A Parallel Framework for Video Super-Resolution

Pedro Garcia Freitas and Aletéia P. F. de Araújo
Department of Computer Science,
University of Brasília (UnB),
Brasília, Brazil
Email: sawp@sawp.com.br, aleteia@cic.unb.br

Mylène C.Q. Farias
Department of Electrical Engineering,
University of Brasília (UnB),
Brasília, Brazil
Email: mylene@ieee.org

***Abstract*—In this paper, we propose a framework for increasing the processing efficiency of super-resolution algorithms. The framework is targeted at super-resolution video processing algorithms, that require a large amount of data processing. We propose a set of strategies that use a combination of data simplification and parallel processing. The simplification strategies are used to decrease the amount of complex data and, consequently, decrease the processing time. The parallel processing strategies are designed so that major modifications of the super-resolution algorithms are not required. As presented in this paper, the framework is fast and makes the video resolution increase timely.**

***Keywords*-Video Super-resolution, Parallel Computing, High-performance Framework**

I. Introduction

The dimensional increase of videos consists in reshaping the frames to higher spatial dimensions, using digital signal processing techniques [1]. Generally, this magnification is done using interpolation techniques. However, this kind of technique introduces distortions in the magnified frames. Such distortions occur because the magnified image is reconstructed using only the samples of the original image, which are insufficient for accurate signal reconstruction. Thus, interpolation techniques reconstruct the magnified image using subsampled coefficients, which produces unsatisfactory visual results for many applications.

One technique to reconstruct this subsampled information is super-resolution, which typically uses information from other images to increase resolution [2]. This term was incorporated in the literature in 1990 by Irani and Peleg [3]. Since then, many super-resolution algorithms have been proposed, using many different approaches [4], [5].

Image super-resolution algorithms are, generally, computationally expensive, since they involve a number of operations over a large data amount. When these algorithms are adapted to magnify video frames, this computational effort is further increased. Thus, approaches that reduce the processing time of these super-resolution algorithms are desirable.

In this paper, we propose a strategy to reduce the processing time of super-resolution algorithms. With this aim, we propose a selective processing approach, in which data is not processed exclusively by super-resolution algorithms. Combined with the selective processing technique, we also propose an approach to parallel data processing. The combination of these two approaches allows us to build a framework to increase video resolution.

II. The Framework

The framework consists of two major approaches. The first one we call "simplification" because it takes the dataset and classifies between complex and simple regions. The complex regions contain more visually significant informations than simple regions. The more costly algorithms, that produce results with more visual quality, are used in complex regions. The less expensive algorithms are used in simple regions. As more simple regions means less resource consumption, simplification tends to increase the number of these regions and decrease the number of complex regions.

The second major approach is the parallel processing of the super-resolution algorithm. The idea is to process, in parallel, the simplification steps. After this, the framework distributes the data and performs the super-resolution algorithm in parallel using a hybrid architecture (an environment comprised of multicomputer and multiprocessor systems).

A. Simplification Steps

Generally, videos have large amounts of data. Thus, strategies to reduce the amount of processing are necessary approaches. Where as video signals have much redundant information, it is expected that this data redundancy implies redundant processing. In our work, the simplifications are made to reduce the amount of processing, generating data that diverge numerically from the non-simplified approach, but in a way that this divergence is not perceived by the human visual system. Thus, the proposed simplifications are the *significant information selection*, *contour-guided processing* and *differential encoding*.

1) Significant Information Selection (SIS): Usually, each video frame has three color channels. Therefore, the naive way to increase the size of these frames could be to use the super-resolution algorithm in each of these three channels. However, to save computational resources, the super-resolution algorithms can be applied only in the channel in which detail information is more relevant. Therefore, the proposed framework always operates on the YUV color space. Thus, the super-resolution algorithms are applied to the luminance channel, and the color channels are resized using interpolation techniques.

2) Contour-guided Processing (CGP): This strategy is to create a region that is enlarged by the costly super-resolution algorithms. To this end, a mask is created using the Canny

edge detector [6], Figure 1-(b), which highlights the edges and high-detail regions. This operation segments and produces two classes of regions in the image. The first class of regions contains edges or details and are tagged as "regions of interest" (ROI), where the super-resolution algorithm are applied. The other class of regions do not contain many details and are classified as secondary regions (R_s). The Figures 1-(c) and (d) shows this classification. In this image, the black areas correspond to R_s and the whites are the ROI.

(a) (b) (c)

Fig. 1. Selection of regions of interest. (a) original, (b) Canny, (c) partitioned using 4×4 blocks.

3) Differential Coding (DC): The information between frames is very similar (redundant). This redundancy is particularly noticeable between consecutive frames. In videos, the difference between frames must be small for smooth movement transition. To exploit this redundancy, the adopted approach consists in processing the differential frames instead of each frame. This approach is well-known in video compression as *differential coding* [7]. As redundancy elimination helps decrease the information amount, we use this approach to simplify the inter-frame data processing.

B. Distribution and Parallel Processing Steps

The first simplification steps are easily parallelizable. The parallel processing steps described for each buffer are *simplification*, *classification*, *distribution* and *reconstruction*, as described below.

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

The **first stage**, e_1, consists of computing the differential frames. In this case, each process $p(t, e_1)$ computes the difference between the frames t and $t+1$, checking the amount of references (non-differential frames) that a user wants to keep. In the **second stage**, e_2, each process $p(t, e_2)$ handles each frame t, detecting the ROIs of the frame t. Similarly, in the **third stage**, e_3, the process $p(t, e_3)$ detects simple and complex regions, classifying each block as "selected" (for the blocks B_c) or "unselected" (for the blocks B_s). Finally, the stages e_1, e_2 and e_3 are illustrated in Figure 2.

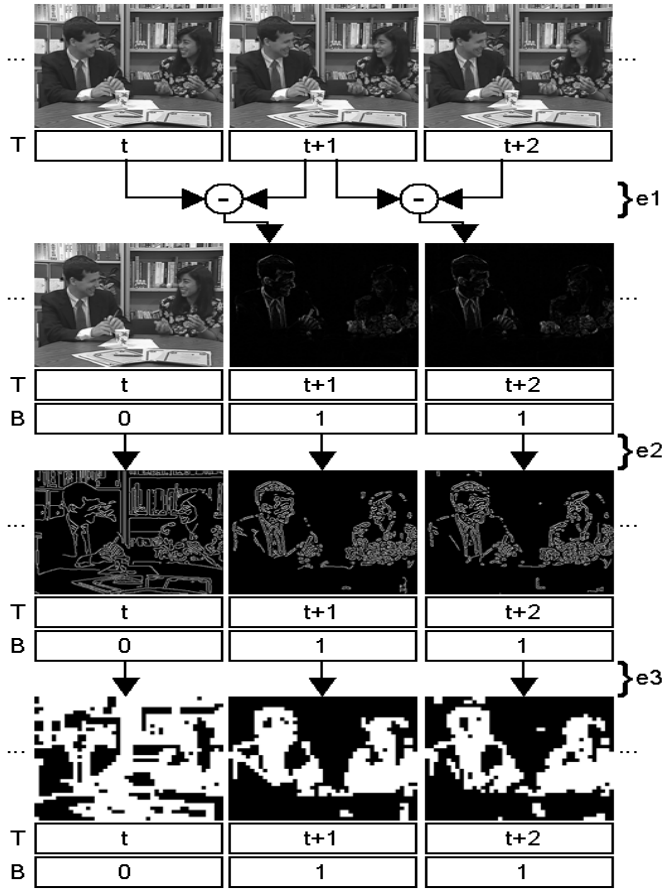

Fig. 2. Parallel processing of simplification and classification steps.

2) Distribution: After the simplification steps, each block is encapsulated into a data structure called "cell". It aims to homogenize the data distributed throughout 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 position of 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 addresses it temporally and spatially.

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

Both F_{in} as well as G_{in} are global shared structures. However, although each enqueued cell in these queues may be accessed by the processes, each process uses its own queue f_{in} or g_{in}, which are not shared. After all blocks are classified and these queues filled, the stages e_1, e_2 and e_3 are executed by the processes independently of each other, as shown in Figure 3.

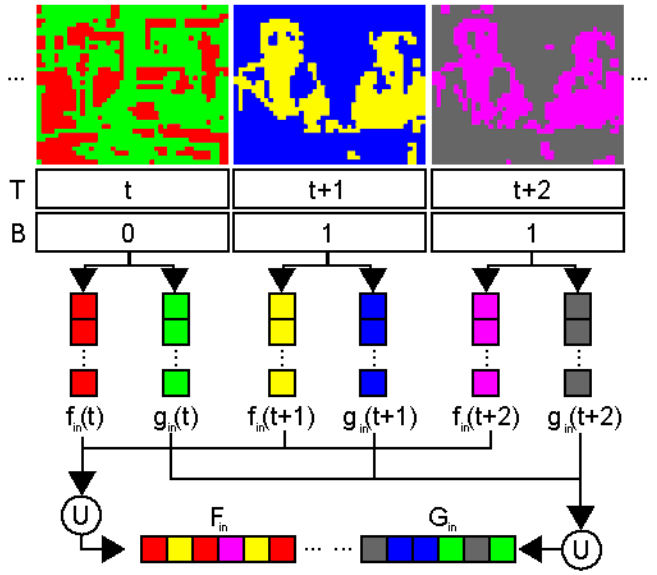

Fig. 3. Block classifications.

Then, the queues F_{in} and G_{in} are distributed over the nodes. The distribution is made delivering a mesh for each node. In this case, a mesh is a collection of cells, proportional to the number of nodes in the platform. In other words, in an environment with K nodes, there will be K meshes. Each node takes a mesh, distributes the cells among the processors and queues the result in a non-shared structure f_{out}.

At that moment, f_{out} contains the cells with the enlarged images, but uncorrelated with the position of the frame. Each input cell is processed and the result is stored into a symmetric cell, which contains the same addressing information of the input (x, y and t), but with the magnified block. This distributed processing approach is shown in Figure 4.

3) Reconstruction: After the f_{in} queuing have been processed and queued in f_{out}, these blocks are then queued in F_{out} by each node. In turn, the inverse sorting process is done, traversing each of these queues and checking each block on which frame belongs. Once the frame which the block belongs to is found, the block is vertically and horizontally properly positioned, as shown in Figure 5.

III. Results

Our tests consisted of reducing the frames of these videos in half and then increasing them using the proposed framework. In our simulations, we used the super-resolution function with three different methods. The first two methods were proposed by Villena *et al* [4]. These methods are variants of one another. Where one uses the total variation (SARTV) and the other uses the norm-1 (SARL1). The third method used in our simulations is the method proposed by Yang *et al* [5] (SRvSC). As the simulation results using different methods were similar, and for reasons of space, follow the results only 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.

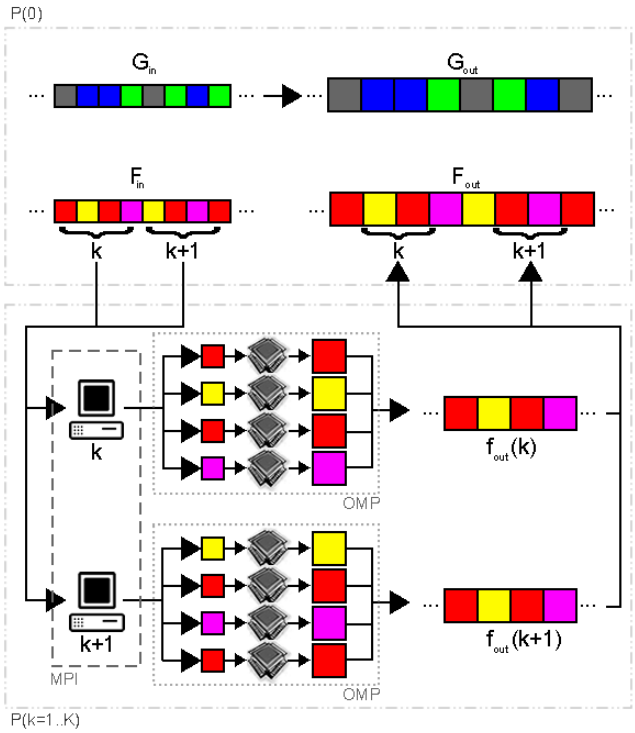

Fig. 4. Distributed processing over the nodes.

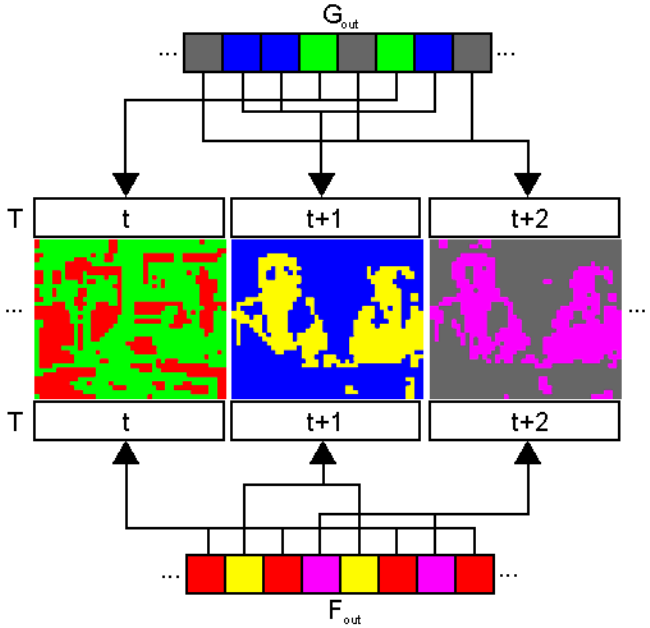

Fig. 5. Block reordering on frame reconstruction.

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 the bilinear interpolation algorithm. Therefore, for more blocks marked as “non-selected”, lower is the amount of the required computational resources.

To compute the performance gain, we borrow the concept of speedup for simplification (simplification speedup). As we can

see from Figure 6, the speedups are less for the larger block sizes. However, while the smaller blocks have greater speedup, these values only illustrates the performance gain when using the simplifications. Considering the minimum time, we have obtained that the optimum size for blocks is 32×32.

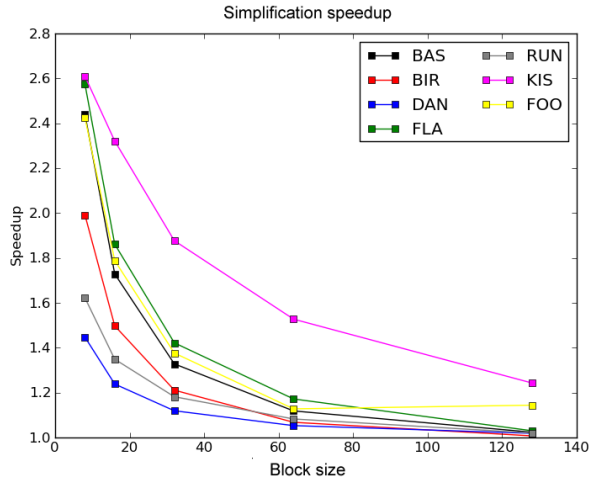

Fig. 6. Simplification performance gain (simplification speedup).

B. Visual Quality Analysis

With 32×32 as the optimum size for blocks, we must ensure that this value generates a result with acceptable visual quality. As observed in the previous section, the smaller the block, the greater the regions increased using interpolation. Therefore, it is expected that degradation is inversely Thus proportional to the block size.

Figure 7 shows that the quality increases in proportion to the block size. However, we can notice that the growth of quality is small for blocks larger than 32×32. Therefore, we can take these dimensions as optimal for block size.

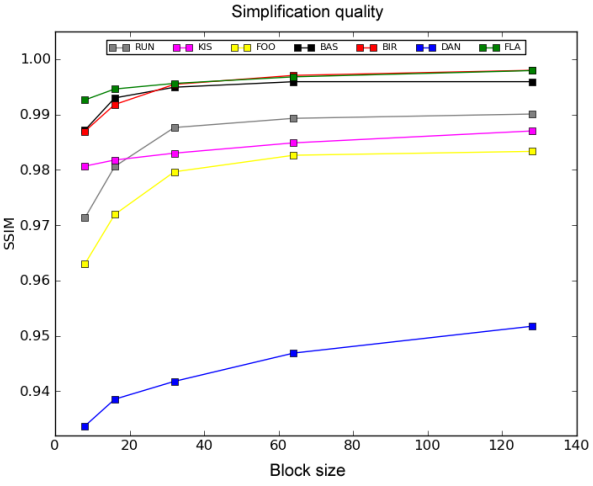

Fig. 7. Quality (SSIM) depending on the block size.

C. Parallel Computing Analysis

Besides the analysis of the quality behavior and the simplification efficiency, we analyze the performance of parallel execution. As the optimal block size is 32×32, we set the block size in these proportions and varying only the number of processes. As we can see from Figure 8, speedup increases with the number of processes in most cases.

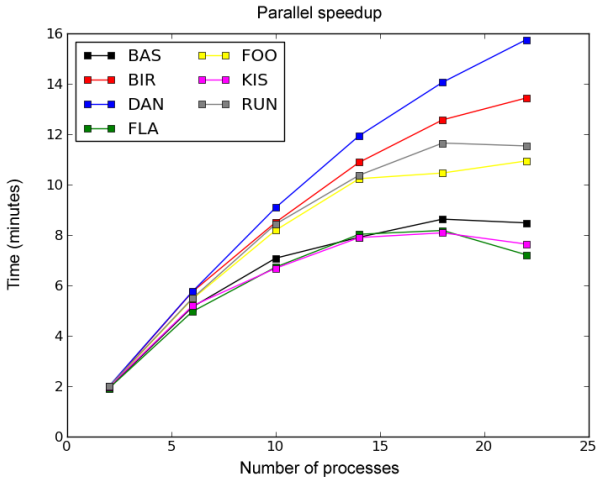

Fig. 8. Parallel speedup.

IV. Conclusion

Section II describes the proposed framework. First, we explain that the proposed simplification strategies are used to decrease the payload data used by the super-resolution algorithms. Then, we describe how these simplifications combine with parallel processing to decrease the runtime even more, as shown in Section III.

The framework as proposed in this work is applicable for many applications. The more evident usage consists of the online hosting video applications. On the other hand, in future studies we will explore ways of satisfying the selection criteria for simplification steps. In Section II, we use the contour-guided processing to classify the regions of interest.

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," *ICPR-C*, vol. 90, pp. 115–120. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.9195

[4] 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

[5] 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.

[6] 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.

[7] D. Marshall, "Differential encoding," http://www.cs.cf.ac.uk/Dave/Multimedia/node232.html, 2001, [Online; accessed 31-Jan-2013].