HAND VEINS FEATURE EXTRACTION USING DT-CNNS

Suleyman Malki[a] and Lambert Spaanenburg[a]
[a]Lund University, Dept. of Information Technology, P.O.Box 118, 22100 Lund (Sweden)

ABSTRACT

As the identification process is based on the unique patterns of the users, biometrics technologies are expected to provide highly secure authentication systems. The existing systems using fingerprints or retina patterns are, however, very vulnerable. One's fingerprints are accessible as soon as the person touches a surface, while a high resolution camera easily captures the retina pattern. Thus, both patterns can easily be "stolen" and forged. Beside, technical considerations decrease the usability for these methods. Due to the direct contact with the finger, the sensor gets dirty, which decreases the authentication success ratio. Aligning the eye with a camera to capture the retina pattern gives uncomfortable feeling. On the other hand, vein patterns of either a palm of the hand or a single finger offer stable, unique and repeatable biometrics features.
A fingerprint-based identification system using Cellular Neural Networks has already been proposed by Gao. His system covers all stages of a typical fingerprint verification procedure from Image Preprocessing to Feature Matching. This paper performs a critical review of the individual algorithmic steps. Notably, the operation of False Feature Elimination is applied only once instead of 3 times. Furthermore, the number of iterations is limited to 1 for all used templates. Hence, the computational need of the feedback contribution is removed. Consequently the computational effort is drastically reduced without a notable chance in quality. This allows a full integration of the detection mechanism. The system is prototyped on a Xilinx Virtex II Pro P30 FPGA.

Keywords: Network on a Chip, Cellular Neural Network, Field-Programmable Gate-Array, Image Processing, Hand-Vein Recognition

1. INTRODUCTION

Modern security systems have to provide fast, accurate and robust personal identification, which implies moving away from traditional and unreliable methods such as PIN codes and smart cards. The use of electronically stored records of human biometrics features seems promising, but national laws on Personal Information Protection enforce the early introduction. The first products have used fingerprints and retina patterns. Fingerprints have been strongly debated because they can be easily forged; the general audience finds iris scanning very intrusive. For such reasons the quest for other means is continuing. Lately first products have been released that are based in hand and finger veins.

In 2004 Hitachi has announced a small device for finger vein authentication[1]. It is based on near-infrared images of veins at the under side of fingers. Light is transmitted from a set of LEDs through the finger to capture a high contrast image, which in turn is matched with a pre-registered profile. The competition from Fujitsu is based on the fact, that the reduced hemoglobin that runs through the hand veins absorbs near-infrared light[2]. Again, light is transmitted from a set of LEDs and runs through the palm into the image sensor. Vein patterns are unique, even for twins, and are not accessible for fraud. Currently these devices are widely gaining acceptance and competition.

In current technology, image matching is used and therefore the application is limited to person authentication. To provide for a more general use, a potentially large database of signatures has to be searched. Suitable features have to be defined and measured in each image to allow for a fast database search. It has already been shown for fingerprint patterns that such features can be found using Cellular Neural Networks (CNN)[3-5]. The system covers all stages of a typical fingerprint verification procedure from Image Preprocessing to Feature Matching. This paper follows the footsteps of the impressive progress achieved by Gao[3-5], by investigating the usage of Discrete-Time CNNs[6] to accomplish the task of hand veins feature extraction. For capturing images of hand veins, we follow the approach used by Hitachi[1]. Near-infrared rays generated by means of LEDs penetrate the hand and are absorbed by the hemoglobin in the blood. Thus, the veins (where the blood flows) appear as dark areas in an image taken by a CCD camera. Then image processing

reconstructs a hand-vein pattern from the camera image. This paper focuses on extracting the features of hand veins from this pattern. The implementation stresses exploiting FPGA as the realization target.

The paper is composed as follows. First the basics of image processing in hardware are reviewed.. Then we discuss the hand vein recognition process, followed by analysis and verification of this process in Matlab. Ensuing we discuss some basic dedicated CNN implementation styles to support the process and conclude that a bit-serial approach with added some basic arithmetic operations will be needed.

2. IMAGE PROCESSING

Image processing has long been studied from an algorithmic perspective. The rising complexity of such algorithms has caused a renewed interest in specialized architectures to boost the performance of actual applications. This has brought the coming of smart cameras, stressing even more the need for hardware that can run normal programs but is especially good in image processing. Typically image-processing hardware differs from normal architectures by the need for vector processing with the attached importance of closing the memory/processor band gap. An early example is the V-IRAM, taking the lessons of the early CRAY vector processor[7]. Overall we find two arrangements dominating the field: the 1-dimensional Very-Long Instruction Word (VLIW) and the tiled architecture. The former excels in the parallel execution of overlapping pixel lines by a single instruction. For instance, the recent IC3D chip has 64 line memories for this purpose[8]. The latter aims for the parallel execution of small programs on groups of pixel data. The recent DaVinci chip family gives an example[9].

An analysis of the hardware needs for the pixel operations in classical image processing is given by Lundgren[10]. It finds that the VLIW architecture is a minimal requirement, together with a broadcasting scheme to bring the pixel information to the individual execution units. Unfortunately the analysis does not take non-linear image processing algorithms, such as the Cellular Neural Network[11], into account. A pipelined VLIW approach can also accommodate a CNN[12], but still tiling brings more flexibility[13].

Dedicated hardware for image processing is based on Low-Level Algorithms that need hardware support[10]. *Linear Filtering*, also called convolution, is often one of the first steps in going from pixel-level data to a higher level of information. In Linear Filtering, every output pixel is a linear combination of the corresponding input pixel and this pixel's closest neighbors. The corresponding input pixel and the pixels in neighborhood that affects the output pixel is called a footprint. The footprint is typically a square with a side size N where N is an odd integer. To describe the linear combination of the pixels in the footprint we use a matrix called a kernel. By placing the kernel over the input image, each kernel value can be multiplied by each underlying image value, and by summing up these products; the value of the output pixel is set. The complete output image is gained by sliding the kernel over the image, and each kernel position corresponds to a single output pixel mapped at the center of the kernel. The process is illustrated in Figure 1.

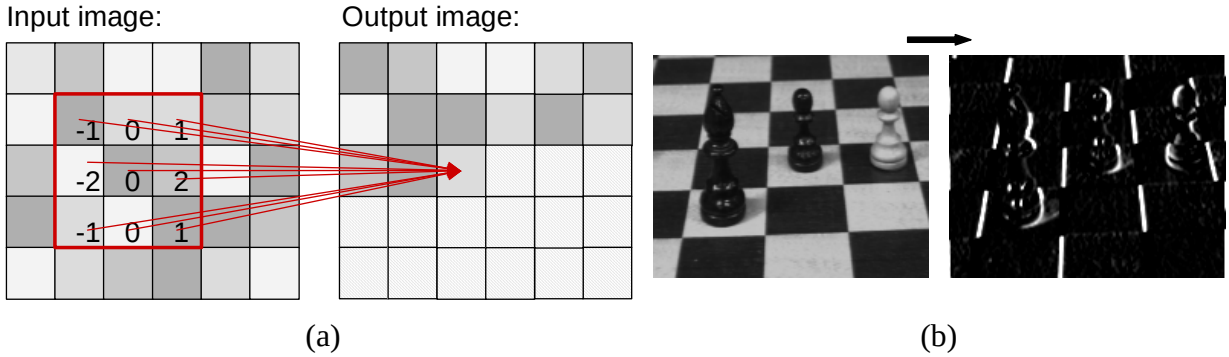

Figure 1 Principle of input-to-output mapping using a 3x3 linear filter (a) and (b) application by a linear Sobel filter.

Examples of other neighborhood-processing algorithms are in morphology and in Median filtering. Next to that, we find in image processing also a range of standard arithmetic operations like addition, absolute value or invert using one or two images. Two images can be added or subtracted pixel-wise from each other, or a constant together with one image can be used. A representative list over the operations is given in Table 1. The mathematical operation is given within

brackets where *src1* denotes the first image and *src2* denotes the second image. All operations are pixel-wise. Such low-level operations are a suitable starting point for more complex ones, such as histogram computation, image correction, min/max computation and moment's computation[10]. In this paper, we focus on the use of such operators within the framework of Cellular Neural Networks (CNN).

Table 1. Arithmetic operations on 1 or 2 images. The mathematical meaning is given in brackets.

One Image	**Two Images**
ABS (|src|)	ADD (src1 + src2)
INVERSE (src != c)	SUB (src1 - src2)
SELECT (src == c)	ABSDIFF (|src1 - src2|)
THRESHOLD (src < c)	AND (src1 & src2)
DOUBLE THRESHOLD (src < c1 & src > c2)	WEIGHTED SUM (c1•src1 + c2•src2) >> c3
	INF (src1 < src2)
	SUP (src1 > src2)
	EQUAL (src1 == src2)

Since their invention, different considerations for cell complexity, cell dynamics and network topology led to the emergence of many CNN models. One of these models is the discrete-time version, the DT-CNN introduced by Harrer and Nossek[6]. A DT-CNN is a regular multidimensional grid of locally connected cells. In a two dimensional DT-CNN, each cell c, which is identified by its position in the grid, communicates directly with its r-neighborhood, i.e. a set of cells within a certain distance *r* to *c*, where $r \geq 0$. For instance, if *r =1* we have a 3×3 neighborhood while in the case of *r =2* a 5×5 neighborhood is obtained. Nevertheless, a cell can communicate with other cells outside its neighborhood due the network propagation effect.

The state of a cell c, denoted x^c, depends mainly on two factors: the time-independent input u^d to its neighbors *d* and the time-variant output $y^d(k)$ of these neighbors. The neighborhood always includes *c* itself. Equation (1) describes this dependence in a discrete time *k*. Output of cell *c* at a certain time step is simply obtained by thresholding the state of the cell.

$$x^c(k) = \sum_{d \in N_r(c)} a_d^c y^d(k) + \sum_{d \in N_r(c)} b_d^c u^d + i^c \qquad \textbf{(1)}$$

Feedback coefficients a_d^c mirror the contribution of neighboring cells' outputs. While control coefficients b_d^c describe the dependency on neighboring cells' inputs, the cell bias i^c is added to adjust the threshold. $N_r(c)$ denotes the r-neighborhood of cell *c*. In short, spatially invariant CNNs are specified by a feedback template $\boldsymbol{A}$ containing a_d^c, a control template $\boldsymbol{B}$ containing b_d^c and finally the cell bias $i = i^c$. These three factors constitute template of the node, $T=<\boldsymbol{A}, \boldsymbol{B}, i>$. The node template determines, together with input image *u* and an initial output *y(0)*, completely the dynamic behavior of the DT-CNN. Each node, and consequently the whole network, will converge to a stable state after a number of iterations *n*, i.e. at time $k = n$.

Both analogue and digital realizations of a CNN have been published[14,13]. The former have a larger pixel capacity and can easily be merged into a focal array, but will be limited in accuracy. The latter have a smaller pixel capacity, but can in principle operate at a quasi-infinite accuracy. Both have their applications, as some have less and some have more computational demands. In the following we will coarsely overview the hand-vein application[15] and subsequently discuss the need for specialized hardware, based on a dual-datapath with support of both linear and non-linear operations.

3. FEATURE EXTRACTION

Normally, the captured hand-vein pattern is gray-scale and subject to noise. Noise Reduction and Contrast Enhancement are crucial to ensure the quality of the subsequent steps of feature extraction[16]. This is achieved by means of:

Binarization that transforms the gray-scale pattern into a black and white image, *Skeletonization* that reduces the width of lines to one pixel and finally *Isolated Pixel Removal* that eliminates the unwanted isolated points. These three steps constitute the procedure of image preprocessing, as shown in Figure 2. In the following we will follow the lines set out by Gao for fingerprint recognition[3-5] and subsequently applied to hand veins by Malki[15].

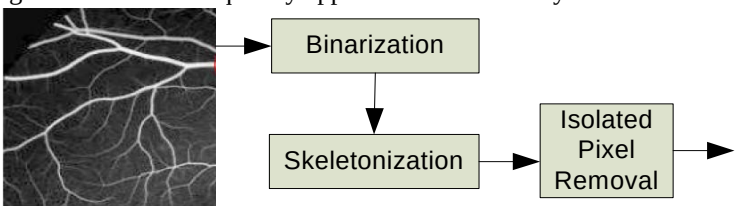

Figure 2 Image Preprocessing

Binarization is based on thresholding. The algorithm of skeletonization performs iteratively in 8 subsequent steps, where each step peels one layer of pixels in a certain direction. As one iteration is accomplished, the pattern is one pixel thinner in all directions. The algorithm stops when no difference between the input and the output is obtained. The order in which the templates of skeletonization are applied influences the type, the number and the direction of extracted features. The unwanted isolated pixels are removed by applying the template of Isolated Pixel Removal. Similar to fingerprint images, hand vein patterns have two main features: endings and bifurcations. The former is the end point of a thinned line, while the latter is the junction point of three lines. Figure 3 illustrates the idea.

Figure 3 Hand vein features: endings and bifurcations

Feature matching of two hand-vein patterns depends on the type, the location and the direction of endings and bifurcations. It is important to point out the existence of false features i.e. two false endings, two false bifurcations or one false ending and one false bifurcation, due to the noise in the original image and artifacts introduced during the procedure of image preprocessing[4]. This implies the introduction of the step False Feature Elimination. The overall design is shown in Figure 4. The detection of bifurcations and endings in the preprocessed image can be carried out in parallel. The intermediate results are added together by a simple Logical OR operation[17] before false features are eliminated. The end of a thinned line has only one black pixel within its neighborhood. As all isolated pixels are already removed during the preprocessing, ending points are easily extracted by applying the template of *Ending Detection* once.

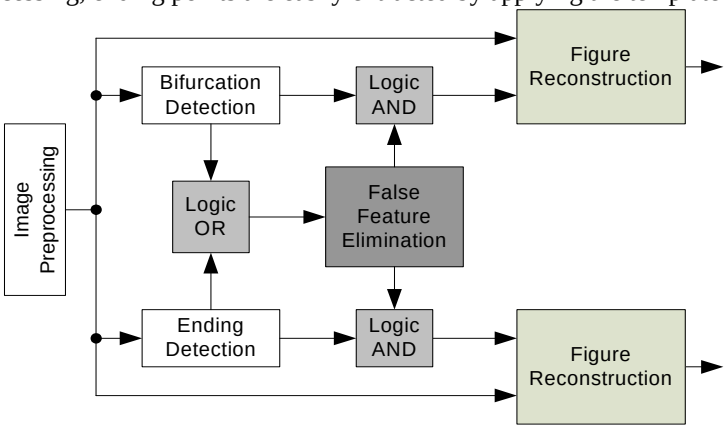

Figure 4 Block diagram of the hand-vein feature extraction

Similarly, *Bifurcation Detection* extracts all points that have at least 3 black pixels within the neighborhood. Three different types of junctions do exist: “real” points, T- and Corner-forms. Extracting real bifurcations from the T- and Corner-forms needs further treatment. We follow the approach introduced by Gao[4] and use the template of Junction Point Extraction that extracts the real junction points but keeps the T- and Corner-forms.

Junction points in T- and Corner-forms are extracted by removing all real bifurcations. Then the template of Isolated Point Extraction is applied in parallel and the result is added to the outcome by means of a Logical OR operation[17]. The order of operation in the procedure of bifurcation detection is depicted in Figure 5.

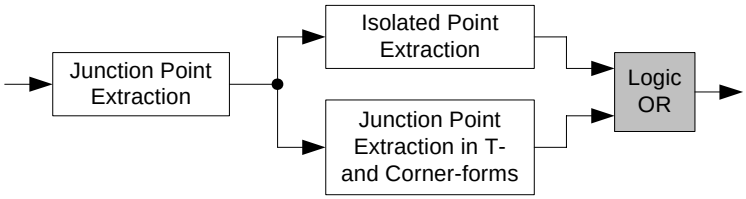

Figure 5 Bifurcation detection uses three different templates in addition ot a Logic OR operation

As mentioned before, we employ Gao's approach[4] to eliminate false endings and bifurcations. In order to remove all false points that are separated by a distance $d \leq n$, it performs the operations dilation and erosion for n/2 iterations each. The block diagram in Figure 6 shows the sequence of the different operations. The fact that two false features are usually close to each other[4], implies the use of low value of n. Actually, in our case n=2 is sufficient. Thus, one iteration is enough for each of the operations.

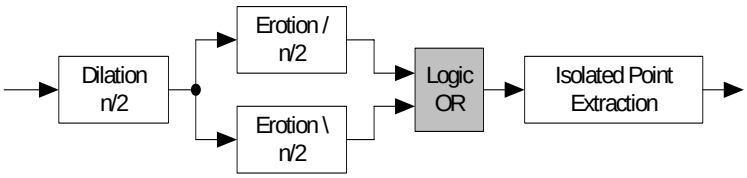

Figure 6 Operations involved in False Feature Elimination

So far, the extracted bifurcations and endings are represented as single points. Thus, only the location of every ending and bifurcation is obtained so far. In order to perform the procedure of Feature Matching, the direction of each feature needs to be known. The template of Figure Reconstruction takes the original image as input and the intermediate image (with the extracted feature) as initial output in order to reconstruct the feature to the limit that makes it comparable. The number of iterations determines the number of pixels that are restored of the three lines leaving a bifurcation and the only line leaving an ending.

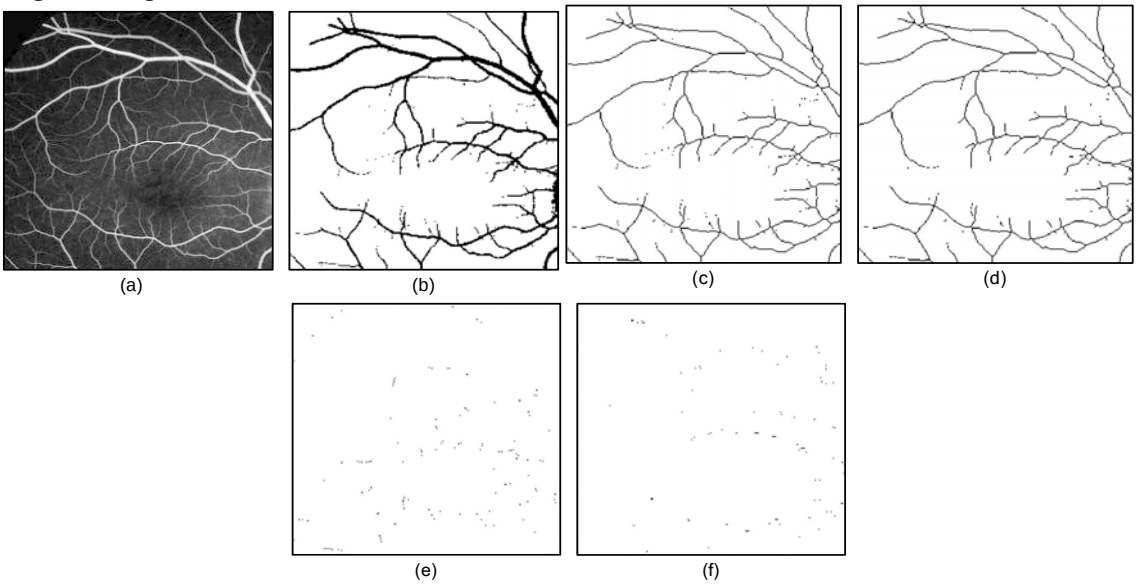

Figure 7 Original image containing the vein pattern is shown in (a). This gives (b) after binarization, (c) after skeletonization, (d) after isolated pixel removal, leaving (e) the ending and (f) the bifurcations.

4. ANALYSIS AND VERIFICATION

Matlab has been widely adopted in academic and industrial communities as an interactive software packet that uses matrices to perform heavy calculations. Furthermore, the results are presented using advanced graphic. Thus, Matlab provides an easy-to-use and feasible environment, on which verification of the aforementioned algorithm is carried out. We start with the image in Figure 7.a, where a pattern of veins is captured. Applying the first operation of preprocessing, i.e. *Binarization*, yields in a black and white image (Figure 7.b). The binary image serves as input to the sequence of *Skeletonization* templates that is applied iteratively 7 times to get the line-thinned image shown in Figure 7.c. As this image undergoes the operation of *Isolated Pixel Removal*, all unwanted isolated points are removed, which is depicted in Figure 7.d. As the stage of preprocessing is accomplished, we move on to the first stage of feature extraction. Ending Detection produces the image shown in Figure 7.e, while Figure 7.f is obtained by means of Bifurcation Detection.

The subsequent stages from eliminating false feature in the ORed image to the reconstruction of bifurcations and endings result in the images shown in Figure 8. The operation of *False Feature Elimination* is crucial for the accuracy of the overall algorithm, as the number of false features as well as the total number of extracted features is affected. A careful look at Figure 8.d shows that 28 of the 30 extracted bifurcations are real while 2 are false features. As the original image has 43 bifurcations in total, the detection rate is 65% with a false detection rate of 5% when the pixel distance n equals 2. By removing the operation of False Feature Elimination, the detection rate is raised to 100% but the false detection rate is also increased to 21%.

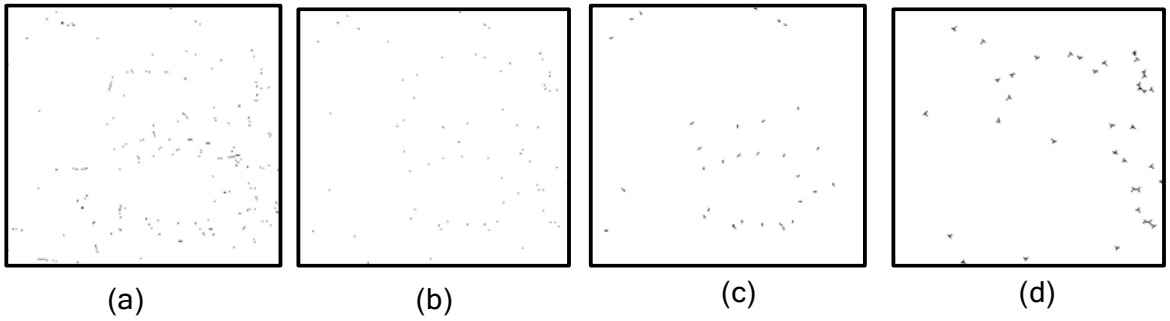

Figure 8 Adding the images with ending and bifurcation points by applying the operation of Logic OR (a) before eliminating the false features (b). Reconstruction of endings (c) and bifurcations (d).

5. HARDWARE IMPLEMENTATION

Two alternatives to build the CNN node appear in literature. In the *state-flow architecture*, we have a pipeline of single multiply-accumulate units (Figure 9 left). Each performs only an operation on a single coefficient/input pair and then moves the result to the next multiply-accumulate unit. This means, that we evaluate each neighboring pixel separately in a pipelined fashion, doing in series as many multiply-accumulates as there are cells in the neighborhood. Interweaving three pipelines, corresponding to a row of three pixels, can reduce the latency (Figure 9 middle). In the *state-scan architecture*, we actively scan the neighborhood for the inputs. In a pre-determined schedule, the coefficient/input pairs are fed through a single multiply-accumulate unit in sequence.

For the application described in this paper we have originally used a packet-switched CNN implementation of the state-scan architecture (Figure 9 right), dubbed Caballero[18]. It employs the approach of switched broadcasting, where packets are transmitted among cells, within a certain neighborhood, using a predefined communication pattern. Thus, the intercommunication is not hard-wired which allows larger neighborhood and unlimited number of iterations. Going towards a time-multiplexed bus can alleviate the communication bandwidth limitation, where the bits of all nodal values are transferred in sequence. To connect this to the existing Caballero node requires buffering to create series/parallel conversion and vice versa. This will be part of the Network Interface (NI) that wraps any design part to become accessible through the network standard. This brings out the basic advantages of the time-multiplexed communication, and is fully in-line with the original Aethereal systematic. But it also presents a degree of overhead that needs to be minimized. Therefore it demands investigation, how the concept of serial processing can be moved further into the node.

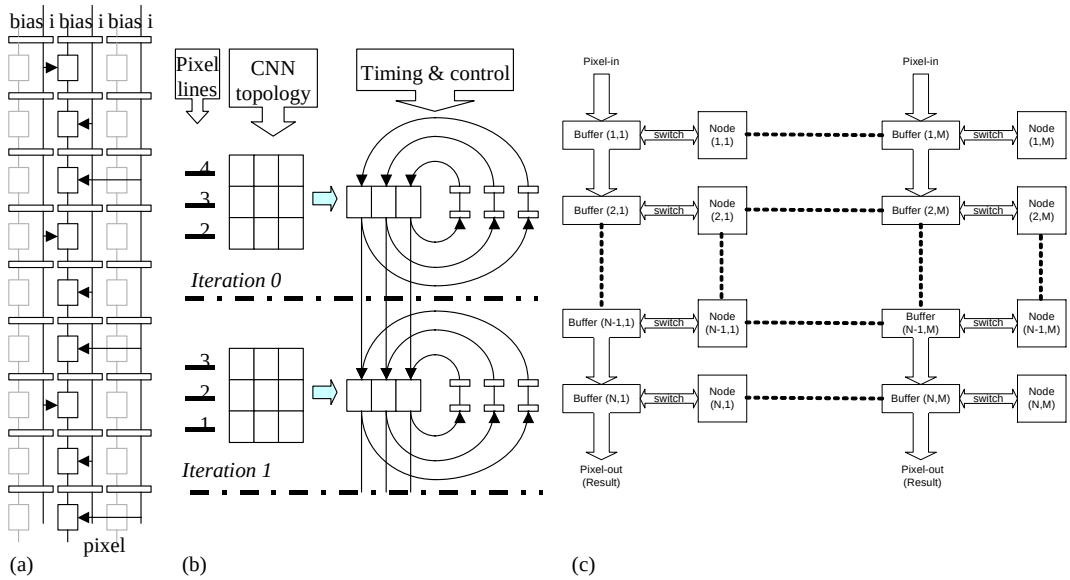

Figure 9 Different state-flow architectures, operating directly on the pixel pipeline (a and b) and c) a state-scan architecture.

For every step, the coefficient bits multiply the single-bit input from each neighbor; the results are added and accumulated. As usual in bit-serial logic, the data-path becomes small but at the expense of a more complicated control. Furthermore we can expect a higher latency, as more clock ticks are needed to get to the result. But that is only true for the single node. The basic 10 clock cycles for a single node have to be repeated for 5 neighboring nodes for reason for bus contention. It does not seem likely that a serial solution that eliminates such bus contention problems will need more. As the small serial node allows for a larger network to be implemented on a single chip, it is worthwhile to evaluate its potential.

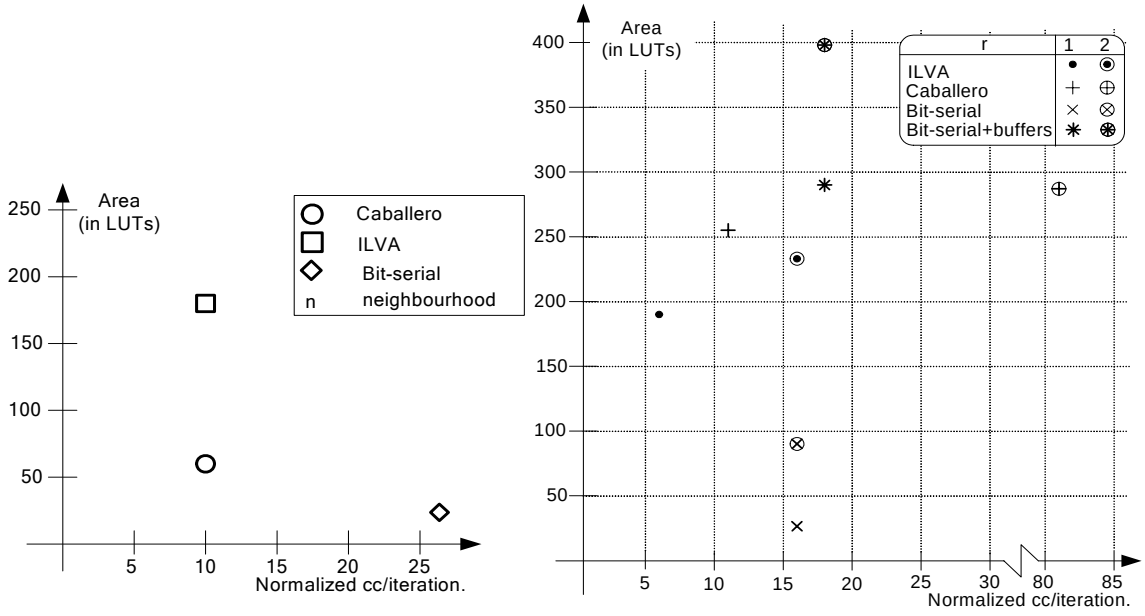

Figure 10 Comparison of different 8-bit architectures (a) without and (b) with taking the communication overhead into account.

Bit-serial schemes provide a scalable architecture, where the internal number representation is not directly reflected in the width of the data-path. This allows for a next degree of freedom, as the inputs and the states of the CNN nodes can be communicated in different accuracies. Moreover such accuracies can be changed on the fly. An alternative is the series/parallel scheme, where an even wider range of area/speed trade-offs is possible. On first sight, it seems that

handling the bits serially leads to maybe to a smaller node, but the price to pay in terms of performance may be too large. Figure 10 shows that this is indeed true for the node itself, but the difference disappears when the communication overhead is taking into account. The main benefit of the serial architecture is the fact that the node is small enough to be duplicated as well as extended with the conventional arithmetic operations. Consequently the algorithmic need to handle two different operations on an image followed by a unifying arithmetic operation does not give rise to complications in the hardware, like additional buffering and/or reloading the image. This algorithmic structure is quite common in CNN applications, and it seems that a bit-serial architecture is more than fitting for such problems.

6. CONCLUSION

Current vein-based products allow single person authentication in less than a second. Being contact-less and non-intrusive they have a growing appeal in the security market in combination with other techniques. The paper describes a hardware implementation that allows vein feature extraction in the order of milliseconds. This efficient pre-processing of vein images severely reduces the overhead in searching large databases and therefore makes real-time person identification a reality. This allows casual operation without the support of other identification techniques for such applications where access limitations must be enforced for public areas, such as for shoplifters in stores and hooligans in stadiums.

REFERENCES

1. Hitachi Engineering Co. Ltd., *About Finger Vein*, http://www.hitachi-hec.co.jp/english/about_fv.htm, March 10, 2006.
2. Fujitsu R&D, *Fujitsu Palm Vein Technology*, http://www.fujitsu.com/za/about/rd/200506palm-vein.html.
3. Q Gao, P. Förster, K. R. Möbus and G. S. Moschytz, "Fingerprint Recognition Using CNNs: Fingerprint Preprocessing, ", *IEEE International Symposium on Ciicuits and Systems*, vol. 2, pp. 433-436, 2001.
4. Q Gao and G. S. Moschytz, "Fingerprint Feature Extraction Using CNNs," *European Conference on Circuit Theory and Design*, pp. 97-100, Espoo, Finland, 2001.
5. Q Gao and G. S. Moschytz, "Fingerprint Feature Matching Using CNNs," *ISCAS*, pp. 73-76, 2004.
6. H. Harrer and J. A. Nossek, "Discrete-Time Cellular Neural Networks," *International Journal of Circuit Theory and Applications*, vol. 20, pp. 453-467, 1992.
7. C.E. Kozyrakis and D.A. Patterson, "A New Direction in Computer Architecture Research", *IEEE Computer*, Vol. 31, No. 11, pp. 24-32, 1998.
8. Fatemi H. et alieni, "Real-Time Face Recognition on a mixed SIMD VLIW Architecture", *Proceedings Progress*, Nieuwegein, The Netherlands (October 2003) 78-83.
9. Gene Frantz, "DaVinci set to drive the future of digital video", *Embedded Systems Europe* (October 2005).
10. A. Lundgren, *Design of a Co-processor that implements several specific smart imaging algorithms*, M.Sc. thesis, Lund University (Lund, Sweden) 2004.
11. L.O. Chua, and L. Yang, "Cellular Neural Networks: Theory", *IEEE Transactions on Circuits and Systems*, Vol. 35, No. 10 (1988) 1257-1272, 1273-1280.
12. Z. Nagy and P. Szolgay, "Configurable Multi-Layer CNN-UM Emulator on FPGA", *Proceedings 7th IEEE Workshop on CNNs and their Applications*, World Scientific (Singapore), pp. 164 – 171, 2002.
13. Suleyman Malki, *Discrete-Time Cellular Neural Networks Implemented on Field-Programmable Gate-Arrays to Build a Virtual Sensor System*, Lic. Thesis, Lund University (Lund, Sweden), 2006.
14. G. Linán et alieni, "ACE4k, An analog I/O 64_64 visual microprocessor chip with 7-bit analog accuracy", *Int. J. Circuit Theory Applicat.*, Vol. 30, No. 2/3, pp. 89–116, 2002.
15. S. Malki, S., Y. Fuqiang, and L. Spaanenburg, "Vein Feature Extraction using DTCNNs", *Proceedings 10th IEEE Workshop on CNNA and their Applications* (Istanbul, August 2006) pp. 307 – 312.
16. A. Jain, R. Bolle and S. Pankanti, *Biometrics: Personal Identification in Networked Society*, Kluwer Academic Publishers, 1999.
17. Tamás Roska et alieni, *CNN software Library (templates and algorithms), vers. 7.3*, Tech. Rep. DNS-CADET-15, Analogical and Neural Computing Laboratory, Computer and Automation Research Institute, Hungarian Academy of Science, 1999.
18. S. Malki, L. Spaanenburg, "On Packet-Switched Implementation of Discrete-Time CNN," *Euromicro symposium on Digital System Design*, Rennes, France, pp. 234-241. 2004.