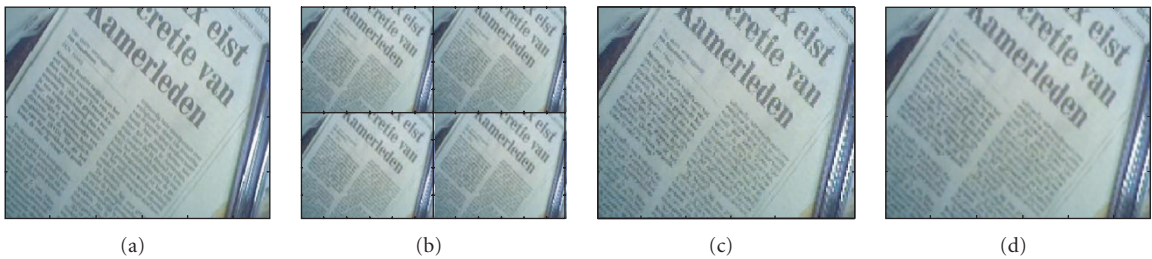


FIGURE 8: (a) Reference Krant picture, (b) the low-resolution input sequence derived from it, (c) the nearest-neighbor, and (d) bilinear interpolations.

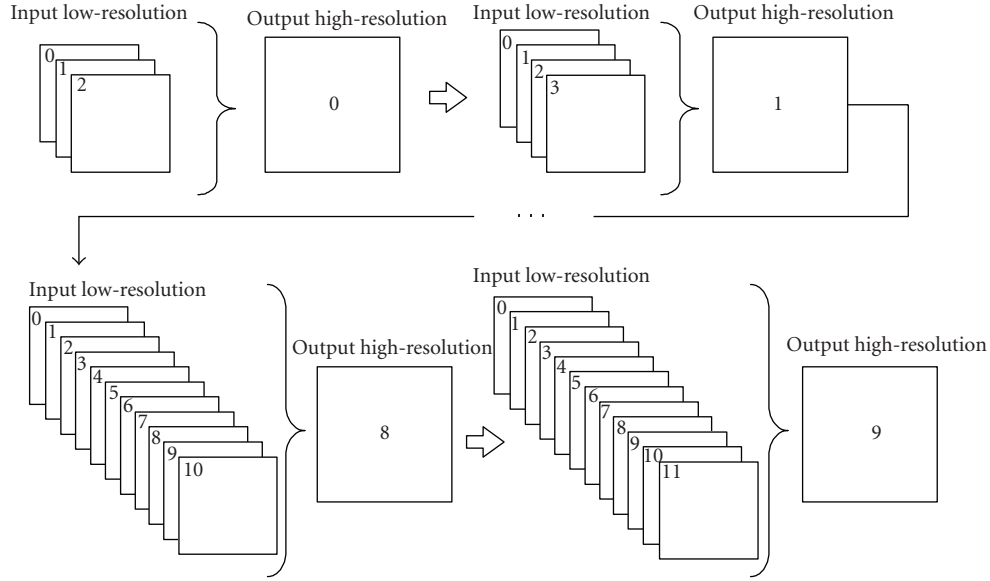


FIGURE 9: Incremental test for assessing the SR algorithms.

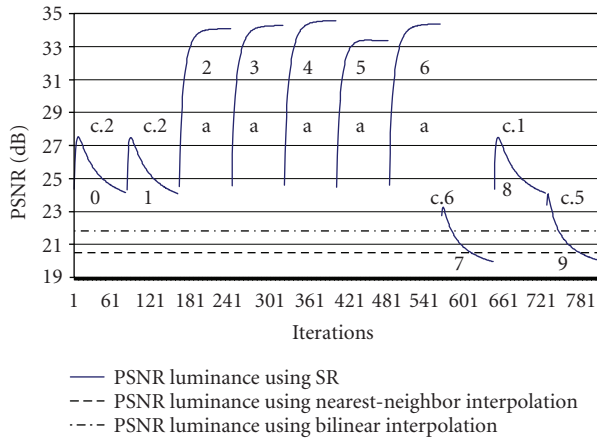


FIGURE 10: Luminance PSNR for 80 iterations of the Kantoor sequence combining 4 input frames. The output sequence has 10 frames.

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 $3 + 4 + 5 + \dots + 12 = 75$ low-resolution frames have been used as inputs to the SR algorithms in order to generate 10 output frames.

5. ISR ALGORITHM RESULTS

In this section the test procedures exposed in the previous section have been applied to the ISR algorithm. Figure 10

shows the luminance PSNR evolution of each frame for the Kantoor sequence during the iterative process. From this chart, it is noticeable that for certain frames (frames 2 to 6) 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 for 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 four 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. Higher or lower PSNR values will be obtained depending on the missing data, decreasing below the interpolation level (frames 7 and 9) when the available data is clearly insufficient. In such cases, the ISR algorithm tries to match the available information to the missing information within the SR frame, producing undesirable artefacts when there is a lack of information. These artefacts cause the motion-vector field between the low-resolution version of the SR image and the low-resolution inputs to get worse with the number of iterations due to the error feedback.

In Figure 11 a classification of the SR frames depending on the available input samples is proposed. The best case is obtained for frames of “a-type” where all the samples are present and the worst cases are for “d-type” frames, where four equivalent motion vectors are generated picking up the same (or equivalent) sample positions four times. According to this classification and inspecting the motion vectors generated for each low-resolution input set, frames 2 to 6 are classified as “a,” frames 0 and 1 as “c.2,” frames 7 and 9 as “c.6” and “c.5,” respectively, and frame 8 as “c.1,” as it is shown in Figure 10.

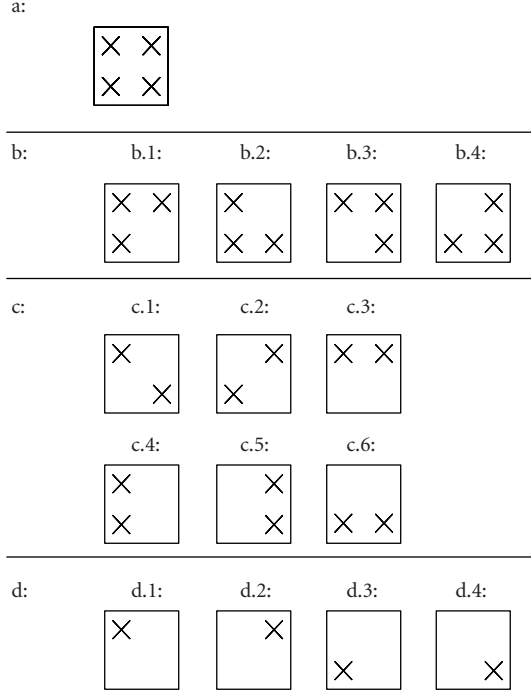


FIGURE 11: SR frame classification depending on the available samples.

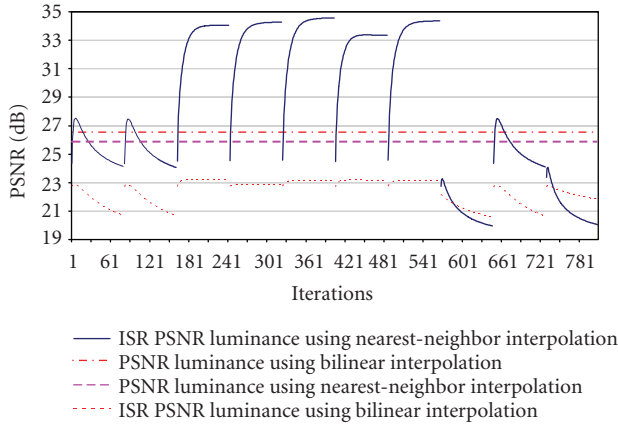


FIGURE 12: Luminance PSNR for 80 iterations of the Kantoor sequence using nearest-neighbor and bilinear interpolations for the upsampling process.

As discussed in Section 2, Figure 12 shows the PSNR of the Kantoor sequence luminance when the upsampling has been implemented using a nearest-neighbor interpolator or a bilinear interpolator. In the second case, the quality of the sequence is lower due to the aliasing removal of the bilinear interpolator. Therefore, for this application a nearest-neighbor interpolator that keeps substantial amounts of aliasing across the SR process is required.

In Figure 13 can be seen the average error of the motion-vector field, computed as the absolute difference between the

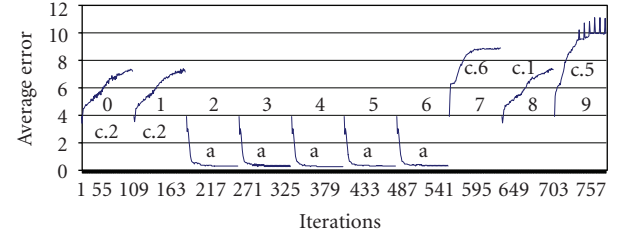


FIGURE 13: Average error of the motion-vector field for 80 iterations of the Kantoor sequence.

real motion vectors and the motion-vectors obtained by the motion estimator. The error is averaged between the horizontal and vertical coordinates and among all the frames. Equations (14) summarize the motion-vector error as it has been computed in this paper, where “ p ” is the number of frames to be combined; “MB $_x$ ” and “MB $_y$ ” are the number of MBs in the horizontal and vertical directions, respectively, and depend on the size of MB upon which the motion estimator is based; “ $mv \cdot x(l)[mv_x, mv_y]$ ” is the motion vector computed for the MB located at (mb_x, mb_y) of frame “ l ” in the horizontal coordinate and “ $mv \cdot y(l)[mv_x, mv_y]$ ” is the counterpart in the vertical coordinate. After the errors in the horizontal ($error_x$) and vertical ($error_y$) directions have been computed, they are averaged in a single number ($error$). It is clear how the error decreases with the iterations for images of type “a.” For the “c.2” and “c.1” image types, the motion error drops in the beginning but it rises after a few iterations. For “c.6” and “c.5” image types, the motion error increases from the first iteration:

$$error_x = \frac{1}{p} \cdot \sum_{l=0}^{p-1} \frac{1}{MB_x \cdot MB_y} \cdot \sum_{mb_y=0}^{MB_y-1} \sum_{mb_x=0}^{MB_x-1} |mv \cdot x(l)[mb_x, mb_y] - mv_real \cdot x(l)[mb_x, mb_y]|,$$

$$error_y = \frac{1}{p} \cdot \sum_{l=0}^{p-1} \frac{1}{MB_x \cdot MB_y} \cdot \sum_{mb_y=0}^{MB_y-1} \sum_{mb_x=0}^{MB_x-1} |mv \cdot y(l)[mb_x, mb_y] - mv_real \cdot y(l)[mb_x, mb_y]|,$$

$$error = \frac{error_x + error_y}{2}.$$
(14)

Taking into account that all the necessary data may not be available (“b,” “c,” and “d” cases), it is better to abort the iterative process after a few initial iterations in order to avoid the quality drops that appear in Figure 10. After examining several sequences, a number of 8 iterations have been selected as

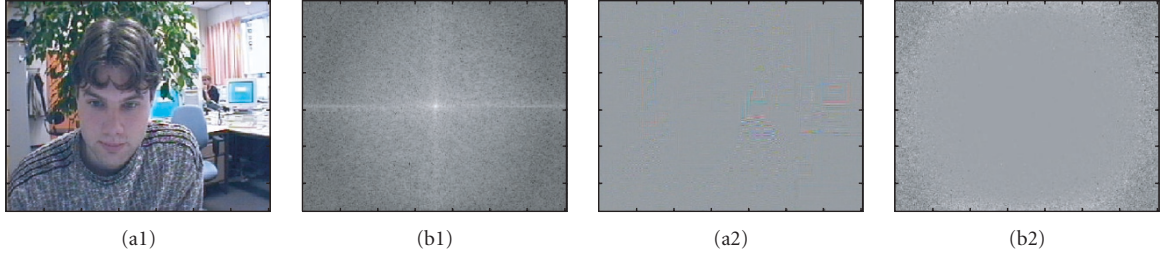


FIGURE 14: Kantoor frame number 4 of “a-type” in (a1) the spatial domain and in (b1) the frequency domain in magnitude, with their associated errors ((a2) and (b2), resp.).

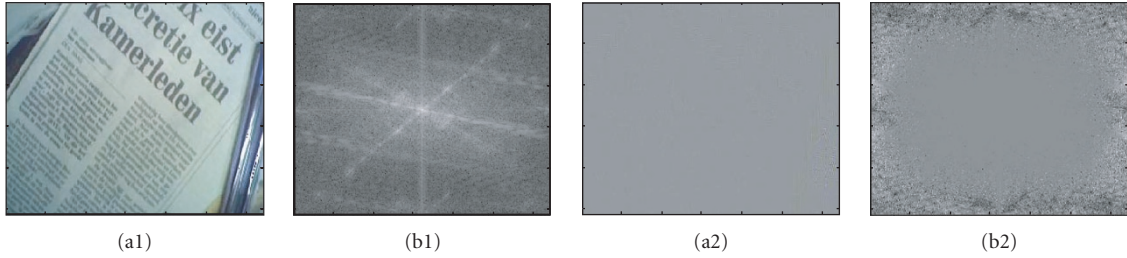


FIGURE 15: Krant frame number 4 of “a-type” in (a1) the spatial domain and in (b1) the frequency domain in magnitude, with their associated errors ((a2) and (b2), resp.).

a reasonable tradeoff between quality and computation effort for the average cases.

If all input data are available, a maximum PSNR of 34.56 dB for frame number 4 (“a-type”) is reached for the Kantoor sequence and 37.59 dB for the Krant sequence. Figure 14 shows the spatial- and frequency-domain images for Kantoor frame number 4 after 80 iterations together with the associated errors with respect to the reference, and Figure 15 shows the same for frame 4 of the Krant sequence. It is clearly appreciated that the low frequencies, located in the central part of image, exhibit lower errors than the high frequencies. The reconstruction process tries to recover as much high-resolution frequencies as possible, but the low-frequency information is easier to recover, mainly because almost all of such information is present prior to the SR reconstruction process.

Figure 16 shows the PSNR for the Kantoor sequence but by limiting the number of iterations to eight. It is easy to see that finally all the frames but one exhibit PSNR above the interpolation levels. Only frame 7 of type “c.6” is below such levels, whereas frame 9 of type “c.5” is just at the interpolation level.

Figures 17 and 18 show some enlarged details of the Kantoor and Krant sequences, respectively, after 80 iterations combining 4 low-resolution frames per SR output frame. In both cases, (a) is the nearest-neighbor interpolation, (b) is the SR image, and (c) is the bilinear interpolation, and also in both cases an important recovery of the high-frequency details is noticeable, as the edge recovery reveals.

Finally, the incremental test described in Figure 9 was applied to the ISR algorithm using the Krant sequence. Initially,

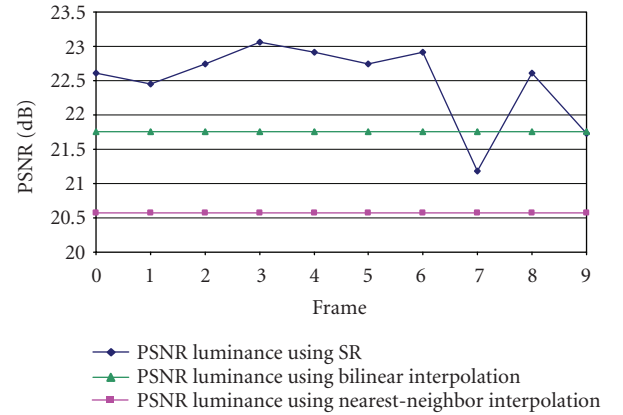


FIGURE 16: PSNR of the luminance for frame 4 of the Kantoor sequence after 8 iterations.

an amount of 80 iterations were settled for every output frame, obtaining the luminance PSNR shown in Figure 19. 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 9 low-resolution frames, the ISR algorithm is able to deliver a type “a” image, manifested in a quality increase as the number of iterations increases. Once again, if the iterations are limited to 8 in order to preclude from an excessive quality dropping, the PSNR



FIGURE 17: Enlarged details of the Kantoor sequence for frame number 4 after 80 iterations, combining 4 low-resolution frames per SR frame. Image (a) is the nearest-neighbor interpolation, image (b) is the SR image combining 4 low-resolution input frames, and image (c) is the bilinear interpolation of the input image.

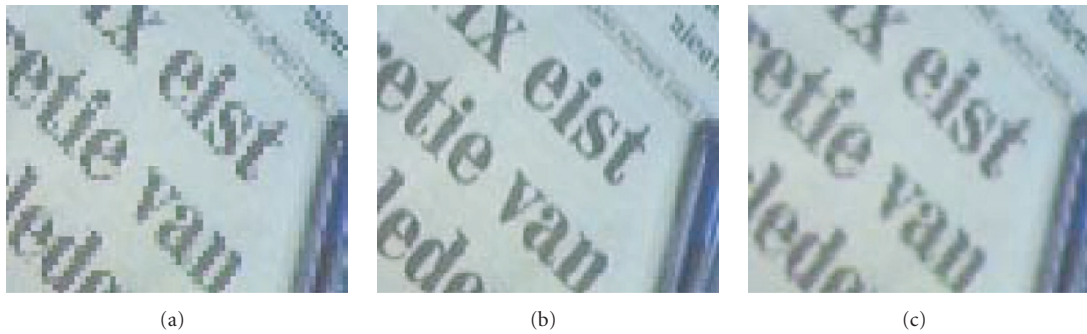


FIGURE 18: Enlarged details of the Krant sequence for frame number 4 after 80 iterations, combining 4 low-resolution frames per SR frame. Image (a) is the nearest-neighbor interpolation, image (b) is the SR image combining 4 low-resolution input frames, and image (c) is the bilinear interpolation of the input image.

represented in Figure 20 is obtained. In the two cases (8 and 80 iterations), both the final and the maximum PSNR values are shown.

In Figure 21 the SR frame number 9 is shown, as a result of the combination of 12 low-resolution frames. Image (a1) is the SR frame in the spatial domain, and (a2) is the error image when compared with the original one. Major errors are located in the edge zones, that is, in the high frequencies. The bidimensional Fourier transform in magnitude is shown in (b1) and the error image is in (b2). As expected, the central zone of the magnitude, corresponding to the lower spatial frequencies, exhibits the lower errors. The phase of the image is shown in (c1) together with the associated error in the frequency domain (c2). Once again, the error is minimal in the lower-frequency zones. Three enlarged details of the pencils of the Krant frame are shown in Figure 22: (a) the nearest-neighbor interpolation, (b) the SR image, and (c) the bilinear interpolation of the input low-resolution sequence.

6. NONITERATIVE SUPER-RESOLUTION

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, that is, a noniterative algorithm suitable for real-time applications. The underlying idea is based on the following considerations:

- (i) every new image adds fresh information that must be combined into a new high-resolution grid;
- (ii) 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 these considerations, a novel noniterative 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 into a high-resolution grid, leaving the unmatched pixels to a zero value. This process will be named as “upsample holes.” As the size increases by a factor of two, both in the horizontal and vertical directions, the location and relationship among the pixels of high and low resolution are as shown in Figure 23.

(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

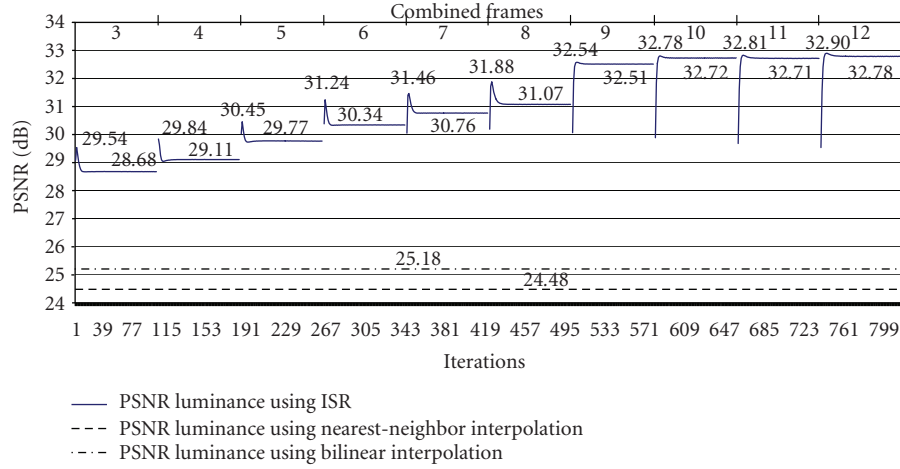


FIGURE 19: PSNR of the Krant sequence with 10 incremental output frames using the ISR algorithm with 80 iterations.

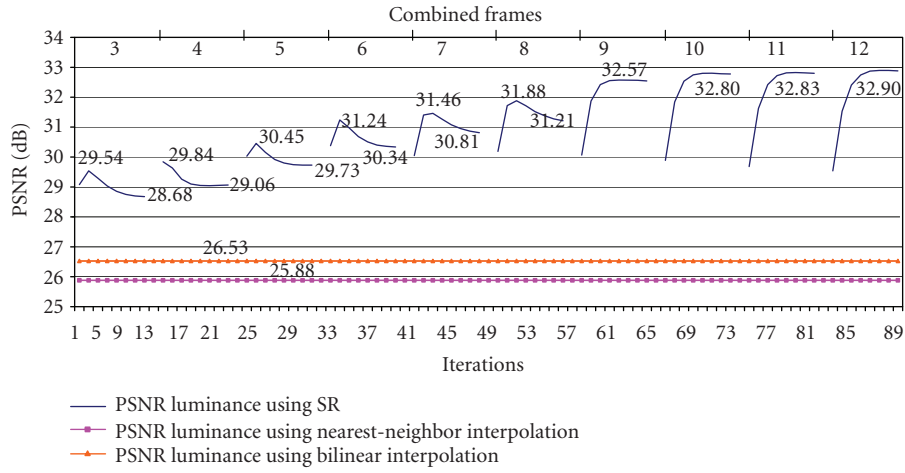


FIGURE 20: PSNR of the Krant sequence with 10 incremental output frames using the ISR algorithm and 8 iterations.

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 $(1/2)$ -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 will be necessary. These contributions are expressed over the high-resolution grid. The contributions of the image in Figure 23 are shown in the left part of Figure 24(b), pointing out that the original pixels have maximum contribution (four) while the rest have zero value.

(3) The relative displacements between the next input image and the first image, considered as the reference one, are estimated. These displacements are stored in a memory, as they will be used later on.

(4) Steps (1) and (2) are applied to the new input image, that is, 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

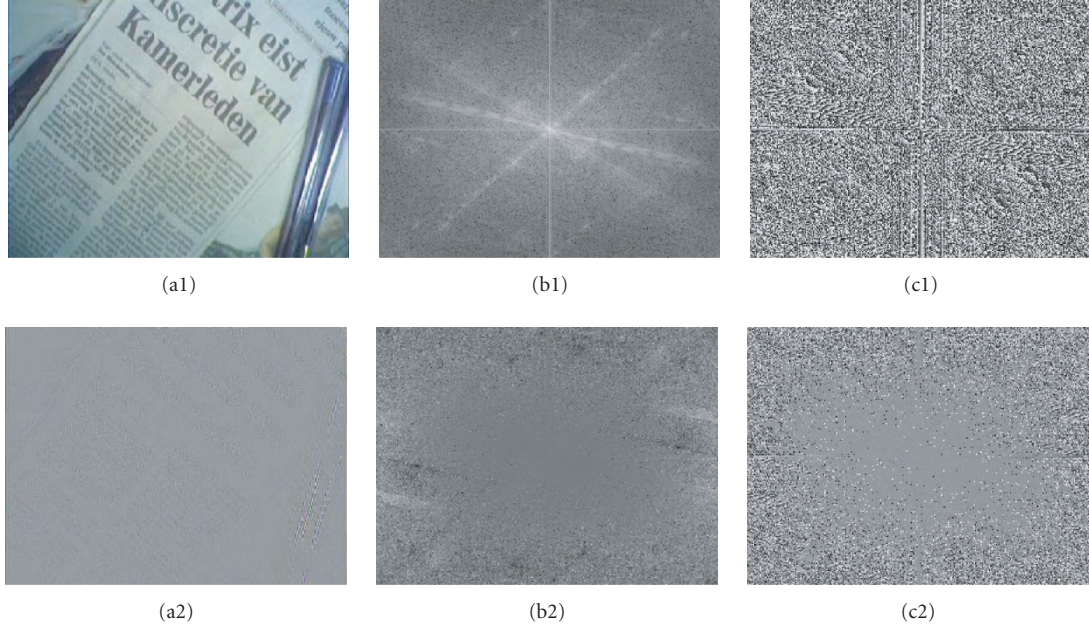


FIGURE 21: Super-resolved image after combining 12 low-resolution frames in (a1) the spatial domain, in (b1) the magnitude frequency domain, and in (c1) the phase frequency domain, together with their associated errors ((a2), (b2), and (c2)) for the Krant sequence.

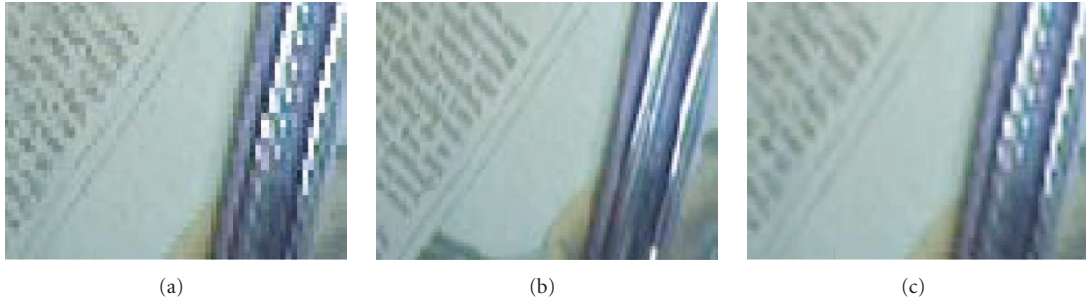


FIGURE 22: Enlarged details of the nearest-neighbor interpolation of (a) the input image, (b) the SR image after combining 12 low-resolution input frames, and (c) the bilinear interpolation of the input image for the Krant sequence.

high-resolution image is adjusted depending on the contributions, as it is indicated in (15), where N is 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.

$$SR(i, j) = N \cdot \frac{\sum_{fr=1}^N \text{Motion_Compensate}(\text{Upsample_Holes}(LR_I[fr]))}{\sum_{fr=1}^N \text{Motion_Compensate}(\text{contributions}[fr])}. \quad (15)$$

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, fitting in 12 bits.

(9) After the adjustment, it is possible that some pixel positions remain empty, because certain positions from the in-

put image set did not add new information. This case will be denoted by a zero, 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. However, the presence of a zero in the image does not necessarily implies that such value must be interpolated, because zero is a possible and valid value in an image. Due to this reason, a pixel will be interpolated only if its final contribution is zero.

(10) 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.

The NISR algorithm is presented in Algorithm 2, where the previous step numbers have been introduced between parentheses as labels at the beginning of the appropriate lines for clearness. In Figure 25 the NISR algorithm data flow is shown, using the memories and the resources of the hybrid

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

Label	NISR algorithm memory		
	Luminance (bits)	Chrominance (bits)	Total (bits)
HR_A	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 16 \cdot 16 \cdot 12)$	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 12)$	$18,432 \cdot MB_x \cdot MB_y$
HR_B	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$12,288 \cdot MB_x \cdot MB_y$
HR_S	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$12,288 \cdot MB_x \cdot MB_y$
HR_S2	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$12,288 \cdot MB_x \cdot MB_y$
HR_Cont	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(2 \cdot MB_x \cdot 2 \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$12,288 \cdot MB_x \cdot MB_y$
3-stripe HR	$(2 \cdot 3 \cdot 2 \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(2 \cdot 3 \cdot 2 \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$36,864 \cdot MB_y$
LR_I[0]	$(MB_x \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(MB_x \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$3,072 \cdot MB_x \cdot MB_y$
LR_I[1]	$(MB_x \cdot MB_y \cdot 16 \cdot 16 \cdot 8)$	$(MB_x \cdot MB_y \cdot 8 \cdot 8 \cdot 2 \cdot 8)$	$3,072 \cdot MB_x \cdot MB_y$
MV_mem[0]	$(MB_x \cdot MB_y \cdot 8)$	0	$8 \cdot MB_x \cdot MB_y$
Total (bits)	$MB_y \cdot (49,160 \cdot MB_x + 24,576)$	$MB_y \cdot (24,576 \cdot MB_x + 12288)$	$MB_y \cdot (73,736 MB_x + 36,864)$

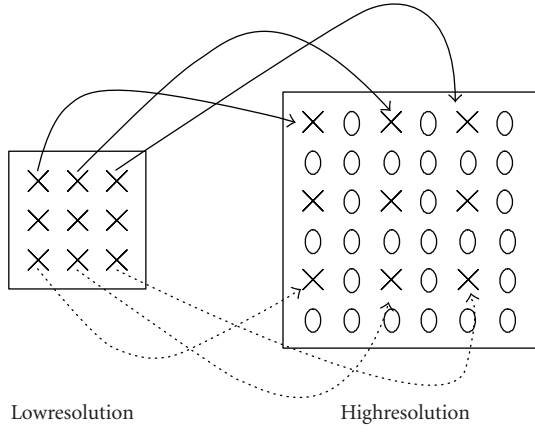


FIGURE 23: Mapping of the low-resolution pixels in the high-resolution grid, leaving holes for the missing pixels.

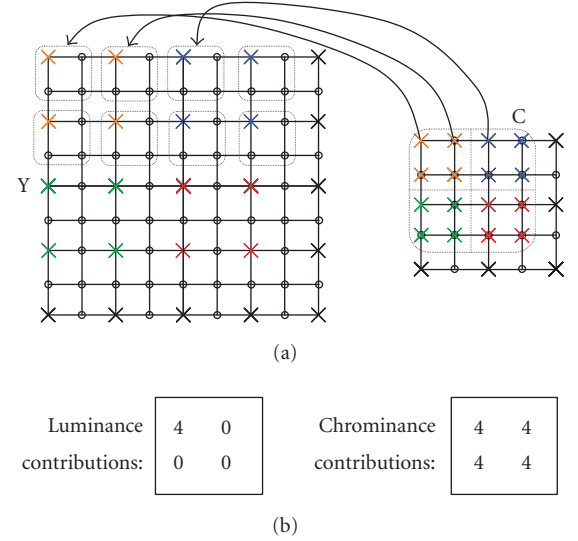


TABLE 4: Memory requirements of the NISR algorithm for different sizes of the input image.

Size	MB_x	MB_y	Memory (KB)
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

video encoder. Memory **HR_A** is in boldface to remark the different bit width when compared to the other memories. The block-diagram has been divided into two different parts: 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 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 relationships among them, the image data flow has been drawn in solid lines, the

FIGURE 24: (a) Mapping of the chrominance C to the high-resolution grid by means of replicating the pixels and its relationship with the luminance Y. (b) Initial contributions of the luminance and chrominance images.

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 an asterisk to point out their different behaviors when they are executing in the SR mode.

6.1. Implementation issues

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


```

(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

```

ALGORITHM 2: Pseudocode of the NISR algorithm implemented on the video encoder.

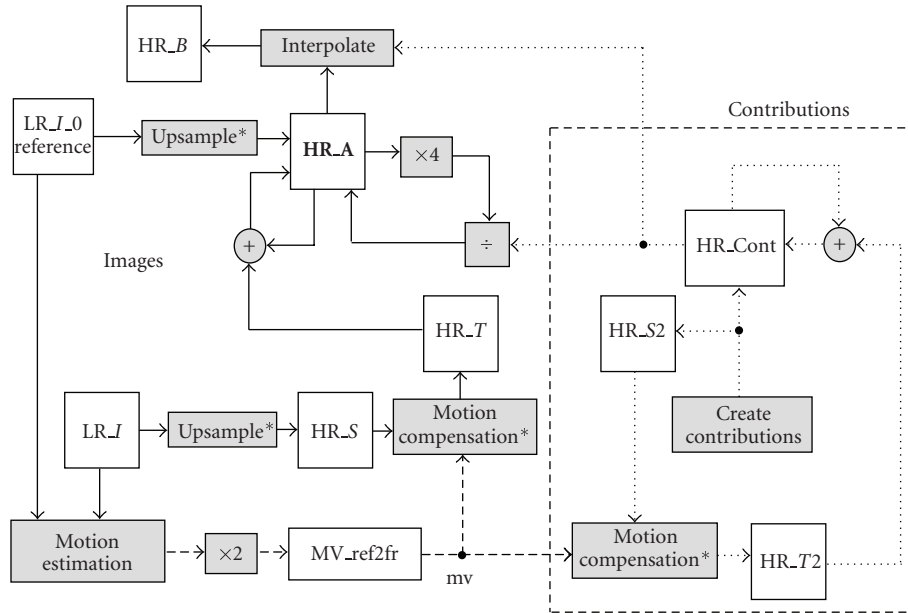


FIGURE 25: NISR algorithm data flow.

Table 3 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. Table 4 summarizes the memory requirements of the NISR algorithm for the most common input sizes.

6.1.1. 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

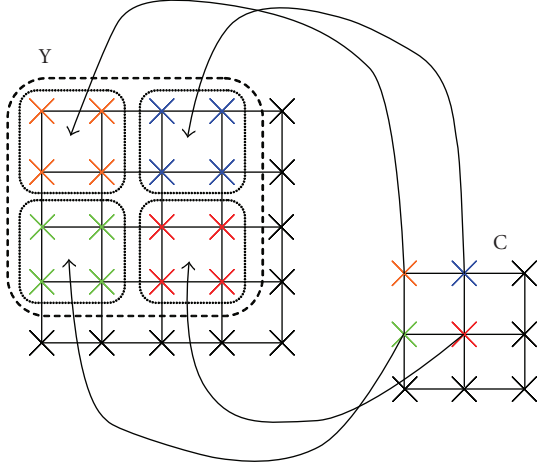


FIGURE 26: Relationship between the chrominance and the luminance for sampling format YCbCr 4 : 2 : 0.

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 nonexistent 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 has an opportunity to fill the holes with valid values coming from other images.

6.1.2. Adjustments in the chrominances

Due to the different sampling schemes used for the luminance and the chrominance components, it is necessary to perform some modifications on the proposed NISR algorithm to obtain SR improvements also in the chrominance images.

First of all, for the chrominance pixels, the way to perform the upsampling process cannot be the same as the one used with the luminance. This fact is reflected in Figure 26, where it can be seen how each chrominance pixel affects four luminance pixels. Therefore, when the chrominance high-resolution grid is generated, every pixel must be replicated four times in order to keep the chromatic coherence as shown in Figure 24(a).

For the same reason, the initial chrominance contributions cannot be the same as the luminance ones. As there is no zero-padding, all the chrominance pixels must initially have the same contribution weights, as is shown in Figure 24(b) where the initial contributions for the luminance and the chrominances are presented.

7. NONITERATIVE RESULTS

In order to enable a suitable comparison between the ISR and the NISR algorithms, firstly the results of combining four low-resolution frames per output SR frame will be shown for both sequences. In Figure 27 the PSNR resulting from applying the NISR algorithm to the Kantoor sequence is shown

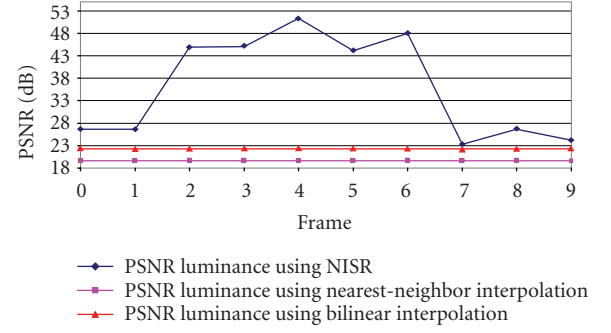


FIGURE 27: PSNR of the NISR when combining four low-resolution images per frame for the Kantoor sequence.

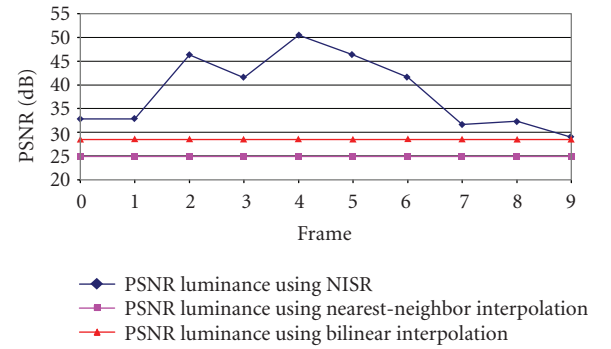


FIGURE 28: PSNR of the NISR when combining four low-resolution images per frame for the Krant sequence.

and Figure 28 shows the SR results for the Krant sequence. As far as the introduced shifts have been the same for both sequences, they exhibit a very similar behavior. The first one has an average PSNR of 36.07 dB, while the second one has an average PSNR of 38.47 dB. The quality differences are due to the inherent nature of these two images: in the same conditions, a lower entropy image will always have higher PSNR. In Figure 29 is shown the resulting frame number 4 in the case of the Kantoor sequence, as it exhibits the higher quality of the ten-frame sequence. As it was commented in Section 5, the error is again lower in the low-frequency region. Compared with the ISR algorithm, the error image for the NISR algorithm is more uniform both in the spatial and in the frequency domain for the same number of combined frames. A detailed observation of the same frame for the Krant sequence offers the same results, as it can be seen in Figure 30. At the same time, some details of these last images containing edges are shown both for Kantoor and Krant sequences in Figures 31 and 32, respectively, increasing in both cases the perceptual quality.

The qualities resulting from applying the incremental test to the Krant sequence are shown in Figure 33. 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.

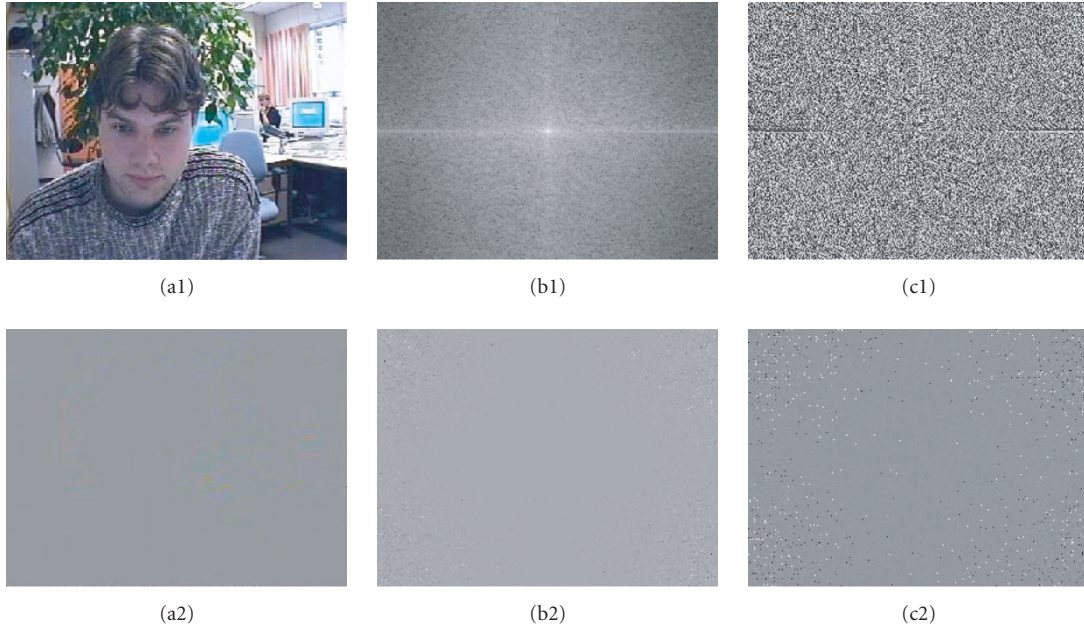


FIGURE 29: PSNR of the NISR when combining four low-resolution frames for the Kantoor sequence.

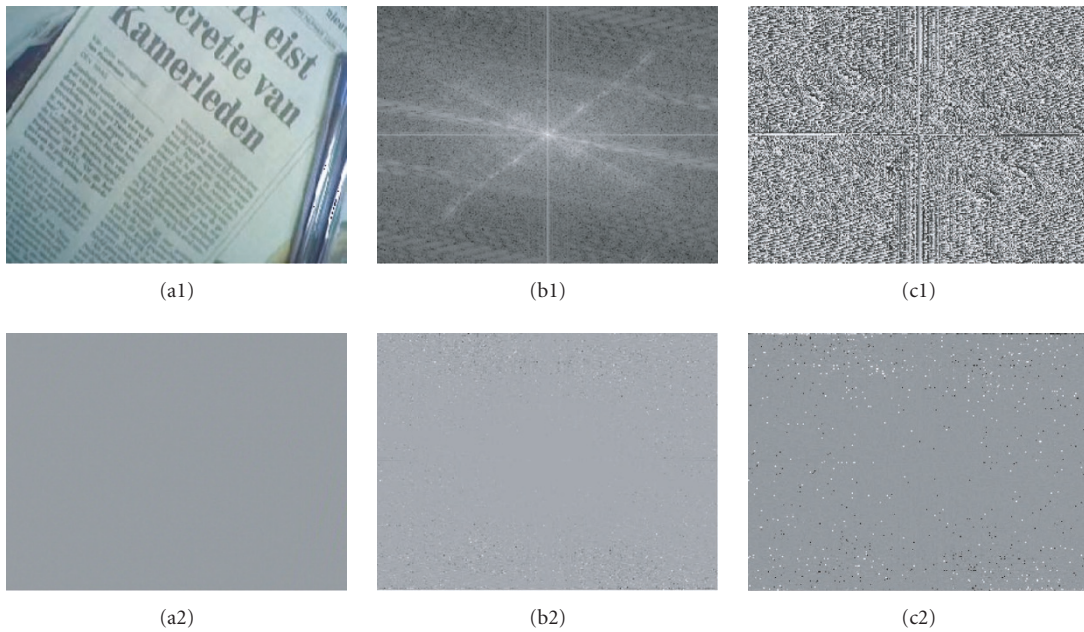


FIGURE 30: PSNR of the NISR when combining four low-resolution frames for the Krant sequence.

The perceptual quality exhibits few variations after combining 6 input frames, that is, when 34.57 dB is reached. In addition, it can be seen that the largest 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.

The results from the combination of 12 low-resolution frames are shown in Figure 34, where it can be noticed that the error image in the spatial domain (a2) is highly uniform, indicating a low error with respect to the reference image. The magnitude of the bi-dimensional Fourier transform shows an almost complete removal of the aliasing (less overlap of the high-energy bands), revealing a minimal image error in the high-power spectral bands (b2). The minimal

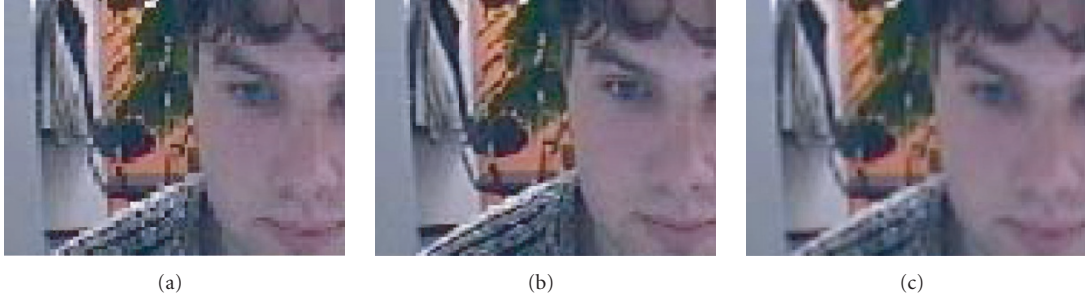


FIGURE 31: Enlarged detail of the Kantoor sequence with 10 output frames when four low-resolution frames are combined.

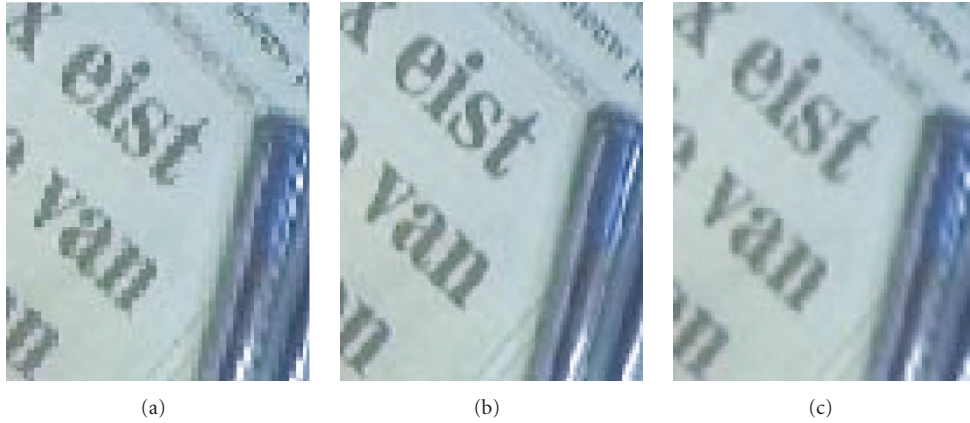


FIGURE 32: Enlarged detail of the Krant sequence with 10 output frames when four low-resolution frames are combined.

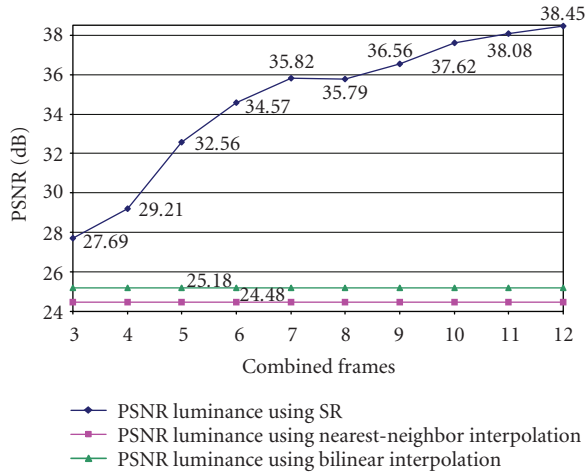


FIGURE 33: Incremental test of the Krant sequence with 10 output frames.

phase spectral error (c2) follows the same trend. When comparing these results with the ones shown in Figure 35 (which have been obtained combining only 3 low-resolution frames) higher amounts of aliasing and consequently, higher error with respect with the original image, are perceived.

In Figure 36 an enlarged detail of the frame shown in Figure 34 is shown, obtained as the gathering of 12 frames by the NISR algorithm. In (a) the nearest-neighbor interpolation is shown, in (b) the SR image and in (c) the bilinear interpolation. The quality improvement of the image, is clearly manifested, especially for those letters on the top right side of the super-resolved image.

8. ALGORITHMS COMPARISON

The two developed SR algorithms have been compared through three main features: the simulation time (measured in milliseconds), the obtained quality (measured in PSNR dB), and the memory requirements (measured in MB).

Figure 37 shows the simulation time of both algorithms applied to two of the test video input sequences: Kantoor and Krant. In all the cases 10 output frames have been generated. The results of the ISR algorithm are obtained from the iteration of 40 low-resolution frames, while the results of the NISR algorithm come from the incremental combination of three to twelve low-resolution frames, that is, a total combination of 75 input frames. Even though the NISR algorithm has processed a higher number of input frames, the total amount of time inverted in the execution of the NISR algorithm is five times lower than that of the ISR algorithm, placing the former in a best situation for a real-time execution scenario.

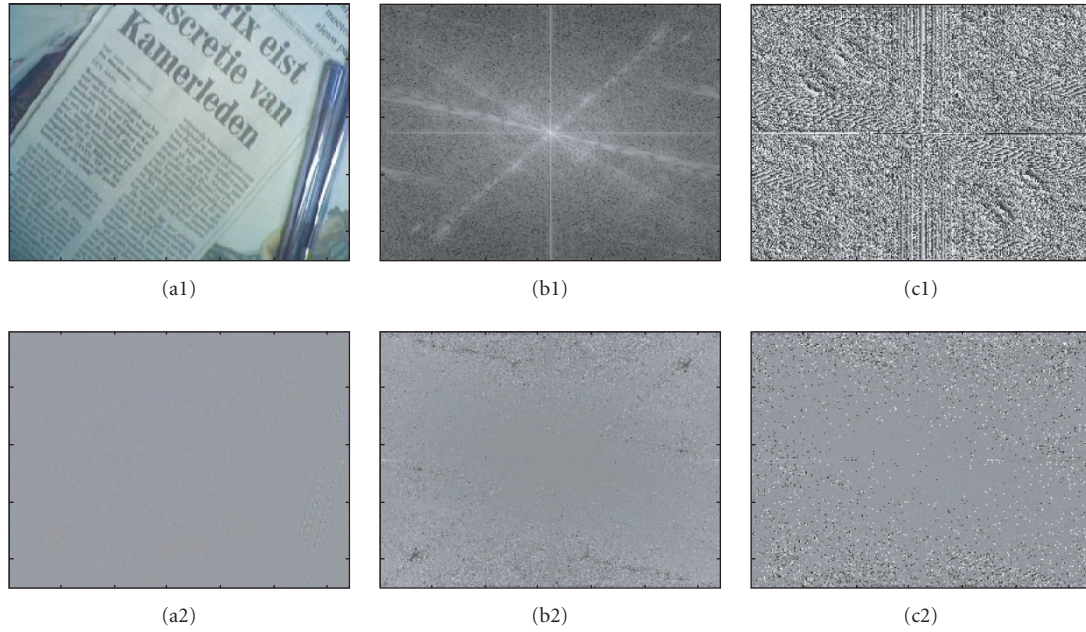


FIGURE 34: Super-resolved frame after combining 12 low-resolution frames in (a1) the spatial domain, in (b1) the frequency domain in magnitude, and in (c1) phase, together with their associated errors ((a2), (b2), and (c2)).

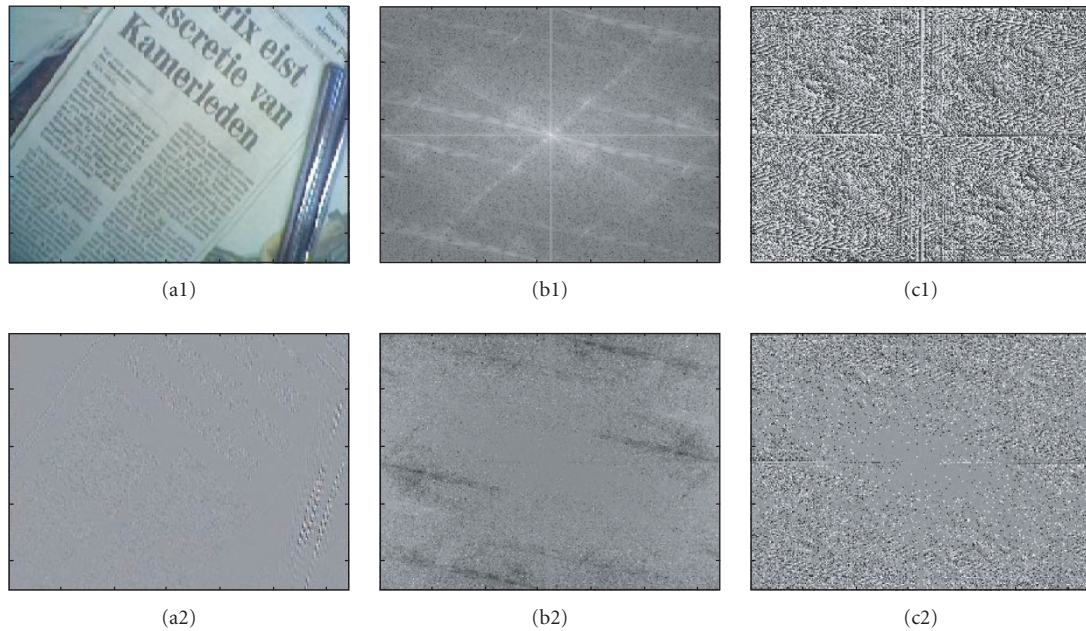


FIGURE 35: Super-resolved frame after combining 3 low-resolution frames in (a1) the spatial domain, in (b1) the frequency domain in magnitude, and in (c1) phase, together with their associated errors ((a2), (b2), and (c2)).

With respect to the quality exhibited by each SR algorithm, it is interesting to remark that it is important to consider not only the absolute value of the PSNR achieved by the algorithm but also the quality increase with respect to the interpolation levels, as it represents the SR gain with respect to a cheaper and faster method to increase the image size. In Figure 38 a comparison between the quality obtained

by the NISR algorithm (Figure 33) and the best results of the ISR algorithm (maximum values are the same for 8 and 80 iterations) is presented. It is clear how the NISR algorithm outperforms the ISR algorithm when the number of low-resolution frames is above four frames. The larger PSNR difference occurs when 12 low-resolution frames are combined, getting a gain of 5.54 dB for the NISR over the ISR algorithm.

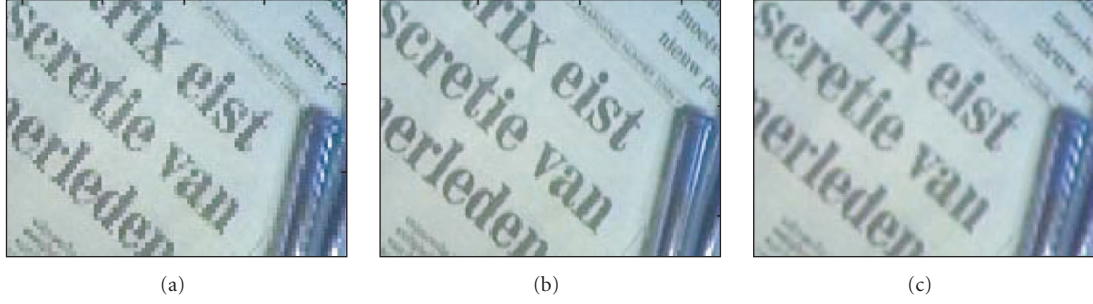


FIGURE 36: Enlarged detail of (a1) the nearest-neighbor interpolation of the input image, (b1) the SR image combining 12 low-resolution input frames, and (c1) the bilinear interpolation of the input image for the Krant sequence.

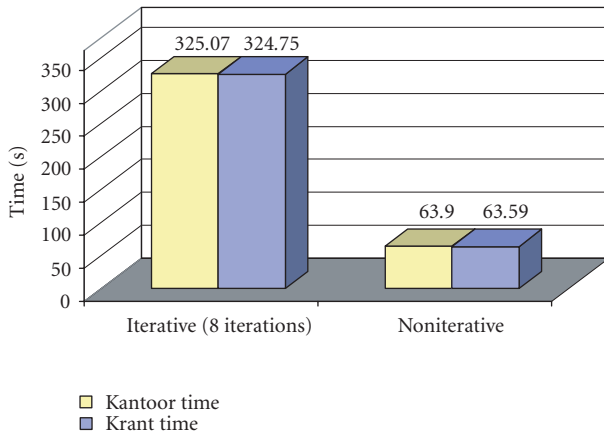


FIGURE 37: Execution time comparison between the ISR and NISR algorithms presented in this paper.

The average difference is of 3.08 dB from the combination of 3 to 12 low-resolution frames.

In Figure 39 a quality comparison between Kantoor and Krant sequences for eight iterations of the ISR algorithm is presented. As the Kantoor sequence has several different focal planes, it is more difficult to compute a motion-vector set that accurately matches the real motion vectors. In the case of the Krant sequence, many MBs belong to objects in the same focal plane and share the same motion-vectors, thus increasing the motion vector consistency and the overall super-resolved quality. Due to this reason, a higher quality is obtained for the Krant sequence for all the output frames. The SR enhancement with respect to the bilinear interpolation level, for frame six, has been marked with two arrows, being 8.45 dB in the case of the Krant sequence and 1.55 dB in the case of the Kantoor sequence. Figure 40 shows the quality comparison between the same two sequences for the NISR algorithm. Once again, the quality of the Krant sequence is always above the quality of the Kantoor one. For instance, after combining 7 frames the Krant sequence exhibits an SR enhancement of 11.37 dB and the Kantoor sequence an SR enhancement of 9.3 dB.

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. Figure 41 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.

It is important to note that although the SR algorithms have been applied both to the luminance and the chrominance, the major improvements with respect to the interpolation levels have been obtained for the luminance component. This is mainly due to the lower entropy of the chrominances in the YCbCr 4 : 2 : 0 sampling format, which does not allow an accurate edge reconstruction as in the luminance case. In other words, the chrominances have deficient high frequency information to reconstruct a high-resolution colour image with a quality higher than the interpolated counterparts.

Finally, in Figure 42 the memory used by the ISR algorithm for different magnification factors ranging from $\times 2$ to $\times 8$ and for several input image sizes is shown. In Figure 43 the memory demanded by the NISR for the same magnification factors and input image sizes is shown. Taking into account that the hardware platform cannot allocate more than 1 Mbyte of SRAM internal memory in order to achieve low-cost goals, only QCIF at $\times 2$ or SQCIF at $\times 2$ and $\times 3$ can be addressed in this implementation.

9. CONCLUSIONS

High-resolution images are often required in nowadays multimedia services and applications. The achievement of high-quality images 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 minimal changes on the architecture, such as using sub-pixel motion estimation, enough loop memory, and additional block-level operator and accumulators. Two different SR algorithms are proposed in this work. The first one, named iterative super-resolution (ISR) algorithm, presents a much higher performance than classical image interpolation

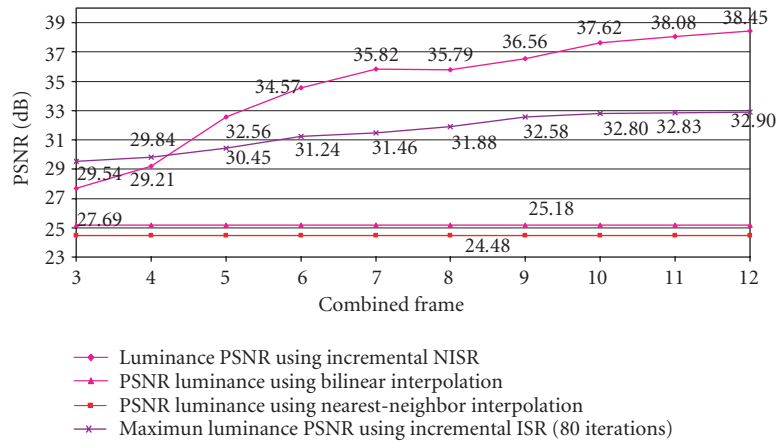


FIGURE 38: PSNR of the Krant sequence with 10 incremental output frames using the NISR and the ISR algorithms with 80 iterations.

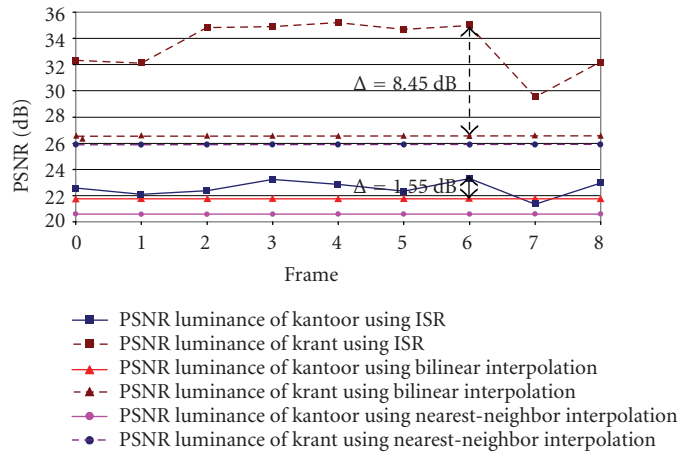


FIGURE 39: PSNR comparison between Kantoor and Krant sequences for the ISR algorithm (8 frames, ten iterations per frame).

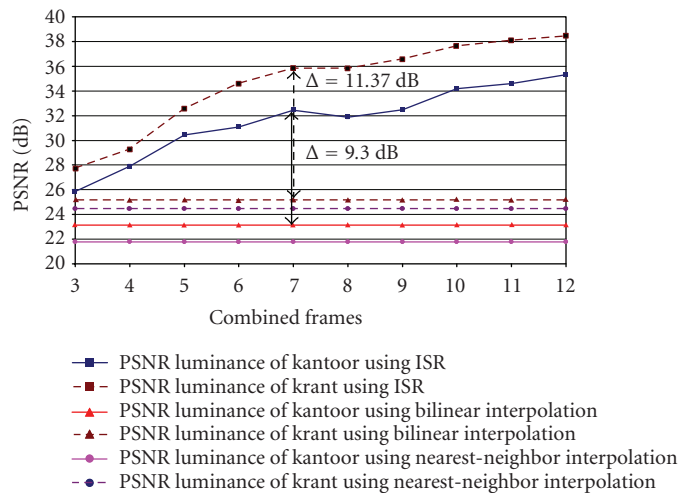


FIGURE 40: PSNR comparison between Kantoor and Krant sequences for the NISR algorithm.

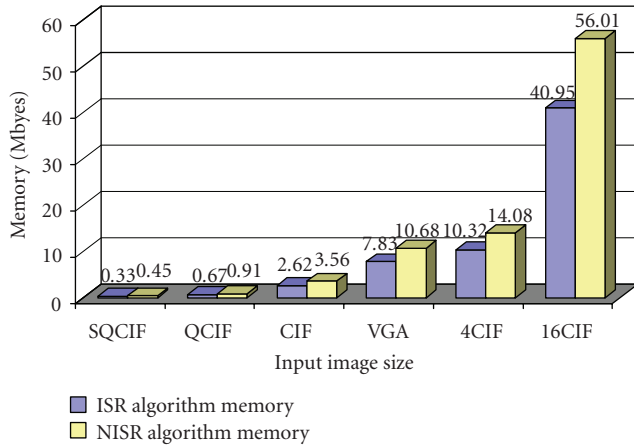


FIGURE 41: Memory requirements for the ISR versus NISR algorithm.

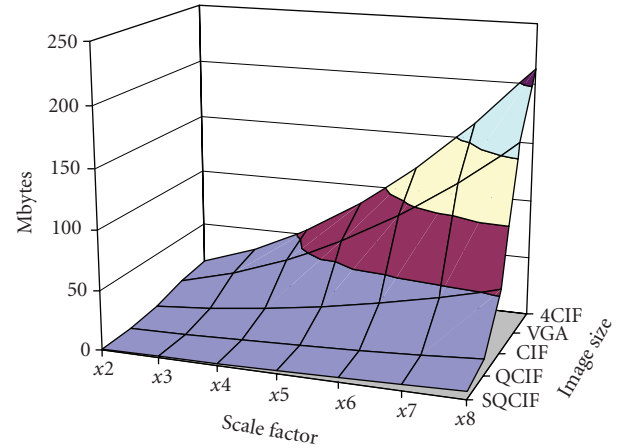


FIGURE 43: Memory requirements for the NISR algorithm.

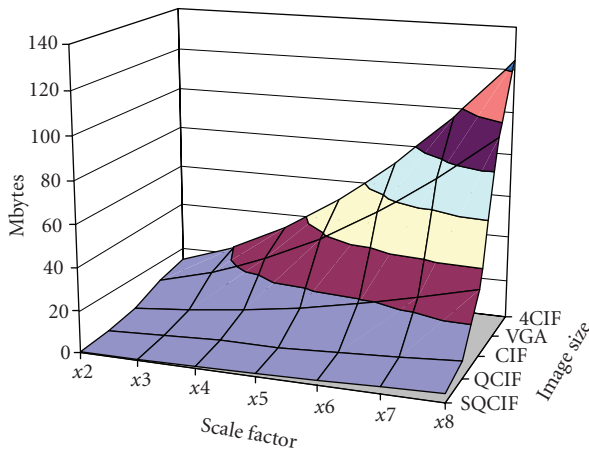


FIGURE 42: Memory requirements for the ISR algorithm.

methods. However, it is not always possible to achieve SR improvements by using this approach, because if all the sampled data are not present, 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 restrict the number of iterations, as it has been stated in this paper. Although the ISR algorithm exhibits a good behavior 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 noniterative super-resolution (NISR) algorithm has been developed. The experiments carried out reveal a clear quality increase of the super-resolved image as the numbers of low-resolution frames to be combined are increased. The introduction of the contribution concept allows the algorithm to be independent of problems related to the image 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 algorithm in a single step, but at the expense 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 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 applied 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 (MPSoC) video platforms.

ACKNOWLEDGMENTS

This work was supported in part by the Philips Research Laboratories in the context of the “Picasso” project and by the Institute for Applied Microelectronics (IUMA) at the University of Las Palmas de Gran Canaria (ULPGC) in the context of projects TIC2003-09687-C02-02 and TEC2005-018138-C02-01/MIC of Spain’s National Research Plan. The authors want to thank Marc Op de Beek and Richard Kleihorst from Philips Research Laboratories, Eindhoven, for the contribution to this work. The reviewers’ technical comments in order to improve the overall quality of the paper are also greatly acknowledged.

REFERENCES

- [1] T. Komatsu, T. Igarashi, K. Aizawa, and T. Saito, “Very high resolution imaging scheme with multiple different-aperture cameras,” *Signal Processing: Image Communication*, vol. 5, no. 5-6, pp. 511–526, 1993.
- [2] D. C. Ghiglia, “Space-invariant deblurring given N independently blurred images of a common object,” *Journal of Optical Society of America A*, vol. 1, no. 4, pp. 398–402, 1984.
- [3] S. P. Kim and W.-Y. Su, “Recursive high-resolution reconstruction of blurred multiframe images,” *IEEE Transactions on Image Processing*, vol. 2, no. 4, pp. 534–539, 1993.

- [4] P. Cheeseman, B. Kanefsky, R. Kraft, J. Stutz, and R. Hanson, "Super-resolved surface reconstruction from multiple images," in *Maximum Entropy and Bayesian Methods*, G. R. Heidbreder, Ed., pp. 293–308, Kluwer Academic, Dordrecht, The Netherlands, 1996.
- [5] V. Avrin and I. Dinstein, "Restoration and resolution enhancement of video sequences," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)*, vol. 4, pp. 2549–2552, Munich, Germany, April 1997.
- [6] P. Eggleston, "Quality photos from digital video: salient Stills' algorithms provide a missing link," *Advanced Imaging Magazine*, vol. 15, no. 5, pp. 39–41, 2000.
- [7] R. R. Schultz, "Super-resolution enhancement of native digital video versus digitized NTSC sequences," in *Proceedings of the 5th IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI '02)*, pp. 193–197, Sante Fe, NM, USA, April 2002.
- [8] D. Sale, R. R. Schultz, and R. J. Szczerba, "Super-resolution enhancement of night vision image sequences," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, pp. 1633–1638, Nashville, Tenn, USA, October 2000.
- [9] B. K. Gunturk, A. U. Batur, Y. Altunbasak, M. H. Hayes III, and R. M. Mersereau, "Eigenface-based super-resolution for face recognition," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '02)*, vol. 2, pp. 845–848, Rochester, NY, USA, September 2002.
- [10] D. Capel and A. Zisserman, "Super-resolution enhancement of text image sequences," in *Proceedings of the 15th International Conference on Pattern Recognition (ICPR '00)*, vol. 1, pp. 600–605, Barcelona, Spain, September 2000.
- [11] J. A. Goyette, G. D. Lapin, M. G. 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, 1994.
- [12] A. K. Katsaggelos and N. P. Galatsanos, Eds., *Signal Recovery Techniques for Image and Video Compression and Transmission*, Kluwer Academic, Dordrecht, The Netherlands, 1998.
- [13] D. Chen and R. R. Schultz, "Extraction of high-resolution video stills from MPEG image sequences," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '98)*, vol. 2, pp. 465–469, Chicago, Ill, USA, October 1998.
- [14] K. J. Erickson and R. R. Schultz, "MPEG-1 super-resolution decoding for the analysis of video still images," in *Proceedings of the 4th IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI '00)*, pp. 13–17, Austin, Tex, USA, April 2000.
- [15] 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, no. 9, pp. 803–811, 2002.
- [16] 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, no. 4, pp. 217–226, 2002.
- [17] C. A. Segall, R. Molina, and A. K. Katsaggelos, "High-resolution images from low-resolution compressed video," *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 37–48, 2003.
- [18] F. M. Candocia and J. C. Principe, "Method using multiple models to superresolve SAR imagery," in *Algorithms for Synthetic Aperture Radar Imagery V*, vol. 3370 of *Proceedings of SPIE*, pp. 197–207, Orlando, Fla, USA, April 1998.
- [19] Y. Cheng, Y. Lu, and Z. Lin, "A super resolution SAR imaging method based on CSA," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS '02)*, vol. 6, pp. 3671–3673, Toronto, Canada, June 2002.
- [20] D. Pastina, P. Lombardo, A. Farina, and P. Daddi, "Super-resolution of polarimetric SAR images of a ship," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS '01)*, vol. 5, pp. 2343–2345, Sydney, NSW, Australia, July 2001.
- [21] R. Wu and J. Li, "Autofocus and super-resolution synthetic aperture radar image formation," *IEEE Proceedings—Radar, Sonar and Navigation*, vol. 147, no. 5, pp. 217–223, 2000.
- [22] 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," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS '01)*, vol. 3, pp. 1101–1103, Sydney, NSW, Australia, July 2001.
- [23] X. Liang, R. Tao, S. Zhou, and Y. Wang, "Multidimensional real aperture radar imaging," in *Proceedings of CIE International Conference on Radar*, pp. 675–678, Beijing, China, October 2001.
- [24] 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," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS '01)*, vol. 7, pp. 3200–3202, Sydney, NSW, Australia, July 2001.
- [25] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.
- [26] H. Nagahara, Y. Yagi, and M. Yachida, "Super-resolution from an omnidirectional image sequence," in *Proceedings of the 26th Annual Conference of the IEEE Industrial Electronics Society (IECON '00)*, vol. 4, pp. 2559–2564, Nagoya, Japan, October 2000.
- [27] H. Nagahara, Y. Yagi, and M. Yachida, "Resolution improving method for a 3D environment modeling using omnidirectional image sensor," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02)*, vol. 1, pp. 900–907, Washington, DC, USA, May 2002.
- [28] M. Elad and A. Feuer, "Super-resolution reconstruction of image sequences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 817–834, 1999.
- [29] M. Elad and A. Feuer, "Superresolution restoration of an image sequence: adaptive filtering approach," *IEEE Transactions on Image Processing*, vol. 8, no. 3, pp. 387–395, 1999.
- [30] T. S. Huang and R. Y. Tsay, "Multiple frame image restoration and registration," in *Advances in Computer Vision and Image Processing*, T. S. Huang, Ed., vol. 1, pp. 317–339, JAI Press, Greenwich, Conn, USA, 1984.
- [31] J. L. Yen, "On nonuniform sampling of bandwidth-limited signals," *IRE Transactions on Circuit Theory*, vol. 3, no. 4, pp. 251–257, 1956.
- [32] A. Papoulis, "Generalized sampling expansion," *IEEE Transactions on Circuits and Systems*, vol. 24, no. 11, pp. 652–654, 1977.
- [33] T. S. Huang, Ed., *Image Sequence Processing and Dynamic Scene Analysis*, Springer, Berlin, Germany, 1983.
- [34] B. R. Hunt and O. Kubler, "Karhunen-Loeve multispectral image restoration, part I: theory," *IEEE Transactions on Acoustics, Speech, Signal Processing*, vol. 32, no. 3, pp. 592–600, 1984.
- [35] M. E. Zervakis, "Optimal restoration of multichannel images based on constrained mean-square estimation," *Journal of*

- Visual Communication and Image Representation*, vol. 3, no. 4, pp. 392–411, 1992.
- [36] A. K. Katsaggelos, "A multiple input image restoration approach," *Journal of Visual Communication and Image Representation*, vol. 1, no. 1, pp. 93–103, 1990.
 - [37] A. K. Katsaggelos, J. N. Driessen, S. N. Efstratiadis, and R. L. Lagendijk, "Spatiotemporal motion-compensated noise filtering of image sequences," in *Visual Communications and Image Processing IV*, vol. 1199 of *Proceedings of SPIE*, pp. 61–70, Philadelphia, Pa, USA, November 1989.
 - [38] A. K. Katsaggelos, R. P. Kleihorst, S. N. Efstratiadis, and R. L. Lagendijk, "Adaptive image sequence noise filtering methods," in *Visual Communications and Image Processing '91: Image Processing*, vol. 1606 of *Proceedings of SPIE*, pp. 716–727, Boston, Mass, USA, November 1991.
 - [39] S.-J. Ho and Y. H. Lee, "Nonlinear spatio-temporal noise suppression techniques with applications in image sequence processing," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '91)*, vol. 1, pp. 662–665, Singapore, June 1991.
 - [40] A. J. Patti, A. M. Tekalp, and M. I. Sezan, "Image sequence restoration and deinterlacing by motion-compensated Kalman filtering," in *Image and Video Processing*, vol. 1903 of *Proceedings of SPIE*, pp. 59–70, San Jose, Calif, USA, February 1993.
 - [41] A. J. Patti, M. I. Sezan, and A. M. Tekalp, "High-resolution image reconstruction from a low-resolution image sequence in the presence of time-varying motion blur," in *Proceedings of the IEEE International Conference on Image Processing (ICIP '94)*, vol. 1, pp. 343–347, Austin, Tex, USA, November 1994.
 - [42] M. Elad and A. Feuer, "Recursive optical flow estimation-adaptive filtering approach," in *Proceedings of the 19th IEEE Convention of Electrical and Electronics Engineers in Israel*, pp. 387–390, Jerusalem, Israel, November 1996.
 - [43] R. R. Schultz and R. L. Stevenson, "Improved definition video frame enhancement," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '95)*, vol. 4, pp. 2169–2172, Detroit, Mich, USA, May 1995.
 - [44] A. J. Patti, M. I. Sezan, and A. M. Tekalp, "High resolution standards conversion of low resolution video," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '95)*, vol. 4, pp. 2197–2200, Detroit, Mich, USA, May 1995.
 - [45] R. R. Schultz and R. L. Stevenson, "Extraction of high-resolution frames from video sequences," *IEEE Transactions on Image Processing*, vol. 5, no. 6, pp. 996–1011, 1996.
 - [46] M. Elad and A. Feuer, "Restoration of single super resolution image from several blurred, noisy, and undersampled measured images," *IEEE Transactions on Image Processing*, vol. 6, no. 12, pp. 1646–1658, 1997.
 - [47] M. G. Kang and A. K. Katsaggelos, "Simultaneous iterative image restoration and evaluation of the regularization parameter," *IEEE Transactions on Signal Processing*, vol. 40, no. 9, pp. 2329–2334, 1992.
 - [48] R. L. Lagendijk and J. Biemond, *Iterative Identification and Restoration of Images*, Kluwer Academic, Boston, Mass, USA, 1991.
 - [49] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, Upper Saddle River, NJ, USA, 1989.
 - [50] W. K. Pratt, *Digital Image Processing*, John Wiley & Sons, New York, NY, USA, 1991.
 - [51] A. Zomet, A. Rav-Acha, and S. Peleg, "Robust super-resolution," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '01)*, vol. 1, pp. 645–650, Kauai Marriott, Hawaii, USA, December 2001.
 - [52] D. C. Youla, "Generalized image restoration by the method of alternating orthogonal projections," *IEEE Transactions on Circuits and Systems*, vol. 25, no. 9, pp. 694–702, 1978.
 - [53] W. Zhao, H. Sawhney, M. Hansen, and S. Samarasekera, "Super-fusion: a super-resolution method based on fusion," in *Proceedings of the 16th International Conference on Pattern Recognition (ICPR '02)*, vol. 2, pp. 269–272, Quebec, Canada, August 2002.
 - [54] M. Irani and S. Peleg, "Motion analysis for image enhancement: resolution, occlusion, and transparency," *Journal of Visual Communication and Image Representation*, vol. 4, no. 4, pp. 324–335, 1993.
 - [55] W. T. Freeman and E. C. Pasztor, "Learning low-level vision," in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV '99)*, vol. 2, pp. 1182–1189, Kerkyra, Greece, September 1999.
 - [56] D. Capel and A. Zisserman, "Super-resolution from multiple views using learnt image models," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '01)*, vol. 2, pp. 627–634, Kauai Marriott, Hawaii, USA, December 2001.
 - [57] S. Baker and T. Kanade, "Limits on super-resolution and how to break them," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 9, pp. 1167–1183, 2002.
 - [58] S. Chaudhuri, Ed., *Super-Resolution Imaging*, Kluwer Academic, Boston, Mass, USA, 2001.
 - [59] S. Farsiu, D. Robinson, M. Elad, and P. Milanfar, "Fast and robust multi-frame super-resolution," *IEEE Transactions on Image Processing*, vol. 13, no. 10, pp. 1327–1344, 2004.
 - [60] M. O. de Beeck and R. P. Kleihorst, "Super-resolution of regions of interest in a hybrid video encoder," in *Proceedings of Philips Conference on Digital Signal Processing (DSP '99)*, Veldhoven, The Netherlands, November 1999.
 - [61] G. M. Callicó, R. P. Llopis, A. Núñez, and R. Sethuraman, "Mapping of real-time and low-cost super-resolution algorithms onto a hybrid video encoder," in *VLSI Circuits and Systems*, vol. 5117 of *Proceedings of SPIE*, pp. 42–52, Maspalomas, Spain, May 2003.
 - [62] R. P. Llopis, R. Sethuraman, C. A. Pinto, H. Peters, S. Maul, and M. Oosterhuis, "A low-cost and low-power multi-standard video encoder," in *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, pp. 97–102, Newport Beach, Calif, USA, October 2003.
 - [63] G. M. Callicó, A. Núñez, R. P. Llopis, R. Sethuraman, and M. O. de Beeck, "A low-cost implementation of super-resolution based on a video encoder," in *Proceedings of the 28th Annual Conference of the IEEE Industrial Electronics Society (IECON '02)*, vol. 2, pp. 1439–1444, Sevilla, Spain, November 2002.
 - [64] R. P. Llopis, M. Oosterhuis, S. Ramanathan, et al., "HW-SW co-design and verification of a multi-standard video and image codec," in *Proceedings of the 2nd IEEE International Symposium on Quality of Electronic Design (ISQED '01)*, pp. 393–398, San Jose, Calif, USA, March 2001.
 - [65] G. M. Callicó, A. Núñez, R. P. Llopis, and R. Sethuraman, "Low-cost and real-time super-resolution over a video encoder IP," in *Proceedings of the 4th IEEE International Symposium on Quality Electronic Design (ISQED '03)*, pp. 79–84, San Jose, Calif, USA, March 2003.

Gustavo M. Callicó was born in Granada, Spain, in 1970. He received the Telecommunication Engineer degree in 1995 and the Ph.D. degree in 2003, both from the University of Las Palmas de Gran Canaria and both with honours. From 1996 to 1997 he was granted with a research grant from the Educational Minister and in 1997 he was hired by the university as an electronic lecturer. In 1994 he joined the research Institute for Applied Microelectronics (IUMA) and in 2000 he stayed for 1 year in the Philips Research Laboratories (NatLab) at Eindhoven, The Netherlands, as a Visiting Scientist, where he developed his Ph.D. thesis. He is actually an Assistant Professor at the University of Las Palmas de Gran Canaria and develops his research activities at the Integrated Systems Design Division of the IUMA. His current research fields include real-time super-resolution algorithms, synthesis-based design for SOCs, circuits for multimedia processing, and video coding standards, especially H.264.



Rafael Peset Llopis has an M.S. degree in physics (1989, cum laude) from the University of Groningen, and a Ph.D. degree in electrical engineering (1993) from the University of Twente, both in the Netherlands. From 1994 to 2002 he worked in Philips Research on design methodologies, and a low-cost, low-power, and multistandard video codec for mobile applications. Currently he is responsible for the Digital VLSI Group of the IC-Lab of Philips Consumer Electronics. His research interests are video compression, system-level design, and low power. He has published more than 20 articles, 2 patents, and several patent filings. He has supervised 2 Ph.D. and more than 10 M.S. students, and has given courses at the Philips Centre for Technical Training.



Sebastian López was born in Las Palmas de Gran Canaria, Spain, in 1978. He received the M.Eng. degree in 2001, obtaining regional and national awards for his CV during his degree. He is actually an Assistant Professor at the University of Las Palmas de Gran Canaria and develops his research activities at the Integrated Systems Design Division of the Research Institute for Applied Microelectronics, where he is finishing his Ph.D. thesis in electronic engineering in the field of architectures and solutions for video motion estimation. His research interests include real-time super-resolution algorithms, video coding standards, and multiprocessor systems-on-chip.



José Fco. López obtained the five-year degree in physics (specialized in electronics) from the University of Seville, Spain, in 1989. Since then, he has conducted his investigations at the Research Institute for Applied Microelectronics (IUMA), where he is part of the Integrated Systems Design Division. He also lectures at the School of Telecommunication Engineering, the University of Las Palmas de Gran Canaria (ULPGC), being responsible for the courses on analogue circuits



and VLSI circuits. In 1994, he obtained the Ph.D. degree and was awarded by the ULPGC for his research in the field of high-speed integrated circuits. He was by Thomson Composants Microondes (now United Monolithic Semiconductor (UMS)), Orsay, France, in 1992. In 1995 he was with the "Center for Broadband Telecommunications" at the Technical University of Denmark (DTU) and in 1996, 1997, 1999, and 2000 he was funded by the Edith Cowan University (ECU), Perth, Western Australia, to make research on low-power, high-performance integrated circuits and image processing. He has been actively enrolled in more than 15 research projects funded by the European Community, Spanish Government, and international private industries. He has written around 70 papers in national and international journals and conferences.

Antonio Núñez was born in Madrid, Spain, in 1951. He received the Higher Engineering degree in 1974 from the School of Telecommunication Engineering at the Technical University of Madrid, and received the Ph.D. degree in 1981 from the same university. From 1974 to 1976 he worked as a Consultant for Telefonica. He was a Research Scientist with the Electrical Engineering Department of EPFL Lausanne, Switzerland, in 1981, working on CMOS processor design, and a Visiting Scientist (1986-1987) and a Visiting Professor (1987-1988) at the School of Electrical Engineering of Purdue University, USA, working on compound-semiconductor high-speed VLSI design. He was appointed a Titular Professor at the University of Las Palmas de Gran Canaria, in 1982, and Professor in 1989. Among other academic positions he has served as a Director of the Department of Electronics, Director of the Institute for Applied Microelectronics, and Dean of the School of Telecommunication Engineering of ULPGC. He has published five books, over 40 journal papers, and 90 conference papers. He has supervised 11 Ph.D. theses. His current research fields include heterogeneous architecture platform-based design for embedded systems, communication architecture synthesis for MPSoC, multimedia processor architectures, DVB, DAB, and DMB codec design, and optimization of integrated circuits. He is a Member of IEEE Circuits and Systems Society, IEEE Standards Association (Design Languages), The European Association for Design Automation, Euromicro, and others.



Ramanathan Sethuraman is a Senior Scientist in the Embedded Systems Architectures on Silicon (ESAS) Group of Philips Research. He received his M.S. (1992) and Ph.D. degrees in electrical engineering (1997) from the Indian Institute of Science for which he received the best M.S. and Ph.D. thesis awards. His research interests include embedded system design, low-power VLSI systems, hardware-software codesign, VLSI systems for multimedia, VLSI signal processing, and RISC/VLIW processor architectures. He has published more than 35 articles and 20 patent filings. He has supervised 2 Ph.D. students and 5 M.S. students. He is a Member of the IEEE Signal Processing Society and IEEE Communications Society.



Roberto Sarmiento is a Full Professor at the Telecommunication Engineering School at the University of Las Palmas de Gran Canaria, Spain, in the area of electronic engineering. He contributed to set this school up and was the Dean of the faculty from 1994 to 1995 and Vice-Chancellor for Academic Affairs and Staff at the ULPGC from 1998 to 2003. In 1993, he was a Visiting Professor at the University of Adelaide, South Australia, and later at the University of Edith Cowan, also in Australia. He is a founder of the Research Institute for Applied Microelectronics (IUMA) and Director of the Integrated Systems Design Division of this institute. Since 1990 he has published over 25 journal papers and book chapters and more than 60 conference papers. He has been awarded with two six-year research periods by the National Agency for the Research Activity Evaluation in Spain. He has participated in more than 30 projects and research programmes funded by public and private organizations, being a leader researcher in 14 of them. Between these projects, those funded by the European Union like GARDEN and the GRASS workgroup had special mention. He has got several agreements with companies for the design of high-performance integrated circuits, the most important being those performed with the Vitesse Semiconductor Corporation, California.

