A Parallel Framework for Video Super-Resolution

Pedro Garcia Freitas
Department of Computer Science
University of Brasília (UnB)
Brasília, Brazil
http://www.sawp.com.br

Mylène C.Q. Farias
Department of Electrical Engineering
University of Brasília (UnB)
Brasília, Brazil
http://www.pgea.unb.br/ mylene/

Aletéia P. F. de Araújo
Department of Computer Science
University of Brasília (UnB)
Brasília, Brazil

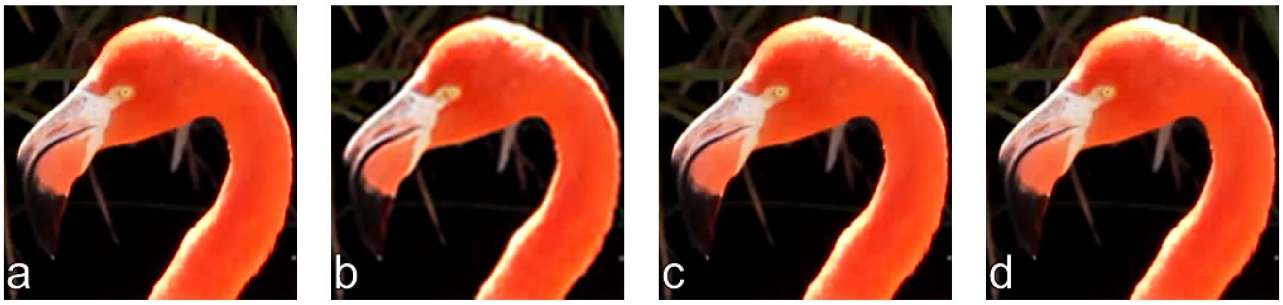


Fig. 1. Sample results of the proposed method: (a) reference frame, (b) enlarged using bicubic interpolation, (c) enlarged using SR for whole frame, and (d) enlarged using proposed method.

***Abstract*—In this paper, we propose a framework for acquiring super-resolution videos from low-resolution originals. Given that super-resolution conversion algorithms require a large amount of data processing, the proposed framework uses a set of strategies to improve performance and computational efficiency. The strategies consists of a combination of data simplification and parallel processing techniques. The simplification strategies are used to decrease the amount of data to process and, consequently, the required processing time. The parallel processing techniques are designed so that major modifications of the super-resolution algorithms are not required. The framework is fast and makes the video resolution increase timely.**

***Keywords*-Video super-resolution; parallel computing; high performance computing network;**

I. Introduction

The techniques used to increase the spatial resolution of videos consist of resizing the individual frames to higher spatial dimensions, using digital signal processing algorithms [1]. The most simple way of obtaining this magnification is using interpolation techniques. However, interpolation techniques are limited and introduce spatial and temporal artifacts in the magnified videos. The most common artifacts in interpolated images are blurring, aliasing, and edge halo.

To overcome these limitations, several techniques have been proposed to increase spatial resolution of images and videos. One of the first works on this topic was the technique proposed by Tsai and Huang [2], which considers the problem of increasing the spatial resolution of still images using a sequence of low-resolution images. Around 1990, the term "super-resolution" was incorporated in the literature by Irani and Peleg [3]. Since then, many super-resolution algorithms have been proposed, using different approaches [4], [5], [6], [7].

Some of the proposed super-resolution methods operate in frequency domain. These frequency-based methods use the shift and scale properties of the transform to obtain a higher resolution image. Tekalp *et al.* [4] use a multivariate statistical technique known as correspondence analysis [8]. Still exploring the frequency domain, Kim *et al.* [5] propose an approach that uses a recursive least-squares method.

There is also a class of super-resolution methods that operate on the spatial (pixel) domain. In this class of methods, different approaches may be used, such as reconstruction of non-uniformly spaced samples [9], backprojection [10], [7], [11], and stochastic models [12], [6], [13].

Image super-resolution algorithms are, generally, computationally expensive, since they involve performing several operations over a large amount of data. When these algorithms are adapted for video signals, the computational complexity is further increased. Therefore, approaches that reduce the processing time of video super-resolution algorithms are necessary.

In this paper, we use a selective processing strategy to reduce the processing time of super-resolution algorithms. In other words, the approach selects a subset of the video frame pixels to be processed by the super-resolution algorithms. Combined with this selective processing technique, we propose a specific parallel data processing approach. The combination of these two approaches allows us to build an efficient strategy to increase video resolution. This proposed strategy is described as a framework because the selective

1530-1834/14 $31.00 © 2014 IEEE
DOI 10.1109/SIBGRAPI.2014.15

processing technique and the parallel data processing scheme are not part of the super-resolution algorithm. Therefore, the proposed framework allows the use of different super-resolution algorithms, without the necessity of re-designing or modifying the algorithms.

The paper is described as follows. Section II describes the strategy for data simplification and the proposed parallel processing method. Section III presents the simulation results. Finally, in Section IV we present our conclusions and future works.

II. Proposed Framework

The framework consists of two major parts, which are described in this section. The first one is called "simplification", which is the classification of data in complex or simple regions. Complex regions are defined as regions with more visually significant information, when compared to simple regions. The computationally complex algorithms that produce better visual results are used only in complex regions. Less complex algorithms (interpolation) are used in the simple regions. If we want to reduce the amount of processing time, we have to classify most of the regions as simple.

The second part of the proposed framework is the parallel processing implementation of the super-resolution algorithm. Parallel processing technique makes the implementation faster, taking advantage of popular multi-core architectures. The proposed technique distributes video data and executes the super-resolution algorithm in parallel. For this, we use a hybrid architecture comprised of multi-computer and multi-processor systems.

A. Simplification Methods

As pointed out earlier, videos carry large amounts of data and, frequently, strategies are necessary to reduce the complexity of signal processing algorithms. Since video signals have a lot of redundant information, this redundancy can be used to reduce complexity. In the proposed work, we use simplifications to reduce the amount of areas that need to be processed by the most complex (and accurate) algorithms. If the selection algorithm is well designed, the videos generated with this method can be hard to distinguish from the signals generated using only the more complex algorithms. With this goal, in this work we use the following data simplifications methods: significant information selection, contour-guided processing, and differential encoding.

1) Significant Information Selection (SIS): Usually, each video frame has three color channels. The naive way to increase the frame size of a video is to use super-resolution algorithms in each of the three color channels. However, to save computational resources, super-resolution algorithms can be applied only to the most important color channel. Since digital videos are frequently encoded using the YUV color space (one luminance channel and two chrominance channels), the SIS simplification consists of using the super-resolution algorithm in the luminance channel and an interpolation algorithm in the two chrominance channels. As mentioned before, the quality of the results generated by interpolation is typically lower than the quality of the results generated by a super-resolution algorithm. However, since the human visual system is more sensitive to details in luminance, the perceptual quality difference is small [14]. This approach greatly reduces the amount of data to be processed by the most computationally expensive (by 50% in 4:2:0 videos) and, consequently, the processing time. is, therefore, considered a data simplification method.

2) Contour-Guided Processing (CGP): This strategy divides the spatial areas of the frames into two types: (1) edges and high-detail regions and (2) mostly uniform areas. For this, a mask is created using the Canny edge detector [15], which highlights the edges and high-detail regions. The regions that contain edges or details are classified as "regions of interest" (ROI), while the mostly uniform regions are classified as secondary regions (R_s). An example of this type of classification is shown in Fig. 2 for the video 'Paris'. Fig. 2-(b) shows the output of the Canny algorithm, corresponding to the original video in Fig. 2-(a). In this figure, black areas correspond to R_s and white areas to the ROI.

After the classification of the regions of the frame as ROI and R_s, for magnification purposes it is necessary to make the data more uniform. To accomplish this, the segmented image is partitioned into $n \times n$ blocks. Then, we verify which blocks contain ROI pixels and mark them as "complex" blocks (B_c). If there is no ROI pixels in the block, we mark it as a "simple" block (B_s). The output of this step is a binary mask that is used as a guide to apply the appropriate magnification algorithm. Complex blocks are processed using the super-resolution algorithm and simple blocks with an interpolation algorithm.

Figs 2-(c) and (d) show the result of the block classification for two different block sizes: 4×4 and 8×8, respectively. The white blocks are B_c blocks (complex information), while the black ones are B_s blocks (simple information). The smaller the partitioning block, the larger the discarded area and the bigger the reduction of complexity.

3) Differential Coding (DC): The CGP and SIS simplifications exploit features in data level, operating only on the spatial domain. But the information in neighboring frames is very similar, what is known as temporal redundancy. To exploit this redundancy, the difference between consecutive frames is calculated and, then, we use the previous simplifications to classify the regions of the frame.

When the CGP step is applied to the difference of two frames, we notice a further reduction of the number of ROI pixels. There is a smaller number of B_c blocks, what naturally reduces the amount of data to be processed by the super-resolution algorithm. Fig. 3 shows an example of using this method for the video 'Paris'. Fig. 2-(d) and Figure 3-(a) correspond to the results of applying CGP (8×8 blocks) to the first and second frames, respectively, of the video. Fig. 3-(b) corresponds to the result of applying CGP (8×8 blocks) to the difference between the first and second frames.

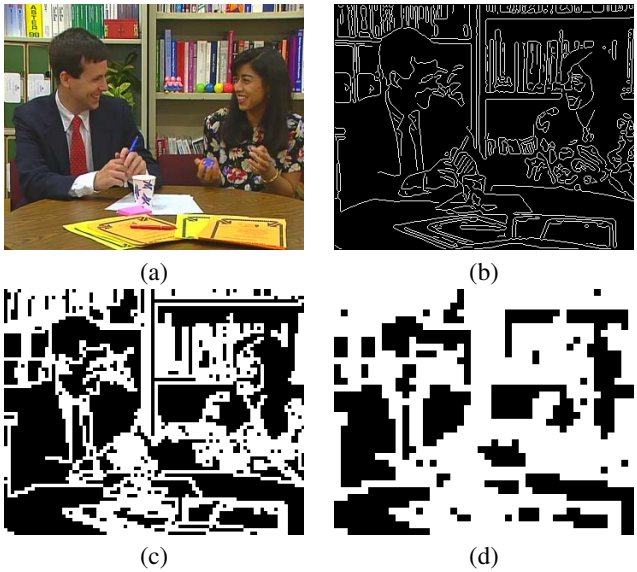


Fig. 2. Results of applying CGP to first frame of video Paris: (a) original, (b) output of Canny algorithm, (c) frame partitioned using 4×4 blocks e (d) frame partitioned using 8×8 blocks. B_c and B_s blocks in (c) and (d) are shown in white and black, respectively.

Fig. 3. Results of applying CGP (8×8 blocks) to: (a) the second frame of the video Paris and (b) the difference between the second and the first frames of the video Paris. Selected blocks are shown in white.

B. Distribution and Parallel Processing Steps

The first simplification steps are easily parallelizable. Initially, a buffer is created containing a set of consecutive frames. Depending on the video size and the available computational resources, several buffers may be created in sequence. The size and the number of these buffers depend on the size of the video.

This section describes the strategy of distributed processing for a single buffer. We assume that there is more than a single buffer available and that all steps are repeated for each buffer. The parallel processing steps described for each buffer are simplification, classification, distribution, and reconstruction, as described below.

1) Simplification and Classification: The simplification step uses a set of consecutive frames, numbered according to their time position. These positions are stored in a queue T, which is a shared structure accessible by all processes.

Then, a process is assigned to a data set, according to the available resources. For example, if a buffer contains K distinct frames in an environment with K processors, then one frame is attributed to each processor. Each process is defined as $p(t, e_i)$, where t is the frame time position and e_i is the *ith*-stage of the simplification step. Besides the selection of significant information, there are three simplification and classification stages: differential coding, contour-guided processing, and block classification.

The first stage, e_1, consists of computing the differential frames. In this case, during process $p(t, e_1)$ the difference between frames t and $t + 1$ is computed, considering the number of reference frames the user wants to keep. In other words, if the user wants to keep r referential frames, the algorithm keeps all frames with position equal to:

$$t \ mod \ \frac{K}{r} = 0. \quad (1)$$

To save the information about the type of frame (if it is a differential or a referential), a queue B is used. It serves as a bitmap, indicating whether the frame is differential or not.

In the second stage, e_2, the process $p(t, e_2)$ detects the ROIs of frame t. Similarly, in the third stage, e_3, the process $p(t, e_3)$ detects simple and complex regions in the frame, classifying them as "selected" (for the blocks B_c) or "unselected" (for the blocks B_s). The stages e_1, e_2, and e_3 are illustrated in Fig. 4.

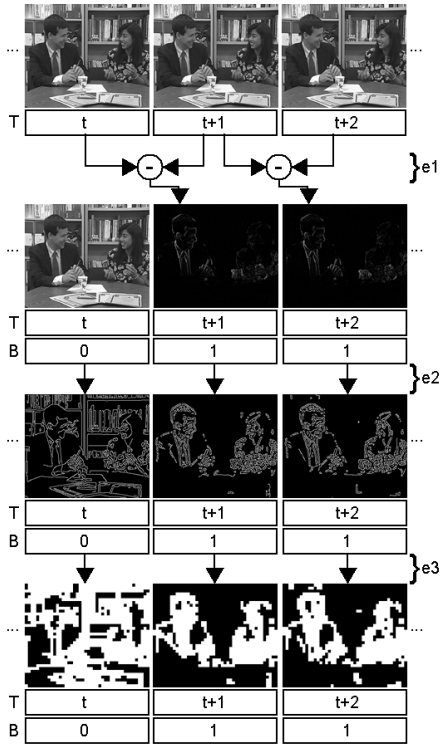


Fig. 4. Parallel processing of simplification and classification steps.

2) Distribution: The partitioning used in the simplification shows that there is a clear independence of data between each video frame. Each set of operations may be performed on each block independently.

On the other hand, when the contour-guided processing is executed in different frames, the number of selected blocks varies among different frames. This happens because different frames have different numbers of high-frequency information (ROI pixels). This difference is particularly noticeable when observing the classification of referential and differential frames, as shown in Fig. 3-(a) and 3-(b). Therefore, the homogeneous division of blocks per frame is not adequate for distribution of data among the different processes.

After the simplification steps, each block is encapsulated into a data structure called "cell" to distribute data uniformly among the processes. The cell structure contains a boolean flag indicating the block class (selected or non-selected), an integer variable indicating which frame the block belongs to, two integer variables to represent the horizontal and vertical spatial position of the block in the frame, and a matrix containing the pixel values of the block. That way, the cells stores the class of the encapsulated block and its temporal and spatial information.

To generate a data distribution with a good load balance, two queues are used: F_{in} and G_{in}. When a cell contains a true flag, it is queued in F_{in}. Otherwise, queuing is done in G_{in}. Therefore, using these queues, the blocks are arranged by demand and not by the order of the frames. Consequently, using both queues the data distribution is balanced.

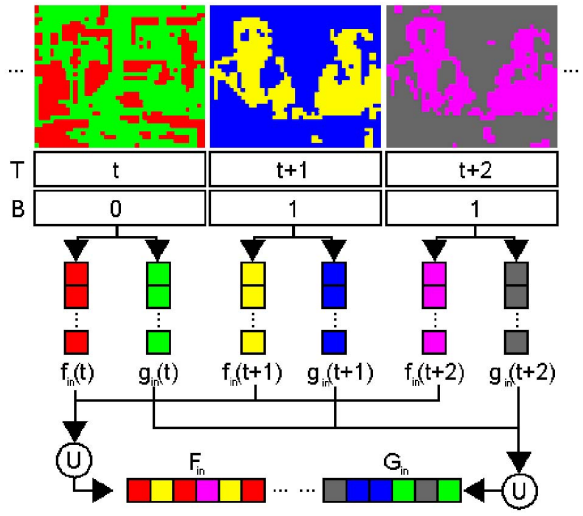
Both F_{in} and G_{in} are global shared structures. However, although each queued cell may be accessed by all processes, each process uses its own queue (f_{in} or g_{in}), which are not shared. After all blocks are classified and the queues are filled, stages e_1, e_2, and e_3 are independently executed by the processes.

Fig. II-B2 shows the distribution and classification of the internal and external data structures. For the three frames shown, each highlighted square is a cell. Different colors are used to indicate the frame class and the block type. Red, yellow, and pink regions are formed by B_c blocks of frames t, $t+1$, and $t+2$, respectively. Green, blue, and gray regions contain B_s blocks of frames t, $t+1$, and $t+2$, respectively. When these cells are queued in F_{in}, the elements may be treated uniformly, allowing this structure to be equally distributed among the processes.

Next, the content of queues F_{in} and G_{in} are distributed over all nodes. The distribution is made delivering a mesh to each node. In this case, a mesh is a collection of cells, which is equal to the number of nodes in the platform. In other words, in an environment with K nodes, there will be K meshes.

To take advantage of heterogeneous architectures, the proposed framework supports a set of processors in each node. This way, the framework can be executed in a distributed memory system, where each node has a set of processors using shared memory. Each node receives a mesh, distributes the cells among processors, and queues the result in a non-shared structure f_{out}.

At this point, f_{out} contains the cells with the enlarged blocks, queued without the original video position. Each input cell is processed and the result is stored in a symmetric cell,



captionBlock classifications in the distribution step.

which has the equivalent address information of the input (spatial and temporal positions – x, y and t), but containing the corresponding magnified block.

Fig. 5 shows the distribution and processing over the nodes, which correspond to sending the content of meshes k and $k+1$ to nodes k and $k+1$. In this figure, each mesh has only four cells, which are distributed to the four processors contained in each node. Fig. 5 also illustrates how the processing of F_{in} and G_{in} is performed in a master-slave paradigm [16]. In this case, G_{in} is processed sequentially at the master node and the results are stored in G_{out}. The slave nodes are used to processing F_{in}, which contains the blocks labeled as "selected" that are magnified using super-resolution algorithms.

3) Reconstruction: After the f_{in} has been processed and sent to f_{out}, the elements of this last queue are queued in F_{out} by each node. Next, the elements are ordered to the original position. The sorting is performed as shown in Fig. 6. First, we iterate through G_{out}, checking the correct position which is encapsulated in each cell structure. Using the position information, we copy the encapsulated block to the output video. Then, we process F_{out} in the same way.

When all blocks are reordered, we must perform the reverse process of that described in Section II-A3. Therefore, the last stage of the framework is to look over the frames and check if it is differential or not. If the frame is differential, we sum it with the previous frame.

Fig. 7 shows this last step. Comparing this figure with Fig. 4, we notice that the computation of differential frames is clearly parallelizable. Moreover, the restoration shown in Fig. 7 is inherently serial because the frame restored on $t+1$ depends of t, $t+2$ depends on $t+1$, and so on.

C. Avoiding the Blocking Effect

The decomposition of the frames in independent blocks generates a blocking effect, which consists of visible edges around the borders of blocks, as illustrated in Fig. 8. To avoid

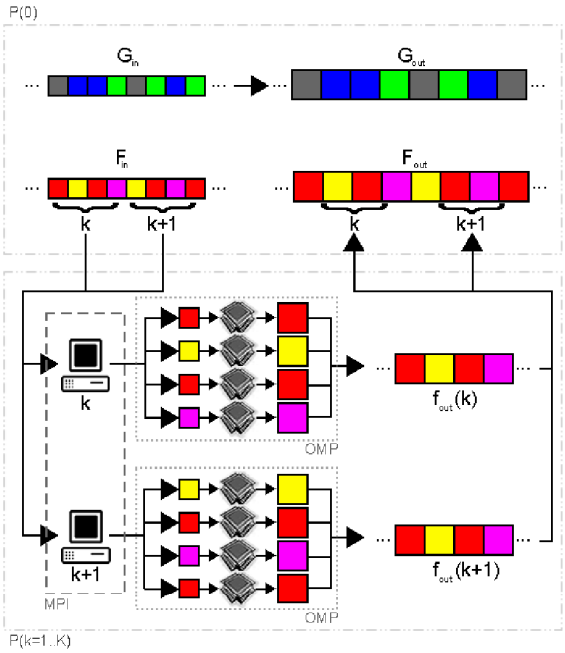


Fig. 5. Distributed processing over the nodes.

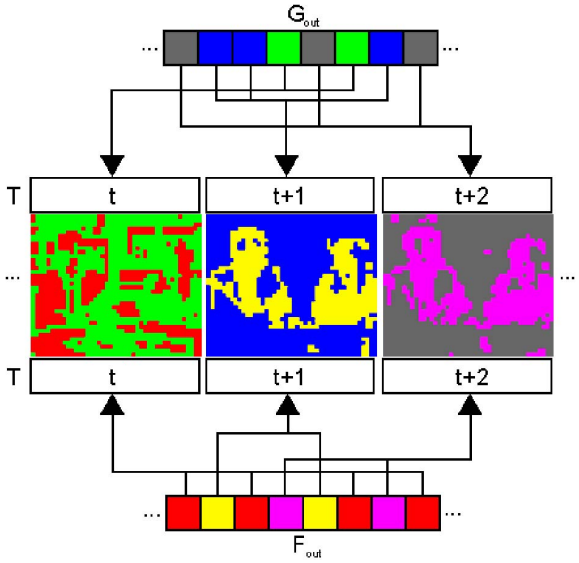


Fig. 6. Block reordering on frame reconstruction.

the blocking effect, we simply introduce an edge of zeros around every block (padding) before performing the resizing (super-resolution or interpolation). After the resizing process, the padding is removed from the magnified block.

III. Results

The parallelization scheme proposed can be portable across different parallel technologies. The load distribution, which corresponds to the division of F_{in} along the nodes, can be

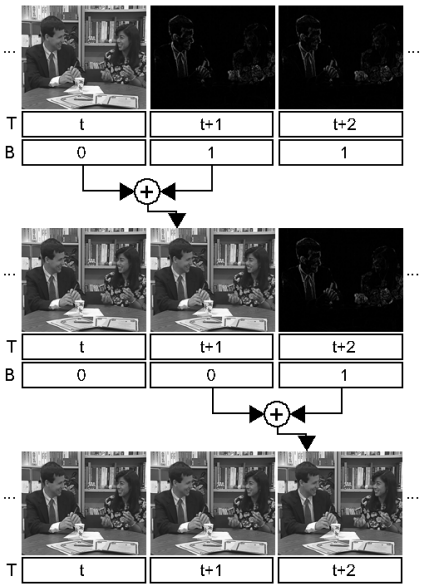


Fig. 7. Reconstruction of differential frames.

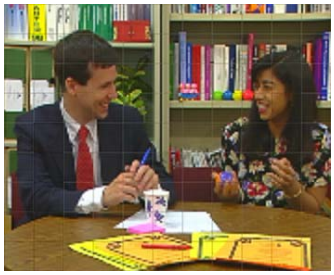


Fig. 8. Reconstructed frame without padding.

made with technologies that support the delivery of data in a multi-computer environment. Message Passing Interface [17] and Parallel Virtual Machine [18] are candidate technologies for this purpose.

In our simulations, we used a Single Program, Multiple Data (SPMD) [19] paradigm for exploiting the data parallelism in distributed memory. The results consist of the proposed framework implemented in Matlab environment. Furthermore, also using this environment, the processing within each node is performed in parallel using the Matlab functions `parfor` and `blkproc`.

The simulations were performed using a set of seven videos with different spatial and temporal characteristics. In order to standardize the performed tests, all chosen videos have 720 lines and 1280 columns, with 24 frames per second (720p24). The length of the videos differs. The video *Basketball* (BAS) has 14 seconds (337 frames), *Birds* (BIR) has 24 seconds (593 frames), *Dancing* (DAN) has 40 seconds (961 frames), *Flamingo* (FLA) has 10 seconds (250 frames), *Football* has 19 frames (457 frames), *Kiss* has 14 seconds (326 frames), and *Running* has 20 seconds (481 frames).

Our tests consisted of resizing the frames of these videos

in a half and then recovering the original dimension using the proposed framework. In our simulations, we used three different super-resolution methods. The first two methods were proposed by Villena *et al.* [20]. The first method uses the total variation (SARTV) optimization and the second one uses the norm-1 (SARL1) optimization. The third method used in our simulations is a method proposed by Yang *et al.* (SRvSC) [21]. As the simulation results using these methods were similar, we report only the results for the first method (SARTV).

A. Simplification Performance Analysis

The first step to test the simplification performance is to count the number of blocks obtained for each region class. This is to get the total data labeled as "non-selected" and "selected". In this case, the "selected" blocks are those processed by the super-resolution algorithm. The "non-selected" blocks are magnified using an interpolation algorithm. Therefore, the amount of required computational resources is lower for a higher number of blocks marked as "non-selected".

Video blocks counting was made using different block sizes. This counting was performed on all frames with and without DC simplification. The idea is to see how differential coding contributes to the reduction of the number of "selected" blocks. As a greater number of blocks labeled as "non-selected" is desirable, we calculate the percentage of these blocks, as shown in Figs. 9.(a) and 9.(b), for frame-by-frame and differential frame classifications, respectively. From this figure, we notice that the smaller the block size, the greater the number of "non-selected" blocks. Therefore, the larger the block, the greater the chance of it being classified as "selected", which implies more data to be processed by super-resolution algorithms.

From Fig. 9, it is possible to see that differential coding increases the percentage of non-selected blocks in most of the cases. Differential coding is responsible for more than half of the data being processed by the least computationally expensive algorithm.

The next step of the analysis is to check how simplification reduces the processing time. For this, the size of the blocks was varied and execution time was measured. The times were measured for two configurations. In the first configuration, we use no simplification. The purpose of this configuration is to collect the run-time using only the super-resolution algorithms. In other words, this configuration serves as a reference to analyze the performance boost of the second configuration.

In turn, the second configuration uses the simplifications proposed in Section II-A. Calculating the processing time using the proposed framework, we calculate the performance gain. Fig. 10 illustrates the processing time for all videos, as a function of the block size, with and without simplification. As can be seen from these plots, the simplifications provide a significant reduction in processing time. Also from this figure, we notice that, for most cases, processing time is smaller for 32×32 blocks.

Fig. 11 shows the *simplification performance gain* (simplification speedup) versus the block size. As we can see from this figure, the speedups are smaller for the larger block sizes.

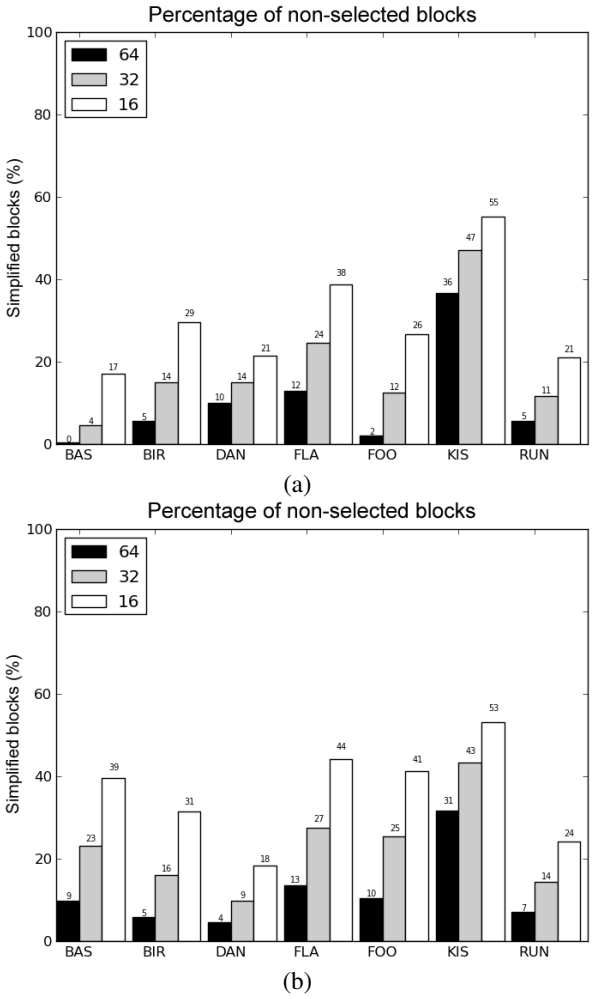


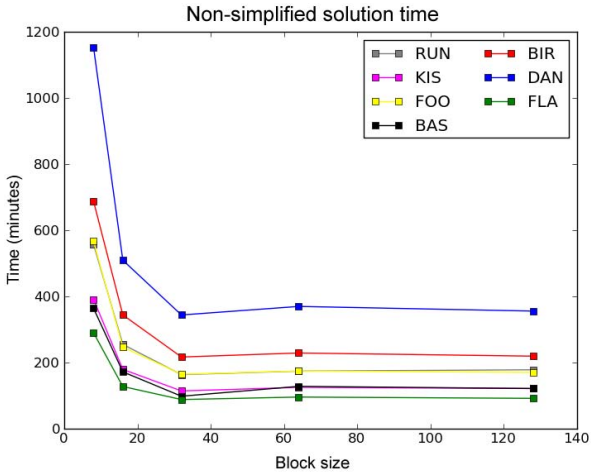
Fig. 9. Percentage of blocks classified as "non-selected": (a) frame-by-frame classification and (b) differential frame classification.

However, while the smaller blocks have greater speedup, these values only illustrate the performance gain when using the simplifications. Considering the minimal time, a good value for block size is 32×32.

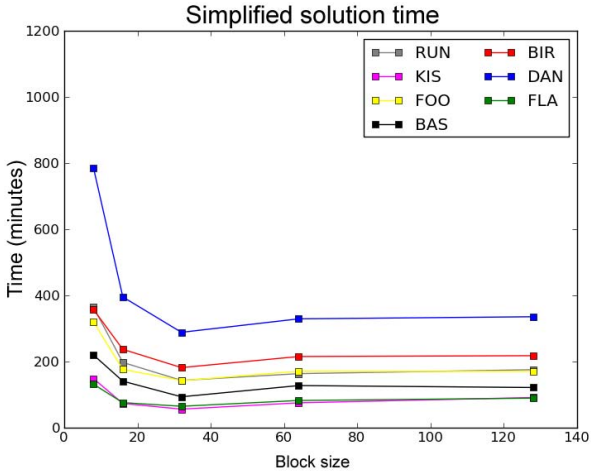
Having the optimal size for the block, an analysis of the impact of simplification on the visual properties is needed. In other words, it is necessary to examine the quality of the signal to determine how it is affected by the simplification. With this aim, we used the structural similarity index (SSIM) [22] as quality metric.

B. Visual Quality Analysis

Setting 32×32 as the block size, we must make sure that this block size generates a result with an acceptable visual quality. As observed in the previous section, for the smaller blocks we have a higher number of regions processed using interpolation. Therefore, it is expected that degradation decreases with block size while quality increases with block size. Fig. 12 shows that ssim values increase with block size. However, we can notice that the growth is small for blocks larger than 32×32.



(a)



(b)

Fig. 10. Processing time for all videos tested, as function of block size: (a) non-simplified and (b) simplified.

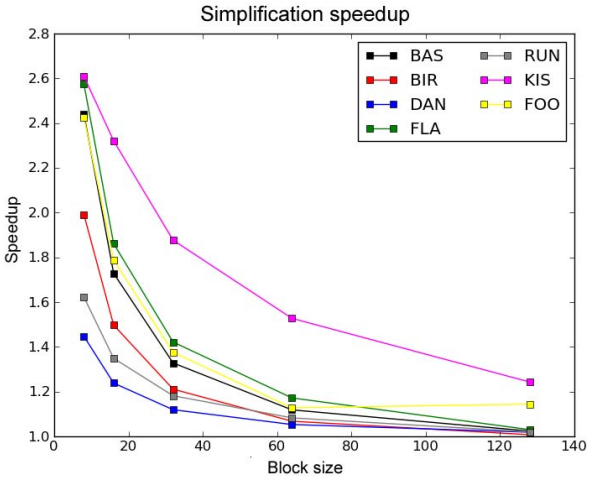


Fig. 11. Simplification performance gain (simplification speedup).

Therefore, we assume that these dimensions are a good choice for block size.

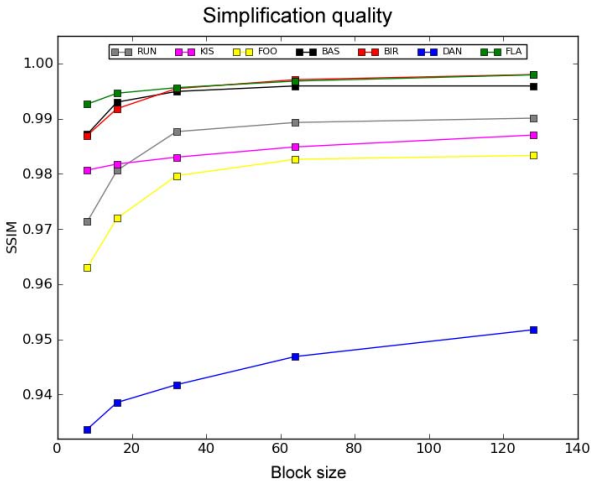


Fig. 12. Quality (SSIM) depending on the block size.

C. Parallel Computing Analysis

Besides the analysis of the quality behavior and the simplification efficiency, we analyze the performance of parallel execution. We set the block size to 32×32 and vary the number of processes. Fig. 13 shows the execution time versus the number of processors. Dividing the runtime on multiple processors by the runtime on a single processor, the speedup of the parallel implementation is computed, as shown in Fig. 14. Notice that, in most cases, speedup increases with the number of processes.

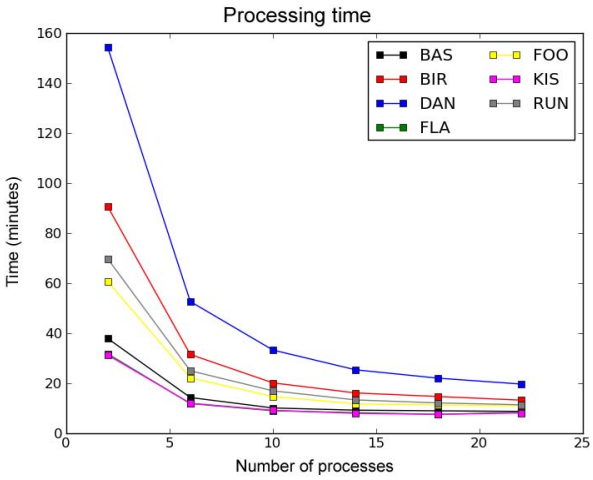


Fig. 13. Processing time depending on the number of processes.

IV. Conclusion

In this paper, we propose a framework for acquiring super-resolution videos from low-resolution originals. Given that super-resolution conversion algorithms require a large amount

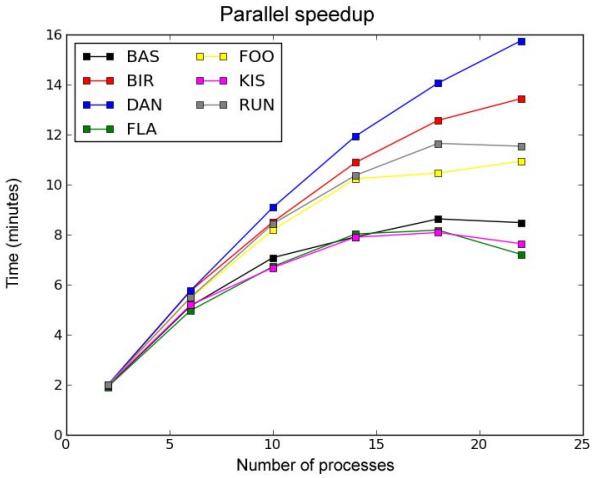


Fig. 14. Parallel speedup.

of data processing, the proposed framework uses a set of strategies to improve performance and computational efficiency. The strategies consists of a combination of data simplification and parallel processing techniques. The simplification strategies are used to decrease the amount of data to process and, consequently, the required processing time.

The framework as proposed in this work is useful in many applications. The most evident application consists of online hosting video applications that require videos in several resolutions. To offer content in high-resolution, these services require that the users submit the content in high-resolution. However, using the framework proposed in this paper, even though the users may be submitting the content in low-resolution, these services could process the videos and deliver it in high-resolution.

In future studies, we will explore other ways of selecting regions for the simplification steps. In Section II, we used the contour-guided processing to obtain the regions of interest. However, the human visual system does not pay attention to random regions. Rather, it tends to direct attention to specific information of the visual signals, as explained *Nadenau et al.* [23]. Thus, using algorithms that model the human visual attention system to extract the regions of interest is an interesting approach.

Also, works involving other parallel architectures can be implemented. For example, porting the proposed framework for technologies such as CUDA or OpenCL allows having a high degree of parallelism without the need of multiple machines. This has a direct implication on the viability of the framework as a product distribution for the end user.

REFERENCES

[1] C. Weerasinghe, M. Nilsson, S. Lichman, and I. Kharitonenko, "Digital zoom camera with image sharpening and suppression," *Consumer Electronics, IEEE Transactions on*, vol. 50, no. 3, pp. 777 – 786, aug. 2004.

[2] R. Tsai and T. Huang, "Multiframe image restoration and registration," in *Advances in Computer Vision and Image Processing*, 1984.

[3] M. Irani and S. Peleg, "Super resolution from image sequences," in *10th International Conference on Pattern Recognition. Proceedings.,*, vol. ii, Jun 1990, pp. 115–120 vol.2.

[4] A. M. Tekalp, M. K. Ozkan, and M. I. Sezan, "High-resolution image reconstruction from lower-resolution image sequences and space-varying image restoration," in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, vol. 3. IEEE, Mar. 1992, pp. 169–172 vol.3.

[5] S. Kim, N. Bose, and H. Valenzuela, "Recursive reconstruction of high resolution image from noisy undersampled multiframes," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 38, no. 6, pp. 1013 –1027, jun 1990.

[6] S. Villena, M. Vega, R. Molina, and A. K. Katsaggelos, "Image prior combination in super-resolution image reconstruction," in *European Signal Processing Conference (EUSIPCO 2010)*. Aalborg (Denmark), August 2010, pp. 616–620.

[7] S. Dai, M. Han, Y. Wu, and Y. Gong, "Bilateral back-projection for single image super resolution," in *Multimedia and Expo, 2007 IEEE International Conference on*, july 2007, pp. 1039 –1042.

[8] H. O. Hirschfeld, "A connection between correlation and contingency," in *Proceedings of the Cambridge Philosophical Society*, vol. 31. Cambridge Univ Press, 1935, pp. 520–524.

[9] F. Marvasti, *Nonuniform sampling: theory and practice*. Springer, 2001, vol. 1.

[10] M. Irani and S. Peleg, "Motion analysis for image enhancement: Resolution, occlusion, and transparency," *Journal of Visual Communication and Image Representation*, vol. 4, pp. 324–335, 1993.

[11] M. Bareja and C. Modi, "An effective iterative back projection based single image super resolution approach," in *Communication Systems and Network Technologies (CSNT), 2012 International Conference on*, may 2012, pp. 95 –99.

[12] S. Villena, M. Vega, S. Babacan, R. Molina, and A. Katsaggelos, "Using the kullback-leibler divergence to combine image priors in super-resolution image reconstruction," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, sept. 2010, pp. 893 –896.

[13] G. Chantas, N. Galatsanos, R. Molina, and A. Katsaggelos, "Variational bayesian image restoration with a product of spatially weighted total variation image priors," *IEEE Transactions on Image Processing*, vol. 19, no. 2, pp. 351–362, February 2010. [Online]. Available: http://decsai.ugr.es/vip/files/journals/final_versiontc.pdf

[14] S. Winkler, M. Kunt, and C. J. van den Branden Lambrecht, "Vision and video: models and applications," in *Vision Models and Applications to Image and Video Processing*. Springer, 2001, pp. 201–229.

[15] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[16] A. S. Tanenbaum, *Modern Operating Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007.

[17] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI-The Complete Reference, Volume 1: The MPI Core*, 2nd ed. Cambridge, MA, USA: MIT Press, 1998.

[18] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. Sunderam, "A user"s guide to pvm parallel virtual machine," Knoxville, TN, USA, Tech. Rep., 1991.

[19] A. A. Kamil, "Single program, multiple data programming for hierarchical computations," Ph.D. dissertation, University of California, 2012.

[20] S. Villena, M. Vega, D. Babacan, R. Molina, and A. Katsaggelos, "Bayesian combination of sparse and non sparse priors in image super resolution," *Digital Signal Processing*, 2012. [Online]. Available: http://decsai.ugr.es/vip/files/journals/2011spnpJPSVsentrevSVRM4.pdf

[21] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[22] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE TRANSACTIONS ON IMAGE PROCESSING*, vol. 13, no. 4, pp. 600–612, 2004.

[23] M. N., M. J. Nadenau, D. Alleysson, and M. Kunt, "Human Vision Models for Perceptually Optimized Image Processing – A Review," in *Proc. of the IEEE*, 2000. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5.2376