High performance Computing Algorithm Applied in Floyd Steinberg Dithering

Anuja Dagar
Asst. Professor
Gurgaon Institute of Technology and Managment

Archana
Asst. Professor
Gurgaon Institute of Technology and Managment

Deepak Nandal
Asst. Professor
PDM P.D.M College Of Engg, Bahadurgarh

ABSTRACT
In this paper, we are going to implement parallel Floyd Steinberg Dithering algorithm for multi core architecture. The algorithm is based on error dispersion. This algorithm is commonly used by image manipulation software, for example when an image is converted into GIF format. This technique generates the best results of any classical method, but it is the slowest because of its sequential computation. This paper brings out FSD algorithm for distributed (multi core) architecture. PFSD (parallel Floyd Steinberg Dithering) algorithm based on master slave architecture. Master collects data information and distributes data to multiple sections whereas slave works on sections which are monitored by master. The proposed algorithm can be extended to different data intensive and complex computing applications for multi core architecture. We expect this research will be very helpful in the field of mobile communication devices as multi core architecture has been introduced in mobile communication devices.

Keywords
Floyd Steinberg Dithering (FSD), PFSD, HPC, Multi core architecture, Bluetooth, Error Diffusion, Image Halftoning Technique

1. INTRODUCTION
The human eyes can perceive a large range of continuous tones, while most of the current print devices are limited to a few tones. To produce images with continuous tones, half toning technology is widely used in the field of printing. The most commonly-used half-toning methods include Ordered Dithering and Error Diffusion [1]. Grayscale images contain at the most 256 shades of gray. The human eye can discern about two dozen or so shades of gray [3]. Dithering is a means of reducing the gray level range of images from 256 to a minimum of two shades. This technique arranges black and white pixels in differing quantities and spatial orientation to approximate intensity images [2]. By Juxtaposing pixels of two shades create an illusion of third shade. There are many different type of Dithering algorithm is present and most are based on error diffusion. Error Diffusion deals with the pixels line by line, quantizing each one to the nearest available dot tone and spreading the resulting error to unvisited neighbor pixels. Less visual artifacts as it generates, it has to deal with large amount of data, also it is hard to be parallelized and pipelining [1]. The best known example of error diffusion dither is the Floyd-Steinberg dither. This was originally introduced in 1970. The Floyd-Steinberg algorithm schematic has been shown in Figure 1**.**

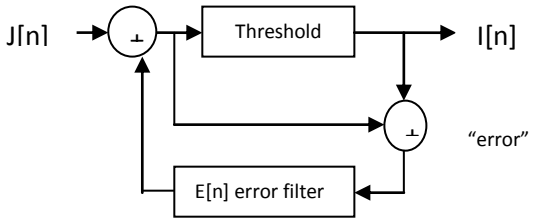

Fig.1. FSDA schematic

The algorithm scans the image from left to right, top to bottom, quantizing pixel values one by one. Each time the quantization error is transferred to the neighboring pixels, while not affecting the pixels that already have been quantized. Hence, if a number of pixels have been rounded downwards, it becomes more likely that the next pixel is rounded upwards, such that on average, the quantization error is close to zero.

$$\frac{1}{48}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & F & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{pmatrix}$$

Fig.2. quant error coefficient matrix

Where F represents the current pixel being scanned the weights represent the proportion of the error being distributed to the pixel in that position. The pixels to the right of F get 7/48 of the error, the pixel right of it gets 5/48 and so on. Like the human ear, the human eye is less sensitive to high frequencies [5] and since calculated error is concentrated in the high frequencies, the changes in the high frequency content of the image is readily visible. There are several different methods of dispersing such as ordered dithering. The Floyd-Steinberg dither usually generates the best results.

FSD is a very time consuming process. In fact, for a single pixel of image it needs 13 floating point operations and 13 memory access operations. Hence for a n.m image size it needs 26n.m operations which make it computationally costly process.

2. RELATED WORK
Ping Wah worked on Error diffusion is a procedure for generating high quality bilevel images from continuous-tone images so that both the continuous and halftone images appear similar when observed from a distance[11]. It is well known that certain objectionable patterning artifacts can occur in error-diffused images adjusting the error-diffusion filter concurrently with the error-diffusion process so that an error

criterion is minimized. The minimization is performed using the least mean squares (LMS) algorithm in adaptive signal processing. Floyd and Steinberg's worked on error diffusion technique is a well-known approach to digital halftoning. The main drawback of this technique is that it is inherently serial. This paper presents a new parallelizable error-diffusion algorithm, called*line diffusion*. In this method, the pixels of the original image are divided into classes line by line, and all the pixels on a line are halftoned simultaneously.

Yuefeng Zhang worked on error diffusion scheme for digital halftoning is proposed. The scheme is an iterative and isotropic error feedback process. Also, to compensate for the nonlinear tone reproduction by laser printers due to ink drop overlaps, we incorporate Pappas-Neuhoff's simplified printer model into the new error diffusion scheme. It appears that the proposed halftoning technique enhances the perceived tone continuity of halftone hard copies, and it alleviates the objectionable structured halftone textures of some existing methods.

3. PARALLEL ALGORITHM

PFSD algorithm is based on master slave and data parallelism (SIMD). Fig.3 depicts the design for Parallel FSD algorithm. FSD algorithm is a computational and data intensive algorithm. We observed that lot of time is consumed in reading and writing the file. PFSDA has been designed by considering all aspects including file writing and reading. PFSDA has been divided into three main blocks, which are processing block, File reading block and File writing block. PFSDA major components:

A. Controller Thread.
B. Processing Block.
C. File Reading Block.
D. File writing Block.

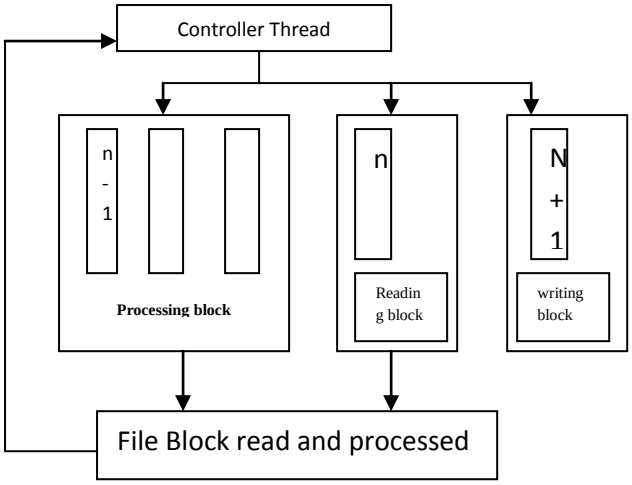

Fig.3. PFSDA for Sequential File System

3.1 Controller Thread

Controller/Master thread collects information about file to be processed and take decision to divide work among different blocks. Decision depends on file size and available processing units (Core). Controller maintains N buffers for processing. At a given time all block will be active, if file reading block is working on N^{th} chunk then processing block will be working on the $(N-1)^{th}$ chunk and writing block will be active on the $(N-2)^{th}$ chunk.

Code snippet:

```
#pragma omp parallel
        default(shared) #pragma
        omp sections
        loadImage(File*,matrix[n])
        ; //end section
        #pragma omp
        section
        processBMP(matrix[
        n-1]); //end section
//end omp parallel
```

3.2 Processing Block

It is the core block where PFSD algorithm has been implemented and processed. Input for processing block is taken from file reading block. One chunk is read by file reading block and given to process block. Process block can itself spawn multiple treads to perform task faster.

#pragma omp parallel

$$\text{For each } i = \left[\frac{height}{total\_num\_threads}\right].thread\_id \text{ to } \left[\frac{height}{total\_num\_threads}\right].thread\_id + 1$$

```
oldpixel := pixel[x][y]
newpixel :=
find_closest_palette_color(oldpixel)
pixel[x][y] := newpixel
quant_error := oldpixel - newpixel
pixel[x+1][y] := pixel[x+1][y] + 7/16 * quant_error
pixel[x-1][y+1] := pixel[x-1][y+1] + 3/16 * quant_error
pixel[x][y+1] := pixel[x][y+1] + 5/16 * quant_error
pixel[x+1][y+1] := pixel[x+1][y+1] + 1/16 *
quant_error
done
```

3.3 File Reading Block

We have considered file system as sequential file system so one thread has been dedicated to read the file in multiple chunks. We achieved parallel execution in file reading by dividing file in chunks (chunk= total file size /number of threads (chunk>50MB)). These multiple chunks are passed to processing block one by one.

3.4 File Writing Block

File writing block takes inputs from processing block. One separate thread is dedicated for writing processed data into file. Execution takes place in parallel to other blocks.

4. RESULTS AND SIMULATION

Fig.4. Original image

Fig.5. Dithered Image

Serial computation vs. parallel computation

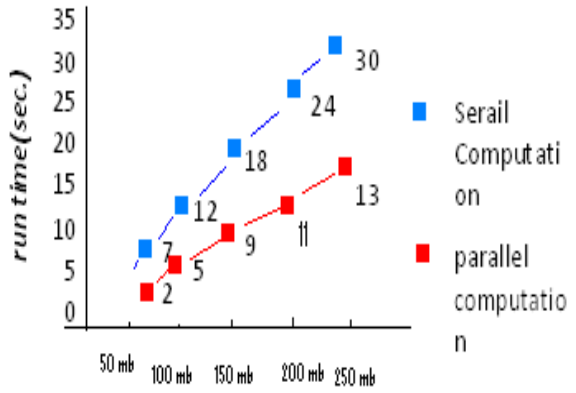

Fig.6. comparison graph of simulation time of serial and parallel computation

Data (MB)	Serial Computation(sec)	Parallel Computation(sec)
50	7	2
100	12	5
150	18	9
200	24	11
250	30	13

Fig.7. comparison table of simulation time of serial and parallel computation

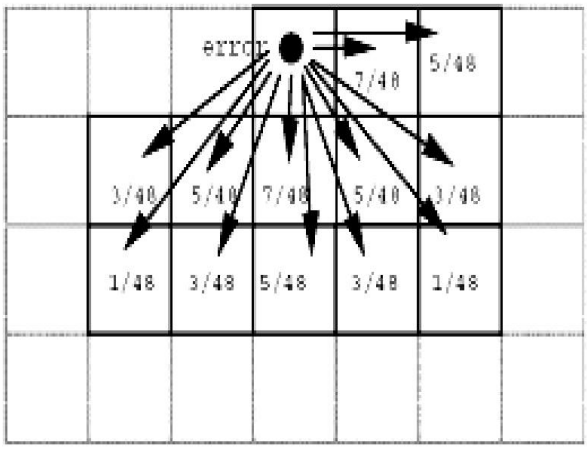

Fig 8. Error diffusion matrix

5. CONCLUSION

In this paper, we have implemented FSD algorithm by serial and parallel techniques. Our practical data is based on 24 bit image on 64 bit architecture. For parallelization open MP directives has been used. We achieved a significant enhancement in the performance of FSD computation. For further research SSE, IPP can be used to increase the performance. It help in speedup the parallel processing.

6. REFERENCES

[1] R. W. Floyd, and L. Steinberg, "An adaptive algorithm for spatial grayscale," in Proceedings for the Society for Information Display, Vol. 17, pp. 75-77, 1976.

[2] J. F. Jarvis, C. N. Judice, and W. H. Ninke, "A survey of techniques for the display of continuous tone pictures on bilevel displays," Comput. Graphics and Image Processing, pp. 13–40, 1976.

[3] J. W. Ahn, and W. Sung, "Multimedia processor-based implementation of an error-diffusion halftoning algorithm exploiting subword parallelism," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 16, pp. 129-138, 2001.

[4] William K. Pratt, Digital image processing, John Wiley & Sons, Inc., New York, NY, 1978

[5] G. O. Young, "Synthetic structure of industrial plastics

(Book style with paper title and editor)," inPlastics, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

[6] Y. Zhang, "Line diffusion: a parallel error diffusion algorithm for digital halftoning," The Visual Computer, Vol. 12, pp. 40-46, 1996.

[7]M. St¨urmer, G. Wellein, G. Hager, H. K¨ostler, U. R¨ude: Challenges and potentials of emerging multicore architectures.

[8] In: S. Wagner et al. (Eds.), High Performance Computing in Science and Engineering, Garching/Munich 2007, 551– 566, Springer (2009).

[9] B. M. Chapman, L. Huang, H. Jin, G. Jost, B. R. deSupinski: Toward Enhancing OpenMP's Work-Sharing Directives. In W. E. Nagel et al. (Eds.): Proceedings of Euro-Par 2006, LNCS 4128, 645–654. Springer (2006).

[10] OpenMP ARB, "OpenMP APP', www.openmp.org

[11] Ping Wah Wong in Image Processing, IEEE Transactions 1996 page 1184 - 1196 Volume: 5, Issue: 7

[12] Xiaolin Wu in Digital halftoning by iterative isotropic error feedback The Visual Computer Volume 11, Number 2 (1994), 69-81, DOI: 10.1007/BF01889977

[13] Yuefeng Zhang in Line diffusion: a parallel error diffusion algorithm for digital halftoning in journal The Visual Computer Pages 40-46(1996)

[14] Ulichney, R.,Dithering with blue noise," Proceedings of the IEEE 76(1), 56{79 (1998).

[15] "General-purpose computation using graphics hardware." http://www.gpgpu.org.