2008 International Symposium on Nonlinear Theory and its Applications
NOLTA'08, Budapest, Hungary, September 7-10, 2008

Implementing the non-linear wave metric on the Q-Eye cellular array processor chip

Dániel Bank†, Ákos Zarándy‡ and Dániel Hillier §

†Faculty of Information Technology, Péter Pázmány Catholic University, Budapest, Hungary
‡Analogic and Neural Computing Laboratory, Computer and Automation Research Institute of the Hungarian Academy of Sciences, Budapest, Hungary
§Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland
Email: banda@digitus.itk.ppke.hu, zarandy@sztaki.hu, daniel.hillier@fmi.ch

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 is a relatively novel approach that can be used to measure both the shape and the area difference between two objects in one single operation. In this paper we present the implementation of the wave metric on the Q-Eye cellular visual microprocessor chip (AnaFocus Ltd.) that combines the benefits of a highly selective metric with high speed, efficient execution.

1. Introduction

Medical imaging, surveillance, etc. rely on image processing algorithms that very often require some kind of comparison between images. Examples include detecting a pathology in a medical image, object search in image databases, or real-time recognition of intruder-like shapes in a surveillance scenario. A very straightforward solution for comparing two images is to compute their pixel-wise difference. On a traditional digital computer architecture the pixel-wise difference operation requires at least twice as much computational steps as the number of pixels in the image. More complex image processing algorithms are composed of many similar image processing steps, thus in most cases the solution of a real-world problem can cost a lot of computing power. 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, pixel-wise difference does not provide enough information for meaningful object comparison thus more complex measures are required. E.g. quantitation of shape differences between objects is often used in image processing and recognition algorithms. The non-linear wave metric, introduced in [11], measures both the shape and the area difference between two objects in one single operation. The execution time of the non-linear wave metric is extremely short when implemented on a multi-layer cellular nonlinear network (CNN) architecture, e.g. on the CACE1k chip [7], but the availability of such devices is limited.

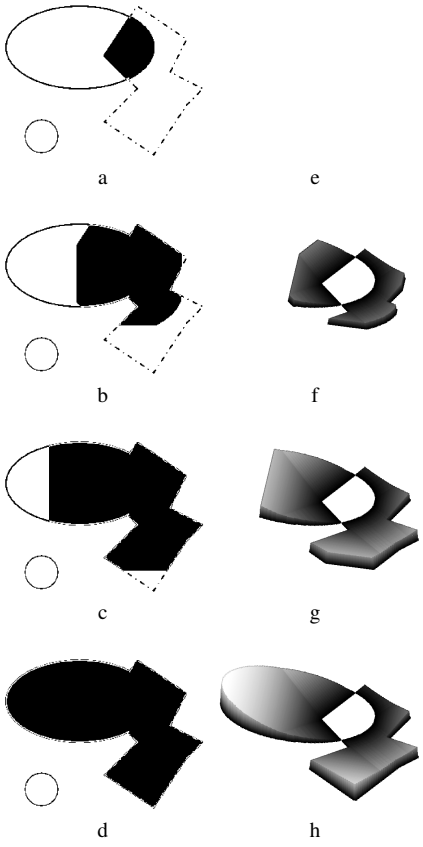

Figure 1: Snapshots showing the binary evolution of $A \cap B$ constrained by $A \cup B$ during the calculation of the wave metric. As shown in (a), there are two objects to be compared denoted by solid and dashed outlines respectively. The propagation is initiated from $A \cap B$ and proceeds till it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

In this paper, we present the implementation of the grayscale version of wave metric on the Eye-RIS Vision System v1.2 (AnaFocus Ltd., www.anafocus.com) that embeds a pixel-parallel topographic processor, the Q-Eye (the successor of the ACE16k chip [10]).

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

2. Object comparison

Object comparison requires a properly defined metric and a reference. The choice of the metric is an intricate task. The pixel-wise difference is a very obvious measure of the degree of coincidence of point sets, also termed as the area difference when binary images are compared. This operator corresponds to the well-known Hamming distance which is the result of a pixel-wise XOR operation on two given finite binary point sets A and B:

$$d_{Hm} = \sum (A \cup B \backslash A \cap B) \tag{1}$$

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

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

where $h(A,B) = \max_{a \in A} \min_{b \in B} \|a - b\|$ and $\|.\|$ 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 $\|.\|$. 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 distance.

2.1. Non-linear wave metric

In [11] the non-linear wave metric was introduced that measures both area and shape differences between two binary objects. Let a binary wave - very similar to the constrained wave operator used in [3,8] - 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 binary wave metric has parallel implementation with about 10 μs running time on [5] or [10].

2.2. 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 objects 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):

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

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. 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 [13, 14].

3. Implementation on the Q-Eye chip

The Matlab code approximating Eq. (3) was used as a standard to which the output of the hardware implementation could be compared. The pseudo code description of the algorithm working on the Q-Eye chip is presented in Algorithm 1.

First, images I_{min} and I_{max} are loaded into the on-chip memory space (line 2,3). The wave metric is calculated via discrete propagation steps (line 6). The result of each propagation step is constrained by the union I_{max} of the two objects to be compared (line 7). The difference between the current state of the wave and the union is calculated in each iteration (line 9,10). The propagation terminates once the sum of this difference gets below a predefined value (line 16,20). The distance of the two input images is obtained in a single value as the global sum of the generated wavemap (line 12), or by summing the number of pixels changed in current iteration (line 14).

c_{offset} is a predefined constant corresponding to the gray level value of the hardware specific offset (which was measured), 4096 is a limit in the current Eye-Ris implementation on the number of countable pixels on a binary image. Δ represents the result of horizontal and vertical propagation within one iteration.

Each processing cell in the Q-Eye chip contains a resistive grid for implementing analog image filters. Horizontal propagation was implemented using masked diffusion. A binary mask was used to drive the diffusion (line 5). Growth of image intensities (propagation onto the second layer on the CACE1k chip [7]) can be easily implemented, by adding intensity values (beside offset and noise) to the object, using the dedicated circuitry in each cell.

Algorithm 1 Wave metric algorithm on the Q-Eye chip

1: **function** METRIC($I_{input}, I_{reference}, n_{itermax}, c_{offset}$)
2: $I_{min} = min(I_{input}, I_{reference})$
3: $I_{max} = max(I_{input}, I_{reference})$
4: **for** $i = 1 : n_{itermax}$ **do**
5: Where $I_{min} > 2 * c_{offset}$ $I_{ObjectMask} = 1$
6: $I_{min} = \Delta \cdot I_{min}$
7: $I_{min} = min(I_{min}, I_{max})$
8: Where $I_{min} < c_{offset}$ $I_{min} = 0$
9: Where $(I_{max} - I_{min}) > c_{offset}$ $I_{ActiveMask} = 1$
10: Where $(I_{max} - I_{min}) > c_{offset}$ $I_{WaveMap} = I_{WaveMap} + 1$
11: **if** $s > 4096$ **then**
12: $s = sum(I_{WaveMap})$
13: **else**
14: $s = sum(I_{ActiveMask})$
15: **end if**
16: **if** $s > 0$ **then**
17: Where $I_{max} > c_{offset}$ $I_{min} = 0$
18: Where $I_{max} > c_{offset}$ $I_{WaveMap} = 0$
19: **else**
20: break
21: **end if**
22: **end for**
23: return s
24: **end function**

In order to obtain the binary mask the grayscale object has to be separated from the background noise and offset. This was solved by setting an initial threshold value at two times the measured offset, to define the object boundaries, thus image intensities under this value do not participate in metric calculation.

The tricky part of the analog implementation is to avoid noise summation in a deeply recursive algorithm. Reusing a picture in a LAM (Local Analog Memory) recursively, the computational noise can be significant after many steps in the background. This effect is compensated at each iteration, by setting the object background to zero intensity level (line 8, 17, 18).

Final metric is calculated from the generated wavemap by counting the number of changed pixels in each iteration. The result is uploaded to the NIOS co-processor where it is weighted and accumulated. To maximally exploit the bit depth of the Q-Eye we generate a wavemap in the first few iterations and sum it up, (line 11). Then in the next iteration steps we count only the changed pixels (which is faster, line 14), and add them to the previous value to get proper wavemetric.

Snapshots from the evolution of the wave on the Q-Eye chip are shown in Fig. 2.

The changed pixel values plotted at each iteration step of the Matlab implementation, and the Eye-RIS implementation of Algorithm 1 can be compared in Fig. 3.

The code was developed in the Eye-RIS IDE and executed on the chip. Global summation was done by the NIOS II 32 bit RISC processor embedded in the Eye-RIS VS. The final and partial results were downloaded from the chip.

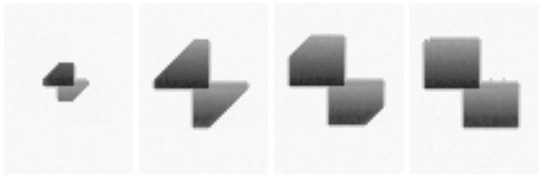

Figure 2: Snapshots showing the evolution of I_{min} during the calculation of the wave metric.

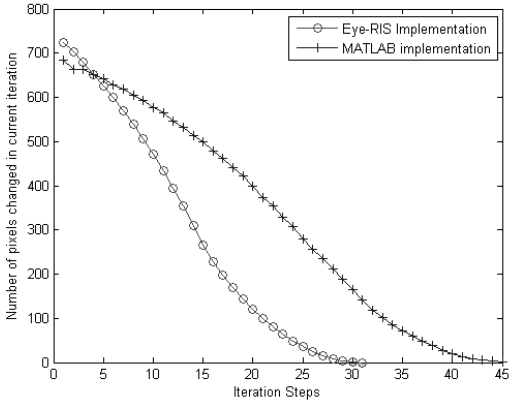

Figure 3: Global sum values $s(i)$ plotted at each iteration during wave evolution. The process stops when the entire grayscale object is filled, when no pixel changes on $I_{ActiveMask}$. We used the same accuracy in the Matlab implementation (8 bit) to compare with the Q-Eye chip (8 bit equivalent). As we can see the convergence is faster on the Q-Eye chip, this is due to the effect of offset and signal dependent noise of the analog processing elements. This effect does not corrupt metric calculation, changes the accumulated value, but it remains still selective to measure with it.

One iteration of the wave evolution takes 93 μs at 144×176 resolution on the Q-Eye, 52 μs at 128×128 resolution on the SCAMP chip (simulation result) and 2.5 ms at 128×128 resolution implemented in C language on a PC with Pentium 4 2.8 GHz processor and 512 Mb RAM.

4. 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 [1] represent a powerful framework for this concept. In many CNN implementations, each individual cell circuitry is a realisation of a 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 [15] on which various difficult image processing problems were solved at high speed [9].

Eq. (3) can be implemented in a single instruction on the CACE1k chip [7], however concerns have been raised related to the accuracy and efficiency of the continuous-time analogue CNN hardware implementations [2]. A straightforward method is available to compensate accuracy problems of continuous-time implementations via tuning CNN templates to the chip instance used [4].

The availability of the Eye-RIS VS makes the wave metric - already shown to be highly useful in solving very hard image processing problems [12] - easily embeddable in high speed image processing algorithms.

A next step will be to implement the wave metric on the ASPA [6] architecture.

Acknowledgement

We wish to thank István Szatmári for sharing the result of his work with us. Dániel Bank wishes to thank the support of Péter Pázmány Catholic University for giving the oppurtunity to work in the Jedlik Laboratory on the Eye-RIS Vision System. We wish to thank to AnaFocus to make the Eye-RIS system available for us, and gave technical support.

References

[1] L.O. Chua and T. Roska. *Cellular neural networks and visual computing*. Cambridge University Press New York, NY, 2002.

[2] P. Dudek. Accuracy and Efficiency of Grey-level Image Filtering on VLSI Cellular Processor Arrays. *Proc. CNNA*, pages 123–128, 2004.

[3] D. Hillier, V. Binzberger, D. L. Vilarino, and Cs. Rekeczky. Topographic cellular active contour techniques: Theory, implementations and comparisons. *International Journal of Circuit Theory and Applications*, 34(2):183–216, 2006.

[4] D. Hillier, S. Xavier-de Souza, J.A.K. Suykens, and J. Vandewalle. CNNOPT: Learning dynamics and CNN chip-specific robustness. *IEEE Int. Workshop on Cellular Neural Networks and their Applications (CNNA)*, 2006.

[5] 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*, 30(2-3):89 – 116, 2002.

[6] 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*, pages 18–23, 2006.

[7] I. Petrás, Cs. 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*, 12(6):691–710, 2003.

[8] Cs. Rekeczky and L. O. Chua. Computing with Front Propagation: Active Contour And Skeleton Models In Continuous-Time CNN. *The Journal of VLSI Signal Processing*, 23(2):373 – 402, 1999.

[9] Cs. 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*, 51(5):864–884, 2004.

[10] 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*, 51(5):851 – 863, 2004.

[11] I. Szatmári, Cs. Rekeczky, and T. Roska. A Nonlinear Wave Metric and its CNN Implementation for Object Classification. *The Journal of VLSI Signal Processing*, 23(2):437 – 447, 1999.

[12] I. Szatmari, A. Schultz, Cs. Rekeczky, T. Kozek, T. Roska, and L. O. Chua. Morphology and autowave metric on CNN applied to bubble-debris classification. *IEEE Trans. Neural. Networks*, 11(6):1385–1393, 2000.

[13] I. Szatmari. Object comparison using PDE-based wave metric on cellular neural networks. *Int. Journal of Circuit Theory and Applications*, Vol. 34, pp. 359-382, 2006.

[14] I. Szatmari. Spatio-temporal Nonlinear Wave Metric for Binary and Gray-scale Object Comparison on Analogic Cellular Wave Computers, *Int. Journal of Functional Differential Equations*, 2005.

[15] A. Zarandy, P. Foldesy, P. Szolgay, Sz. Tokes, Cs. Rekeczky, and T. Roska. Various implementations of topographic, sensory, cellular wave computers. *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 5802–5805, 2005.