Realtime Feature Extraction using MAX-like Convolutional Network for Human Posture Recognition

Bo Zhao, Shoushun Chen
VIRTUS IC Design Centre of Excellence, School of EEE, Nanyang Technological University, Singapore

***Abstract*—This paper presents a realtime feature extraction processor based on MAX-like convolutional network. Due to the massive parallel MAX operations across multiple layers of feature maps, conventional implementation requires a vast amount of memory access as well as computation circuits. By exploring the overlapped data and reusing the intermediate computation results between consecutive "neurons", tremendous saving in both memory bandwidth and hardware resource has been achieved. Experimental results show that the number of logic gates drops from 402k to 170k, compared to conventional approach. The proposed feature extraction processor can be integrated with a custom-designed motion detection image sensor and a hardware-accelerated classifier to perform realtime human posture recognition.**

I. Introduction

Understanding human activity is of great importance to a number of applications, such as human-computer interaction, assisted living and health care. Especially in the case of rehabilitants monitoring, any sudden and drastic gesture change of a patient should immediately trigger medical physicians' attention for taking necessary action and required follow-up medical treatments. Considerable work has been devoted to this area. Notably, wearable devices such as gyroscopes and accelerometers [1] have been exploited. However, this kind of approach requires the devices being mounted to the patients and can only detect a few abrupt activities. Vision-based surveillance is a more popular, non-invasive solution [2], [3]. Unfortunately, the relative success of many vision systems is primarily based upon laboratory measurements in well-controlled environments. In the real world application, the obstacle is mainly related to the need of multiple sensors to cover a large surveillance area and different view angles, together with the need of high frame rate video acquisition devices to detect extreme activities such as falling down. Conventional cameras, with little computing capabilities, fall short of meeting the requirements. Massive quantities of primitive, redundant image data have to be transmitted and processed before the features of interest are obtained.

On the signal processing side, existing systems have been focusing on algorithmic-software implementations, involving complex computations such as active contours, hidden Markov models (HMM), and subvolume matching [4], [5]. On the other hand, primates vision is extremely accurate and efficient in the categorization of objects. The current theory of the cortical mechanism responsible for object categorization has been pointing to a hierarchical and mainly feedforward organization [6], where short-range feedback is believed to play a secondary role. This organization can provide hierarchical features of increasing complexity and invariance to size and position, making object categorization a multi-layered and tractable problem. In recent years many artificial systems have been implemented using biological-like processing [7]. However, realizing these models into realtime systems remains a challenge. Parallel MAX operations across multiple layers of feature maps with different sizes and orientations require a vast amount of memory access and computation circuits.

In this paper, we propose a hardware-reuse methodology to efficiently implement the MAX-like convolutional network into digital VLSI circuits. By exploring the overlapped data and reusing intermediate computation results between consecutive "neurons", both memory access and computation resource are largely reduced. A realtime feature extraction unit is realized and integrated with a custom motion detection image sensor [8]. The proposed approach is innovative due to its tremendous saving in hardware, realtime feature extraction and improved recognition efficiency. The rest of the paper is organized as follows: Section II introduces the system overview. Section III discusses the MAX-like convolutional network and its hardware implementation considerations. Section IV presents the VLSI architecture and in particular the hardware reuse methodology. Section V reports the experimental results and section VI concludes this paper.

II. System Overview

Fig.1 illustrates the architecture of the overall human posture recognition system, which includes a customerized image sensor, MAX-like feature extraction unit and a classifier. A known set of posture library is used for evaluating the recognition performance. We use a temporal difference image sensor [8] as input device, which can filter the background information and alleviate the recognition processor from background subtraction computation. The output of the image sensor is a stream of ON/OFF motion events. Each of the motion events is sent to a network of oriented Gabor filters, and convolution operation is performed on the fly. The responses of the filters are analogous to the lowest-level "neurons" (S1) in visual cortex. After that, a MAX-like operation is applied in order to find the maximal response among the feature maps and gain invariance to size and position translation. Each

survival "neuron" after the competition represents a contour line segment in the image. In this way, the original motion object (or human posture) is translated into a set of vectorial line segments, which are then fed to a classifier based on modified line-segment Hausdorff-distance scheme [9]. This paper only focus on the feature extraction part.

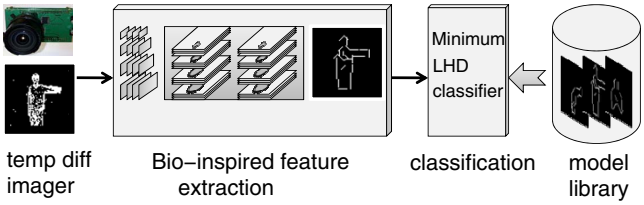

Fig. 1. Block diagram of the overall human posture recognition system.

III. MAX-like Convolutional Network

A. Network organization and MAX operation

The organization of the convolutional network is shown in Fig.2. The overall data flow can be summarized as $motion\ events \rightarrow S1\ maps \rightarrow C1\ maps \rightarrow line\ segments$.

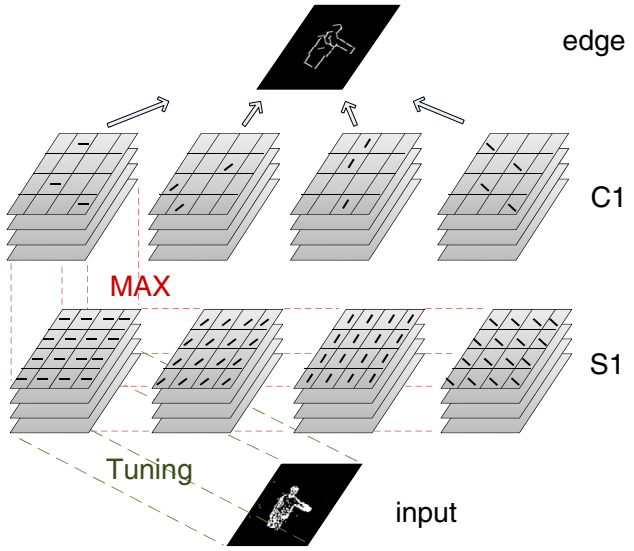

Fig. 2. MAX-like convolutional network for edge feature extraction. The input events are convolved with a network of Gabor filters to get a set of S1 maps, in which each pixel is considered as a "neuron". The S1 "neurons" compete with others in the process of MAX operation, and only those who best match the edge features can survive in the C1 layer.

Simple cells ($S1$) are used to build object-selectivity. This is done by convolving the motion events with a network of Gabor filters. Each filter models a "neuron" cell with certain size of receptive field and responses best to basic feature at certain orientation. For the sake of hardware friendly implementation, we trade-off the network to 4 scales (ranging from 3 to 9, with a step length of 2) and 4 orientations (0^o, 45^o, 90^o, 135^o). The function of Gabor filter can be described as:

$$G(x,y) = \exp\left(-\frac{X^2+\gamma^2Y^2}{2\sigma^2}\right) \times \cos\left(\frac{2\pi}{\lambda}X\right)$$

where $X = x\cos\theta + y\sin\theta$ and $Y = -x\sin\theta + y\cos\theta$. The filter parameters (orientation θ, aspect ratio γ, effective width σ and wavelength λ) have been well tuned in pioneering work [10], and here we adopt a similar set of these parameters. For the consideration of hardware implementation, we normalized the floating-point filter values into integer numbers.

After the convolution, we get 16 $S1$ maps. For a certain feature (say a bar) in the input image, each "neuron" in each of the 16 maps gives a response. Due to the property of the Gabor filter, a "neuron" reaches the maximum response only when its size, position and orientation matches the feature.

Based on this observation, we perform two-step MAX operation to extract the feature. The first step, MAX operation is performed across local neighborhood (Fig.3) to find the center of the feature. Each "neuron" competes with its local neighbors within its receptive field and will only survive when itself is the MAX. Then the processing moves to the successive "neuron", which will compete and strive for existence in the same way as last "neuron" did. The second step, MAX operation is performed over orientations and scales (Fig.4). The "neuron" that won the first MAX operation will compete against other "neurons" of the same orientation but different sizes (vertically, as shown in Fig.4), to find the size of the feature. At the same time, comparison is also performed across the counterparts of the same size but different orientations (horizontally, as shown in Fig.4), to detect the direction of the feature.

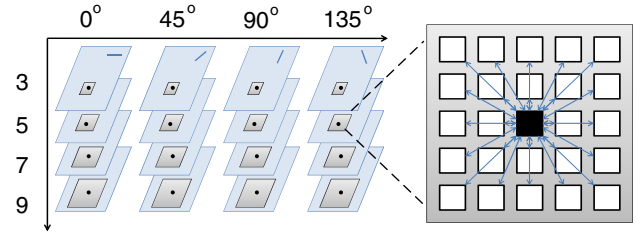

Fig. 3. MAX over local neighborhood. "Neurons" located in different-scale S1 maps have different receptive fields, such as 3×3, 5×5, 7×7 and 9×9. Each "neuron" competes with all the other "neurons" located within its receptive field. It can survive only when itself is the MAX in this area.

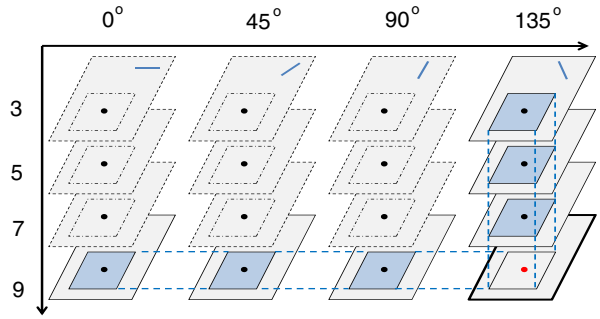

Fig. 4. MAX over orientations and scales. Here only shows the competition for a scale-9 orientation-135^o "neuron" (the dot in the bottom right map). This "neuron" competes with all the "neurons" that reside in the 6 shadowed 9×9 regions, and can only survive if itself is the MAX among these neurons.

After the two-phase MAX operation, each survival "neuron" represents a feature, i.e, a line segment with certain size and

orientation.

B. Hardware Implementation Considerations

Due to the vast amount of memory access and comparison, special care is required to efficiently implement the MAX network into hardware.

Firstly, the event stream produced by the motion image sensor are to be processed in parallel by a battery of S1 convolutional filters. Conventional implementation requires the whole frame of pixels first being stored in one set of memory, then each n×n (n=3, 5, 7, 9) pixels undergo convolution with a kernel, and finally the response has to be written into another set of memory. Thanks to the nature of event-based representation of the image sensor, we can adopt on-the-fly convolution scheme [11]. Within each "neuron" map, the "neurons" are updated on the fly, reducing both memory cost and computation time.

Secondly, during MAX operation a large number of "neurons" are to be read out from the memory and get compared in parallel. For the largest size "neuron", each MAX operation involves $9^2 \times 16 = 1296$ operands, producing a big challenge to the memory throughput. On top of this, the number of computation units (i.e. comparators) is another design trade-off. Without any optimization, the number of comparison can reach up to

$$N_c = \sum_{i=1}^{4} 4(n_i^2 - 1) + \sum_{i=1}^{4} [16(n_i^2 - 1) + 24] = 3296$$

where $n_i = (3, 5, 7, 9)$. Full parallel implementation will cost large amount of hardware resource.

IV. VLSI IMPLEMENTATION

A. Hardware Architecture

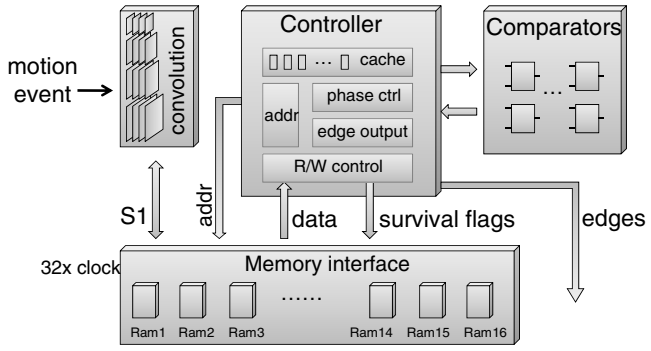

Fig. 5. Block Diagram of the feature extraction unit

Fig.5 shows the block diagram of feature extraction unit. Each motion event is sent in parallel to 16 sets of "on-the-fly-Gabor + memory". Each memory has 4096 words (the sensor features a resolution of 64×64) and 11-bit width, among which 10 bits represent S1 response value and the other 1-bit stores a survival flag. Upon the arrival of the last motion event, the two-step MAX operation is initiated. As discussed earlier, this procedure involves a large amount of comparison in parallel. In order to provide high throughput, a memory interface is designed to wrap the memory array, allowing each SRAM to output 32 words concurrently by running internally at a 32× clock. The central controller produces SRAM read/write addresses and delivers S1 values to an array of comparators. In the first phase MAX operation, when a "neuron" fails the competition, the controller will toggle its corresponding survival flag and write it back into the memory. In the second phase competition, each survival "neuron" represents a line segment with certain size and orientation.

B. Hardware reuse methodology

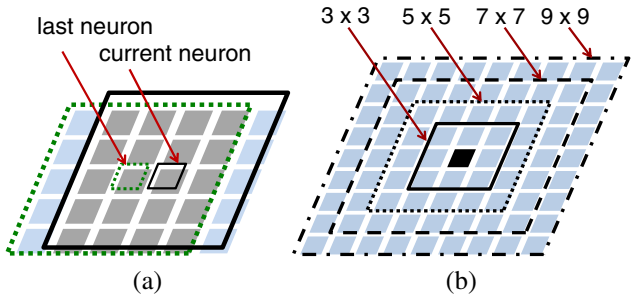

Fig. 6. Hardware reuse methodology for MAX operation. (a) shows the case of MAX over local neighborhood (taking the "neurons" with receptive field of 5×5 as example). The current "neuron" shares a large number of operands (the 4 shadowed columns) with the last "neuron". We find MAX column by column, store the rightmost four column-max in last neuron's computation, and reuse them for current neuron's processing. (b) illustrates the reuse technique in MAX over orientations and scales. The maximum value of the 3×3 block can be reused in computing the maximum of 5×5 region, which can then be reused in finding the MAX of 7×7, and similar rule applies to 9×9. Each time when the "neuron" size increases, only the new peripheral data need to be read in and get compared.

To alleviate the heavy requirement on hardware resource, we proposed a pipelined technique to reuse both the computation circuits and memory data. The principle is illustrated in Fig.6. For the sake of clarity, Fig.6(a) shows two 5×5 consecutive "neurons" during the 1^{st} stage MAX operation. One can note that when the MAX operation moves to a new "neuron" (denoted as "current neuron" in the figure), it shares a large number of operands (shadowed) with the previous "neuron". A straightforward way to optimize the memory access is to "cache" the overlapped data and each time only read in a new column. To further reuse the computation results, we perform the comparison in a column-wise manner by finding the maximum value within each column. The centering "neuron" only needs to compete with 5 column-max values, of which the rightmost 4 will be "cached" and reused by a new "neuron". In this way, the number of "cache", memory access and comparison is largely reduced.

Similar technique is applied to the 2^{nd} phase MAX operation. As introduced earlier (Fig.4), competition is performed over all orientations and scales. The procedure appears as full parallel and requires huge hardware resource. To reduce the hardware cost, We propose a bottom-up approach, doing the processing from the smallest size to the largest (i.e., 3→5→7→9). The comparison is performed in a array-wise manner (as shown in Fig.6(b)). For the smallest size 3, we

find the maximum value of each 3×3 block. The centering "neuron" only needs to compete with 6 block-max values (over orientations and scales). Next, the "neuron" size grows to 5. One can note that the maximum value of the overlapping 3×3 block can be reused. Therefore, each time when the "neuron" size increases, computation is only limited to find out the maximum value among the new "peripheral" data. The most computation-intensive scenario happens when "neuron" size grows from 7 to 9, where the maximum value among $9^2 - 7^2 = 32$ numbers will be calculated.

In summary, with the above hardware reuse methodology, the maximum parallel comparison is only $(9^2 - 7^2)/2 \times 16 = 256$, a dramatic reduction of 92% has been achieved (as compared to 3296 using straightforward implementation).

V. Experimental Results

The proposed edge feature extraction algorithm has been implemented using UMC 0.18μm CMOS technology. Table I summarizes the implementation results. The design can operate at a maximum clock frequency of 14.8 MHz. Each extraction procedure involves on-the-fly convolution, 2-step MAX operation and totally needs about $70 \times 4096 = 286720$ clock cycles. Therefore a maximum frame rate of 51 (64×64 resolution) can be attained, which means the design can support most realtime applications. In addition, we also estimated the total hardware saving by the proposed reuse technique. It was shown that the number of logic gates drops from 402k to 170k, a tremendous reduction of 58% has been achieved.

TABLE I
HARDWARE IMPLEMENTATION RESULTS

Technology	UMC 0.18 μm CMOS
gate count	170k
MAX freq.	14.8 MHz
processing capability	51 fps
power consumption	90 mW @30fps

We then further integrate the realtime feature extraction unit into the recognition system. In order to evaluate the system performance, we have created a dataset of postures (http://www3.ntu.edu.sg/home2009/zhao0130/dataset.htm). It includes 6 sets (500 images/set), namely "bend", "hand1", "hand2", "squat", "stand" and "swing". Fig.7 shows the line segments extraction results. Compared to the previous work [12], the new feature extraction algorithm brings forth enhancement in the form of less noise together with reduced number of edges. This further leads to improved classification performance (higher recognition rate and shorter computation time) as indicated in Table II (using modified Line-segment Hausdorff Distance classifier as reported in [12]). The tests are run on a desktop computer with Intel Core 2 Quad 2.66GHz CPU and 3G RAM. Our future work includes full hardware accelerated classifier for a complete realtime system.

VI. Conclusion

This paper reports the algorithm and hardware implementation of a MAX-like convolutional network for edge feature extraction. Compared to previous work, the proposed realtime feature extraction processor brings forth enhancement in the form of less noise and reduced number of edges. In addition, a hardware reuse methodology is proposed by exploring the overlapped data and the intermediate computation results between consecutive "neurons". Both memory access and computation resource are therefore largely reduced. The presented feature extraction processor will be integrated with a hardware accelerated classifier in the near future to build a complete realtime recognition system.

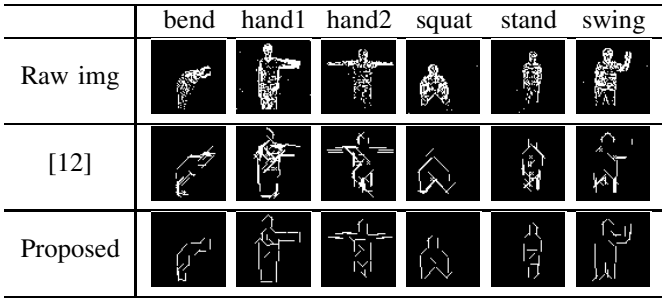

Fig. 7. Edge feature extraction results

TABLE II
COMPARISON OF THE RECOGNITION RATES AND COMPUTATION TIME

	Recognition Rate	Time/test (s)
[13] (C2 feature + SVM)	0.80	7.82
[12] (LHD classifier)	0.82	10.6
this work (LHD classifier)	0.87	2.22

Acknowledgment

This work was supported by Nanyang Assistant Professorship (M58040012) and ACRF Project (M52040132).

References

[1] R. Jafari, *et al.*, "Physical Activity Monitoring for Assisted Living at Home," *International Workshop on Wearable and Implantable Body Sensor Networks*, vol. 13, pp. 213-219, 2007.

[2] S.K.F Gupta, "Learning Feature Trajectories Using Gabor Filter Bank for Human Activity Segmentation and Recognition," *IEEE International Symps. on Multimedia Software Engineering*, pp. 546-556, 2004.

[3] M. Singh, *et al.*, "Human Activity Recognition Based on Silhouette Directionality," *IEEE Trans. on Circuits and Systems for Video Technology*, pp. 1280-1292, 2008.

[4] P. Turaga, *et al.*, "Machine Recognition of Human Activities: A Survey," *IEEE Trans. on Circuits and Systems for Video Technology*, vol.18, no.11, pp.1473-1488, Nov. 2008.

[5] Buccolieri F., *et al.*, "Human Posture Recognition Using Active Contour sand Radial Basis Function Neural Network," *IEEE Conf. on Advanced Video and Signal Based Surveillance*, pp. 213-218, 2005.

[6] S. Thorpe, *et al.*,"Reverse engineering of the visual system using networks of spiking neurons," *IEEE ISCAS*, vol. 4, pp. 405-408, 2000.

[7] Y. LeCun, K. Kavukcuoglu and C. Farabet, "Convolutional Networks and Applications in Vision," *IEEE ISCAS*, pp. 253-256, 2010.

[8] S. Chen, *et al.*, "A 64×64 pixels UWB wireless temporal-difference digital image sensor," *IEEE ISCAS*, pp. 1404-1407, 2010.

[9] Yongsheng Gao, Maylor K.H. Leung, "Line segment Hausdorff distance on face matching," *Pattern Recognition*, vol.35, pp. 361-371, 2002.

[10] T. Serre, "Learning a Dictionary of Shape-Components in Visual Cortex: Comparison with Neurons, Humans and Machines," Ph.D Thesis of Brain and Cognitive Sciences Department, MIT, April, 2006.

[11] A. Linares-Barranco, *et al.*, "On the AER Convolution Processors for FPGA," *IEEE ISCAS*, pp. 4237-4240, 2005.

[12] S. Chen, *et al.*, "A Biologically Inspired System for Human Posture Recognition," *IEEE BIOCAS*, pp. 113-116, 2009.

[13] T. Serre, *et al.*, "Object recognition with features inspired by visual cortex," *CVPR'05*, vol. 2, pp. 994-1000, 2005.