

Implementing the grayscale wave metric on a cellular array processor chip

Dániel Hillier*[‡] Piotr Dudek[†]

*Jedlik Laboratory, Faculty of Information Technology
Péter Pázmány Catholic University, 1083 Budapest, Práter u. 50/a, Hungary
e-mail: hillier@digitus.itk.ppke.hu

[†]School of Electrical and Electronic Engineering, The University of Manchester
PO Box 88, Manchester M60 1QD, UK e-mail: p.dudek@manchester.ac.uk

[‡]Neural Circuit Laboratory, Friedrich Miescher Institute for Biomedical Research
Maulbeerstrasse 66, 4058 Basel, Switzerland

Abstract—Algorithms designed for machine vision applications such as medical imaging, surveillance, etc., very often require some kind of comparison between images. The non-linear wave metric can measure both the shape and the area difference between two objects in one single operation. We present the implementation of the wave metric on the SCAMP chip that combines the benefits of a highly selective metric with high speed, efficient execution.

Index Terms—SCAMP, wave metric, wave computing, Cellular Nonlinear Networks

I. INTRODUCTION

Algorithms designed for machine vision applications such as medical imaging, surveillance, etc., very often require some kind of comparison between images. Examples include detecting a pathology in a medical image, object search in image databases, or live recognition of intruder-like shapes in a surveillance scenario. The simplest solution for comparing two images is to compute their pixel-wise difference. On a personal computer (PC), the required number of computational operations for this simple comparison is at least twice the number of the pixels in the image. Complex image processing algorithms are composed of many operations and are in most cases very time consuming. Therefore, real-life problems can hardly be solved on-line or real-time at a reasonable cost using current PC architectures, impeding the application of high speed machine vision algorithms in many fields. In most cases, more complex measures are required for meaningful object comparison. Quantitation of shape differences between objects is often used in image processing and recognition algorithms. The so-called non-linear wave metric, introduced by Szatmári et al. [1] measures both the shape and the area difference between two objects in one single operation. The non-linear wave metric can be calculated in a single instruction on a multi-layer cellular nonlinear network (CNN) architecture, e.g. on the CACE1k chip [2], but the availability of such devices is limited.

In this paper, we present the implementation of the wave metric on a general-purpose single instruction multiple data

(SIMD) cellular processor array chip (SCAMP [3]). This implementation makes use of the massively parallel processing and global operation capabilities of the device thus combining the benefits of a highly selective metric with high speed, efficient execution.

This paper is organized as follows. First, the non-linear wave metric is outlined. Next, the SCAMP implementation is described and then compared to implementations on other architectures.

II. OBJECT COMPARISON

Object comparison requires a properly defined metric and a reference. The choice of a robust metric is an intricate task. The most obvious criterion of the degree of coincidence of point sets is a measure of symmetrical difference, i.e. the area difference. This is a natural choice and it is the well-known Hamming distance which on two given finite binary point sets A and B is the result of a pixel-wise XOR operation:

$$d_{Hm} = \sum (A \cup B \setminus A \cap B) \quad (1)$$

Another often-used distance is the Hausdorff distance. The Hausdorff distance is defined as

$$d_{Hs} = \max(h(A, B), h(B, A)) \quad (2)$$

where $h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|$ and $\|\cdot\|$ is some norm on the points of A and B . The function $h(A, B)$ identifies the point $a \in A$ that is farthest from any point of B and measures the distance from a to its nearest neighbor in B using the given norm $\|\cdot\|$.

Although the Hamming and Hausdorff distances are commonly used in image processing applications for object comparison and classification, they have several disadvantages. Hamming distance measures the area difference, but does not reveal anything about shape difference. Hausdorff metric measures shape difference but cannot tell anything about shape properties, like average distance between two objects, e.g. a one pixel sized noisy spot can drastically modify the Hausdorff

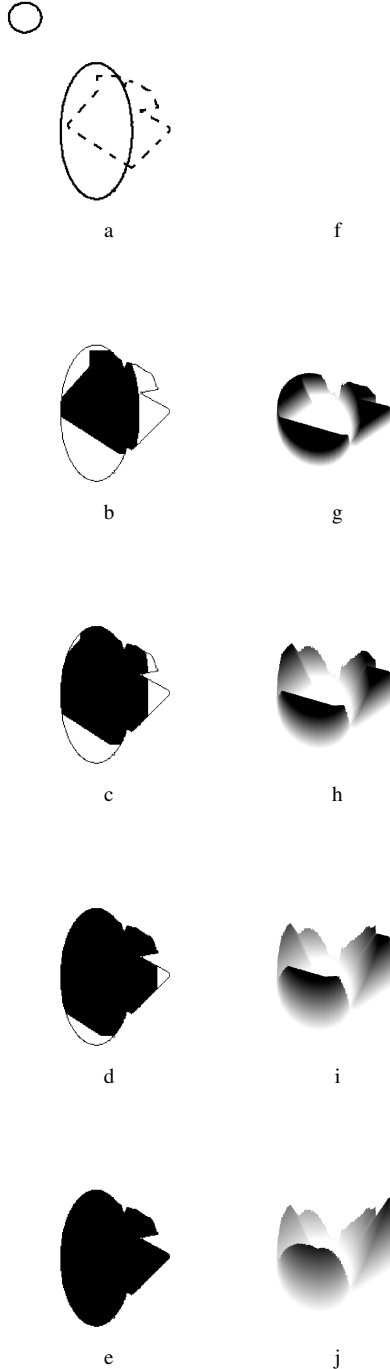


Fig. 1: Snapshots showing the binary evolution of $A \cap B$ constrained by $A \cup B$ during the calculation of the wave metric. As shown in (a), there are two objects to be compared denoted by solid and dashed outlines respectively. (c) shows that the propagation is initiated from $A \cap B$ and proceeds till it fills $A \cup B$ shown in (e). (f)–(j) show the evolution of the wave map d_{lHS} storing the local Hausdorff distances.

distance. The wave metric was shown to have higher sensitivity and selectivity to object shift and scale changes compared to

both the Hamming and the Hausdorff distances [8].

A. Non-linear wave metric

In [1] the non-linear wave metric was introduced. The wave metric measures both area and shape differences between two binary objects. Let a binary wave - very similar to the constrained wave operator used in [4], [5] - be started from $A \cap B$ and propagating till it matches the points of $A \cup B$. The time required for the wave to occupy $A \cup B$ measures the difference between the shapes A and B . During wave evolution a grayscale image d_{lHS} is created in which a pixel value corresponds to the time required for the wave to reach that pixel from $A \cap B$. The sum of these local Hausdorff distances gives the wave-type metric $d_W = \sum d_{lHS}$.

The process of calculating the wave metric can be seen in Fig. 1. Object parts not connected to $A \cap B$ are not filled by the wave thus the wave metric is robust to noise appearing as small spots on the background. In addition to capture both area and shape differences, this metric has parallel implementation with about $10 \mu s$ running time on chips presented in [6] or [7].

B. Grayscale version

The binary version of the wave metric can be extended to grayscale images. The grayscale version of the wave metric is formulated in a PDE model where an image - either binary or gray-scale - is defined as a real function $I(x, y) : [0, N]^2 \rightarrow [0, 1]$, zero values stand for background. The two object to be compared are two images I_{in} and I_{ref} with identical dimensions to I . The dynamical equation defining the grayscale wave metric comparing two images is a nonlinear partial differentiation equation (PDE):

$$\begin{aligned} \frac{\partial I_1(x, y, z)}{\partial t} &= D \cdot \Delta I_1 + v \cdot (I_{max} - I_1) \\ \frac{\partial I_2(x, y, z)}{\partial t} &= w \cdot (I_{max} - I_1) \end{aligned} \quad (3)$$

where $I_{max}(x, y) = \max(I_{in}(x, y, 0), I_{ref}(x, y, 0))$ contains the pixel-wise maximum of I_1 and I_2 , $I_{min}(x, y) = \min(I_{in}(x, y, 0), I_{ref}(x, y, 0))$, $I_1(x, y, 0) = I_{min}$, $I_2(x, y, 0) = 0$, $v > 0$ and $w > 0$ are constants. Δ is the Laplace operator and $D > 1$ is a constant. The final wave map is the steady state solution of I_2 . The interested reader can find more details on the grayscale version of the metric in [8].

III. IMPLEMENTATION ON THE SCAMP CHIP

The grayscale version of the wave metric was implemented as follows. First of all, images I_{min} and I_{max} are initialized. The wave propagation is performed in an iterative way. Each iteration is started by one propagation step. The result of the propagation is constrained by the union I_{max} of the two objects to be compared. The sum of the difference between the current state of the wave and the union is calculated in

each iteration using the global sum operation. The propagation terminates once this difference gets below a predefined value. The distance of the two input images is obtained in a single value that is obtained by cumulating the global sum values of each wave evolution iteration.

The Matlab code implementing Eq. (3) was used as a standard to which the output of the SCAMP simulator and hardware could be compared. The pseudo code description of the algorithm working on the SCAMP chip is presented in Algorithm 1.

Algorithm 1 Wave metric algorithm on the SCAMP chip

```

1: function METRIC( $I_{in}, I_{ref}, i_{max}, thr$ )
2:   set global sum settings
3:    $I_{min} = \min(I_{in}, I_{ref})$ 
4:    $I_{max} = \max(I_{in}, I_{ref})$ 
5:    $I_{min}^0 = I_{min}$ 
6:   for  $i = 1 : i_{max}$  do
7:      $I_{min} = \Delta \cdot I_{min}$ 
8:      $I_{min} = \min(I_{min}, I_{max})$ 
9:      $I_{min} = \max(I_{min}, I_{min}^0)$ 
10:    Where  $I_{max} > 0$ , replace  $I_{min}$  with  $I_f$ 
11:     $s(i) = GlobalSum(I_{max} - I_{min})$ 
12:    if  $s(i) < thr$  then
13:      BreakFor
14:    end if
15:  end for
16:  return  $s$ 
17: end function

```

Line 2 changes the parameters of the flexible read-out operation to perform global grayscale summation. The code was developed in the SCAMP simulator using 'Level 2' error modeling, i.e. signal dependent and fixed pattern noise was included in the model.

The main challenge was to work around the effect of signal dependent noise. This type of noise can destroy meaningful operation when the data is processed recursively. This is the case for the current implementation of the wave metric. Lines 5,9 and 10 minimize the effect of signal dependent error by setting back the value of pixels not changed in the current iteration to their previously stored value. I_f in Line 10 contains the background intensity value used to keep inactive pixels at their original value. Snapshots from the evolution of the wave are shown in Fig. 2. The global sum values of each iteration of the Matlab implementation, the uncompensated - i.e. Algorithm 1 without line 5,9,10 - and the compensated SCAMP algorithm can be seen in Fig. 3.

Two practical ways of extracting the result of the wave metric algorithm are mentioned. The first way is to transfer the current state of the propagation to the host computer where the wave map can be reconstructed representing the steady state of I_2 in Eq. (3), shown as a grayscale image in Fig. 1j. The outcome of the object comparison can also be represented by the values of the analog grayscale sums. In such a case there

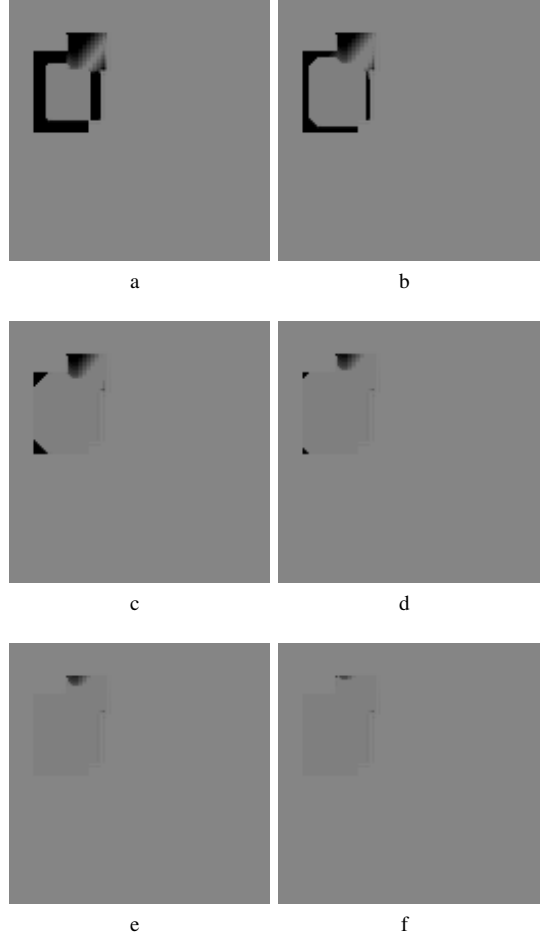


Fig. 2: Snapshots showing the evolution of $I_{max} - I_{min}$ during the calculation of the wave metric.

is no need to transfer any image to the host but information on the spatial dissimilarity of the two objects is lost.

One iteration of the wave evolution at 128×128 resolution takes $52 \mu s$ on the SCAMP whereas the corresponding steps implemented in C language take 2.5 ms on a PC with Pentium 4 2.8 GHz processor and 512 Mb RAM.

IV. DISCUSSION AND CONCLUSION

Low-level image processing operators like filtering, edge detection, binary hole filling, feature extraction, etc. are computationally intensive. These operations are inherently pixel-parallel, i.e. identical, localized operations are performed on every pixel. Efficient image processing systems can be designed by associating each image pixel with an image processing circuitry and allowing local connections between neighboring processing cells. Each cell can have local memories and can perform basic arithmetic and logic operations on pixel values of their local neighborhood. CNNs [9] represent a powerful framework for this concept. In many CNN implementations, each individual cell circuitry is a realisation of a

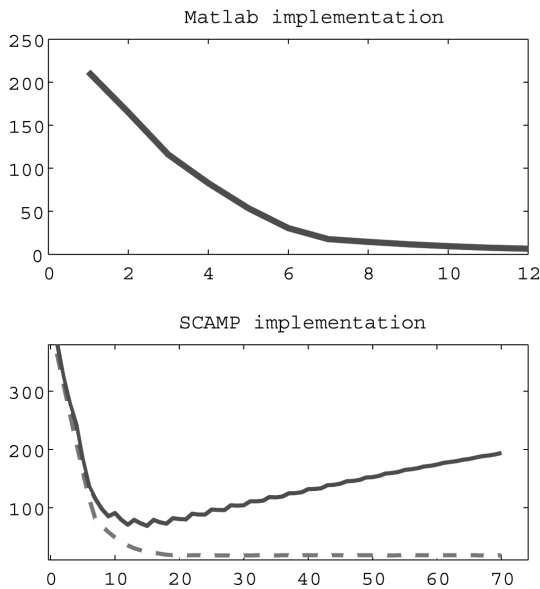


Fig. 3: Global sum values $s(i)$ plotted at each iteration during wave evolution. The process stops when $s(i) < thr$ where thr must be chosen above the hardware noise. The solid curve in the bottom plot shows the evolution of $s(i)$ during the execution of the uncompensated implementation, the dashed curve corresponds to the compensated implementation.

nonlinear ordinary differential equation, i.e. CNNs can be used to approximate solutions of PDEs. A number of different CNN processor implementations are available for parallel image processing [10] on which various difficult image processing problems were solved at high speed [11].

Eq. (3) can be implemented in a single instruction on the ACE16k chip [2], however concerns have been raised related to the accuracy and efficiency of the continuous-time analogue CNN hardware implementations [12]. A straightforward method is available to compensate accuracy problems of continuous-time implementations via tuning CNN templates to the chip instance used [13].

The implementation on the SCAMP [3] discrete-time analogue cellular processor array presented in this paper balances the requirements of a highly selective metric and efficient, robust execution exploiting the power of the SCAMP chip, an architecture characterized by small cell area, low power dissipation and the ability to execute a variety of image processing algorithms. The SCAMP implementation outperforms the Pentium implementation by 1.5 order of magnitude. In addition, the SCAMP chip draws 250 mW power whereas the Pentium draws about 80 W.

The illustrations and timing results were obtained using the SCAMP simulator. The code was also executed successfully on the SCAMP hardware. The result of this study reinforces earlier results on the efficiency of the SCAMP chip for high

speed machine vision. Also, the availability of the SCAMP box makes the wave metric - already shown to be highly useful in solving very hard image processing problems [14] - easily embeddable in high speed image processing algorithms.

A next step will be to implement the wave metric on the ASPA [15] architecture.

ACKNOWLEDGMENT

We thank I. Szatmári for his help. Research supported by the Incoming Short Visit Grant of the Royal Society. D. Hillier acknowledges the financial support of the RET program via the Faculty of Information Technology of the Péter Pázmány Catholic University.

REFERENCES

- [1] I. Szatmári, C. Rekeczky, and T. Roska, "A Nonlinear Wave Metric and its CNN Implementation for Object Classification," *The Journal of VLSI Signal Processing*, vol. 23, no. 2, pp. 437 – 447, 1999.
- [2] I. Petrás, C. Rekeczky, T. Roska, R. Carmona, F. Jimenez-Garrido, and A. Rodriguez-Vazquez, "Exploration of spatial-temporal dynamic phenomena in a 32×32 -cell stored program two-layer CNN universal machine chip prototype," *Journal of Circuits, Systems, and Computers*, vol. 12, no. 6, pp. 691–710, 2003.
- [3] P. Dudek and S. J. Carey, "A general-purpose 128x128 simd processor array with integrated image sensor," *Electronics Letters*, vol. 41, no. 12, pp. 678–679, June 2006.
- [4] C. Rekeczky and L. O. Chua, "Computing with Front Propagation: Active Contour And Skeleton Models In Continuous-Time CNN," *The Journal of VLSI Signal Processing*, vol. 23, no. 2, pp. 373 – 402, 1999.
- [5] D. Hillier, V. Binzberger, D. L. Vilarino, and C. Rekeczky, "Topographic cellular active contour techniques: Theory, implementations and comparisons," *International Journal of Circuit Theory and Applications*, vol. 34, no. 2, pp. 183–216, 2006.
- [6] G. Linan, S. Espejo, R. Dominguez-Castro, and A. Rodriguez-Vazquez, "ACE 4 k: An analog I/O 64×64 visual microprocessor chip with 7-bit analog accuracy," *International Journal of Circuit Theory and Applications*, vol. 30, no. 2-3, pp. 89 – 116, 2002.
- [7] A. Rodriguez-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro, and S. E. Meana, "ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs," *Circuits and Systems I: Regular Papers, IEEE Transactions on [see also Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on]*, vol. 51, no. 5, pp. 851 – 863, 2004.
- [8] I. Szatmari, "Object comparison using PDE-based wave metric on cellular neural networks," *INTERNATIONAL JOURNAL OF CIRCUIT THEORY AND APPLICATIONS*, vol. 34, no. 4, p. 359, 2006.
- [9] L. Chua and T. Roska, *Cellular neural networks and visual computing*. Cambridge University Press New York, NY, 2002.
- [10] A. Zarandy, M. Foldesy, P. Szolgay, S. Tokes, C. Rekeczky, and T. Roska, "Various implementations of topographic, sensory, cellular wave computers," *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 5802–5805, 2005.
- [11] C. Rekeczky, I. Szatmari, D. Balya, G. Timar, and A. Zarandy, "Cellular multiadaptive analogic architecture: a computational framework for UAV applications," *Circuits and Systems I: Regular Papers, IEEE Transactions on [see also Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on]*, vol. 51, no. 5, pp. 864–884, 2004.
- [12] P. Dudek, "Accuracy and Efficiency of Grey-level Image Filtering on VLSI Cellular Processor Arrays," *Proc. CNNA*, pp. 123–128, 2004.
- [13] D. Hillier, S. Xavier-de Souza, J. Suykens, and J. Vandewalle, "CNNOPT: Learning dynamics and CNN chip-specific robustness," *IEEE Int. Workshop on Cellular Neural Networks and their Applications (CNNA)*, 2006.
- [14] I. Szatmari, A. Schultz, C. Rekeczky, T. Kozek, T. Roska, and L. O. Chua, "Morphology and autowave metric on CNN applied to bubble-debris classification," *IEEE Trans. Neural. Networks*, vol. 11, no. 6, pp. 1385–1393, 2000.

- [15] A. Lopich and P. Dudek, "Global operations in SIMD cellular processor arrays employing functional asynchronism," *Computer Architecture for Machine Perception and Sensing, 2006. CAMP 2006. International Workshop on*, pp. 18–23, 2006.