Analysis of Digital Image Processing with Parallel with Overlap Segment Technique

Mohd.Iqbal
Information Technology
Samrat Ashok Technological Institute, Vidisha 464001
(M.P), India

Sandeep Raghuwanshi
Information technology,
Samrat Ashok Technological Institute Vidisha 464001
(M.P), India

Abstract
Image processing computing time is an issue for research from the beginning and different method has been used but no one gives the efficiently result with different method. Our research in this paper calculates the image processing computation time. Using parallel processing and overlap segment technique. It can reduce the computation time. Discussing in this paper divided image frame into a number of sections by overlap technique. Using DSP resources and keeping high accuracy and speed main target. It is avoiding a traditional segment technique. It uses an Overlap segment technique is best for sectioning and grouping. Overlap segmentation technique remove problem of filtering operation. This technique implemented with parallel processing for computation time and enhances output data. Parallel computing is very useful for performance evaluation and design time calculation as well as advantages of not using data bus as a common data bus transfer. This parallel design gives the advantages of applying different algorithm on image frame.

Keyword
DSP, FFT, Overlap technique, parallel processing

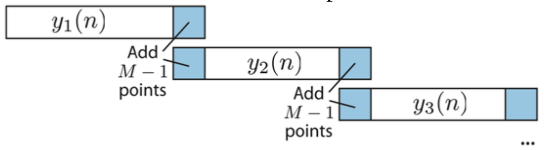
I. INTRODUCTION

Researchers main aim from long time ago was reducing processing time and increasing data computation speed, reaching real time or even real time still the main goal. Two different trends tried to achieve the same target; first method tried by increasing clocking frequency for the operating processor as maximum as possible, other trend took another methodology by using parallelism. Increasing clock frequency drawback were power consumption and high temperature, researchers eyes went again to parallel computation. Concurrent advantage of VLSI technology is allowing very large number of components to fit chip. Changing computing architecture for parallelism is one of the method for parallel processing technique.

Our contribution in this paper is: defining a loosed coupled parallel computing design method for fast image processing in which processing time will not exceed for image frame size. It can get computing time with different kernel size and using different filtering algorithm.

Studying parallel computation capabilities: Defining parallel computing as many calculations are carried out simultaneously each is a processing element considering in principle that large problems can often be divided into independent smaller ones and each processing element can execute its parts, which are then solved concurrently (in parallel). Also processing element can be diverse and included resource such as single computer with multiple processor, several networked computer specialized hardware, or any combination of the above. Concluding of the above in other words would say that, parallel computing is a “collection of processing elements that communicating and cooperate to solve large problems fast”

Studying overlap segment technique: The Overlap-Add method is based on the observation that when we consider two discrete-time signals, say xk(n) and h(n), with support L and support M, respectively (note: the support is the length of the smallest consecutive stretch of points that contains all non-zero signal elements), the resulting convolution yk(n) = xk(n) * h(n), has a support of L+M-1. For example, say the support for xk(n) is n = 0,…, L-1 and the support for h(n) is n = 0, …, M-1, then the support for yk(n) is at n = 0, …, L+M-2. The questions at the end of this lab will elucidate this concept.



Using this idea suppose our input stream x(n) is an infinite sequence starting at time n = 0. Divide x(n) into L-length blocks and convolve each L-block with h(n) (using linear convolution). Then sum all the convolution outcomes along the L-boundaries (we elaborate more soon). This works because of the additive property of convolution which states (x1(n) + x2(n)) * h(n) = x1(n) * h(n) + x2(n) * h(n), and is visually depicted in Figure 1.

Manuscript received June 5, 2014
Manuscript revised June 20, 2014

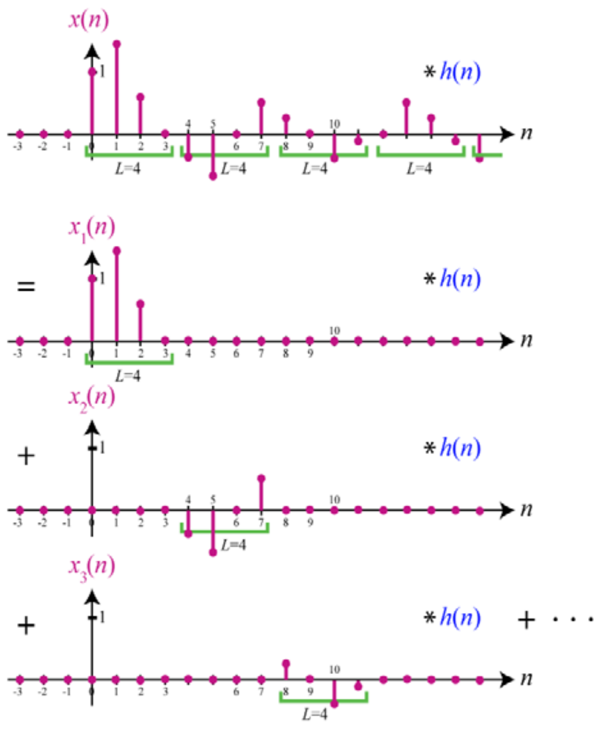
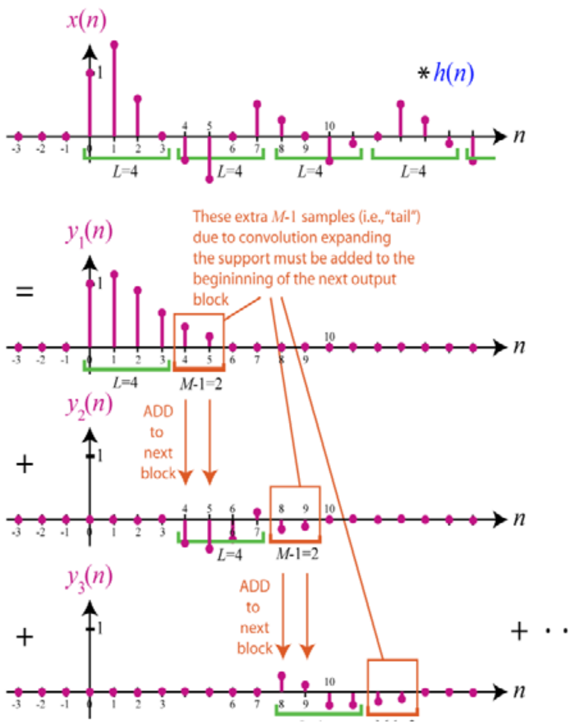


Figure 1: Basic Idea Behind Overlap Method. The kth L-block of x(n) is denoted xk(n).

In Figure 1, the operation of convolving a very long x(n) with h(n) is equivalent to the operation of convolving each L-block of x(n), denoted xk(n) for the kth block, with h(n) and then conducting addition judiciously to deal with the “tail” region from each block convolution as we discuss next. An important aspect is that after convolving each block with h(n), the resulting intermediate signal is L+M-1 samples in length as discussed earlier, and thus the extra M-1 samples at the end due to convolution expanding the support (called the “tail”) must be added to the first M-1 samples of the next convolved block. This is illustrated in Figure 2, where the right hand side (RHS) of the equality shows the result (graphically) after convolution. Specifically, the kth output block is given by: yk(n) = xk(n) * h(n) and the last M-1 samples of yk(n) must be added to the first M-1 samples of yk+1(n) to produce the appropriate output signal y(n) = x(n) * h(n).

One main challenge in a real-time processing scenario is that the timing of completing a block convolution needs to be appropriately synchronized with the overall output speed so that that tail region may be added to the next block at the right time. If the process of convolving each block is slower than outputting the samples of blocks already convolved, then the tail region will not have the opportunity to be added to the next block (because the next block is still in the process of going through convolution), resulting in the erroneous output of the samples 0 to M-2 and L to L+M-2 for each block. One way to deal with this is to slow down the rate of output, which may fail to meet the timing requirements for a given application. Another, more attractive approach, is to speed up the process of convolution. This can be achieved through the use of the FFT for convolution. A reminder of how to use the FFT algorithm to filter a block of input to perform convolution is summarized here (note: this is not the entire Overlap-Add algorithm as the output blocks must be combined at the end of this


x(n)
*h(n)
L=4
These extra M-1 samples (i.e.,"tail") due to convolution expanding the support must be added to the begininning of the next output block
y1(n)
=
M-1=2
y2(n)
ADD to next block
+
y3(n)
+ · ·
n

It is a fast convolution While implemented linear convolution in FIR filters, the input signal sequence x(n) is much longer than the impulse response sequence h(n) of a DSP system. Circular convolution can also be used to implement linear convolution by padding zeros. The output cannot be obtained until the entire input signal is received and hence there will be characteristic delays. Also, as the signal N1+N2-1 gets longer, implementation and the size of the memory needed become impractical. In order to eliminate These problems while performing filtering operation in the frequency domain, two signal segmentation methods, namely the overlap add and overlap save segments method can be used to perform fact convolution by sectioning and grouping the long input sequence into block or batches of samples and the final convolution output sequence can be obtained by combining the partial convolution result generated from each block.

Studying FFT Algorithm: There are a number of FFT algorithms that can be used for Implementation of the fast convolution, most of which have limitations on the sizes of image they can operate on. The particular FFT algorithm used for this implementation was a variation of the mixed-radix FF T transform algorithm [8]. This algorithm, while not the Fastest, is capable of processing a wide variety of different sized images. It operates by breaking the width and height of the images down into their prime factors to arrange the image into groups of pixels to operate on . The largest prime factor that can be used is 19. Any image which has a height or width with a prime factor greater than 19 or that is odd, must be increased in size until they are even with prime factors less than 19. If an image has to have its dimensions increased the existing data is written into the top left-hand corner of the area. This occurs when an image is increased to allow its dimensions to be accepted by the FFT, and to bring the structuring element image up to the same size as the image being used by the FFT algorithm. The mixed-radix FFT algorithm should in theory show approximately logarithmic behavior [4]. However we found that the operation of the algorithm to accommodate the different sizes of input image introduces a variation in computing time and overlap-save algorithms, that speed up computation time and allow the convolution of a large image with smaller kernel to be performed [2] without having to pad up the kernel. The two algorithms are similar and are performed in the spatial domain. The input image i subdivided into N segments (slices) of the kernel, the\convolution is evaluated for each segment, and then all the separate convolution results are recombined in the spatial domain. In between these two operations any convolution algorithm can be used. In this case, our fast convolution is used, employing the multiplication of two equal sized Fourier transforms in the frequency domain

Studying kernel size and overlap factor: Kernel size reduces number of input and operation calculation more than half of input and consequently number of operation needed. It was the start of our research to discuss the difference in pixel output when using different kernel size such as 3x3, 5x5, and different overlap factor

II. PARALLEL COMPUTING

The computing speeds up is strongly attached to data dependency, so understanding data dependencies is fundamental in implementing parallel algorithm. No program can run more quickly then the longest chain of dependent calculations, since calculation that depend upon prior calculations in the chain must be executed in order. However, most algorithms do not consist of just a long chain of dependent calculations; there are usually opportunities to execute independent calculation in parallel if it satisfies the following;

Let Pi and Pj be two program segments. Bernstein's conditions [14] describe when the two are independent and can be executed in parallel. For Pi, let Ii be all of the input variables and Oi the output variables, and likewise for Pj. P i and Pj are independent if they satisfy

- $I_j \cap O_i = \varnothing,$
- $I_i \cap O_j = \varnothing,$
- $O_i \cap O_j = \varnothing.$

Violation of the first condition introduces a flow dependency, corresponding to the first segment producing a result used by the second segment. The second condition represents an anti-dependency, when the second segment (Pj) produces a variable needed by the first segment (Pi). The third and final condition represents an output dependency: When two segments write to the same location, the result comes from the logically last executed segment.

III. PARALLEL PROCESSING ON DSP FOR IMAGE PROCESSING ALGORITHM

The parallel Reviewing parallelization classification and trying to find the suitable method for image processing as the various architectures can be used to solve vision problems, as the pipeline, Multiple Instruction and Multiple Data stream machine, Single Instruction stream-Multiple Data stream.

The parallel architectural for Image Processing, Image processing community has been relying for many years on special purpose computers to accelerate their computations; they start to turn towards parallel processing technology. They were searching, if they can get benefit simultaneously from reliable and fast hardware, and software programming tools that allow implementing more complex image and vision applications. The commercial success of any computer architecture significantly depends on the availability of tools that simplify software development. Consequently, many different programming tools have been developed that attempt to solve difficulties and obstacles facing software programming languages design for parallel and distributed computers. However, our research study uses the ability of connecting Matlab 2007a with FPGA software tool ISE 10.1 backage from XILINX.

The selected Digital Signal Processor TMS320C6414T-1000 is from Texas Instruments. It is a device from the C6000 family and one of the cutting-edge fixed point DSPs in this series. It runs with a clock of 1 GHz and provides 8000 million MAC (Multiply-Accumulate) operations per second. An example will show the performance improvement of partly hand optimized code. Some tests are carried out with the function which implements the Gaussian Pyramid. The digital filtering of an image requires the MAC operation on every pixel with a sliding scope over the image. The first unoptimized implementation of the Gaussian pyramid fits only the needs of the functionality, but without any performance aspects. The filter coefficients are derived from an array. All coefficients are multiplied with the corresponding pixels.

The products are summed up and divided by the sum of the coefficients derived from the array, Again the Gaussian Filter is a neighborhood function, which means that an ROI is used. During the test, the image.size of 256x256 pixels is used, where the ROI window has a size of 252x252 pixels, because a border of 2 pixels is required for the algorithm. This results in a destination image with the size of 126x126 pixels and obtains a performance of 86.11 ns/pixel. The functions PfeGetPix8u and PfeSetPix8u are a part of the PfeLib (Performance Primitives Library) [23]. Without using these two optimized functions to load and store pixels from the memory, the performance of the functional behavior (unoptimized) will be much worse. A common method to gain execution time performance is to inline the function. It means that the C/C++ source code for the called function is inserted at the place of the function call. This technique is useful for platforms with multi-staged-pipelines and especially for VLIW processors. Execution time on VLIW processors is improved, because function calls inside of loops would cause the optimizer to not parallelize loops.

we are investigating how we can make use of the advantages of both technologies to build a new platform which is based on both DSPs and FPGAs. That platform would enable us to split algorithms and to execute parts of the algorithm on the processing unit (DSP or FPGA) which is better suited for.

Studying peak to signal ratio: Peak Signal-to-Noise Ratio, often abbreviated PSNR, is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale. PSNR is most commonly used to measure the quality of reconstruction of lossy compression codecs (e.g., for image compression). The signal in this case is the original data, and the noise is the error introduced by compression. When comparing compression codecs, PSNR is an approximation to human perception of reconstruction quality. Although a higher PSNR generally indicates that the reconstruction is of higher quality, in some cases it may not. One has to be extremely careful with the range of validity of this metric; it is only conclusively valid when it is used to compare results from the same codec (or codec type) and same content

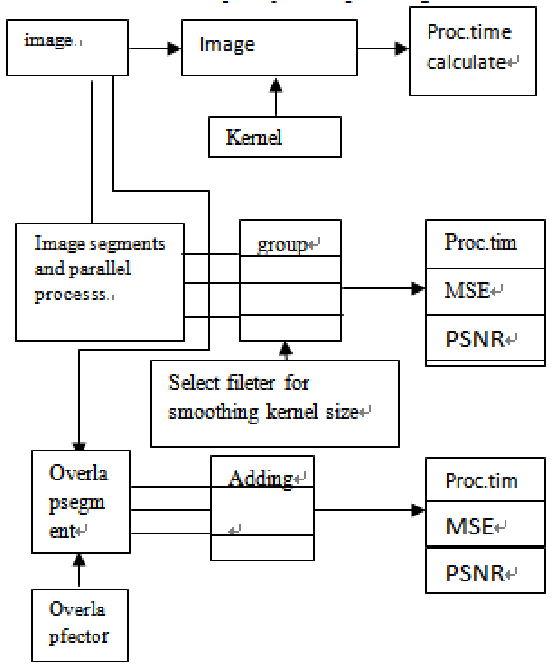
PSNR is most easily defined via the mean squared error (MSE). Given a noise-free m×n monochrome image I and its noisy approximation K, MSE is defined as

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \end{aligned}$$

Proposed computational model;

This computational model calculates the result of different kernel size and different filter .But it is more depend upon number of segments. If number of segments are more than number of parallel pipelines is increasing then processing time reduced. First step image processing without parallel processing calculate the time different filter size and different kernel size Then second step image processing with parallel then applies different filter and different kernel size. In second step also calculate other parameter as MSE it shows how much error occur in output image and PSNR calculate the quality of output image.



Then a third step. Image processing with parallel processing .But divided the image into a number of parts by the overlap segment method. It gets the segment by an overlap factor then proceeds. It is reduced processing time and MSE, PSNR .Then compare the result each other with different parameter. Shows in fig.

Experimental Results:

Different experiments were carried out to test the correctness and the timing of the algorithm. For the overlap segment technique with parallel processing . It can compare the result based on different kernel and different filter size

Filter type	Kernel size	Proc. time (msec)
Gaussian filter	3x3	0.59017
Median filter	3x3	0.17763
Average filter	3x3	0.02652
Motion filter	3x3	0.1015

Sequential computation Table.1

Number of segment	Filter type	Kernel size	Process time (msec)	Mse	Psnr
3	Gaussian filter	3x3	0.006390	8.884	37.8
3	Median	3x3	0.003927	6.005	39.5
3	Average	3x3	0.002948	4.818	40.4
3	motion	3x3	0.002998	11.79	36.5

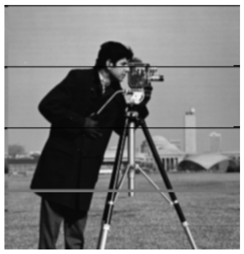
Parallel computing Table.2

Number of segment	Overlap fector	Processing time(msec)	mse	Psnr
3	4	0.0014504	0	Inf
4	5	0.0050847	0	Inf

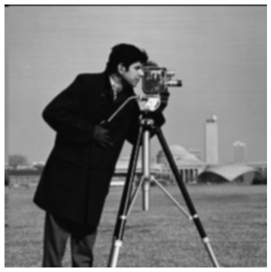
Parallel processing with overlap segment table.3

Original image

side effect image



Result



Conclusion

In this work a performance analysis of image processing with parallel and overlap segment technique it is implemented on DSP system. The advantages of overlap segment technique. It is eliminate the problem of filtering operation It is gives the better result is compare to tradition segment method. It can analysis the result by some parameter such as filter type, segment size, and overlap factor, MSE(Mean square error), PSNR(Peak to signal noise ratio) these parameter decided the image processing computing time and quality of image. In this approach parallel processing concept is a very important because it is reducing the process time of image processing. Analysis the segment size of image. it can identify the results by comparative model which parameter gives the efficient results for image processing. In this model comparing with Gaussian filter, median filter, average filter motion filter Overlap segment technique faster is compare to traditional segment technique different size segment and different window size for processing. The selected Digital Signal Processor TMS320C6414T- 1000 [18] is from Texas Instruments.

References

[1] M. Razaz and D.M.P. Hagyard “efficient convolution based algorithms for erosion and dilation school of information systems University of East Anglia Norwich England

[2] Shiping Zhu. An Image Segmentation Algorithm in Image Processing Based on Threshold Segmentation .SITIS, 07 Third International IEEE, Conferences

[3] R.C.Gonzalez and R.E Wood. Digital image Processing, Second Edition. Pearson Education International, 2002

[4] van Herk, M., “A fast algorithm for local minimum and maximum filters on rectangular and octagonal kernels”, Pattern Recognition Letters, Vol. 13, pp 517-521, 1992.

[5] B. Kisacanin, C. Schonfeld, “A Fast Thresholded Linear Convolution Representation of Morphological Operations”, IEEE Transactions on Image Processing, Vol. 3, No. 4, Pages 455-457, 1994.

[6]] I. S. Koc. Design considerations for real-time systems with dsp and risc architectures. In Proceedings of the EU-SIPCO2005 13th European Signal Processing Conference, 2005.

[7] D. Baumgratner, P.Rossler, W.Kubinger, Performance Benchmark of DSP and FPGA Implementation of low level vision algorithms, IEEE, 2007

[8] D.M.P. Hagyard M. Razaz and P.Atkin, “A Fast algorithm for computing morphological image processing primitive “, proc. IEEE Nonlinear signal & image processing ,sept. 1997

[9] R. C. Gonzalez and R. E. Woods. Digital Image Processing, Second Edition. Pearson Education International, 2002

[10] D. Baumgartner, P. Rossler, W. Kubinger, Perferomance Benchmark of DSP and FPGA Implementations of low level vision algorithms, IEEE, 2007

[11] J.A. Kalomiros, J. Lygouras, Design and evaluation of a hardware/ software FPGA based system for fast image processing, Microprocessors and Microsystems, P. 95-106, 2008.
[12] Sima, D. et al. (1997) Advanced Computer Architectures: A Design Space Approach. Reading, MA/New York: Addison-Wesley/Longman.
[13] Batcher, K.E. (1987) Design of a massively parallel processor. IEEE Transactions on Computers C-29: 1523–1538.
[14] Duff, M.J. et al. (1988) Review of the CLIP image processing system. Proceedings of the National Computer Conference, pp. 1055–1060.
[15] Kung, S.Y. et al. (1988) VLSI Array Processor. Englewood Cliffs, NJ: Prentice-Hall.
[16] 5. Maresca, M. et al. (1990) Connection autonomy in SIMD computers: a VLSI implementation. Journal of Parallel and Distributed Computing 7: 302–320.
[17] Oldfield, D.E. et al. (1985) An image understanding performance study on the ICL distributed array processor. Proceedings of the IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database, pp. 264–265.
[18] Loughead, R.M. et al. (1980) The cytocomputer: a practical pipeline image processor. Proceedings of the 7thAnnual International Symposium on Computer Architecture, pp. 271–277.
[19] Kent, E.W. et al. (1985) PIPE: Pipeline Image Processing Engine. Journal of Parallel and Distributed Computing 2:50–78.
[20] Me´ rigot, A. et al. (1991) Architectures massivement paralle` les pour la vision artificielle. Annales des Telecommunications 46: 78–89.
[21] Uhr, L. (1987) Parallel Computer Vision. Boston, MA: Academic Press.
[22] Siegel, H.J. et al. (1981) PASM: A partitionable SIMD/ MIMD system for image processing and pattern recognition. IEEE Transactions on Computers C-30: 934–947.
[23] Annaratone, M. et al. (1987) The Warp computer: architecture, implementation and performance. IEEE Transactions on Computer C-29: 1523–1538.
[24] Borkar, S. et al. (1988) iWarp: an integrated solution to high-speed parallel computing. Proceedings of Supercomputing' 88, pp. 330–339.
[25] Hillis, W.D. (1985) The Connection Machine. Cambridge, MA: MIT Press.
[26] Blank, T. (1990) The MasPar MP-1 architecture. Proceedings of IEEE Compcon Spring, pp. 20–24.
[27] Raman, S. & Clarkson, T. (1990) Parallel image processing system – a modular architecture using dedicated image processing modules and a graphics processor. IEEE, Conference on Computer and Communication Systems, September 1990, Hong Kong, pp. 319–323.