A Fully-Differential CMOS Implementation of Oja's Learning Rule in a Dual-Synapse Neuron for Extracting Principal Components for Face Recognition

Ronald G. Spencer and Edgar Sánchez-Sinencio
Analog and Mixed-Signal Center
Texas A&M University
College Station, TX. 77843

Abstract- **A fully-differential, CMOS implementation of a self-organizing, dual-synapse neuron with on-chip learning for real-time facial feature extraction is presented. The adaptation of the network follows Oja's learning rule and the synaptic weight vector is shown to adapt to the principal component vector of the set of two-dimensional input vectors.**

I. INTRODUCTION

Principal component analysis (PCA) is a powerful tool for extracting features for face recognition. Mathematically, PCA is a method of calculating the eigenvector of an autocorrelation matrix with the largest eigenvalue; i.e. the eigenvector that is most representative of the data that make up the autocorrelation matrix. In one sense, PCA is useful because it reduces the dimensionality of multi-dimensional data to one dimension; a compression of sorts.

PCA is also useful for generating features for face recognition. When applied locally, to a small neighborhood of pixels, PCA holds several advantages: 1) a combination of the results of many local analyses can be rearranged into slightly different permutations, much like that of the elastic graph matching to compensate for distortion such as smiling, frowning, perspective changes, etc. [1] and 2) it lends itself to analog VLSI implementation.

II. BACKGROUND

A. Principal Component Vectors

A principal component vector has two important characteristics. In terms of *analysis*, the principal component vector *preserves the most variation* when the data are projected onto it. In terms of *synthesis*, the principal component vector does the best job of *reconstructing* the original data from such aforementioned projection. Such characteristics have profound implications in the simplest model of a linear neuron with instar and outstar synaptic weight vectors, shown in Fig. 1. While the inner product between the input vector and the synaptic instar vector of the neuron performs *analysis*, the resulting neural activation multiplied times the synaptic outstar vector performs *synthesis*. If both weight vectors happen to be equal to the principal component vector of the set of input vectors being presented, the difference between the input and reconstructed output will be minimal in the least squares sense.

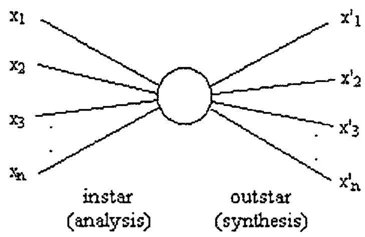


Fig. 1. Abbreviated signal flow graph of a single-neuron, dual-synapse network for extracting the principal component of a set of input vectors.

Mathematically, the process is described by (1), (2), and (3) and illustrated in Fig. 2. Analysis is performed by the inner product between the input and instar vectors, yielding a scalar which represents a projection of a two-dimensional pattern into a one-dimensional subspace. The original two-dimensional input vector is then approximately reconstructed as the outstar vector is activated to the same degree that the neuron was excited by analysis.

$$y = \mathbf{w}^t\mathbf{x} \tag{1}$$

$$x_j' = yw_j = \left(\mathbf{w}^t\mathbf{x}\right)w_j \tag{2}$$

$$\mathbf{x}' = y\mathbf{w} = \left(\mathbf{w}^t\mathbf{x}\right)\mathbf{w} \tag{3}$$

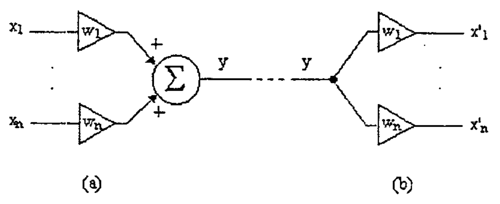


Fig. 2. Signal flow graphs of (a) analysis and (b) synthesis parts of a single-neuron.

B. Oja's Learning Rule

A neural system that is designed to find the principal component vector *iteratively*, must employ the Oja learning rule [3], as given by (4) and (5) where w_j is the j^{th} weight, x_j is the j^{th} input, y is the inner product between the input and instar vectors, and c is the learning rate.

$$\dot{w}_j = cy\left(x_j - yw_j\right) \tag{4}$$

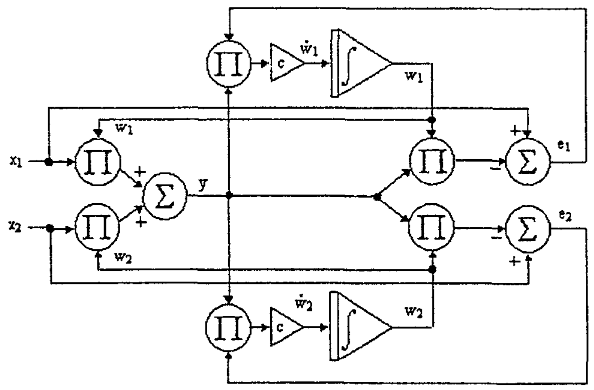
$$y = \sum_{j=1}^{n} w_j x_j \tag{5}$$

The Oja rule is typical of the self-organizing paradim; i.e. after the network decomposes the input (analysis) into its own internal representation, recomposes it (synthesis), and calculates the difference between the two. This difference is treated as error which drives the adaptation. Perfect reconstruction leads to zero error and no adaptation. Less-than-perfect reconstruction over all input vectors is normal and the neuron's synaptic weight vectors usually never reach the point where reconstruction error is zero, but they will converge to a vector that minimizes reconstruction error over the entire data set. For a set of input vectors with components whose variance is centered around zero, this vector *is* the principal component vector.

The Oja learning rule only allows for the extraction of the principal component vector. For a network that extracts every eigenvector of the data, one should see [4].

III. METHODS

The block diagram of a two-dimensional adaptive neuron with Oja learning is shown in Fig. 3. The adaptation of the synaptic weights is implemented by a master-slave architecture as the outstar vector adapts and the instar weight vector follows.



ıg. 3. *Signal flow graph of a two-dimensional linear neuron with master-slave adaptation.*

The multipliers were implemented by a fully-differential variant of the Gilbert cell multiplier, shown in Fig. 4. For more information on multipliers of this type the reader is directed to [5]. Common-mode feedback was added to lock the common-mode of the output to a fixed reference voltage. The loop consists of a balanced differential pair with common-mode sensor, and a single-ended operational transconductance amplifier (OTA) connected to the gate of the tail current transistor and a capacitor. When the common-mode of the output lowers, charge is taken out of the capacitor, lowering the potential of the gate of m_{cl} thus increasing the current flowing into both output nodes raising the common-mode. The single-ended OTA consisted of a simple differential pair with current mirrors and high output resistance output stage.

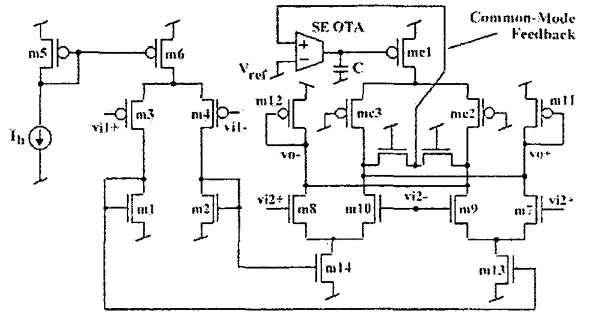


Fig. 4. Fully-differential Gilbert cell multiplier with common-mode feedback.

The integrators were realized by the fully-differential OTA shown in Fig. 5. Source degeneration transistors (m_{rl} and m_{rr}) were used for linearizing the transfer function and cascoded outputs were used for increasing the output resistance. Capacitors shown at the positive and negative output terminals were used to realize an OTA-C integrator.

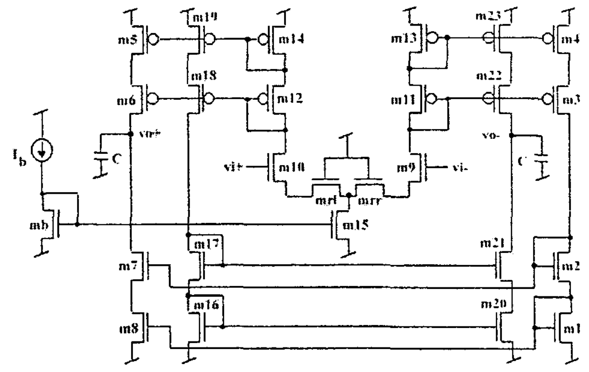


Fig. 5. Fully-differential OTA-C integrator for synaptic weight adaptation.

Using the blocks shown in Fig. 4 and Fig. 5, the block diagram of Fig. 3 was laid out and sent for fabrication in a 1.2 micron, double-metal, double-poly n-well AMI process. The lay-out of the 40 pin tiny chip is shown in Fig. 6.

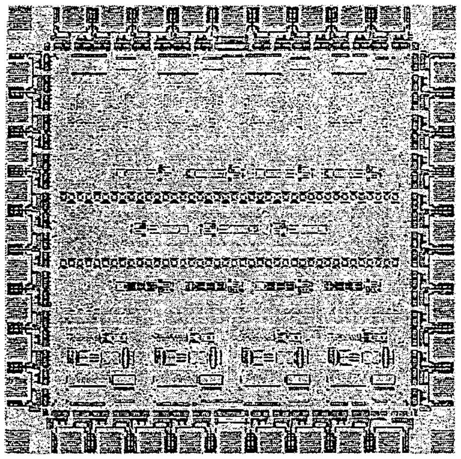


Fig. 6. Fully-differential dual-synapse neuron with on-chip adaptation.

IV. Simulation Results

At the time of this writing, the chips had just been received and testing just begun. Therefore, all results presented here are from HSPICE simulation of extracted blocks. A simulation was conducted to demonstrate that the weight vector converges to the principal component vector of the input data. The results are shown in Fig. 7. The learning rate was determined by the differential voltage on a multiplier, acting as the intensity of the image pixel whose x-y coordinates where presented at the two inputs. This graph shows the time evolution of the synaptic

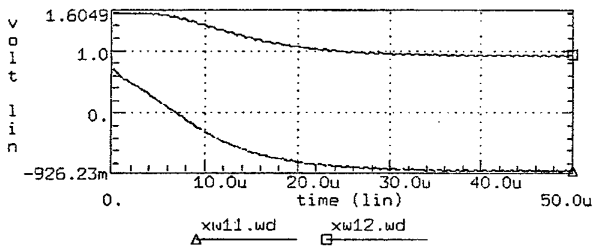


Fig. 7. Transient HSPICE results showing the convergence of the synaptic weights on the principal component vector.

weights given repeated presentation of the points (-.3,.3) and (.3,-.3). These data had zero mean in both dimensions. The weights converge to the direction of the most significant eigenvector of (-1,1) which is shown more clearly in Fig. 8. This simulation demonstrated how the system would react to a local neighborhood of two pixels with equal intensity.

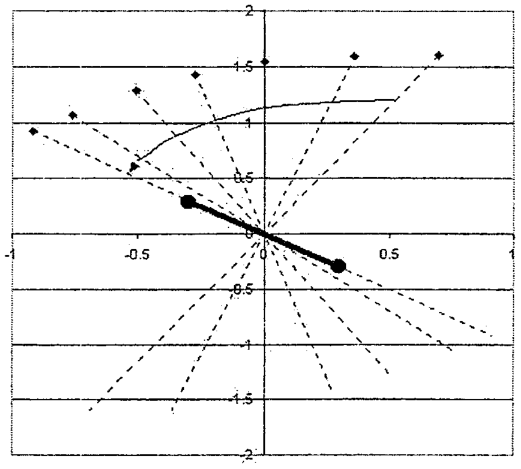


Fig. 8. X-Y plot of transient HSPICE results showing the convergence of the synaptic weights on the principal component vector of (-.3,.3) and (.3,-.3).

Acknowledgments

Funding for the fabrication of the integrated circuit came in part from the National Science Foundation (NSF). Circuits were designed using HSPICE and GSI from Avanti Corporation.

References

[1] M. Lades, J.C. Vorbruggen, J. Buhmann, J. Lange, C. von der Malsburg, R.P. Wurtz, and W. Konen, "Distortion invariant object recognition in the dynamic link architecture", *IEEE Trans. on Computers*, vol. 42, no. 3, Mar., 1993.

[2] S. Grossberg, "Some networks that can learn, remember, and reproduce any number of complicated space-time patterns", *Journal of Mathematics and Mechanics*, vol. 19, pp. 53-91, 1969.

[3] E. Oja, "A simplified neuron model as a principal component analyzer", *Journal of Mathematics and Biology*, vol. 15, pp. 267-273, 1982.

[4] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network", *Neural Networks*, vol. 2, pp. 459-473, 1989.

[5] G. Han and E. Sánchez-Sinencio, "CMOS transconductance multipliers: A tutorial", *IEEE Trans. on Circuits and Systems II*, vol. 45, pp. 1550-1563, Dec. 1998.