QoS Scheduling Algorithm for Videoclips Denoising

SORIN ZOICAN
POLITEHNICA University of Bucharest, ROMANIA

Abstract: This article presents a general frame-work for scheduling videoclips denoising processes ensuring the quality of service (QoS). In general, a denoising algorithm has two phases which are run sequentially: the first one determines the noisy pixels in the videoclip frames and the second applies a median filtering over the each frame considering the only good pixels. In all such denoising algorithms, the first phase is run for multiple times depend on the noise power. The second phase also may be executed more than one time but this depends on the specific algorithm. The issue in such applications is the denoising process may not terminate within its deadline. The proposed solution adapts the execution time in such way so the deadline to be respected by determining the remaining time to the deadline before running each phase and reducing the number of runs in each phase in order to not exceed the deadline. The goals of the article are the following: presents the QoS scheduling algorithm and proposes an implementation solution of based on Blackfin microcomputer with support of Visual DSP kernel (VDK). The article is organized in 5 sections: a briefly introduction to set up the general context of quality of services in videoclips denoising applications and to present the original video processing algorithm, two sections that present the proposed solution and its VDK implementation, the performance evaluation and the conclusions.

Key Words: QoS scheduling algorithm, deadline, Blackfin implementation

Received: November 26, 2020. Revised: April 2, 2021. Accepted: April 24, 2021. Published: April 30, 2021.

1. Introduction

In a multimedia application image is often corrupted by impulsive noise due the errors in the transmission channel. Impulsive noise, called “salt and pepper”, is caused by camera sensors, faulty hardware memory locations, or because errors occurred during communication channels transmitting images, affecting randomly a fraction of the total number of pixels, leaving other pixels unchanged. It is important to eliminate this type of noise in the images before they can apply other subsequent processing methods such as contour detection, object recognition or image segmentation. Many denoising algorithms exists almost all based on median filtering. However, the median filter may cause blurred in the reconstructed image. To overcome this phenomena a noised pixel detector is applied before median filtering. In such way the edges in the image will be preserved. The noised pixels detector is repeatedly applied over the image in order to achieve better results.[1] Unfortunately, this additional phase added to the classical median filter increases the computation time and it is possible that the application not run in real time. Moreover, many image denoising algorithms have a second phase that computes median value adaptively using the results from the first phase.

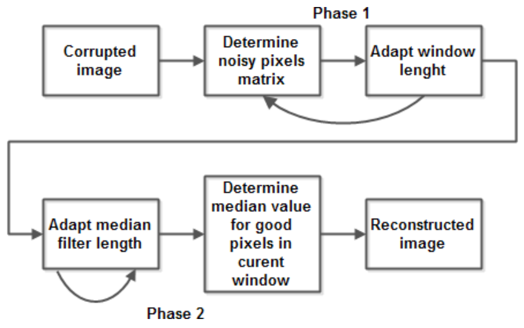
Phase 1
Corrupted image
Determine noisy pixels matrix
Adapt window lenght
Adapt median filter length
Determine median value for good pixels in curent window
Reconstructed image
Phase 2

Fig. 1. The image denoising algorithm

The Figure 1 illustrates an image denoising algorithm with two phases. For videoclips, the algorithm is applied each frame.

2. The QoS Scheduling Algorithm

In real-time systems, there are specific deadlines to be met. In particular, for a video processing application, deadlines are determined by the number of frames per second. If processing time exceeds the deadline then the following methods may be used to maintain system functionality: reserving of additional resources, tasks skip, adaptation of the task activation period and an adaptation of task execution time. In embedded systems, with limited resources, additional resources reserving can not be a viable option. More, the activation period is fixed and it not be modified without a severe degradation in performance. For example, a videoclip can not play at a different (larger) frame per second rate, different from the original rate. In such systems only the task skipping and execution time adaptation may lead to deadline meeting while preserving the functionality. The videoclip processing should be divided into two task category: mandatory tasks and additional tasks.[6] The mandatory task ensures a basic quality of the videoclip and the additional tasks improves the quality. In case that the deadline may be exceeded some of additional tasks (or all of them) will be skipped. The execution time adaptation is as similar methods. In this situation, the videoclip processing may be divided into several tasks consists of iterative sequences. The image processing performance increases with the number of iterations of each task. In this paper the execution time adaptation is used. The scheduling algorithm should be aware about the remaining execution time of each task before it reaches its deadline. If the remaining time is less than a threshold the scheduling algorithm notify the task to modify

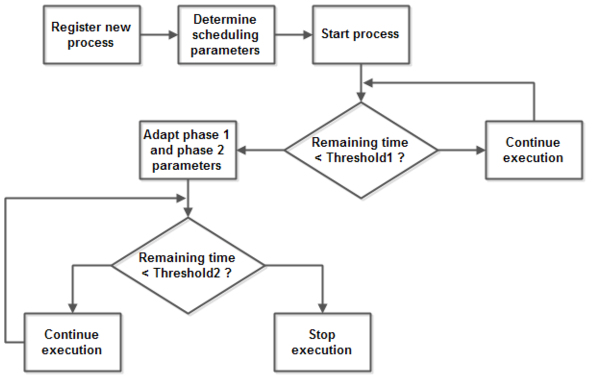
Register new process
Determine scheduling parameters
Start process
Remaining time < Threshold1 ?
Continue execution
Adapt phase 1 and phase 2 parameters
Remaining time < Threshold2 ?
Continue execution
Stop execution

Fig. 2. The denoising algorithm with QoS

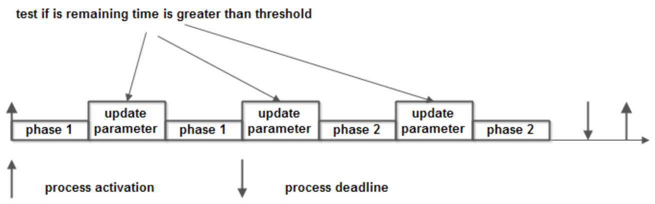

Fig. 3. A running example

the execution parameters in phase 1 and in phase 2 so that the execution time to be reduced and the deadline to be not exceeded. The scheduling algorithm is illustrated in the Figure 2. After each iteration in phase 1 and phase 2, the remaining time to the deadline is evaluated and the computations for the current phase may be interrupted. A running example is shown in Figure 3. Using this scheduling algorithm, each frame will be processed as well is possible and therephore the quality of service (QoS) is ensured.

3. The Implementation

The detailed algorithm is: In order to meet the time constraint, two thresholds are tested after each iteration in *Phase 1* and *Phase 2*.In this manner the quality of service is assured and the videoclip is played in the best conditions. Without testing the deadlines the quality of videoclip will be dramatically degraded due the fact that some very noisy frames will be not processed. The above presented algorithm was implemented on Blackfin digital signal processors family[9], [3] with Visual DSP Kernel (VDK) [4]support that provides critical kernel features: preemptive scheduler (time slicing and cooperative scheduling), thread creation, semaphores, interrupt management, inter thread messaging, events, and memory management. The image processing algorithm may be implemented easily using VDK functionality: each phase in algorithm will be separately coded in a dedicated task and a time measurement mechanism will be defined using a periodic semaphore. The following VDK primitives will be involved to measure the current execution time: *MakePeriodic()* and *GetSemaphoreValue()*. A semaphore is defined and it is declared as periodic semaphore by calling *MakePeriodic* primitive (the semaphore is posted every tick). In the processing image task,

Algorithm 1 The noise removal algorithm with QoS

```
Initial data
Corrupted image:$\mathbf{I} = \{I(i,j)|i = 0,..N-1, j = 0,...,M-1$
Noise matrix: $\mathbf{N} = \{N(i,j)|i = 0,..N-1, j = 0,...,M-1$
Current value good pixel count:$C = 0$
Previous value of good pixel count:$C_1 = 0$
Noisy pixels counter in current window: $S = 0$
Scanning window length: $L = 3$
Maximum scanning window length: $L_{max} = 5$
Reconstructed image: $\mathbf{J} = \{J(i,j)|i = 0,..N-1, j = 0,...,M-1$
Phase 1
For $i = 0..N-1 \, and \, j = 0..M-1$
1. Update the current window, centered of the current pixel $p = I(i,j)$, $\mathbf{W} = \{I(i+k_1, j+k_2)\}, k_1, k_2 \in \{-L, -L+1,..,0,..,L-1, L\}$
2. Compute minimum and maximum element in the current window: $w_{min} = min(\mathbf{W} - \{p\})$,$w_{max} = max(\mathbf{W} - \{p\})$
3. Compute the noise matrix elements and good pixel counter: $If \, (p \in [w_{min}, w_{max}]) \, then \, (N(i,j) = 0 \, and \, C = C+1) \, else \, N(i,j) = 1$
End For
$If \, (remaining \, time < threshold \, 1) \, then$
$If \, (C < C_1 and \, L \leq L_{max}) \, then \, (C = C1 \, and \, L = L+1 \, Repeat \, Phase \, 1)$
$else \, goto \, Phase \, 2$
Phase 2
For $i = 0..N-1 \, and \, j = 0..M-1$
4. $L = 1$, $S = 0$
5. Update the current window, centered of the current pixel $p = I(i,j)$, $\mathbf{W} = \{I(i+k_1, j+k_2)\}, k_1, k_2 \in \{-L, -L+1,..,0,..,L-1, L\}$
6. Count the noisy pixels in the current window: $If \, (N(i+k_1, j+k_2) = 1) \, then \, (S = S+1)$, $k_1, k_2 \in \{-L, -L+1,..,0,..,L-1, L\}$
7. Test the noisy pixel count:
$If \, (remaining \, time < threshold \, 2) \, then$
$If \, (S < S_1 \, or \, L < L_{max}) \, then \, goto \, 5 \, else \, goto \, 8$
$else \, goto \, End \, Phase \, 2$
8. Compute median value and set current pixel in reconstructed image: $J(i,j) = median(\{\mathbf{W}^*\})$, $\mathbf{W}^* = \mathbf{W}|N(i+k_1, j+k_2) = 0$, $k_1, k_2 \in \{-L, -L+1,..,0,..,L-1, L\}$
End For
End Phase 2
```

the *GetSemaphoreValue* is called and the value returned is used as a local execution time. If this value is greater than thresholds discussed above, the proper actions will be taken (the processing parameters for *Phase 1* and *Phase 2* will be modified in order to deadline to be not exceeded). The following threads have been defined: *Main*, *P1* and *P2*. The *Main* thread creates *P1* thread and set the periodic semaphore then it is destroyed. The *P1* thread creates *P2* thread , read the periodic semaphore value and executes an iteration of *Phase 1* of algorithm. Before continuing with other iteration in this

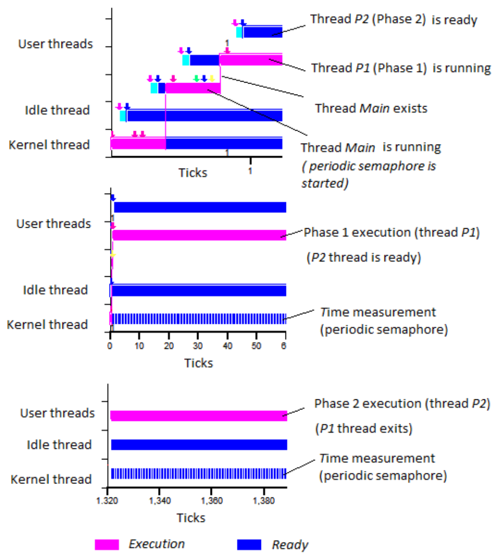

Fig. 4. Tasks execution and periodic semaphore in VDK

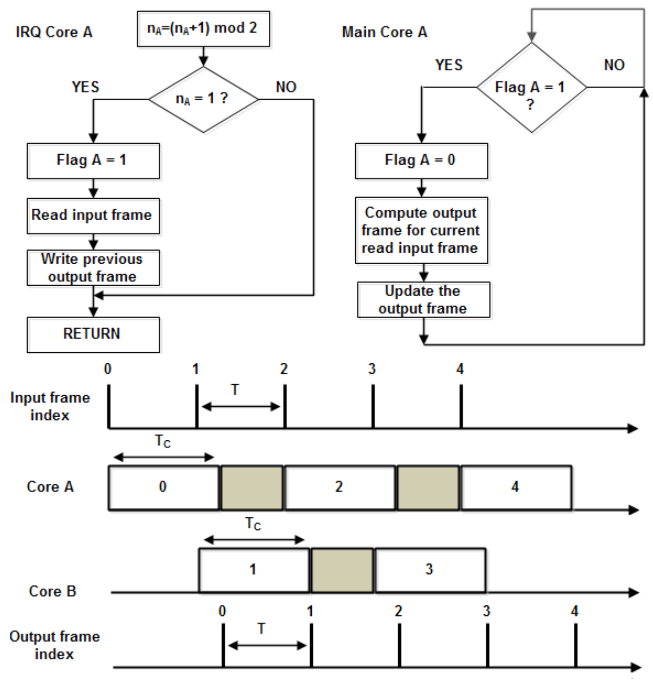

Fig. 5. The image processing flowchart

phase, *P1* thread read again the periodic semaphore value and calculates the time interval to deadline. If this interval is greater enough, a new iteration of *Phase 1* is running, else the *P1* task exits and *P2* task will be executed. The *P2* task performs similar operations as *P1* but it implements the *Phase 2* of the algorithm, instead *Phase 1* as *P1* task. An identical mechanism to measure the time interval to the deadline is used as in *P1* task. Figure 4 illustrates how periodic semaphore is involved in time measuring. The main issue here is to achieve a real time implementation. The proposed algorithm controls the execution of *Phase 1* and *Phase 2* to reduce the number of iterations of each of them with respect of the deadlines. If the frame processing is complex, (if the impulsive noise has high power or the frame size is large) then the QoS scheduling algorithm will reduce the execution time so the tasks can complete within their deadline. However, at least one iteration of each phase must be completed. In certain situation that may take a large execution time, therefore special methods to optimize the tasks execution should be involved. These methods will be discuss below for achieving a real time implementation using the digital signal processing Blackfin microcomputer family[3]. The Blackfin processor has a dual multiply and accumulate (MAC) signal processing engine, an orthogonal instruction set and single instruction multiply data (SIMD) instructions. Issuing parallel instructions and using vector operations may be used to obtain a real time functioning. The Blackfin processor does permit up to three instructions to be issued in parallel: one 32-bit DSP instruction and two 16-bit instructions (load/store, DSP load). A powerful feature of Blackfin processors is the existence of instructions that manipulate video pixels. Such instructions perform 8-bit pack and unpack, quad 8-bit subtract operation that can be used to compute minimum and maximum values in four windows simultaneously. Also, four values in the noise matrix**N**, are updated in parallel. Using assembly language implement the iterative instruction as a hardware loops that save processor cycles. Additionally, dual-core Blackfin processors, (BF561 and BF60x) may be involved. Each dual-core Blackfin processor has two cores, each with its own internal memory. There is a common memory shared between the two cores, and both cores share access to external memory. Each core functions independently. The frame processing task is designed as dual-core application that allows for splitting the main code on the two cores and for all of the shared memory areas to be used efficiently by both cores. Common routines and data will be placed in shared memory without the need for explicit positioning. Two successive input frames may be processed in the two cores of the processor, as illustrated in the figure 5. In figure 5, T is the frame rate and T_c represents the computation time for the current frame. Each core in the Blackfin processor has its own interrupts system. The input frames are acquired from a serial port that generates a common interrupt for both cores (indicated in the figure 5 as *IRQ_core_A* and *IRQ_core_B*). Each core implements the frame processing algorithm in its own main program (denoted as *Main_core_A* and *Main_core_B*) if an appropriate flag (*flag_A* or *flag_B*) is set to 1. These flags are set in the interrupt service routines for *core A* or *core B*. The main programs process the odd or the even input frames only. A necessarily functioning condition is $T_c < 2 \cdot T$. (the flowchart for *core B* is not shown, it is similar to flowchart of *core A*). Two counters, $n_A$and $n_B$were defined to determine the even and odd frames.

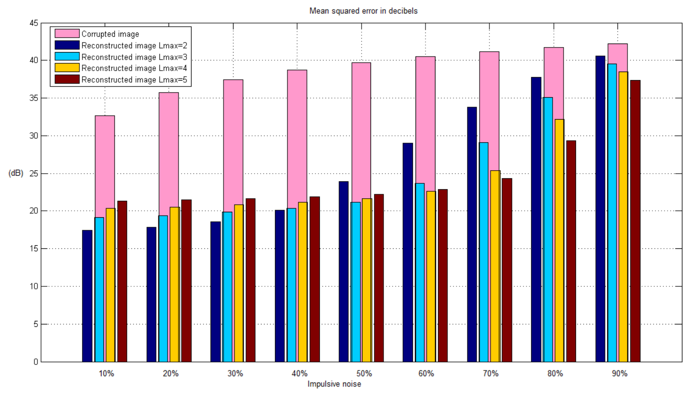

Fig. 6. The MSE of restored image

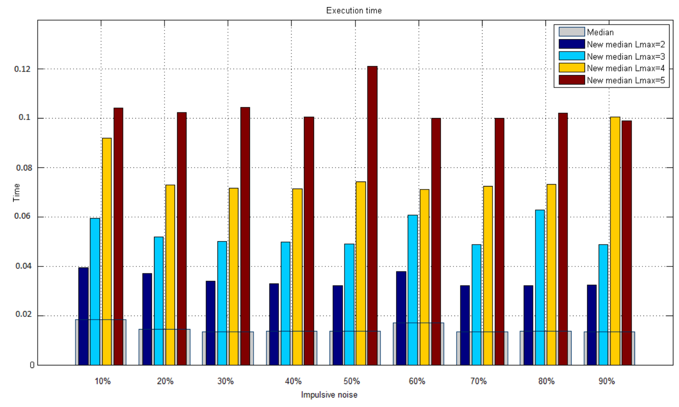

Fig. 7. The execution time (320 x 240 image size)

4. The Performance Evaluation

This section evaluates the performance achieved using the proposed scheduling algorithm. Noise removal algorithm with QoS scheduling was implemented using the Blackfin BF561 microcomputer and run with different values of impulsive noise and maximum size of the scanning window. The above presented techniques for code optimizing were involved. The maximum size of the window was considered because even if the noise has high power, the algorithm must end after a maximum number of iterations. In this way, the execution time will be calculated in the worst case.The image processing performance and the execution time were evaluated. The figures 6 and 7 illustrate the mean squared error $MSE = 10log_{10}\{\sum_{i=0}^{i=N-1}\sum_{j=0}^{j=M-1}[I(i,j) - J(i,j)]^2\}/(N \cdot M)$ of restored images and the execution time for various windows length and various noise powers. One can observe, from Figure 6, that for low levels of impulsive noise error does not vary significantly depending on the maximum size of the window $\mathbf{W}$. In contrast, if impulsive noise is relatively high (60% -70%), the error decreases relatively (up to 6 dB) if the window size increases. Execution time increases when impulsive noise is high because the number of iterations in *Phase 1* and *Phase 2* will increase. One can observe that the improvement is small if the number of iterations is increased over a specific limit. Figure 7 illustrates that the execution time is decreased very much but the noise reduction is very similar. The execution time is greater than for median filtering, but the noise removal is better for the new median filter. [8], [2] The results in Figure 7 support the idea that reducing the number of iterations in *Phase 1* and *Phase 2* leads to a small degradation in performance for noise removal algorithm but significantly reduce computation time which allows tasks to fulfill deadlines. Table 1 illustrates the average execution time, in frames per seconds, for various image sizes. In this table the maximum size of the scanning window, $L_{max}$is considered as parameter. The average time represents the arithmetic mean of the execution time for noise levels ranging from 10% to 90%. One can observe that the new median filtering can be used for color image size about 320×240, if the admitted frame per second is minimum 25. For this limit there are a possibility to obtain a real time functioning for the minimum scanning window.

TABLE I
EXECUTION TIME IN FPS FOR VARIOUS GRAY SCALE IMAGE SIZES

Image size	**FPS**		
	$L_{max} = 1$	$L_{max} = 2$	$L_{max} = 3$
176×144	266	95	48
320×240	87	31	16
480×320	43	15	8
640×480	21	7	4
960×540	13	4	2

5. Conclusion

This work proposed a scheduling algorithm for image processing tasks which have two computations steps both of them depending of noise power. Each phase is running for multiple times in order to efficiently eliminate the noise. The scheduling algorithm trades off between the quality of the restored image and the constraint to meet the deadlines. A real time implementation using digital signal processing Blackfin microcomputers from Analog Devices is possible for medium color image sizes. The paper presents a framework for real-time implementation of such image processing algorithms that ensure a reasonable quality of videoclips in systems prone to impulsive noise but meeting deadlines.

Acknowledgment

This work has benefited the support of FP7 project ALICANTE—FP7—ICT—2009—4 no. 248652.

References

[1] Manohar Annappa Koli , “Robust Algorithm for Impulse Noise Reduction ”, International Journal on Computer Science and Engineering (IJCSE), Vol. 02, No. 07, 2010, 2375-2377
[2] Sorin Zoican,” Adaptive algorithm for impulse noise suppression from still images and its real time implementation”, Telecommunication in Modern Satellite Cable and Broadcasting Services (TELSIKS 2011), 5-8 Oct. 2011 Nis, Serbia, pp. 337-340
[3] Analog Devices, Inc., Blackfin Processor Programming Reference, 2012
[4] Analog Devices, Inc., VisualDSP 5.0 Kernel (VDK) Users Guide , 2011

[5] Analog Devices, Inc.,VisualDSP++ 5.0 Blackfin C/C++ Compiler and Library Manual, 2011
[6] Real Time Systems, Architecture, Scheduling and Application, Seyed Morteza Babamir, editor, Ed. Intech 2012, ISBN 978-953-51-0510-7
[7] Embedded Systems and Wireless Technology, Raul Aquino Santos and Arthur Edwards Block, editors, CRC Press, 2012, ISBN 978-1-57808-803-4
[8] Zhou Wang and David Zhang, "Progressive Switching Median Filter for the Removal of Impulse Noise from Highly Corrupted Images", IEEE Transactions On Circuits And Systems—II: Analog And Digital Signal Processing, vol. 46, no. 1, Jan. 1999, pp. 78-80
[9] Woon-Seng Gan Sen M. Kuo, "Embedded Signal Processing with the Micro Signal Architecture", John Wiley & Sons, Inc, 2007, ISBN: 978-0-471-73841-1