Implementation of a CNN based Object Counting Algorithm on Bi-i Cellular Vision System

Selcuk SEVGEN[1], Fethullah KARABIBER[2], Eylem YUCEL[3] and Sabri ARIK[4]

[1,2,3,4]Istanbul University, Department of Computer Engineering, Turkey
sevgens@istanbul.edu.tr, fetullah@istanbul.edu.tr, eylem@istanbul.edu.tr, ariks@istanbul.edu.tr

Abstract

Object counting has been used in many areas such as medical and industrial applications. It is a challenging problem to count the target objects in high speed. It is useful to implement image processing applications using the high capability computational power offered by Cellular Neural Network type analog processor named as ACE16k. In this paper, we implement an efficient object counting algorithm working on ACE16k chip. Our results have proved that the proposed algorithm can count objects on a given image rapidly and accurately.

1. Introduction

Counting objects in images is necessary in various applications, such as counting people on roads or birds in the sky, packaging, and quality control in industrial systems, etc. Object counting is a simple task with human efforts. But it is necessary to develop automated methods for object counting using computer vision to reduce human efforts.

In generally, the studies related to task of object counting are based on detection of target objects. Objects on an image are labeled and indexed by a finding algorithm. This process is very time consuming [13]. Using an algorithm based on Cellular Neural Networks (CNNs) for counting objects reduces time consuming. In previous works, Seiler [11] and Fasih et al. [9] developed different object counting methods using CNN. These works are based on software simulations. Our proposed algorithm works on Bi-i Cellular Vision System which has two type microprocessors - Ace16k and Digital Signal Processor (DSP).

CNN based applications are performed using templates. Template design is one of the main problems in CNN based image processing application. In this study, we have trained necessary template for object counting using Iterative Annealing optimization method.

This paper is organized as follows. In section 2, the CNN architecture, ACE16k chip and Bi-i Cellular Vision System are examined. Section 3 explains template design tool. In section 4, object counting algorithm is given. Finally, the concluding remarks are given in Section 5.

2. CNN Architecture and Bi-i Cellular Vision System

2.1. CNN structure

The key feature of a Cellular Neural Networks (CNN), introduced in [1] [2], is that it is a locally interconnected analog processor array. Since CNN has two dimensional (2D) grid structure, it is a suitable platform for developing image processing algorithms.

Based on the mathematical modeling of CNNs, a programmable CNN, called CNN universal machine (CNN-UM) [3] has been developed. The CNN-UM is programmable array computer with real time and super computer power in a single chip. Since these chips have huge computational power and capability of parallel processing, it is possible to perform image processing tasks in a high speed in comparison to conventional architectures. The latest hardware implementation of the CNN is Analog Focal Plane processor called ACE16k [4].

Program instructions called templates have most important role in the CNN applications. The dynamical behavior of a CNN is completely determined by the templates. The design of suitable templates is one of the fundamental tasks in CNN area. It is also important to find optimal values for template elements so that a CNN performs a desired task.

Standard CNN consists of *MxN* rectangular array of cells. The smallest part of CNN is called *cell* (*C(i,j)*) with Cartesian coordinates (i,j) (*i=1,2,3...M, j=1,2,3...N*). Each cell can be defined by the following linear and non-linear mathematical equations [9];

$$\frac{dx_{ij}}{dt} = -x_{ij} + \sum_{C(k,l)\in S_r(i,j)} A(i,j;k,l)y_{kl} + \sum_{C(k,l)\in S_r(i,j)} B(i,j;k,l)u_{kl} + z_{ij} \quad (1)$$

$$y_{ij} = f(x_{ij}) = \frac{1}{2}\left|x_{ij}+1\right| - \frac{1}{2}\left|x_{ij}-1\right|$$

where,

$x_{ij} \in R$; State variable of cell *C(i,j)*,

$y_{kl} \in R$; Outputs of cells,

$u_{kl} \in R$; Inputs of cells,

$z_{ij} \in R$; Threshold,

A(i, j ; k, l) ; Feedback operator,

B(i, j ; k, l) ; Control operator.

y_{ij} ; Output of cell *C(i,j)*.

The sphere of influence, $S_r(i,j)$, of the radius r of cell *C(i,j)* is defined to be set of all neighborhood cells satisfying the property

$$S_r(i,j) = \left\{ C(k,l) \,\middle|\, \max_{1\le k\le M,\,1\le l\le N} \{|k-i|,|l-j|\} \le r \right\} \quad (2)$$

The total number of the template parameters in a CNN is 19 when *r=1* (a threshold parameter z_{ij} , 9 parameters a_{kl} , 9

parameters b_{kl}). The general structure of the CNN templates is as follows

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix},\ \mathrm{B}=\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \text{and Z} \quad (3)$$

2.2. ACE16k Chip

ACE16k is a CNN-UM implementation. CNN-UM is an analog and logic computer that consists of many interconnected parallel processor units on its main processor. ACE16K can be basically described as an array of 128x128 identical, locally interacting, analog processing units designed for high speed image processing tasks. The system contains a set of on-chip peripheral circuitries that, on one hand, allow a completely digital interface with the host, and on the other provide high algorithmic capability by means of conventional programming memories where the algorithms are stored [4].

Although ACE16K is essentially an analog processor (computation is carried out in the analog domain), it can be operated in a fully digital environment. For this purpose, the prototype incorporates a bank of Digital-to-Analog (for input) and Analog-to-Digital (for output) converters at the images I/O port [4].

ACE16K is conceived to be used in two alternative ways. First, in applications where the images to be processed are directly acquired by the optical input module of the chip, and second, as a conventional image co-processor working in parallel with a digital hosting system that provides and receives the images in electrical form [4].

2.3. Bi-i Cellular Vision System

The Bi-i cellular vision system which contains ACE16k chip and Digital Signal Processor (DSP) is a high-speed, compact and intelligent camera for training. Most important interface of Bi-i is 100 Mbit Ethernet. Programs to be run on Bi-i are loaded over Ethernet, and the host computer can write or read data to or from the Bi-i over Ethernet. *Instant Vision* Libraries and Bi-i SDK (Software Development Kit) are set of C++ programming library for developing Bi-i applications. These libraries can be used with the development environment for the DSP and ACE16k called Code Composer Studio. Functions in the SDK are operations on different components of the Bi-i hardware such as operating the CMOS sensor. *TACE_IPL* library is an image processing library for ACE16k chip [5] [10].

3. Template Design Tool

In this section, we briefly explain the template design tool. We have developed on chip training system using Iterative Annealing method on procedure shown by the block diagram in Fig.1. Nort-West corner detection template which is used in the object counting algorithm has been trained by this tool. Information about IA method and on chip training procedure are given below. Details of the template design tool can be examined by [12].

3.1. Iterative Annealing

Iterative Annealing (IA), a kind of Simulated Annealing [6], is an optimization method specially developed for CNN [7]. The algorithm of Iterative Annealing is shown below:

1. Choose initial values
   $x_0^k, s_{\max}, j_{\max}, T_0, \tau, j = 0$
2. Calculate step size $\nu = (\tau / T_o)^{j_{\max}/s_{\max}}$
3. $T = T_0$, $i = 0$
4. $y_i^k = x_i^k + u^k.T$ (u^k : Unit distribution U[-0.5, 0.5])
5. If $f(\vec{y}_i) < f(\vec{x}_i)$ then $\vec{x}_{i+1} = \vec{y}_i$
6. Reduce temperature $T = \nu.T$
7. $i = i + 1$
8. If $i < (s_{\max} / j_{\max})$ then Go To 4
9. $j = j + 1$
10. If $(j < j_{\max})$ Then Go To 3

The function $f(\vec{x})$ represents the error measure which must be minimized and $\vec{x}$ represents the D-dimensional parameter vector. $s_{\max}$ is the maximum number of iteration steps and $j_{\max}$ determines how many reruns, have to be carried out. Having the physical effect of Annealing in mind, we call T_0 the start temperature. The minimal temperature τ determines the accuracy of the parameter vector at the global minimum. At every step the temperature is chilling, leading to a decreasing search area until T reaches τ . Then the process restarts with $T=T_0$. Finally a global minimum is found [7].

3.2. On Chip Training with Iterative Annealing

Iterative Annealing (IA) method was modified to work on a PC with the ACE16k chip. This means that we can obtain templates which are stable and robust without inaccuracies of CNNUM hardware realization. ACE16k chip as an external process unit obtains output images for variable template configurations during training process. IA algorithm consists of two loops. The inner loop contains annealing procedure. The outer loop controls iterative behavior. The function to minimize is an error measure calculated between a given reference image R and an output image O obtained from the chip [8]. This function is

$$f(R,O) = \frac{\sum_{i=1}^{a}\sum_{j=1}^{b}\left|r_{i,j} - o_{i,j}\right|}{g * a * b} \quad (4)$$

Here a and b are the image dimensions, $r_{i,j}$ and $o_{i,j}$ the pixel grey values. g is a factor for normalization to obtain values between 0 and 1.

We can modify the Iterative Annealing algorithm to adapt to the chip by adding the following steps into the inner loop: 1. Templates are generated dynamically out of the parameter vector to load and perform them directly on the chip. 2. The chip output image saved for computing the distance to a reference image. This algorithm generates templates using parameter sets. Then, it loads and runs these templates to ACE16k chip, saves

output images and compares them with desired output using error measure function [8].

4. Object Counting Algorithm on Bi-i System

In this section we describe the algorithm for object counting on Bi-i System based on work of Fasih et al [9]. We have developed and adapted the algorithm to ACE16k chip. This algorithm can count objects rapidly in a grey level image because of high-speed offered by Bi-i System. Block diagram of the algorithm is shown in Fig.1. In this algorithm, we use the north-west corners to count the number of the objects. Aim of the first three steps is to prepare the image to detect the corners. The algorithm accepts 128x128 pixels grey level images (Fig.2a.).

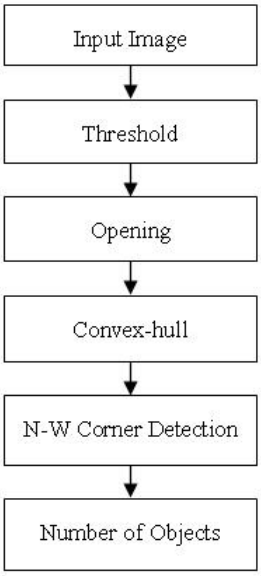
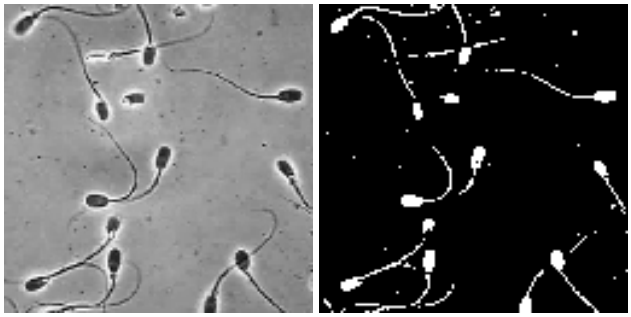


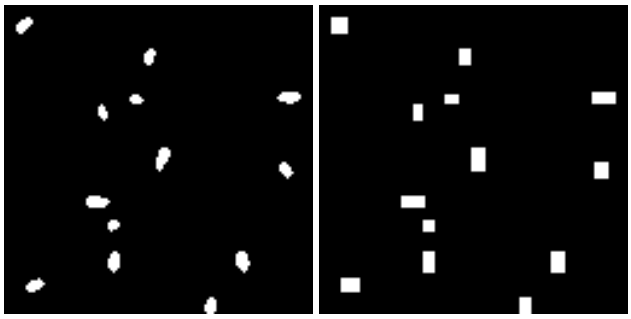
Fig.1. Block Diagram of Object Counting Algorithm on Bi-i System



a) **b)**

Fig.2. a) Input Image b) Threshold Result

In the first step of the algorithm, input image is converted to binary image by the threshold operator called *ConvLAMtoLLM*. This function in SDK Library, converts a grey image in a LAM memory to a binary image in a LLM memory on ACE16k. Pixels above the threshold, are converted to ON, others are converted to OFF [5]. Output image is given in Fig.2b.



a) **b)**

Fig.3. a) Opening Result b) Convex-Hull Result

In next step of the algorithm, small objects on binary image are eliminated by *Opening4* function in SDK Library. These objects are the remains of thresholding operation which also can be called as a *noise*. In addition, this function *Opening4* function performs 4-connectivity binary opening using dilation and erosion functions [5]. Noiseless image is given in Fig.3a.

In the following step, objects on noiseless image obtained in previous step are converted to rectangular objects to find North-West corners easily of each object. This operation is perfomed by *ConvexHull* in SDK Library. This function thickens concave objects to convex ones. The result depends on the type of parameter. Using parameter of *CONVEX_90_DISCON*, the objects on the image are involved into a square [5]. Result of this function is shown in Fig.3b.

After ConvexHull operation, North-West corner detection template is applied on image in order to represent each rectangular object as a corner. We trained this template with the template design tool that we have developed. Training procedure was explained in Section 3.2. In order to detect corners, an input image (Fig.2b.) and trained N-W corner template are sent to ACE16k chip. Output of ACE16k chip gives corners to us. N-W template and corresponding output image of this template are shown below.

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4.76 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 3 & -2.93 & 1.63 \\ -1.87 & 3.51 & -0.19 \\ -0.49 & -0.12 & 2.3 \end{bmatrix} \quad Z = 5.49 \qquad (4)$$

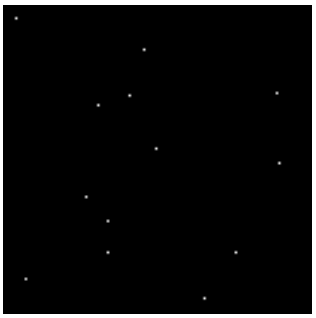


Fig.4. N-W Corner Template Result

Last step of the algorithm is counting white pixels on the image shown in Fig.4. This operation is performed using loop operation (*for* function) in C++ programming language.

3.1. Analysis of Execution time

We have performed the algorithm on different platforms such as ACE16k and DSP in Bi-i and Matlab in order to show the computational power of ACE16k. We have used a PC (Core2Duo 2.0GHz, 2GB RAM). First three steps of the algorithm are implemented on all platforms. Duration of threshold operation is almost same on these three platforms. The shortest execution times of Opening and Convex-Hull operations belong to ACE16k. Template execution step can not be realized on DSP. In addition, template execution time obtained using *MATCNN* (A toolbox for CNN implementations on Matlab) [14] was given as reference; because N-W corner template does not work on Matlab. This template was trained to work on ACE16k.

When total execution time of ACE16k is compared with Matlab's time, computational power of ACE16k chip appears clearly. ACE16k is faster about 60 times than Matlab.

Table 1. Execution times of counting algorithms on different platforms

	ACE16k	DSP	Matlab
Threshold	641 µs	841 µs	670 µs
Opening	425 µs	1673 µs	36000 µs
Convex-Hull	1362 µs	96154 µs	9200 µs
N-W Corner	423 µs	-	390000 µs
Counting	4169 µs	4169 µs	760 µs
Total	7020 µs = 0.007020 s	102837 µs = 0.102 s	436630 µs = 0.436 s

5. Conclusions

We have implemented an object counting algorithm on Bi-i Cellular Vision System and showed how to work on a sample image. We have calculated execution times of this algorithm on ACE16k, DSP and Matlab. Obtained results have shown that the algorithm can rapidly and truly count objects on a given image. However obtained execution times points to high performance of ACE16k chip and shows that Bi-i Cellular Vision System is fast and therefore, easily applicable to image processing algorithms in real time.

6. References

[1] L. O. Chua and L. Yang, "Cellular neural networks: Theory", *IEEE Trans. Circuits Syst., 35,* pp.1257–1272, 1988.

[2] L. O. Chua and L. Yang, "Cellular neural networks :Applications", *IEEE Trans. Circuits Syst., 35*, pp.1273–1290, 1988.

[3] T. Roska and L. O. Chua, "The CNN Universal Machine: An Analogic Array Computer", *IEEE Transactions on Circuits and Systems- II: Analog and Digital Signal Processing, Vol. 40*, pp. 163–173, 1993.

[4] G. Liñàn, R. Domìguez-Castro, S. Espejo, and A. Rodrìguez-Vàzquez, "ACE16k: A programmable focal plane vision processor with 128_128 resolution", *Eur. Conf. Circuit Theory and Design, vol. 1, Espoo, Finland*, pp. 345-348, 28–31 Aug. 2001.

[5] Bi-i Vision System : User Manual.

[6] S. Kirkpatrick, C.D. Gelatt, Jr. and M.P. Vecchi, "Optimization by Simulated Annealing", *Science*, vol. 220, no. 4598, pp.671-679, 1983.

[7] D. Feiden, R. Tetzlaff, "Iterative annealing a new efficient optimization method for cellular neural networks", in *ICIP 2001*, Thessaloniki, Greece, 2001, pp. 549-552.

[8] D. Feiden, R. Tetzlaff, "On-Chip Training for Cellular Neural Networks using Iterative Annealing", in *Microtechnologies for the New Millenium 2003*, Gran Canaria, Spain, 2003, pp.470-477.

[9] A. Fasih, J. Chedjou and K. Kyamakya, "Ultra Fast Object Counting Based-on Cellular Neural Network", *First International Workshop on Nonlinear Dynamcis and Synchronization (INDS´08)*, pp. 181-183, 2008.

[10] A. Zarandy and C. Rekeczky, "Bi-i: a standalone ultra high speed cellular vision system.", *IEEE Circuit and Systems Magazine*, pp. 36-45, 2005.

[11] G. Seiler, "Small Object Counting with Cellular Neural Networks", *Cellular Neural Networks ant Their Applications, CNNA-90*, pp.114-123, 1990.

[12] S. Sevgen, F. Karabiber, S. Arik, "Implementation of On-Chip Training System for Cellular Neural Networks Using Iterative Annealing Optimization Method", *International Symposium on INnovations in Intelligent SysTems and Applications INISTA'09*, 2009 (accepted).

[13] T. Kobayashi, T. Hosaka, S. Mimura, T. Hayashi and N. Otsu, "HLAC Approach to Automatic Object Counting", *Bio-inspired Learning and Intelligent Systems for Security BLISS '08*, pp.40-45, 2008.

[14] www.mathworks.com