Real-time object classification on FPGA using moment invariants and Kohonen neural networks

Deepayan Bhowmik, Balasundram P. Amavasai and Timothy J. Mulroy
Microsystems and Machine Vision Laboratory
Materials and Engineering Research Institute
Sheffield Hallam University, Pond Street, Sheffield S1 1WB, United Kingdom
deepayan.bhowmik@yahoo.com,{b.p.amavasai, t.j.mulroy}@shu.ac.uk
http://www.shu.ac.uk/meri/mmvl

Abstract - *In this paper, the use of moment invariants and Kohonen neural networks for real time object classification is addressed. The implementation of such a scheme using a reconfigurable hardware FPGA (Field Programmable Gate Array) device is described. In the image processing stage, the Hu's moment invariants algorithm has been implemented in hardware, and the issues surrounding this implementation is discussed. Following the image processing stage, a neural network is employed for the classification stage. By using the Kohonen unsupervised neural network the system is essentially self-supervised and it is able to perform an all parallel neural computation for classification purposes. A discussion of the concept and real simulation results are provided.*

Keywords: Moment invariants, FPGA, real-time, object recognition, Kohonen self-organising maps.

1 Introduction

The requirement for the recognition and classification of objects in real-time is important for many real world tasks, especially in robotics and industrial-type applications. A variety of blob and shape based algorithms exist, but many of these do not meet real-time constraints.

In recent times, with the advent of sophisticated software tools, the use of Field Programmable Gate Arrays (FPGAs) has changed from being simply a glue-logic type component to a vehicle for complete delivered solutions. FPGAs are made up of programmable logic components and programmable interconnects. Unlike other technologies, that force the programmer or designer to make critical decisions in the early part of development, FPGAs allow the development of the application to be adapted and improved over time. Developing FPGA solutions is comparable to developing solutions in software (rather than hard-coding).

This paper, describes an attempt to implement an object classification paradigm that is able to classify objects in real-time using an FPGA solution. The process makes use of two well studied algorithms in the area of machine vision and neural networks, namely Hu [1] moment invariants and Kohonen unsupervised neural networks. The computation of moment invariants has been implemented in hardware. The Kohonen neural network algorithm is divided into two parts, namely training mode, and detection mode. Currently the training of the system is performed using MATLAB. The result of the training is used for object classification and this has been implemented (and simulated) on an FPGA device.

2 Background

The domain of application of FPGAs has traditionally been in digital logic and digital signal processing (DSP). DSP-type problems are easily mapped to FPGAs due to the fact that most DSP functions are made up of sum-of-product type operations that consist of simple logic. Since images are essentially 2-D signals, image processing algorithms have also been widely implemented on FPGA.

The implementation of higher level machine vision algorithms that consist of a number of decision making stages is more limited. This is largely due to the limited number of directly mapped arithmetic functions available and the complexity in designing algorithms that are able to adapt or optimise online, since FPGA designs are often static. Parameterising the algorithms and using external memory to store these parameters can overcome this problem.

In recent times Hirai *et. al.* [2] have designed complete single-task vision systems on FPGA, with the objective of detecting the position and orientation of distinctly visible planar objects. Their system consisted of three parts, namely the computation of image gravity centre, the detection of object orientation using radial projection and the computation of the Hough transform, albeit discrete. They reported being able to process images at a rate of around 200 fps, far more than the standard frame-rate of a PAL camera.

In a separate study, Arribas and Maciá [3] implemented the Santos-Victor paradigm to compute motion fields for use in robot guidance applications. The technique required the computation of optical flow fields in real-time. The flow fields were computed using both standard differential and correlation methods.

Recent approaches have been made towards the implementation of real-time object recognition using reconfigurable hardware. Most of these systems have been implemented using multiple FPGA boards or similar devices together with a sequential computer, due to the limited number of gates available on each device. For instance, Neema *et. al.* [6] have used multiple DSP processors and a sequential computer for their Automated Image Recognition system. Yasin *et. al.* [7] have demonstrated an effective FPGA prototype for iris recognition. However, the system itself is not completely hardware based, since images are pre-processed using MATLAB, and FPGAs are only used for the recognition part. In their design, the authors make use of multilayer perceptrons (MLPs) for classification. The numbers of neurons are limited due to hardware constraints and hence the computation complexity had to be reduced for implementation on an FPGA board. Jean *et. al.* [8] implemented a system to accelerate the recognition process in infra-red images using an FPGA device. However once again, the system is dependent on a separate computer and in order to reduce complexity, the mathematical calculations have been performed with full precision integer values instead of floating point operations.

Neural networks are an ideal example where the use of FPGAs can offer an advantage. This is due to the fact that the operations carried out are largely parallel. A number of efforts have been made for mapping neural networks onto hardware. For example Aibe *et. al.* [9] implemented a probabilistic neural network in FPGA hardware which exhibited a parallel architecture.

3 Methodology

3.1 Object classification

The method of describing 2-D objects using moment invariants is well studied. Hu descriptors are based on non-orthogonalised central moments that are invariant towards rotation, translation and scale. Hu descriptors are thus computed from normalised centralised moments up to the third order, and consist of seven moments in total. The seven formulae given in (1) are normally used for algorithmic implementation.

$$I_1 = \eta_{20} + \eta_{02}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})]$$

$$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{1}$$

The derivation of these seven invariants and the computation of η is available in [1], and so will not be described here. The combination of these seven Hu descriptors is unique given a specific pattern or shape. Hence it can be used in conjunction with a matching scheme to uniquely identify the object being input to the system. Computing these descriptors is straightforward and fast, and hence it is widely used in the area of real-time vision in robotic applications [4].

Directly comparing the seven descriptors of an object against values in a stored database is an obvious method of classifying an incoming object. However, artefacts such as noise and camera focus can affect the outcome of classification, albeit marginally.

In the case of large numbers of data classes, artificial neural networks (ANNs) have been found to be quicker than traditional search methods. Neurons in an ANN can operate in parallel to provide a further increase in speed. By training a classifier on multiple instances of an object in a variety of poses, it is possible to improve the quality of the classifier. In this paper we propose the use of a Kohonen ANN to cluster the objects online for storage in a reference database.

The Kohonen ANN is essentially a self-organising unsupervised mapping system that can map input vectors of arbitrary length onto a lower dimension map. It is frequently described as a sheet-like neural network array. Patterns that are close together in Euclidean space will remain close in the final map, and are topologically ordered. The learning process of Kohonen ANNs optimises the mapping until the weight change becomes negligible.

Initially, all the neurons were mapped in a two dimensional space. Each neuron consists of seven weight vectors that are initialised with a random value. The seven Hu moment invariants are the input vector for the system, and the seven weights of the individual neuron are related to seven inputs vectors of the Kohonen ANN. Instead of the Euclidean distance measure, the Manhattan distance measure is used to ease and simplify the complexity of the computation and this has been found to give a reliable result.

The Kohonen training method clusters the neurons with similar weight vectors. The identification and numbering of the clusters is then performed using a clustering algorithm. This clustering algorithm allocates an identification number to each set of classes.

The training algorithm has been programmed in MATLAB whereas the detection of the object has been implemented in hardware. Since the current architecture of the Kohonen ANN only allows for classification in hardware, currently, the process of implementing onboard training on hardware is being studied and implemented.

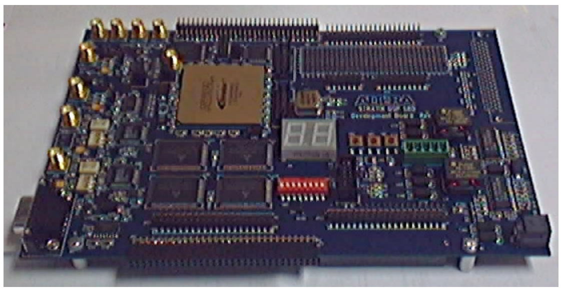

Figure 1: the Altera EP1S80 development board

When detecting objects on the FPGA device using the Kohonen ANN, the parallel neurons are initialised with proper weight values following the training phase. An unknown image containing an object is captured and then passed through the Hu's moment computation process on the FPGA. Once the seven invariants are computed, they are treated as the input for the ANN. Once again a distance measurement procedure is performed and the neuron associated with minimum distance is considered as the winning node. The tagged identification of the winning neuron is considered as the class of the unknown incoming object.

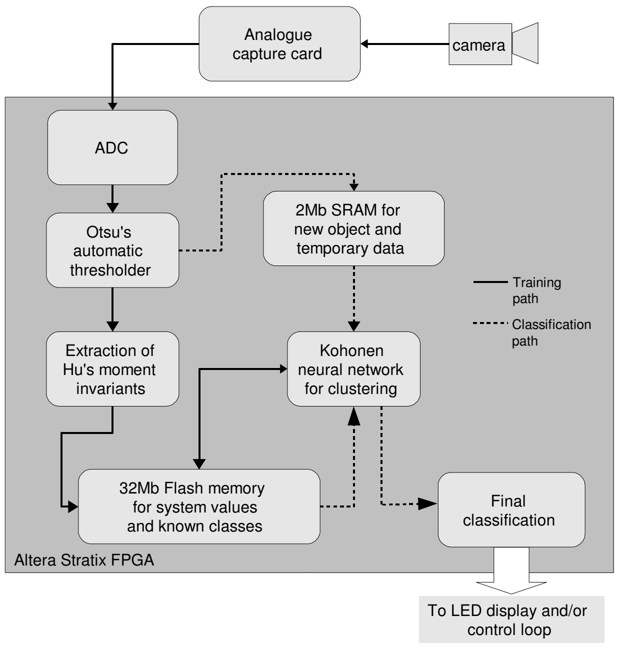
Analogue capture card
camera
ADC
Otsu's automatic thresholder
2Mb SRAM for new object and temporary data
Training path
Classification path
Extraction of Hu's moment invariants
Kohonen neural network for clustering
32Mb Flash memory for system values and known classes
Final classification
Altera Stratix FPGA
To LED display and/or control loop

Figure 2: Training and classifying objects (the proposed system).

3.2 Hardware mapping

The FPGA hardware used in this paper is the Altera Stratix EP1S80[1], shown in Figure 1. The EP1S80 device from Altera Corp. is one of the largest 0.13-micron FPGA devices available. This board consists of 79,040 logic elements (LEs), 7.2 Mbits of embedded RAM, and 1,238 user I/Os. This board is a powerful development platform for digital signal processing (DSP) designs, and features the Stratix EP1S80 device in the fastest speed grade (-6) 956-pin package. It consists of two 12-bit 125-MHz A/D converters, two 14-bit 165-MHz D/A converters, single-ended or differential inputs and single-ended outputs, 2 MBytes of 7.5-ns synchronous SRAM configured as two independent 36-bit buses, 64 Mbits of flash memory, dual seven-segment display, one 8-pin dipswitch, three user-definable pushbutton switches, one 9-pin RS-232 connector, two user-definable LEDs, on-board 80-MHz oscillator and a single 5-V DC power supply. For debugging interfaces, two Mictor-type connectors for Hewlett Packard (HP) logic analyzers and several 0.1-inch headers are available.

[1] Funded by EPSRC Nanorobotics project GR/S85696/01

The basic system diagram of the complete architecture of the proposed vision system is shown in Figure 2. As mentioned previously, classifying of objects consists of two modes, the training mode and the classification mode. The Otsu thresholder is commonly used to automatically binarise images by optimising a threshold level.

The whole implementation procedure has been divided in 3 parts, namely i) Moment invariant computation, ii) Training of the system & iii) Classification mode.

i) **Moment invariant computation:** In this mode, sample 2-D objects (patterns) are captured via a composite camera. The gray-valued image is then thresholded and the binarised values are presented to the system. The Hu descriptors that represent the pattern in the image are then extracted. Floating point calculations have been achieved by using the Altera Megacore library or by using the author's own floating-point implementation. Implementation in hardware has been achieved using VHDL code with Altera Quartus II software. To ease and reduce computation, the moment computation is simplified by replacing all floating point powers with its nearest integer values. This has been experimentally observed to produce good results.

Virtually all the operations of the algorithm have been mapped in parallel. This parallel operation significantly increases the speed of the system. Single precision floating point values have been used for all computation and conform to the IEEE 754 standard.

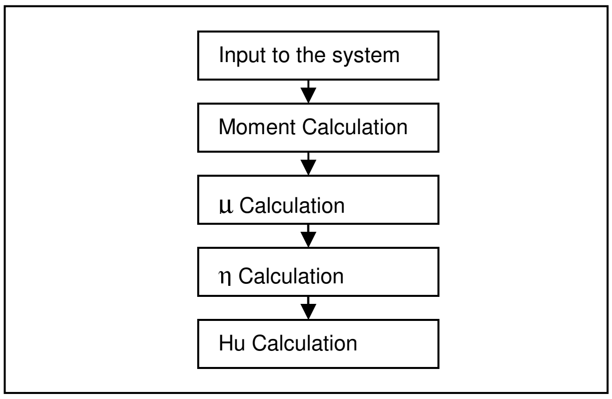

Figure 3: Algorithm for moment calculation.

As this work is a proof of a concept implementation, the input image matrix is hardcoded in VHDL. The computed seven moments have been used in the next stage of the system either for training (in the training mode) or classification (detection mode). The algorithm for the moment computation is shown in figure 3. The input data is iterated through in a sequential manner and then the different moment values have been calculated. After obtaining all moments the computation of the μ parameters (essentially high order moments) are computed in parallel mode. The μ values are required to compute η as shown in equation (1). The final seven Hu moment invariants are calculated and passed on to the next phase.

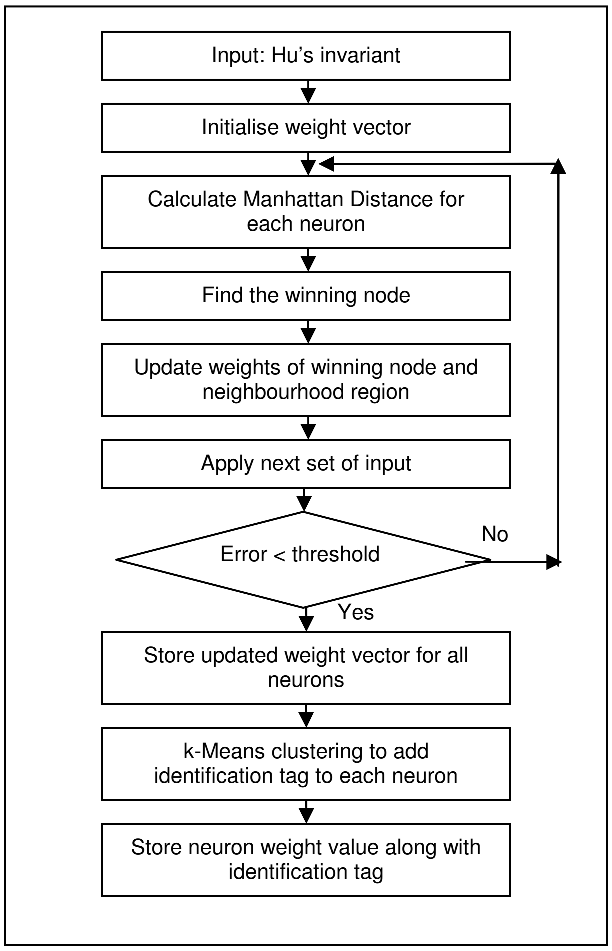
Input: Hu's invariant
Initialise weight vector
Calculate Manhattan Distance for each neuron
Find the winning node
Update weights of winning node and neighbourhood region
Apply next set of input
Error < threshold
No
Yes
Store updated weight vector for all neurons
k-Means clustering to add identification tag to each neuron
Store neuron weight value along with identification tag

Figure 4: Algorithm for Kohonen training and clustering

ii) **Training of the system:** The training of the Kohonen ANN is currently implemented in MATLAB but the generated weight vectors are used in hardware. The proposed system is organised as follows. The Kohonen neural network will self-organise around the descriptors. The map that is produced is then stored onto non-volatile flash memory, so that it may be used later during the classification. The algorithm for the training part is shown in figure 4. A 2-dimensional neuron map is mapped with a random weight initialisation. Seven weights are associated with each neuron. The input parameters are then fed to the ANN to find the Manhattan distance (2) for each neuron. The Manhatten distance is given as:

$$\text{Manhattan distance ("L}_1\text{ Norm"):} \quad \sum_{i=1}^{k} | x_i - y_i | \qquad (2)$$

The neuron corresponding to the minimum distance is the winning node. The weights of the winning node and the neighbourhood region around the winning node is updated with the following update rule:

$$w_{ij}(n+1) = w_{ij}(n) + \lambda(n)(x_i(n) - w_{ij}(n)) \qquad (3)$$

$$\text{for } 0 \le i \le N-1$$

where $w_{ij}(n+1)$ is the updated weight, $\lambda(n)$ is the learning rate and the term $(x_i - w_{ij})$ represents the error. The value of the learning rate normally lies between 0 and 1 and it controls convergence speed and stability. The learning process is iterative and continues until the network has converged satisfactorily.

To identify the different sets of neurons, a k-Means clustering algorithm is applied and one identification tag is attached to each each neuron. The Manhattan distance measurement is applied again for clustering purposse.

iii) **Classification mode:** In classification mode, the weight vector map is first recalled from flash memory. Incoming patterns are stored onto the high speed SRAM in order to increase performance. The new pattern is then matched against the patterns stored on the Kohonen map and a final classification is produced. As described earlier a set of sixteen parallel neurons have been used for the detection of an unknown object. After obtaining the moment invariants of the unknown images it is passed to the inputs of the neurons. The Manhattan distance is then measured for each neuron. The minimum distance is now detected and the associated neuron with the minimum distance is declared as the winning node. The identification tag is then simulated and corresponding outputs are activated. Figure 5 represents the algorithm for classification mode.

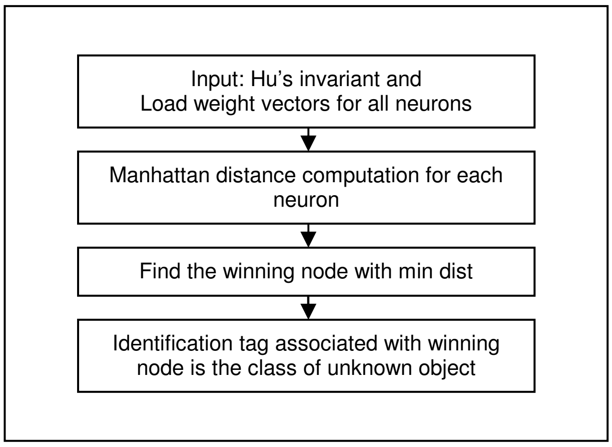
Input: Hu's invariant and Load weight vectors for all neurons
Manhattan distance computation for each neuron
Find the winning node with min dist
Identification tag associated with winning node is the class of unknown object

Figure 5: Algorithm for classification mode

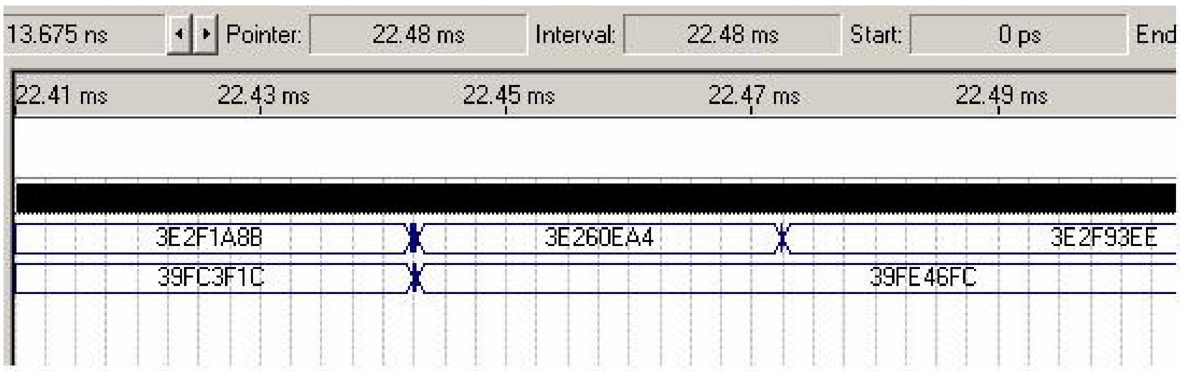

Figure 6: Hu's Moment invariant timing diagram

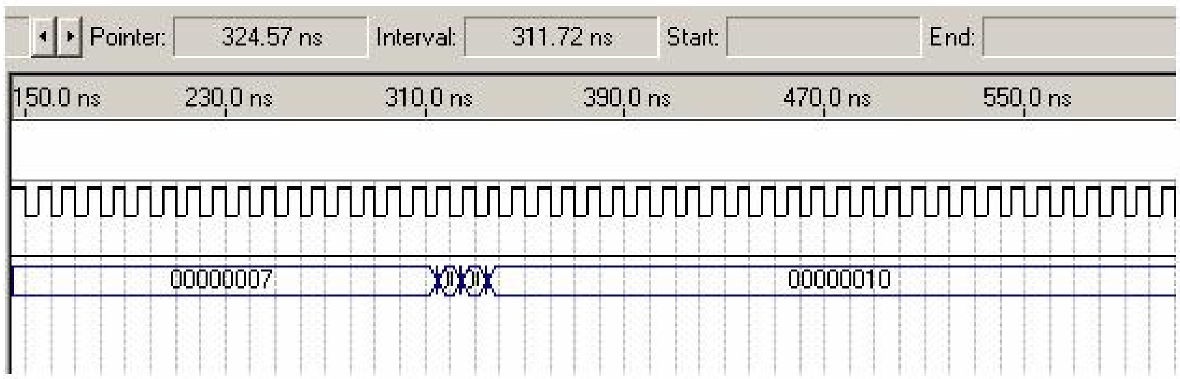

Figure 7: Timing diagram of classification mode

The complete system response and timing analysis is discussed in the results section. It has been observed that the response time for the system yields a good result due to the parallel design.

4 Results and discussion

The algorithm has been implemented in synthesizable VHDL code and a timing analysis has been performed. The simulation results exhibit a very good system timing performance, and the results indicate that the concept design fulfils the requirement for real time object recognition.

The timing analysis of Hu's moment invariant calculation is shown in figure 6. The processing of input data consumes the most time in this part. For a 20x20 pixel input image it took an average of 22.47 milliseconds whereas the rest of the computation of Hu's invariant required approximately 800 nanoseconds to complete (refer to table 1). This is due to the number of sequential operations performed. If the input variables can be processed in parallel, the time response could be significantly improved. The floating point calculations, however, have been implemented efficiently as can be seen from the results.

During the training phase conducted in MATLAB, it has been observed that the classification stage has an accuracy of above 85%.

The object classification mode indicates a good response time with parallel operation. For this application, eight neurons are designed to run in parallel, providing a response time of 310-360 nanoseconds, as shown in figure 7.

Table 1: Average time for different calculations

	Hu's Moment			Classification Mode
	i/p cal	other cal	total	
Time	23 ms	800 ns	24 ms	310-360 ns

The timing analysis shows that the system requires an average of 24 milliseconds (maximum) for the detection of an unknown object. This indicates that even with the sequential computations in parts of the design, a recognition rate of 40 fps is achievable.

5 Conclusion and future work

In this paper, we have demonstrated a possible solution of classifying objects in real-time using an FPGA solution. Object features are represented using Hu's moment invariant method, and an onboard Kohonen artificial neural network is used for initial clustering. The implementation uses the Altera Stratix FPGA device, which has the capability of performing complex mathematical operations. The estimated time to recognise an unknown object (20x20 pixels) is 24 milliseconds. We have shown that the classification of objects can be achieved using a single Altera Stratix board. The use of multiple FPGA boards working in parallel, however, can enhance system performance. At present, the input pixels to the system and the initial moment calculations are performed sequentially. As this is the most time consuming aspect of the process, overall system performance can be further improved by using a parallel implementation.

In future work, the onboard training of the Kohonen ANN will be addressed. The updated weight vectors of the neurons will be stored in the memory of the board, and will be made accessible during the classification mode. The current constraint of the system is the capacity of the FPGA board. By using a larger FPGA, the number of neurons can be increased and hence more parallel operations can be performed. The use of multiple boards for a single system will also enhance the system performance.

We have demonstrated that it is possible to build a full imaging system on a single FPGA board. As a proof of concept, the VHDL simulation has been conducted using the Altera Quartus II software environment. The timing analysis has shown promising results, indicating the proof of the system concept.

References

[1] M-K. Hu, "Visual pattern recognition by moment invariants", IRE Trans. on Information Theory, vol 8, pp. 179-187, 1962.

[2] S. Hirai, M. Zakouji and T.Tsuboi, "Implementing Image Processing Algorithms on FPGA-based Realtime Vision System", Proc. 11th Synthesis and System Integration of Mixed Information Technologies (SASIMI 2003), pp. 378-385, Hiroshima, April, 2003.

[3] P.C. Arribas and F.M-H. Maciá, "FPGA implementation of Santos-Victor optical flow algorithm for real time image processing: an useful attempt", Proceedings of SPIE's International Symposium on Microtechnologies for the New Millennium 2003 - Conference on VLSI Circuits and Systems, pp. 23-32, Canary Islands, Spain, May 19-21, 2003.

[4] P.J. Sanz, R. Marin and J.S. Sanchez, "Including efficient object recognition capabilities in online robots: from a statistical to a Neural-network classifier," IEEE Transactions on Systems, Man and Cybernetics, Part C, vol.35, no.1, pp. 87- 96, February, 2005.

[5] T. Kohonen, "Automatic formation of topological maps of patterns in a self-organizing system", Proceedings of 2nd Scandinavian Conference on Image Analysis, Espoo, Finland, pp. 214--220, 1981.

[6] S. Neema, J. Scott, T. Bapty, "Real time reconfigurable image recognition system", Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference, 2001. Volume 1, 21-23 May 2001, pp. 350 - 355 vol.1

[7] F. Mohd-Yasin, A.L. Tan, M.I. Reaz, "The FPGA prototyping of iris recognition for biometric identification employing neural network", Proceedings of The 16th International Conference on Microelectronics, ICM 2004. 6-8 Dec. 2004, pp. 458 – 461

[8] J. Jean, Liang Xiejun, B. Drozd, K. Tomko, "Accelerating an IR automatic target recognition application with FPGAs", Proceedings of Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM '99. 21-23 April 1999, pp. 290 - 291

[9] N. Aibe, M. Yasunaga, I. Yoshihara, J.H. Kim, "A probabilistic neural network hardware system using a learning-parameter parallel architecture", Proceedings of the 2002 International Joint Conference on Neural Networks, IJCNN '02., Volume 3, 12-17 May 2002, pp. 2270 – 2275