H. A. Cohen, *Architecture and Algorithms for Effective Utilisation of DSP Chips in Image Processing,* Proceedings of IREE (Australia) Conference, "Image Processing and the Impact of New Technologies" Canberra, Dec 18-20, 1989

Architecture and Algorithms for Effective Utilisation of DSP Chips in image Processing

H.A Cohen
Computer Science Department
La Trobe University, Bundoora Vic

Abstract

The effective use of a (single) Digital Signal Processor (DSP) chip as an image processing accelerator for a conventional microcomputer such an AT is critically examined. The focus is on non-linear and "adaptive" window based image operations, for which two distinct algorithms, termed here "simple" and "recursive" are possible. A careful complexity analysis is performed for the two classes of algorithm, as applied to two dimensional rank filtering, and also to a contrast limited variety of rank filtering also known as clipped local area histogram equalization. The analysis is intended as a prototype for like analyses to determine optimum implementation of other image processing operations.

These considerations, and other features of an available DSP were applied to a DSP accelerator for the AT which has been constructed in-house.

Introduction

Contemporary digital signal processing (DSP) chips can perform 32 bit multiplication in 50 ns, and can operate with minimal (or even zero) branching overhead, far exceeding the capabilities of todays general purpose microprocessors and their mathematical coprocessors. Such DSP chips also feature modest but significant in-chip RAM. In image processing and computer vision, a major portion of the computational task involves simple algorithms applied to images of upwards of 512*512 pixels. In hyperspectral image analysis similar operations are performed but on much larger images of multidimensional data where each pixel is a byte array. There is clearly real scope for the application of DSP's as image processing accelerators for otherwise conventional microcomputers such as AT's or 68020 based micros.

This paper is concerned with window based image processing operations wherein a rectangular window traverses the entire image and pixel values within that window mathematically determine a required intermediate result. A major category of window based image processing operations is that of image transformations including both linear convolutions and non-linear and adaptive operations: in image transformations the pixel values within a window determine a transformed value for the central pixel. Certain image transformations are decomposable into the product of 1-dimensional operations, but in general this is not possible.

The focus of this paper is explain a strategy of effective program design to utilise a DSP in conjunction with a slower processor to perform window based image processing operations. It is assumed that the slower processor has access to the entire image, and "feeds" the required pixels to a high speed cache on which the DSP operates. The architecture involved is shown in figure 2.

It is clear that because of the disparity in processor speeds, during each subsequent window cycle the master CPU should only "update" the copy of the window in the cache, by supplying new row or column consequent on window movement.

In considering how the DSP should operate in such an environment it is clear that there are two distinct possibilities:

(1) **The simple window algorithm:**
A complete copy of the window is maintained in the cache, and during each window cycle the DSP accesses the entire window.
For each window cycle the master CPU transmits the update pixels, while the DSP maintains the window through pointer movement or actual data movement.

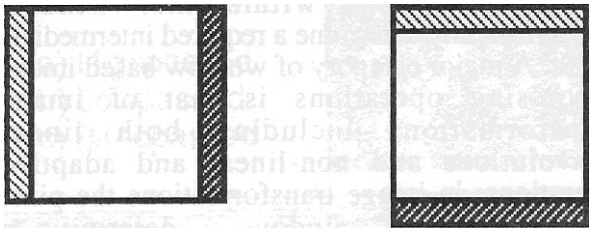
(2) Recursive window algorithm:

During each window cycle the DSP uses update data ("new" and "old" pixels plus movement vector) in conjunction with previously derived result(s).

For each window cycle the master CPU transmits the update pixels, which are used by the **DSP.**

With regard to a recursive window algorithm, there must be some initial processing step during which the master CPU transmits to the cache a full window of image data.

The distinction between the two algorithms is highlighted in figure 1.



(a) **(b)**

Fig 1: As the window slides over the image, either horizontally as in (a), or vertically as in (b) a new column (or row) enters the window. In **simple** window algorithms, the DSP requires to access a complete copy of window each "window **cycle". In recursive** window algorithms, the DSP requires only the "old" and "new" pixels plus a movement vector to perform a computation.

In this paper a careful complexity analysis is performed for the two classes of algorithm, in the context of two dimensional rank filtering. We also analyse a contrast limited variety of rank filtering introduced by Pizer et al [1] [2] [3] known as clipped local area histogram equalization. The analysis is intended as a prototype for like analyses to determine optimum implementation of other image processing operations.

Finally an in-house **DSP** accelerator for the AT is described.

Rank Filtering

In rank filtering the rank(q,W) of the pixel of gray scale q at (i,j) image location, for window W, or more strictly Wii centred about (i,j) is determined. The output gray scale p is then proportional to this rank:

$$q \dashrightarrow p = \frac{\text{rank}(q, W_{ij})}{\text{rank}(q_{max} W_{ij})} \; p_{max}$$

This mapping may for algorithmic purposes be expressed in terms of the histogram functions $h_w(q)$ and the cumulative histogram $H_w(q)$ where both histograms are defined within the window W centred on pixel (ij). Using these quantities, one may equivalently specify local rank filtering by

$$q \dashrightarrow p = \frac{H_W(q)}{H_W(q_{max})} \; p_{max}$$

Thus there are two equivalent algorithms for rank filtering:

(1) simple: comparision count
(2) recursive: histogram update

Both algorithms are described in detail below, where the complexity is calculated, omitting minor overheads, including branching costs, using access times:

S = read or write time master processor to cache
D = fetch or write time for DSP to cache
F = fetch or write time for DSP to on-chip memory
I = increment/decrement DSP register
C = time for comparision plus conditional counter increment

The discussion is in terms of a 16*16 window, of 256 pixels, for which the output gray scale, for 8 bit data, equals the rank.

Simple Algorithm: COMPARISION COUNT

The simplest algorithm, in which a full copy of the window is loaded by the master CPU into the cache. The DSP accesses every pixel within the window, and comparing each such pixel with the "central" pixel to determine its rank.. For a window of 256 pixels, this involves 256 cache reads, with 255 comparisions.

In each window cycle the master CPU must perform 16 cache writes to update the window for the next pixel. This window updating might also require some DSP activity, but details are very hardware dependent. In sum then,

Simple Algorithm for Rank Filtering:
Complexity: 256D +255C.
(ignoring window update costs)

Recursive Agorithm: HISTOGRAM UPDATE

When a 16* 16 window slides just one pixel, the histogram needs to altered by reading the 16 pixels to be dropped (16 accesses to cache, 32 to histogram) and then reading the 16 pixels to be added to the histogram (another 16 cache, 32 histogram accesses).

In this recursive algorithm, the DSP maintains a histogram on the in-chip DSP RAM. When a

16*16 window slides just one pixel, the histogram needs to altered by reading the 16 pixels to be dropped (16 accesses to cache, 32 to histogram), and then reading the 16 pixels to be added to the histogram (another 16 cache, 32 histogram accesses). Hence the number of DSP accesses to produce the histogram for the next position by this recursive strategy is just 32 to cache, 64 to histogram. Note that when a histogram table entry is read into the DSP it will be incremented or decremented at a cost of I before being written back to the table. From the histogram, the rank may be computed using, on average, 128 accesses to histogram entries, and performing a count operation that has a cost H, where (most DSP's) H < 128C. Summing these costs yields:

Recursive algorithm for rank filtering:
Complexity = 321+ 32D + 192F + H
(ignoring window update costs)
and where (conservatively) H = 128F + 128I

It becomes clear on examining the above calculation of complexity that because the recursive algorithm makes so much less reads to the cache than the simple algorithm, the magnitude of D is less critical. In other words if the recursive algorithm is used it is not vital to use the highest possible speed RAM (for the cache).

Contrast Limited Rank Filtering

The rank filter is also known as local area histogram equalization, and even as adaptive histogram equalization. The disadvantage of simple rank filtering is that where there are large number of pixels of the same gray scale, after such processing there may be some excessive contrast jumps, leading to speckle and also to misleading contouring.

Pizer and co-workers in a series of papers, [1][2] and references in [3] has posited contrast limited variants of this transform as the transform of first choice for medical images. Recently Cohen and Suter [3] have reached similar conclusions with regard to S.E.M. (scanning electron microscope) images of biological interest.

To perform clipped a.h.e. Inherently requires the production of a cumulative histogram for the window involved. Thus, for timing purposes, the critical portion of the simple and recursive algorithms for contrast limited rank filtering is the computation of histogram.

Histogram Computation - Simple Algorithm:

The algorithm for a 16*16 window is to first purge the histogram table by writing zero throughout:

256* {index write histogram elt = 1D
increment table pointer = 1 I }

followed by a direct creation of table entries:

256* {read window byte = 1D
index read histogram elt = 1F
increment histogram eh = lI
index write histogram elt = 1F
}

Histogram - Simple Algorithm::
Complexity = 512D + 512I +512F

Histogram Calculation - Recursive Algorithm:

Assuming a histogram as a byte table is held in DSP on-chip RAM, and already holds the "old" histogram, then following a window movement 16 "old" pixel values have to be removed from table, and 16 "new" values added:

32*{ Read cache byte	ID
Index read histogram	1F
Decrement/Increment	1I
Index write histogram	1F}

Histogram - Recursive algorithm:
Complexity = 32D + 32I + 64F

DSP Accelerator Boards

Several commercial DSP boards for the AT are available, including boards from Techtronix and Burr and Burr. At La Trobe a board has been constructed to serve both the purpose of real-time fractal generation, and the requirements of image processing. The DSP chip used is the Texas Instruments 320C30G. A modest 128*8 bit FIFO located on an AT Port and on the DSP's 32 bit is used for communicating data between the two processors. This scheme was adopted for simplicity, despite significant overheads, as the DSP chip must spend time moving each data item from the queue to ordinary RAM for convenient processing. The boot program for the DSP is a 35 ns EPROM, based on TMS 27C292 chip. The RAM which constitutes the cache is 2K*32 bit word with 55 ns access - leading to a single wait state for DSP access. However further fast RAM for the DSP is available on-chip - and this is sufficient for many image processing purposes.

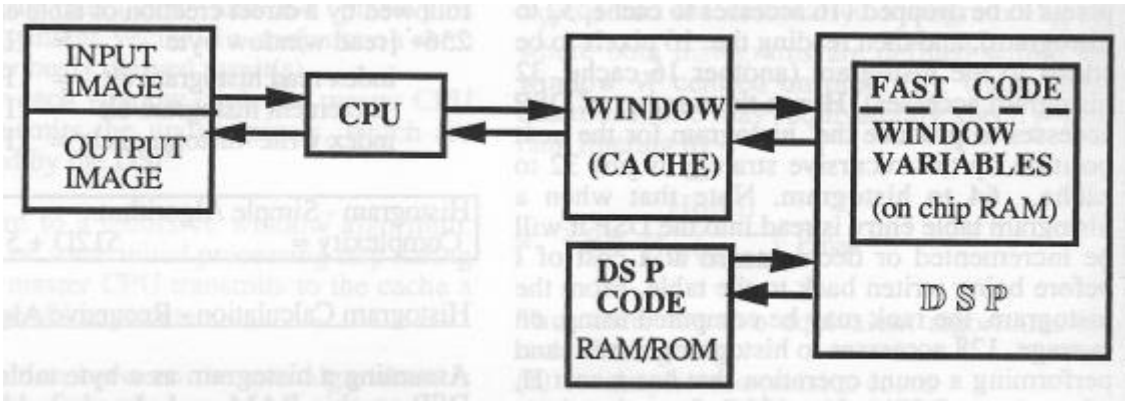


Figure 2. Basic architecture for use of a DSP as an image processing accelerator for a supermicrocomputer. Communication between the master microprocessor and the DSP is via a fast RAM cache, which holds a window of the image.

Discussion

In discussing the utilisation of DSP's in image processing a complexity analysis has been presented directed at non-linear window based operations. This has been done as such operations are both computationally truly expensive and representative of image processing needs. A measure of the power of a machine for image processing - or at least image processing research - must lie in its capability to perform such operations.

A quite different perspective on what are the truly significant image processing operations that merit inclusion for instance in a benchmark may be found in a recent paper by Preston [5]. Preston states that "Conventional benchmarks don't suit image processing computers", an observation which this writer agrees. However, the benchmark, "The Abingdon Cross", that Preston proposed consists of a sequence of near trivial operations. Thus the benchmark includes median filtering, a much simpler transformation than the simplest version of rank filtering discussed in this paper, and only in the form of median transform over a 17-pixel horizontal run, and also over a 17-pixel vertical run, but not over a rectangular window.

The results presented here are in the form of complexity costs. Surprisingly very few such analyses have been published, and none are reviewed in the classic texts [6] [7118]. The particular results presented are directly relevant to DSP accelerators, although the analysis can be interpreted in terms of algorithms on single processor systems. What has not been done in a significant way is an analysis on networks, such as transputer networks [4]. As image processing advances into the era of truly powerful and certainly affordable workstations further such analyses are warranted.

Acknowledgements

The support of Texas Instruments Incorporated through the award of a T.I. Technology Award is acknowledged.

References

[1] S.M. Pizer, J.B. Zimmerman, and R.E. Johnson, "Concepts of the display of medical images" IEEE Trans of NS-29 pp 561-580, 1982.

[2] S.M. Pizer et al, "Adaptive Histogram Equalization and Its Variations", Computer Graph. Vision Im. Proc., Vol. 39 pp. 355-368 , 1987.

[3] H.A.Cohen and D. Suter, "Adaptive Enhancement of Perceived Contrast in Diffuse Images: Case Study: S.E.M. Electron Microscope Images" to appear in Proceedings IEEE International Conference on Image processing, ICIP'89, Singapore, 5-8 September, 1989.

[4] D. Suter, X. Deng, Harvey A. Cohen, and T. Dillon, "Development and Implementation of Parallel Vision Algorithms", Proceedings SME VISION'89, Section 13, pp 1-14 (1989)"

[5] K. Preston, "The Abingdon Cross Survey" IEEE Computer July 1989 pp 9-18.

[6] K. Castleman, "Digital Image Processing" Prentice-Hall, New Jersey 1979.

[7] R.C. Gonzalez and P. Wintz, Digital Image Processing", Addison-Wesley, Reading, Mass 1979.

[8] W.K. Pratt, "Digital Image Processing" Wiley, New York, 1976