Image Processing using CNNs and FPGAs: Initial Results

D. AMANATIDIS, D. TSAPTSINOS, P.R GIACCONE*, G.A JONES*
School of Mathematics, *School of Computer Science
Kingston University
Penhryn Road, Kingston-upon-Thames, Surrey KT1 2EE
UK

Abstract: In this contribution, we propose the use of Cellular Neural Networks as an application for the image segmentation of cinematographic image sequences. The proposed approach is based on a Cellular Neural network cost function that takes into account motion and colour. Cellular Neural Networks are of particular interest for hardware implementation due to the inherent parallelism and initial results using an FPGA simulator are also presented.

Key-Words: Cellular neural networks, image segmentation, optical flow, motion, FPGA, Handel-C

1 Introduction

Regions of image segmentation should be uniform and homogeneous with respect to some features such as pixel brightness or texture. Region interiors should be simple and without many small holes. Adjacent regions of segmentation should have different values, with respect to the feature on which they are uniform. Boundaries of each segment should be simple, not ragged, and must be spatially accurate. Achieving all these desired properties is difficult because strictly uniform and homogenous regions are typically full of small holes and have ragged boundaries. Insisting that adjacent regions have large differences in values can cause regions to merge and boundaries to be lost, producing under-segmented or over-segmented images.

Image segmentation techniques are basically *ad hoc* and differ precisely in the way they emphasise one or more of the desired properties and in the way they balance and compromise one desired property against another. Broadly speaking, there are two approaches to image segmentation. The classical/statistical and the neural/fuzzy approach. Several attempts have been made to develop image segmentation algorithms using neural network models [1,2,3,4]. These algorithms work well even in a highly noisy environment and they are capable of producing outputs in real time. Though many algorithms are available for colour image segmentation, the literature is not that rich as it is for grey level images. Moreover, most of them consider static images and not sequences of them.

The framework of study here is to use a Cellular Neural Network (CNN) for the segmentation of a sequence of colour images. In particular, we are interested in the segmentation of differently moving elements in the images of a film sequence. Segmentation in film post-production is often carried out laboriously by hand, thus making automated segmentation techniques particularly beneficiary to the film industry.

1.1 Cellular Neural Networks

CNN [5] is a multi-disciplinary research area with broad applications in image and video signal processing. CNNs have a continuous-time feature that enables them of processing signals in real time and also they have a local interconnection feature that makes them tailor-made for VLSI implementation. Using CNNs, each cell interacts directly with only neighbouring cells within a specified radius *r*, and through state signal propagation with the remaining network cells. In this way, CNN perform global parallel computation, a feature that makes them very attractive architectures, from the point of view of performance, for image processing applications.

1.2 Reconfigurable Computing

Today, one of the major challenges for both hardware and software engineers and scientists, is that of Reconfigurable Computing. Reconfigurable computers are systems that their hardware architecture can change under software control to suit a different application. The core component of reconfigurable computers is a Field Programmable Gate Array (FPGA), a silicon chip containing a large number of gates as in microprocessors and

RAM chips, whose logical arrangement can be reprogrammed on the fly. Despite that FPGAs were initially introduced sometime ago, only recently a large number of gates can be contained on a single chip. The second key technology is that of hardware compilation. Handel-C, designed by the Oxford Hardware Compilation Group, is such a smart tool. Handel-C is a programming language designed for compiling programs into hardware implementations. A compilation and simulation system has been built around it, which maps Handel-C programs into description of hardware as gate-level netlists. Commercial software can be then used to map these netlists onto FPGAs. Handel-C is closely related to ANSI C, but extended with the parallel and channel communication constructs and arbitrary width variables and expressions. More information can be downloaded from the Celoxica Limited web site: http://www.celoxica.com/

2 Previous Work

Previous work was based on statistical-based approaches for the segmentation of differently moving elements in the images of a film sequence, and in particular the separation of foreground elements, such as actors, from arbitrary backgrounds. In image sequence data, properties of the image such as motion, colour and texture often differ in the foreground and the background. Segmentation techniques have been developed based on the differences of these global characteristics.

2.1 Motion

Visual motion promises to provide a powerful cue to image segmentation. Many of the commonly used motion models are formulated by examining the three-dimensional motion of a point in the view volume. In our case, a depth-independent, 8-parameter linear visual displacement model was derived. After defining the motion model, an appropriate motion estimation procedure was chosen. The classical optical flow procedure [6] was used to generate motion fields between successive images. In order to utilise motion; an iterative estimator was generated taking into account three consecutive frames (Fig.1).

Fig. 1 Three consecutive frames (Cathy sequence)

Global motion estimates were then obtained, representing the motion of the background (the largest independently moving part of the image). The backgrounds of *first-frame* and *last-frame* with that of *middle-frame* were aligned and produced two motion-compensated frames, which when subtracted from *first-frame* and *last-frame* provide images of *motion residuals* known as *forward* and *backward displaced frame differences* (DFDs). Having computed the global motion of the image, each pixel depending on the magnitude of its residuals in the two DFDs is then classified into one of four classes:

- background pixels that belong to the global motion;
- uncovered background pixels that were occluded by the foreground in the previous frame;
- covered pixels that were previously background pixels and now occluded by the foreground and
- foreground pixels that belong to object occluding the background.

Since noise can a have a significant effect on this per-frame classification approach, a temporal dimension was also introduced in the process by including knowledge about the previous classification of a pixel into the calculation of a posterior probabilities. A synopsis of the previous work was highlighted above. The interest reader can consult [7,8] for further information.

3 CNN and Bayesian methods

Experimental comparison of neural networks and Bayesian methods [9] has shown that neural networks do offer a very useful means of classifying images without the need for a statistical model of the features. The major disadvantage is the lack of a clear strategy for choosing between the many training paradigms. A method based on the convergence of the two worlds is therefore desirable.

The results obtained using the statistical-based approach outlined in Section 3 were quite satisfactory. The boundary between foreground and

background was determined accurately and the number of mis-classified pixels was fairly low. However, the iterative nature of the approach meant that it was computationally slow. Here, we propose to use a neural network approach instead. The segmentation problem is posed as an optimisation process based on the same evidence of motion residuals and colour information. A CNN was used to determine the set of pixel labels that minimised a particular cost function, thereby representing the most appropriate labelling of an image.

3.1 Cost Function

The following cost function was derived for each pixel *i* of an image *I* and for each class *k*:

$$\begin{aligned} e(p_{ik}) = & -\sum_{i\in I}\sum_{k\in L} p_{ik}\, p_{ik} \\ & - a\sum_{i\in I}\sum_{k\in L}\sum_{j\in N^t(i)}\sum_{l\in L} C_{ij}^{kl}\, p_{ik}\, p_{jl} \\ & - b\sum_{i\in I}\sum_{k\in L}\sum_{l\in L} T_{kl}\, p_{il}\, p_{ik} \\ & + g\sum_{i\in I}\left(1 - \sum_{k\in L} p_{ik}\right)^2 \end{aligned} \qquad (1)$$

The cost function consists of four terms that constrain the probabilities p_{ik}. The selection pressure coefficient B_{ik} (with range [0,1]) acts an input bias on the neurons, determining which probabilities are to be selected (B_{ik} high) and which suppressed (B_{ik} low). A pixel's selection pressure is based on its motion residuals and colour; e.g., for the covered class B_{ik} is defined as:

$$\begin{aligned} p_{ik} &= p(k = C \mid \acute{a}_i) \\ &= p(e_i^b \mid l_{UN})\, p(e_i^f \mid l_{CH})\, p(c_i \mid l_{\overline{M}}) \end{aligned} \qquad (2)$$

where $\mathbf{\acute{a}_i} = (e_i^b, e_i^f, \mathbf{c_i})$ is the evidence vector for the pixel, made up of the motion residuals at the pixel in the backward and forward DFDs respectively and the colour of the pixel; l_{UN} and l_{CH} are the class associated with unchanged (those having low residual in a DFD) and changed (those having high residual) pixels; and $l_{\overline{M}}$ are pixels belonging to the background of the colour mask.

The second term constrains the labels assigned to pixels *I* and their neighbours, $N^I(i)$, to be compatible. The coefficient C_{ij}^{kl} measures the compatibility between pixels *i* and *j* when assigned labels *k* and *l* respectively. When the labels of neighbouring pixels are compatible the associated probabilities are selected together; when the labels are incompatible, the higher probability is accepted and the lower rejected.

Similarly, the third term seeks compatibility between a pixel's label *k* in the current image I_t and its label *l* in the previous image, where p_{il} is the probability that *l* is the correct label for *I* in the previous image. The coefficient T_{kl} measures the compatibility between the pixel's previous label and each of the possible labels it may have in the current image.

The fourth term (error term) seeks to force the sums of the label probabilities for a pixel to sum to 1. Squaring the deviation from 1 ensures that the term is positive and the resulting cost function is quadratic. The weights a, b, g allow the relative effects of the contributing terms to be controlled.

Expanding the error term and collecting up like terms gives the following form for the cost function:

$$gn - \sum_{i\in I}\sum_{k\in L} C_{ik}\, p_{ik} - \sum_{i\in I}\sum_{k\in L}\sum_{j\in N(i)}\sum_{l\in L} G_{ij}^{kl}\, p_{ik}\, p_{jl} \qquad (3)$$

where

$$N(i) = N'(i) \cup \{i\}$$

$$C_{ik} = p_{ik} + b\sum_i T_{kl}\, p_{il} + 2g$$

$$G_{ij}^{kl} = \begin{cases} -aC_{ij}^{kl} & : \quad j \in N'(i) \\ g & : \quad j = i, k = l \\ 2g & : \quad j = i, k \neq l \end{cases}$$

3.2 Optimisation

Optimisation was performed using a CNN having the following dynamic equation:

$$\frac{du_{ik}}{dt} = \frac{1}{C}\left[I_{ik} - \frac{u_{ik}}{R} + \sum_{i \in I} \sum_{j \in N(i)} G_{ij}^{kl}\, g(u_{ij}) \right] \quad (4)$$

where C and R are respectively the capacitance and resistance associated with the CNN, u_{ij} is the state of the neuron representing label l in pixel j and $g(.)$ is a sigmoid function that converts the state u_{ij} into the probability p_{ij}.

The minimum of the cost function is then obtained by the following algorithm, applied for all i 0 I and for all k 0 ϖ:

1. Initialise u_{ij} to $g(B_{ik})$
2. Solve the dynamic equation for u_{ij} using a Runge-Kutta update procedure
3. Compute $p_{ik} = g(u_{ik})$
4. Evaluate the cost function e_{ik}^{t} for the given probabilities, where t is the current iteration
5. If the relative error $\left| e_{ik}^{t+1} - e_{ik}^{t} \right| / \left| e_{ik}^{t+1} \right|$ in the cost function is greater than some fixed percentage (e.g., 1%), return to step 2.

Figure 2 shows the label images for Kathy, initially and after 100 iterations.

Fig. 2 Label images - initial and after 100 iterations

4 CNN Simulation using Handel-C

Some results of experimenting with the Handel-C compiler are presented in this section. The task is edge detection of 256 gray-level images (synthetic and natural) with various sizes. The two methods under discussion are: A traditional edge detection algorithm based on thresholding (threshold value of 16) the gray level difference of neighbouring pixels, modified to work with images of arbitrary size and a CNN edge detection algorithm.

The three 8-bit test bitmaps (hardware, Lena and lighthouse) can be seen in Figure 3. Their size is 320x240, 128x128 and 80x60 respectively. Handel-C compiler was running on a Pentium II, clocked at 233 MHz with 128 MB of main memory and 4MB of graphics memory.

Fig. 3 Test images

Parameters for the experiments were: the amount of hardware generated; the total number of clock cycles; the amount of real time needed. In all cases, the simulator was set to display the state of the program variables once every 1000 cycles, so the total number of cycles is the truncated result of the true number. Figures 4, 5 and 6 show the results obtained with the two methods. The original images are on the top left corner, the outputs from the thresholding method are on the top right corner, and the bottom images are the outputs from the CNN algorithm after 1 and 3 iterations.

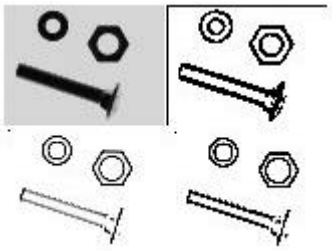

Fig. 4 Hardware results

Fig. 5 Lena results

Fig. 6 Lighthouse results

The thresholding edge detection is faster and produces less hardware. However, with the present hardware design technology standard, a number such as 2000 gates is not a major problem. The compensation for the CNN method is its polymorphic character. By changing the template set the same program can have a completely different functionality. It can also be run in real-time, interactive way. We can monitor the output on every iteration and stop when we think that the result is satisfying, for instance if we would prefer stronger or weaker edges. It can also be tuned for a number of variables, such as the bias or the integration stepsize, whereas in the traditional approach there is only the threshold value.

5 Future Work and Conclusions

The purpose of this contribution was to present the work undertaken so far. Although the area of image segmentation is well explored not much has been reported about sequences of colour images. The CNN approach looks quite promising and we expect that the FPGA implementation will accelerate the performance of the network.

References:

[1] K.J. Cios and I. Shin, Image recognition neural networks: Irnn, *Neurocomputing*, Vol.7, 1995, pp. 159-185.

[2] A. Ghosh, N.R. Pal and S.K. Pal, Image segmentation using a neural network, *Biological Cybernetics*, Vol.66, 1991, pp. 151-158.

[3] A. Goltsev, Brightness image segmentation by a neuron-like network, *Neurocomputing*, Vol. 4, 1992, pp. 9-15.

[4] C.C. Lee and J. Pineda de Gyvez, Color image Processing in a Cellular Neural Network Environment. *IEEE Transactions on Neural Networks*, Vol.7, No.5, 1996, pp. 1086-1098.

[5] L.O Chua, *CNN: A paradigm for complexity*, World Scientific Series on Non-linear Science, 1998

[6] M. Sonka, V. Hlavac and R. Boyle, *Image Processing, Analysis and Machine Vision*, PWS Publishing, 1990

[7] P.R. Giaccone and G.A Jones, Spatio-temporal approaches to the computation of optical flow, *Proceedings of the British Machine Vision Conference*, Colchester, UK, 1997, pp. 420-429.

[8] P.R. Giaccone and G.A Jones, Segmentation of global motion using temporal probabilistic classification, *Proceedings of the British Machine Vision Conference*, Southampton, UK, 1998, pp. 619-628.

[9] K. Richards and G.D Sullivan, Colour and texture in cloud identification: An experimental comparison of neural network and bayesian methods, *Proceedings of the British Machine Vision Conference*, Guildford, UK, 1993, pp. 468-478.