Binary Object Recognition System on FPGA with bSOM

Kofi Appiah, Andrew Hunter, Patrick Dickinson and Hongying Meng
Lincoln School of Computer Science
University of Lincoln
Brayford Pool
Lincoln LN6 7TS
England
Email: {kappiah,ahunter,pdickinson,hmeng}@lincoln.ac.uk

***Abstract*—This paper introduces the implementation of an FPGA-based tri-state rule binary Self Organizing Map (bSOM), which takes binary inputs and maintains tri-state weights, with a node labelling algorithm which makes it capable of object classification. The bSOM is used for appearance-based object identification during tracking in video sequences. It is designed to provide part of an end-to-end surveillance system implemented wholly on FPGA. It is trained off-line using a labelled training data set for nine objects, using binary signatures extracted from the colour histogram, and successfully used for appearance-based identification of objects in approximately 85% of cases in a fairly challenging data set. The paper identifies how this preliminary work can be extended to provide full on-line appearance-based identification and tracking.**

I. Introduction

This paper addresses the issue of appearance-based identification of objects in video-based surveillance [1] using Field Programmable Gate Arrays (FPGAs). Identification is the process of recognizing individual objects so that they may be successfully tracked. This includes frame to frame tracking, tracking through occlusions, re-acquiring tracks on objects which leave the scene and re-enter, and tracking across multiple cameras. Identification is typically performed as part of a pipeline that begins by segmenting objects, uses identification to trace individual objects, and then applies further processing to the resulting activities. Identification is performed by extracting a variety of appearance-based features from objects, which may include: size, shape, intensity, colour and texture features. A robust identification system should be invariant to changes of lighting condition, viewing angle and distance from camera, and should be able to operate in real-time.

The research in this paper forms part of the development of a system for real-time surveillance that is designed to be fully implemented on FPGA. There are significant cost and complexity advantages in realizing a full system on a single reconfigurable architecture. The early stages of the final system, which detect and segment objects (using background differencing and connected components analysis) are described in [2], although in the current work we perform these stages using a CPU-based system [3], [4]. For efficient FPGA-based identification we use binary signatures (appearance feature vectors) which are easily processed.

We use a tri-state rule binary Self Organising Map (bSOM) based on the system described in [5], modified to perform identification. The bSOM is capable of a wider variety of anomaly detection and classification tasks, justifying its use for identification among other tasks, while the FPGA based implementation makes it possible to design a single integrated on-chip system. Section II presents related work in this area, followed by a workflow of the bSOM identification system in section III. Result of experiments conducted on the system are presented in section IV, followed by the FGPA based implementation in section V. We conclude with some comments on future work in section VI.

II. Related Work

Identification is closely related to classification, as an identification system for N known objects may be implemented using a classifier. Sang et al. [6] used a Bilateral Weighted Linear Discriminant Analysis to classify objects which automatically adjusts to different scenes. Similarly, Bose et al. [7] used scene-specific context features, such as position and direction of motion to train a scene-invariant classifier to identify vehicles and pedestrians. Zhang et al. [1] used an appearance-based method to achieve real-time robust object classification in diverse camera viewing angles on detected moving objects. Landabaso et al. [8], presents an object classification system, capable of distinguishing between a single person, group of people or a vehicle.

Takala et al. [9] represented colour cues and texture properties with local binary patterns to build a unifying distance measure, which is subsequently used for tracking and event classification. In [10], Ling et al. presents a colour-based object tracking system using colour information extracted cluster-by-cluster from moving objects to compare the object's similarity across image sequences. Cho et al. [11] presents a parallel architecture for face detection using AdaBoost algorithm [12], which identifies a sequence of Haar-classifiers [13], [14] to indicate the presence of a face. The FPGA based face detector has performance improvement up to 37.39 times the software based version for a VGA sized image. Wei et al. [15] presents an FPGA based a real-time face detection using AdaBoost

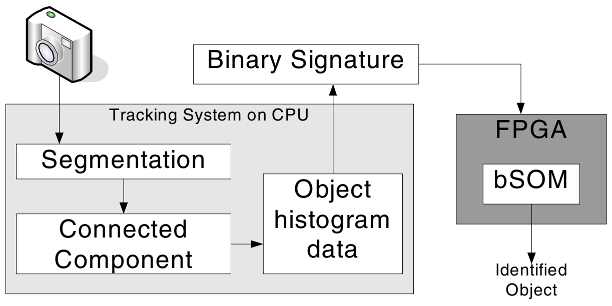

Fig. 1. An overview of the FPGA based object recognition system. Moving objects are segmented and tracked to extract their corresponding binary signatures, which is then fed onto the FPGA to identify the object.

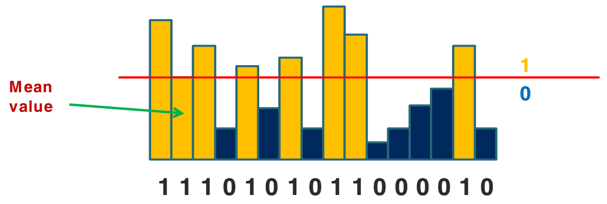

Fig. 2. A sample 16 bin histogram, showing how the binary feature vector can be extracted. The bins in yellow (grey) are the bins greater or equal to the threshold value θ (mean of all bins) and the blue (black) bins are bins less than the threshold value. The yellow bins give a binary output of 1 and the blue a binary output of 0.

algorithm. Appiah et al. [5], presents a bSOM clustering algorithm and demonstrates it fast training rules on FPGAs.

Lefebvre and Garcia [16] used SOM to measure image similarity in face recognition. Recognising people automatically (e.g. by face, gait, iris or DNA), is an interesting area of research with many challenges [17]. The quality of images extracted from video sequence contributes significantly to the effectiveness of most facial recognition systems. Facial recognition is non-intrusive but is affected by changes in facial expressions.

A number of authors have developed identification techniques based on binary signatures, which are amenable to rapid processing. It is very difficult to represent the face as a binary signature. In contrast, fingerprints, a well developed biometric for uniquely identifying individuals, and the Iris [18], [19] can easily be transformed into binary signatures. Unfortunately, fingerprints extraction is intrusive and requires some form of contact with the object [17], and Iris recognition requires close-up imaging [18], [19]. However, this approach can be usefully modified for identification-based tracking in video sequences.

In this paper we use a very simple colour-based binary signature. The colour distribution of a moving object is used to generate a histogram, which is an effective invariant appearance-based method of object representation [20]. The histogram is computationally inexpensive to create and can easily be transformed into a binary signature, as described below. We train the bSOM to identify individual objects based on these signatures. Figure 1 gives an overview of the FPGA based object recognition system presented in this paper. The system relies on a robust tracking algorithm [3], [21] capable of extracting the colour histogram for every moving object. Binary signatures extracted from the output of the tracker are fed into the FPGA via USB to train the bSOM.

III. Binary Classification and Recognition

In the vast majority of implementations, the SOM input data and neurons are represented by real numbers (or fixed point representation), making them difficult to implement efficiently on FPGA. However, in many applications the data is either naturally presented as a binary string, or may be conveniently recoded as such (a "binary signature"). The bSOM [5] takes a binary vector input, and maintains tri-state vector weights with {0, 1, # } as the possible values. The # represents a "don't care" state, which signifies that the corresponding input vector bit may be either set or clear. The weight vectors have the same length as the input binary vector. The bSOM has the same essential structure as a standard SOM [22], [23], with an input layer and a competitive layer. Given a binary input vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, all the units in the competitive layer are "connected" by corresponding prototype vectors, $\mathbf{w_j} = (w_{j1}, w_{j2}, \ldots, w_{jn})$. Each neuron in the bSOM has a tri-state vector of the same length as the input binary signature, with the similarity between a neuron and an input binary signature determined by using the Hamming distance (a # being treated as a match to either 0 or 1).

A. Feature Vector Construction

Given the silhouette of a segmented moving object, a 768 bin histogram is generated – 256 bins for each of the three RGB colour components. This is converted into a binary signature by thresholding at the average number of bin entries as shown in equation 1; any histogram bin with a value greater than or equal to θ is represented as binary 1; binary 0 otherwise (see figure 2).

$$\theta = \frac{\sum_{i=1}^{768} bin_i}{768} \tag{1}$$

A binary feature vector, $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ for $n = 768$ is generated as in equation 2.

$$x_i = \begin{cases} 1 & \text{if } bin_i >= \theta \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

B. Training and Recognition

To test the performance of the bSOM using signatures extracted from the colour histogram, a limited number of objects (nine people, 2248 instances) are used to train a fixed size bSOM off-line; see figure 3.

In a simple scene each object might ideally be assigned a unique representative signature. However, due to partial occlusion, camera jitter, over-segmentation and under-segmentation, the silhouette and hence the histogram for a particular object may vary more widely than desired. We therefore allow each object to be represented by more than one neuron, and use 40 neurons (empirically selected).

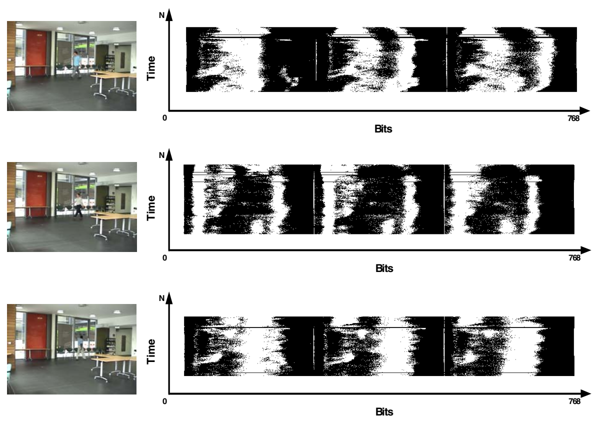

Fig. 3. Three of the nine objects used to train the bSOM with their corresponding binary signatures. On the signature graphs, each row represents a particular time-step, and the consistency and evolution of the signature over time can be clearly seen

Iterations	Average Results	
	cSOM	bSOM
10	81.84%	**84.41%**
20	83.06%	**84.56%**
30	84.50%	**84.85%**
40	84.05%	**84.05%**
50	83.98%	**85.03%**
60	84.70%	**85.91%**
70	85.03%	**85.74%**
80	**85.01%**	84.58%
90	**85.20%**	84.40%
100	**85.15%**	84.58%
200	84.68%	**86.44%**
300	**86.71%**	84.23%
400	**87.33%**	86.05%
500	**87.42%**	86.89%

TABLE I
THE AVERAGE PERFORMANCE OF THE TWO SOM IMPLEMENTATIONS (cSOM AND bSOM) FOR 14 DIFFERENT ITERATIONS, FOR 10 REPETITIONS EACH. THE PERFORMANCE OF THE ORIGINAL SOM INCREASES AS THE NUMBER OF ITERATIONS INCREASES WHILE THE PERFORMANCE LEVEL OF THE TRI-STATE SOM REMAINS FAIRLY CONSTANT.

For this preliminary work, the training set was labelled by an operator (i.e. to identify which of the nine subjects each instance corresponded to). The win frequencies (count of the number of times a particular object gets associated with a neuron in a winner-takes-all competition) is obtained. A node is assigned the most frequent object label associated with it through this process. The bSOM is ready for identification after the node labelling process. To test the performance, 1,139 independent manually labelled test instances are used. During the testing phase, the binary signatures are used to identify the neuron with the minimum Hamming distance, and the label associated with the neuron is assigned to the object from which the signature is extracted. If the minimum Hamming distance exceeds a threshold value set during training, the object is classified as unknown.

IV. EXPERIMENTAL RESULTS

The binary recognition system has been tested with video data recorded over a period of two hours with a total of 18,122 frames. Note, not all the frames have moving objects; hence the difference in number of frames to number of binary signatures. The video was recorded in an indoor environment, very close to the exit of the building. Typically people enter the building and leave at the same exit point. The scene has normal office furniture, which partially occludes the moving object. The scene has some variations in lighting intensity, particularly the effect of the wide transparent windows, see figure 3. Frames from the first 30 minutes with nine different persons entering the building are used to train the system. Signatures from the silhouettes are extracted and manually labelled to distinguish between noise and real objects. Objects with less than 768 pixels are filtered as noise, which also avoids values of $\theta < 1$ in equation 1. Figure 3 shows three of the nine objects that are used to train the bSOM. Due to segmentation problems, the binary signatures for any particular object may vary from frame to frame, as shown in figure 3. Using nine different objects to train the network requires a minimum of nine neurons. Due to signature variation from frame to frame, 40 neurons are required for efficient performance.

The recognition level of the modified bSOM is benchmarked against the conventional SOM (cSOM) originally proposed by Kohonen [22], [23]. Tests were conducted with neurons ranging from 10 to 100 in increments of 10. For networks with more than 50 neurons, the recognition level for both the bSOM and cSOM exceeds 90%, but some neurons do not get used. The performance of the two implementations of SOM (cSOM and bSOM) using 40 neurons are presented in table I. The training and test data have 2,248 and 1,139 binary signatures respectively.

A. Statistical Significance of Results

This section discusses the statistical significance of the performance of the two SOM implementations (cSOM and bSOM) presented in the previous section. The experiment was repeated ten times at each number of iterations. From table I, bSOM appears to outperform cSOM for smaller number of iterations (10 to 70). A Wilcoxon rank-sum test is used to determine whether there is any significant difference between the recognition levels of the different algorithms. One-tailed tests are conducted to test whether the higher mean performance of bSOM at lower iterations, and of cSOM at higher iterations, are statistically significant. From table II, bSOM outperforms cSOM for smaller iterations (10–70), with the exception of iteration 40. Similarly cSOM outperforms bSOM for higher iteration (80–500), with the exception of iteration 100 and 200. There is no statistically significant difference at iteration 100. We conclude that bSOM trains more quickly than cSOM, but ultimately cSOM has higher performance.

V. FPGA ARCHITECTURE

The most critical aspect of any hardware design is the selection and design of the architecture that provides the most

Iteration	Mean Rank		Significance	
	cSOM	bSOM	z	
10	5.50	15.50	-4.00	$\succ$
20	6.50	14.50	-3.19	$\succ$
30	5.50	15.50	-4.00	$\succ$
40	12.50	8.50	1.66	$\prec$
50	6.50	14.50	-3.19	$\succ$
60	5.50	15.50	-4.00	$\succ$
70	6.50	14.50	-3.19	$\succ$
80	15.50	5.50	4.00	$\prec$
90	14.50	6.50	3.19	$\prec$
100	12.50	8.50	1.58	$-$
200	5.50	15.50	-4.00	$\succ$
300	14.50	6.50	3.19	$\prec$
400	15.50	5.50	4.00	$\prec$
500	15.50	5.50	4.00	$\prec$

TABLE II
RESULTS OF THE ONE-TAILED WILCOXON RANK-SUM TEST CONDUCTED ON CSOM AND BSOM FOR THE EXPERIMENTAL RESULTS PRESENTED IN TABLE I AT A SIGNIFICANCE LEVEL OF 5%. $\succ$ IS USED TO SHOW SIGNIFICANCE PERFORMANCE BETWEEN CSOM AND BSOM FOR CASES WHERE BSOM PERFORMS BETTER AND $\prec$ FOR CASES WHERE CSOM PERFORMS BETTER. WHEN THERE IS NO SIGNIFICANT DIFFERENCE $-$ IS USED. THERE IS SIGNIFICANT DIFFERENCE IF ϱ IS LESS THAN 0.05, WITH A SIGNIFICANCE LEVEL OF 5%.

Network Size	40 neurons
Input vectors	768 bits
Neuron vectors	768 bits
Initial weights	Random
Maximum neighbourhood	4 neurons

TABLE III
SPECIFICATION OF THE BSOM AS IMPLEMENTED ON FPGA. THE DESIGN HAS 40 NEURONS, EACH WITH 768 BITS.

efficient and effective implementation [24]. The specifications of the circuit implemented on FPGA is given in table III with its corresponding block diagram in figure 4. The circuitry is made up of five basic blocks: the weight initialization, pattern input, Winner Take All, neighbourhood update and display blocks. Three of the five blocks run in parallel. These are the pattern input, Winner Take All and display (output) block. The weight initialization block is triggered only at start-up. Similarly, the neighbourhood update block is triggered when a winning node is identified for an input binary vector. Details of the five basic blocks are presented in the following sections.

A. Weight Initialization block

This block is used to randomly initialize all the weight vectors in the network. All the neurons in the network are initialized in parallel bit-by-bit; hence it takes as many clock cycles as there are bits in the binary input vector to complete the initialization. The hardware architecture presented here has been tested with binary signatures fed into the FPGA in the form of binary images of size 32×24, totalling 768 bits. The sizes of the input and weight vectors are all set to 768 bits and can easily be altered for any image size. The presented implementation takes exactly 768 clock cycles to completely initialize all the neurons.

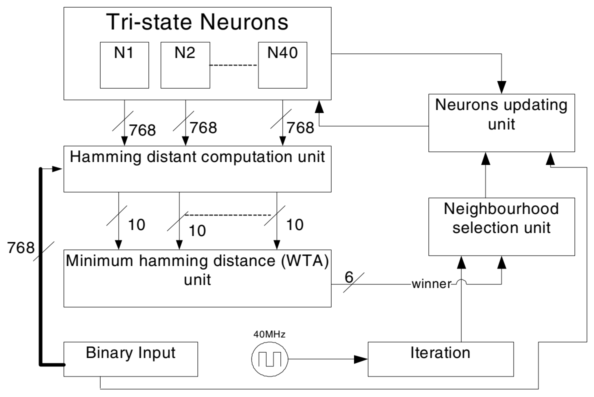

Fig. 4. A block diagram of the FPGA design. The Hamming distances between the input and all the 40 neurons in the network are computed in parallel. The forty 10 bit Hamming distances are fed into the WTA block to evaluate the neuron with minimum Hamming distance to the input. The winning neuron and selected neighbours are then updated.

B. Pattern Input block

This block is used to acquire the binary input vector (or binary image) from an external camera. The size of the input vector, 768 (taken from a binary image of size 32×24), is pre-programmed and the input is complete when a total of 768 bits is read from the camera. This binary data is stored in the input vector and then passed onto the WTA block for further processing.

C. Winner Take All block

This block is made up of two parts: the Hamming distance computation unit and the winning neuron unit. The distance computation unit is used to compute the Hamming distance between the input binary vector and all neurons in the bSOM. The Hamming distance between the input vector $\mathbf{x}$ and a neuron $\mathbf{w}_j$, as shown in equation 3 is a bitwise operation, and hence takes as many clock cycles as there are bits in the input vector. Since the Hamming distance for all the 40 neurons are computed in parallel, it takes exactly 768 clock cycles to compute the Hamming distance for all the neurons in the network.

$$H_{ij} = \sum_{k=1}^{768} H_{ijk}, \text{ where } w_{jk} \neq \#. \tag{3}$$

where k iterates through the bits in the input vector and $j \in (1 \cdots 40)$ is the address of the neuron. It is worth noting that the neuron vector is tri-state and the # state is ignored when computing the Hamming distance. Thus, for a neuron with 768 #'s, the Hamming distance will always be 0. The winning neuron unit uses the results from the Hamming distance computed in the distance computation unit to identify the winning neuron. The design, as shown in figure 5, uses a series of comparators to select the minimum of a pair of two input Hamming distances. For an implementation with 40 values, the design takes exactly seven clock cycles to compute the node with the minimum Hamming distance.

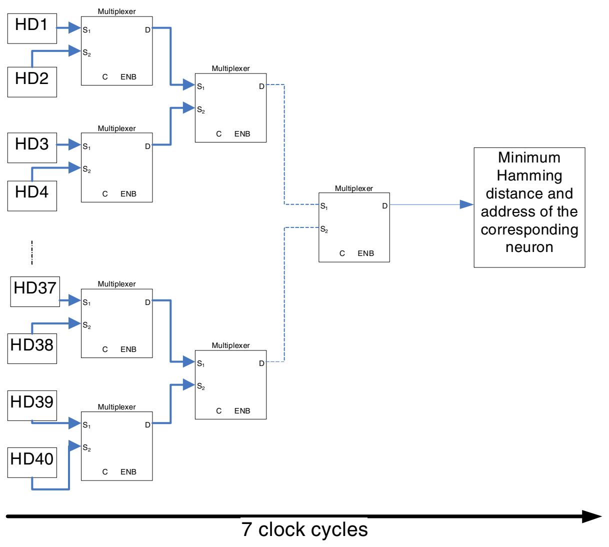
HD1
HD2
HD3
HD4
HD37
HD38
HD39
HD40
Multiplexer
S1
S2
D
C ENB
Minimum Hamming distance and address of the corresponding neuron
7 clock cycles

Fig. 5. Structure of the WTA unit. This unit uses seven cycles to compute the minimum Hamming distance. The first cycle requires fifty comparators, which is halfed every cycle to one in the seventh cycle.

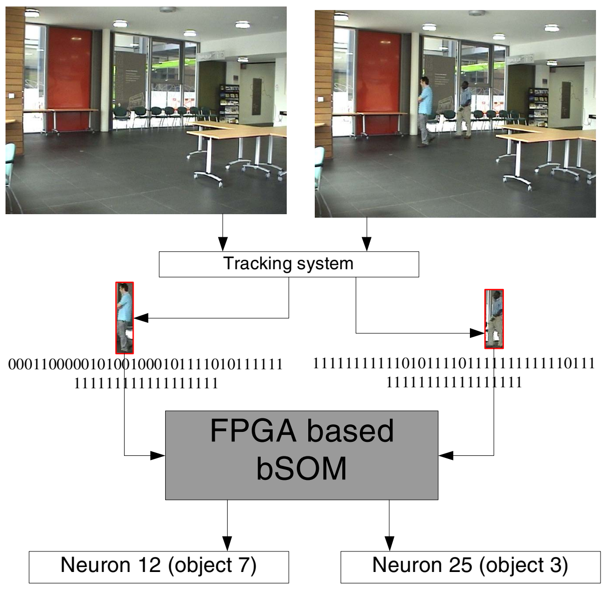

Fig. 6. A figure showing the bSOM based object recognition system on FPGA. Binary signatures extracted from individual moving objects are fed onto the FPGA based bSOM to associate them with neurons. Neurons are labelled after training the bSOM.

D. Neighbourhood update block

This block is used to select the neighbourhood of the winning neuron and to update the neurons in the specified region. The size of the neighbourhood reduces as training progresses. In the hardware implementation the maximum size of the neighbourhood is set to 4, and decreases as training progresses. The iteration count determines the size of the neighbourhood; for example, if the total number of iterations is set to 100, then for the first 25 iterations the neighbourhood is set to 4, then 3 in the second 25 iterations then 2 for next 25 (thus iteration 26 to 50) and then 1 in the last 25 iterations.

E. Output display blocks

The output display block displays the neurons (weights) as a binary image on an external Video Graphics Array (VGA) for visual verification. It runs in parallel with the input and WTA blocks. It runs at the refresh rate for the VGA used, typically 60Hz. The bSOM architecture discussed here has been implemented on a Xilinx Virtex-4 FPGA chip (XC4VLX160) with approximately 152,064 logic cells with embedded RAM totalling 5,184 Kbits. The design and verification was accomplished using the Handel-C high level descriptive language. Compilation and simulation were achieved using the Agility DK design suite. Synthesis – the translation of abstract high-level code into a gate-level net-list – was accomplished using Xilinx ISE tools.

The entire design can be clocked up to 40MHz, making it possible to train the binary Self Organizing Map with up to 25,000 patterns of size 768 bits in a second after initialization. The clock frequency of 40MHz also includes the design for controlling the external logic for the VGA and the camera. This is the actual hardware test and the most stable clock frequency. The frequency could be much higher without the requirement to interface these devices. Table IV gives the details of the resource utilization of the FPGA implementation.

Resource		Total Used	
Name	**Total**	**Used**	**Per.(%)**
Flip Flops	135,168	4,095	3
4 input LUTs	135,168	18,387	13
bonded IOBs	768	147	19
Occupied Slices	67,584	11,468	16
RAM16s	288	43	14

TABLE IV
IMPLEMENTATION RESULTS FOR THE BSOM, USING VIRTEX-4 *XC4VLX160*, PACKAGE *FF1148* AND SPEED GRADE *-10*.

F. Post Training

After successfully training the bSOM with offline binary signatures extracted using a PC based object tracker [21], the bSOM is ready for real-time identification of known objects. The neurons (weights) of the bSOM are stored onto BlockRAM on the FPGA chip. Labels are associated with each neuron to identify them with the objects from which the binary signatures have been extracted. Binary signatures of all moving objects in the camera view are sent onto the FPGA as a 32×24 sized image. Hamming distances between the input signature and all the bSOM weights are computed in parallel. The neuron with the minimum Hamming distance is selected as the winning neuron and the label of that neuron is associated with the object in the camera view.

The bSOM recognition as presented relies on the tracking system to extract the colour histogram of individual moving objects. The frame level binary signatures are fed onto the FPGA at the camera rate of $30fps$. The system is capable of recognising $25,000$ signatures per second, far more than the tracking system can reliably provide [21], and of training with several thousand patterns in less than a second. Figure 6 shows the use of the bSOM implemented on FPGA architecture for real-time recognition of objects after the training. The bSOM recognition has less than 15.97% error, as shown in table I. The ability of the bSOM to run and train in real-time on an FPGA is very attractive for embedded surveillance systems.

VI. Conclusion

A tri-state SOM (bSOM) which maintains tri-state weights and accept binary inputs has been presented with its FPGA implementation. The bSOM is particularly well-suited to FPGA implementation, trains quicker than the original SOM and can be used in clustering and classifying binary signatures. A demonstration of the potential use of the bSOM in security surveillance systems as an identification system using binary signatures extracted from colour histograms has also been presented in this paper. A full implementation of an identification system would require on-line training and automatic labelling. The additional stages required to reach this point are: to use the novelty detection capability of the bSOM to identify previously-unlabelled objects; to use positional tracking to follow such objects for a period and to record the corresponding signatures; and to update the bSOM through on-line training when sufficient new signatures are available. Similarly, clean positional tracking may be used to maintain consistent labelling of an object when the identity shifts sufficiently to require a new bSOM node. Future work will also include: the use of more sophisticated invariant features for identification; testing with a larger number of mutually-occluding objects; and a full FPGA-based implementation of an end-to-end integrated surveillance system.

Acknowledgements

This work was supported by Technology Strategy Board (TSB) under the BRAINS Project. Special thanks goes to the project team; Prof. Derek Sheldon, Nigel Priestley, Mervyn Hobden, Peter Hobden and Cy Pettit.

References

[1] L. Zhang, S. Z. Li, X. Yuan, and S. Xiang, "Real-time object classification in video surveillance based on appearance learning," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[2] K. Appiah, A. Hunter, P. Dickinson, and H. Meng, "Accelerated hardware video object segmentation: from foreground detection to connected components labelling," *Computer Vision and Image Understanding*, April 2010. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2010.03.021

[3] J. Owens, "Neural networks for video surveillance," Ph.D. dissertation, University of Sunderland, UK, November 2002.

[4] K. Appiah, H. Meng, A. Hunter, and P. Dickinson, "Binary histogram based split/merge object detection using FPGAs," in *Proceedings of CVPR 2010 (6th IEEE Workshop on Embedded Computer Vision)*. IEEE, June 2010.

[5] K. Appiah, A. Hunter, H. Meng, S. Yue, M. Hobden, N. Priestley, P. Hobden, and C. Pettit, "A binary self-organizing map and its FPGA implementation," in *IEEE International Joint Conference on Neural Networks*, June 2009.

[6] J. Sang, Z. Lei, S. Liao, and S. Li, "Adaptive object classification in surveillance system by exploiting scene context," *Computer Vision and Pattern Recognition Workshop*, pp. 1–7, 2009.

[7] B. Bose and E. Grimson, "Improving object classification in far-field video," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2004, pp. 181–188.

[8] J. Landabaso, L. Xu, and M. Pardas, "Robust tracking and object classification towards automated video surveillance," in *Proceedings of the International Conference on Image Analysis and Recognition*, 2004, pp. II: 463–470.

[9] V. Takala and M. Pietikainen, "Multi-object tracking using color, texture and motion," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, pp. 1–7, 2007.

[10] T. S. Ling, L. K. Meng, L. M. Kuan, Z. Kadim, and A. A. B. Al-Deen, "Colour-based object tracking in surveillance application," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, March 2009.

[11] J. Cho, B. Benson, S. Mirzaei, and R. Kastner, "Parallelized architecture of multiple classifiers for face detection," in *ASAP '09: Proceedings of the 2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*. IEEE Computer Society, 2009, pp. 75–82.

[12] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[13] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," *Sixth International Conference on Computer Vision*, pp. 555–562, 1998.

[14] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[15] Y. Wei, X. Bing, and C. Chareonsak, "FPGA implementation of adaboost algorithm for detection of face biometrics," in *IEEE International Workshop on Biomedical Circuits and Systems*, December 2004, pp. 17–20.

[16] G. Lefebvre and C. Garcia, "A probabilistic self-organizing map for facial recognition," in *19th International Conference on Pattern Recognition*, 2008.

[17] M. S. Nixon, J. N. Carter, M. G. Grant, L. G. Gordon, and J. B. Hayfron-Acquah, "Automatic recognition by gait: progress and prospects," *Sensor Review*, vol. 23, pp. 323–331, 2003.

[18] J. Daugman, "How iris recognition works," *IEEE Transaction on Circuits and Systems for Video technology*, vol. 14, no. 1, pp. 21–30, January 2004.

[19] ——, "New methods in iris recognition," *IEEE Transaction on Systems, Man, and Cybernetics –Part B*, vol. 37, no. 5, pp. 1167–1175, October 2007.

[20] D. Guillamet and J. Vitria, "A comparison of global versus local color histograms for object recognition," in *Proceedings of the International Conference on Pattern Recognition, ICPR*. IEEE, 2000.

[21] J. Owens, A. Hunter, and E. Fletcher, "A fast model-free morphology-based object tracking algorithm," in *Proceedings of the British Machine Vision Conference*. British Machine Vision Association, 2002.

[22] T. Kohonen, *Self-Organizing Maps*. Springer,New York, 1995.

[23] ——, *Self-Organizing Maps*, third extended edition ed. Springer Series in Information Sciences, 2001.

[24] C. Chang, M. Shibu, and R. Xiao, *Self Organizing Feature Map for Color Quantization on FPGA*. Springer, 2006.