Accuracy and Efficiency of Grey-level Image Filtering on VLSI Cellular Processor Arrays

Piotr Dudek

Department of Electrical Engineering and Electronics
University of Manchester Institute of Science and Technology (UMIST)
PO Box 88, Manchester M60 1QD, United Kingdom

ABSTRACT: *This paper discusses issues related to the efficiency of silicon implementations of cellular processor arrays executing basic grey-level image processing operations - linear convolutions with 3×3 kernels. Speed, accuracy, power consumption and circuit area are considered. It is demonstrated, that the sequential execution, characteristic of SIMD machines, may offer certain advantages over the parallel one, characteristic of CNN-based processors. The discussion is illustrated using processing results from an analogue cellular processor array chip.*

1. Introduction

The cellular neural networks (CNNs) were proposed not only as a paradigm for complexity, but also as an architecture suitable for the design of VLSI processing arrays [1]. The concept of the CNN-Univeral Machine (CNN-UM), which combined a CNN core with an algorithmically programmable cellular computer, lead to the development of a number of silicon implementations of general-purpose visual microprocessors [2,3]. Other sensor/processor arrays have been developed [4,5], which use general-purpose analogue processor arrays in SIMD (single instruction multiple data) configuration, and are not based on continuous-time spatio-temporal processing associated with CNNs.

Since the CNN paradigm is very general, it is important to clarify here that when we are referring to a "CNN-based processor" we mean an architecture, which is largely based on the original Chua-Yang CNN model [1]. At present many VLSI circuits are being built which implement this general architecture (though usually with some modifications) [2,3], and many image processing algorithms are being published for these type of machines, e.g. [6-8]. Here we consider CNNs with nodes on a rectangular grid, and a 3×3 neighbourhood. Our discussion can be easily extended to other topologies. Furthermore, most of the discussion applies not only to CNNs, but generally to all analogue processor arrays, where the convolution operation is executed in parallel.

1.1. Image Processing on CNN chips

It is often said, that the present-generation CNN-UM chips are providing the TeraOPS (10^{15} operations per second) computing power [3,9]. However, when it comes to practical implementations of image processing algorithms using the CNN-UM, it is apparent that the *effective* computing power (i.e. the amount of operations required to achieve the equivalent image processing result) is much smaller. This is because of the fact, that the TeraOPS figure is calculated as "the number of operations that would be required to solve numerically the kind of partial differential equations (PDEs) which are solved by a CNN". There is a small number of image processing algorithms, that are indeed expressed in terms of PDEs [10], there are also interesting modelling problems [11] where a solution of PDEs is explicitly required. However, a great majority of image processing algorithms proposed for the CNN-UM systems in the literature, such as the ones in [6-8], do not require solving PDEs. They can be efficiently executed by a combination of arithmetic and logic operations. Two particular, and arguably most often required operations, are linear convolutions with 3×3 kernels and binary image processing via "propagating" templates. Virtually every image processing problem is being solved as a combination (i.e. sequential execution) of a number of these. Furthermore, since it is difficult to provide the required amount of hardware (requirement of multi-layer CNNs, and more complex operations) in general-purpose CNN-UM

implementations, the more complex algorithms that actually solve PDEs are still executed on a CNN-UM by a combination of more primitive tasks executed in an iterative fashion [10-12]. Obviously, any "universal machine" type of cellular array can implement (at least in principle) any algorithm. The choice of a set of primitive tasks does not impair the functionality of the network – but it will affect the parameters of the implementation: speed, silicon area, accuracy and power consumption. As these are of prime importance when it comes to engineering practical vision systems, it is important to consider how the trade-offs between these parameters, and the choice of a specific cell circuitry, can be efficiently made. In this paper we will limit our discussion to pixel-parallel visual microprocessors performing convolutions with 3×3 kernels, as defined in the following section. In another paper [13] we discuss other fundamental image processing operations often performed on cellular processor arrays: binary "propagation" algorithms.

2. Convolutions on Cellular Processor Arrays

A convolution with a 3×3 kernel is a basic component of a large number of grey-scale image processing algorithms. The operation can be expressed in terms of pixel-wise operations as follows:

$$q_{xy} = \sum_{\substack{i=1..3 \\ j=1..3}} c_{ij} u_{x-2+i,\,y-2+j} \tag{1}$$

where u_{xy} are pixels in the input image, q_{xy} are values of pixels in the output (result) image and c_{ij} are the nine coefficients of the convolution kernel. Two typical kernels, often used in image processing are shown below:

$$\mathbf{d} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \mathbf{v} = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2}$$

A CNN processor can execute the convolution operation in a single step (via B-template), since it contains 9 multiplier circuits working in parallel, as illustrated in Fig.1a. A simple SIMD processor, such as the one illustrated in Fig.1b, contains a single processing unit only (ALU) and will require a sequential execution of a number of instructions: neighbour transfers, additions and multiplications, to calculate the weighted sum of neighbours given by (1).

2.1 Accuracy

It is often said, that low-level image processing algorithms do not require great accuracy, and for many grey-scale operations the equivalent accuracy of perhaps 6-bits is sufficient. It has to be

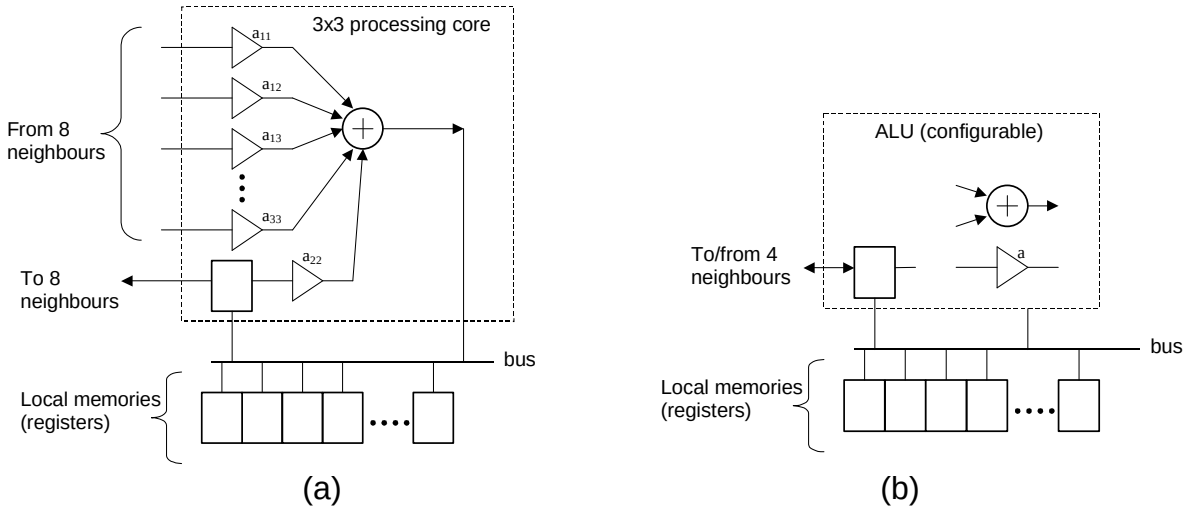


Figure 1. Conceptual diagrams of two alternative cellular processor architectures used for 3×3 convolutions: (a) kernel-based, like a CNN-UM (b) sequential, like a simple SIMD machine

recognised, however, that analogue processors are prone to error accumulation effects, and so the overall error of processing is larger than that of a single operation error, or memory storage error. On the other hand, systematic error effects (e.g. offset and gain errors) can often be ignored or easily compensated for. Even relatively large offset and gain errors result in contrast/brightness adjustment only. It is the random errors (spatial and temporal noise) that are limiting the accuracy of analogue processors, since they corrupt the spatial information in the image.

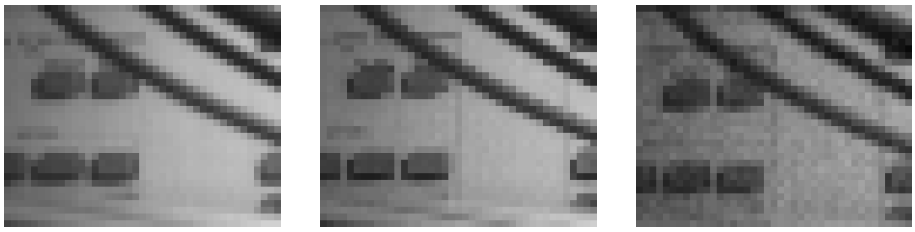
Of particular concern are spatially distributed errors, resulting in fixed-pattern-noise (FPN) effects. This is because they are correlated on pixel-wise basis, and thus the error accumulation results in error addition. The random noise errors, on the other hand, are uncorrelated, and so the accumulation follows a root-of-sum-of-squares law. Consider illustration in Fig.2, where a random noise σ_n=0.09% and fixed-pattern-noise σ_{fpn}=0.04% are applied iteratively to an image. After N iterations the total random noise is $\sqrt{N}\sigma_n$ while the total FPN is $N\sigma_{fpn}$; for example after 100 iterations we get random noise of 0.9% and fixed-pattern-noise of 4%. Furthermore, the averaging can be applied to reduce the noise, and the spatial noise seems to be more critical than the temporal noise, especially that it is usually preferable to increase the time of processing, rather than reduce the spatial resolution of the image.

The fixed-pattern error in analogue processor arrays is due to mismatch between circuit components (mostly transistors). The mismatch is caused by local fluctuations of physical parameters, and can be minimised by using large-area devices. Thus a trade-off between circuit area and accuracy exists, and it becomes critical in cellular pixel-per-processor arrays, where one of the imperatives is to minimise the silicon area of a single cell (large-resolution arrays are required), and thus mismatch problems are severe. Typical size of the entire processing cell in a state-of-the art vision chip is below 250μm^2, and it is desirable to decrease this area even further. This area has to contain photodetectors as well as local memories, arithmetic circuitry and all associated control circuits. Typical current mismatch for a saturated 10μm^2 transistor is in the order of 2%.

The solution shown in Fig.1a requires nine multiplier circuits in a single cell, while the solution in Fig.1b requires one multiplier only. Given equal silicon area used to implement the processing core, we can expect the circuit in Fig.1b to provide significant improvement in terms of accuracy. This is the basic, and most fundamental observation, but consideration of some more detailed issues provides further arguments in favour of using the circuit in Fig.1b.

In some cases, for example in the case of the convolution kernel **v** in (2), which is used to detect vertical edges in the Sobel algorithm, it is more important to ensure that the positive and negative terms are of equal magnitude – the accuracy of their absolute values (even if it changes from pixel to pixel) is of lesser importance. The processor in Fig.1b allows all coefficients to be calculated by the same multiplier circuit, if required, thus ensuring perfect matching between the coefficients.

Furthermore, some simple and often required operations, such as neighbour transfers (which correspond to image translations), are implemented in the Fig.1a via the multipliers, and thus prone to FPN errors introduced by the multipliers. In contrast, neighbour transfers in Fig.1b do not



(a) (b) (c)

Figure 2. Error accumulation on the analogue processor array [3]: (a) original image, (b) image after 25 transfers between two analogue registers, (c) image after 100 transfers between two analogue registers. The majority of the visible noise has a fixed-pattern character. (The results come from different captures, they do not correspond to the same image frame).

introduce these errors, and are only limited by the mismatch error of the analogue memory cell, which is typically very small (e.g. 0.09% in the chip reported in [14]).

There is, however, another aspect of the sequential solution – the possibility of improving the accuracy beyond the limit set by the device mismatch – which sets the solutions of Fig.1a and Fig.1b much further apart in terms of the achievable accuracy. This aspect is exemplified by the design used on the SCAMP-2 chip, where an "ALU-free" philosophy has been adopted [14]. The current-mode processing is used to perform all arithmetic operations in the analogue memory/bus system, without the need for any extra ALU circuitry. The instruction set is reduced to summation, inversion, and division by 2. Using these primitives all arithmetic operations are performed. Although this somewhat restricts the practical values of coefficients in convolution kernels, nevertheless this is usually not a problem. The area-savings offered by the "ALU-free" approach can be used to improve the accuracy of the analogue memories, by using more cell area to implement memories. If these analogue memories have better matching, this improves the accuracy of arithmetic operations also. It also reduces random noise. But, most importantly, a sequential error-correction scheme described in [15] can be used to achieve accurate division operation, thus significantly reducing the overall FPN associated with mismatch of coefficients in convolution kernels and cell-to-cell mismatch. Figure 3 illustrates the results of the Sobel edge detection algorithm (based on two 3×3 convolution kernels) executed on the SCAMP-2 processor. In case illustrated in Fig.4b the accuracy of division is 2.3%, which corresponds to the limit set by the transistor mismatch. In Fig.4c the accuracy has been increased to 0.2%, significantly beyond the transistor mismatch accuracy, through the error correction algorithm.

Of course, in each implementation of a cellular processor array the accuracy depends on a particular circuit design, and so it might be expected that even more optimal solutions than the one presented above exist. For example, using transistors in ohmic region should improve their matching properties [16]. Also, for each implementation, the absolute level of accuracy can be adjusted according to trade-offs between application requirements, total circuit area, speed and power consumption. What we hoped to illustrate, however, is that significant improvements in FPN, beyond transistor mismatch, are only possible through a sequential process, using the same device several times during the calculation, as opposed to a parallel execution of the convolution kernel.

It has to be said, that the improvements in terms of FPN are achieved at a cost of increased processing time. This trade-off certainly exists. However, given that usually not that many 3×3 grey-scale convolutions are required per image frame, for a majority of applications the sequential solution seems to be the more optimal one. Especially, when we consider that the parallel multiplication offered by the circuit in Fig.1a does not provide as great speed-up as it might be expected. This issue will be briefly exposed in the next section.

2.2 Efficiency

Let's assume, that the effective number of elementary arithmetic operations performed by the CNN based processor to calculate 3×3 convolution is equal to 17 (9 multiplications and 8 additions) per cell. Note that this corresponds to a processor shown in Fig.1a, rather than a PDE-solving CNN. If

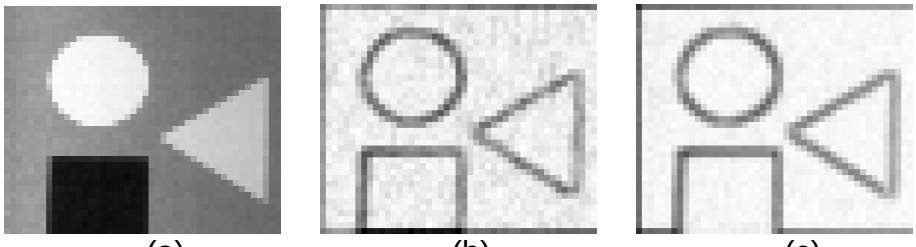


Figure 3. Edge detection on the analogue processor array [3]: (a) original image, (b) result with 2.3% division error (transistor mismatch limit), (c) result with 0.2% division error (sequential error compensation applied)

we were to use the TeraOPS figures sometimes quoted for the CNN-UM chips the number of "operations" would be much higher. We also ignore other CNN-specific arguments, such as bias term. We consider here 8 additions, as opposed to 1 summation operation, because the power consumption will usually depend on the number of arguments of the arithmetic operation.

While the 17 operations are in general required to implement any 3×3 kernel, the CNN solution is still inherently inefficient, as compared with a sequential SIMD machine. This is because in many cases the minimum number of operations required to implement convolutions is smaller, due to the possible decomposition of the convolution kernels that can be exploited when implementing them in a sequential way. As an example, consider the vertical Sobel edge detection kernel **v** and the smoothing kernel **d**, shown in (2). Possible sequential implementations of these convolutions are shown in Table 1. From these implementations it can be seen, that the total number of operations (additions, subtractions and multiplications) is equal to 5 for the edge detection kernel and 8 for the smoothing kernel. The CNN core, however, will always perform 17 operations

The inherent inefficiency of the CNN core is even more prominent if we consider, that a number of templates used for typical CNN image processing tasks are sparse, with only a few non-zero elements. Consider an extreme case of a B-template, used for example in [6]:

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{3}$$

A 3×3 kernel-based processor will still perform 17 operations to implement this template in case when only one neighbour transfer operation is all that is required. It shall be pointed out, that multiplication by zero still consumes power on the state-of-the-art CNN-UM chips! Indeed, the peculiarities of analogue VLSI design often lend themselves to the situation where arithmetic zero is represented by some non-zero biasing current or voltage level (for example, the one-transistor synapse strategy [16], which is the key to the compact circuit implementation of the CNN-UM architecture [2] requires zero to be represented by a fixed current).

It should be stated, that a particular SIMD processor implementation may require more elementary operations per arithmetic operation, due to the limitations of its instruction set (e.g. neighbour transfers may be allowed only via a particular register) or constraints imposed by the accuracy of processing (e.g. error cancellation schemes on analogue microprocessors). For example, on the SCAMP-2 chip [14] the kernels **v** and **d** are implemented with 19 and 32 elementary instructions respectively. On the other hand, practical CNN-UM implementations require additional calibration steps to achieve reasonable accuracy levels [2], and these should be also considered.

In the end, as it was the case when we considered the accuracy, the particulars of the implementation will have a great effect on the overall efficiency of the system and imaginative circuit solutions may be deployed to improve overall power consumption of CNN chips. Nevertheless, it can be said that the CNN processing cores are inherently inefficient when executing grey-scale convolutions. They will usually perform more operations than it is needed, and these

Table 1. SIMD programs for the implementation of two examples of convolution kernels (2)

*** v - SOBEL VERTICAL:**	*** d – SMOOTHING:**
A = I/4	A = I/4
C = A + A(north)	C = A + A(north)
C = C + A + A(south)	C = C + A + A(south)
C = C(east) – C(west)	A = C/4
	C = A + A(east)
	C = C + A + A(west)

operations will consume power and also take up silicon space (since hardware to perform these operations must be provided).

3. Conclusions

Although detailed comparisons between specific implementations of the two architectures can be made (if complete performance data are available), we have not compared particular chip designs in this paper. Instead, we tried to emphasise how some fundamental features of the two architectures affect the parameters of the implementation. We have reasoned that the parallel execution of 3×3 convolutions, as performed on CNN-based processors, is not optimal, in terms of accuracy and efficiency. The sequential execution, as performed on SIMD machines, does increase processing time, but allows the optimisation of the power consumption by performing fewer elementary operations. At the same time, it leads to the reduction in circuit area, while enabling accuracy improvements. Since the reduction of feature size will make it even easier to achieve high speeds of processing in future generation of analogue processors, for reasons of accuracy and efficiency outlined above, it might be expected that the techniques that use sequential rather than parallel execution of convolution kernels will be of greater use in the design of analogue cellular processors.

References

[1] L.O.Chua and L.Yang, "Cellular neural networks: Theory and applications", *IEEE Transctions on Circuits and Systems*, vol 35, pp.1257-1290, Oct 1988.

[2] G. Liñán *et. al*. "Architectural and Basic Circuit Considerations for a Flexible 128x128 Mixed-Signal SIMD Vision Chip", *Analog Integr. Circuits and Sig. Proc.* , vol.33, pp.179–190, 2002

[3] A.Paasio, A.Kananen, K.Halonen and V.Porra, "TOPS Information Processing on a Single Chip", *Circuits and Devices Magazine*, pp.13-15, May 1998

[4] P.Dudek, "A Processing Element for an Analogue SIMD Vision Chip", Proc. European Conference on Circuit Theory and Design, ECCTD'03, vol.III, pp.221-224, September 2003.

[5] Jacques-Olivier Klein *et al.* "A Universal Switched Capacitor Computation Cell Applied to a Programmable Vision Chip", Proc. Conf. ECCTD'03, vol III, pp.225-228, September 2003

[6] G.Grassi *et al.* "Object-Oriented Image Analysis Using the CNN Universal Machine: New Analogic CNN Algorithms …", *IEEE Trans. on Cicrcuits and Systems – I*, vol.50, no.4., pp.488-499, April 2003

[7] P.Arena, L.Fortuna and L.Occhipinti, "A CNN Algorithm for Real Time Analysis of DNA Microarrays", *IEEE Trans. on Cicrcuits and Systems – I*, vol.49, no.3., pp.335-340, March 2002

[8] P.Arena, A.Basile, M.Bucolo and L.Fortuna, "An Object Oriented Segmentation on Analog CNN Chip", *IEEE Trans. on Cicrcuits and Systems – I*, vol.50, no.7., pp.837-846, July 2003

[9] T.Roska, "Computer-Sensors: Spatial-Temporal Computers for Analog Array Signals, Dynamically Integrated with Sensors", *Journal of VLSI Signal Processing*, 23, pp.221-237, 1999

[10] T.Kozek and D.L.Vilariño, "An Active Contour Algorithm for Continuous-Time Cellular Neural Networks", *Journal of VLSI Signal Processing*, 23, pp.403-414, 1999

[11] D.Balya, Cs.Rekeczky, T.Roska, "A realistic mammalian retinal model implemented on complex cell CNN universal machine", ISCAS 2002, vol IV, pp.161-164, 2002

[12] I.Szatmari and Cs.Rekeczky, "A Nonlinear Wave Metric and its CNN Implementation for Object Classification", *Journal of VLSI Signal Processing*, vol. 23, pp.437-447, 1999.

[13] P.Dudek, "Fast and Efficient Implementation of Trigger-Wave Propagation on VLSI Cellular Processor Arrays", CNNA'2004

[14] P.Dudek "A 39x48 General-Purpose Focal-Plane Processor Array Integrated Circuit", ISCAS 2004

[15] J.-S. Wang and C.-L. Wey, "Accurate CMOS Switched-Current Divider Circuits", Proc. ISCAS'98, vol I, pp.53-56, May 1998.

[16] A.Rodríguez-Vázquez *et al.* "MOST-Based Design and Scaling of Synaptic Interconnections in VLSI Analog Array Processing VLSI Chips", *Journal of VLSI Signal Processing*, 23, pp.239-266, 1999