Visual Sensors for Focal Plane Computation of Image Features

Thesis by

Alberto Pesavento

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2002

(Submitted May 17, 2002)

Acknowledgements

First and foremost, I would like to thank my parents, Paolino and Caterina, and my brother Luca for their encouragement and emotional support throughout the **way too many** years I have been in school.

I would also like to give many thanks to Prof. Christof Koch for accepting me in his lab and for giving me the freedom and encouragement to pursue my projects.

I thank my candidacy and thesis committee: Prof. Chris Diorio, Prof. Pietro Perona, Prof. Demetri Psaltis, and Prof Yu-Chong Tai for reading my dissertation and providing a supportive environment.

I thank the Office of Naval Research (ONR) and the National Science Foundation's Engineering Research Center (ERC) for Neuromorphic Engineering at Caltech, for funding my research over the years.

I would like to thank again Prof. Pietro Perona and Dr. Arrigo Benedetti for the many discussions about the feature detection sensors.

I am grateful to Prof. Chris Diorio and Prof Timothy Horiuchi for the stimulating collaboration on the adaptation in focal-plane arrays project.

I thank Prof. Reid Harrison, Prof. Chuck Higgins, Dr. Oliver Landolt, Ania Mitros, and Theron Stanford, the members of the ToyLab, for all the interesting ideas on new sensors and, especially, for all the discussions on what is and what IS NOT Neuromorphic Engineering.

Finally, I would like to thanks the many friends at Caltech that made the last five years of my life so enjoyable. They are George Barbastathis, Arrigo Benedetti, Jean-Yves Bouguet, Francesco Bullo, Marco Casari, John Choi, Enrico Di Bernardo, Gianfranco Doretto, Luis Goncalves, Serena Guarnaschelli, Chris Moser, Barbara Polloni, and Melissa Saenz.

Abstract

Feature detection and tracking is a fundamental problem in computer vision research. By detecting and tracking features in an image sequence it is possible to recover information about both the motion of the viewer and the structure of the environment. The selection of features is a computationally intensive task. We derived two low-complexity algorithms that are suitable for integration in a CMOS sensor with focal-plane processing. We review the two algorithms and the circuits that implement them. We present results from accurate simulations and experimental results from the testing of these CMOS sensors.

Contents

Chapter 1 Introduction

With weights of the order of few grams and power consumptions well below one watt, integrated CMOS image sensors are becoming increasingly attractive as a replacement for the standard combination of CCD camera and image processor in such applications as machine vision systems and embedded systems.

Conventional systems are put at a disadvantage by the separation between a camera for *seeing* the world, and a computer for *figuring out* what is seen. The main advantage of CMOS image sensors, in fact, is the ability to integrate sensing and processing on the same chip. This advantage is especially important for implementing imaging systems requiring significant processing such as digital cameras and computational sensors.

In the field of digital cameras, Active Pixel Sensors (APS) using standard CMOS technologies have attracted a considerable amount of attention in the past few years. Nowadays CMOS APS imagers can be found at both the low end, and the high-end of the market spectrum. In APS sensors, processing can be integrated at the chip level using a "system-on-chip" approach, at the column level by integrating an array of processing elements each dedicated to one or more columns, and at the pixel level by integrating a processing element at each pixel or group of neighboring pixels. At present, chip and column level processing are the most widely used in APS sensors. Pixel level processing is generally dismissed as resulting in pixel sizes that are too large to be of practical use in such imaging systems.

The work on computational sensors, on the other hand, involves the integration of analog processing at the pixel level. By distributing and parallelizing the processing, speed can be reduced to the point where analog circuits operation in weak inversion (subthreshold) can be used, yielding a substantial reduction in systems power.

Some of the previous work in computational sensors has focused the attention on visual motion.

The analysis of motion using focal-plane pixel-parallel continuous time circuits is particularly appropriate because it eliminates the temporal aliasing problem found in sampled motion systems. Starting with the work of Mead and Tanner in 1986 [44] many works have been presented in this field of research [39].

While many of these sensors showed creativity and great insight, almost all of them failed to generate enough interest among potential users like computer vision specialists and researchers in autonomous robotic platforms. One of the main reason of this disconnection between sensors designers and users is the fact that most of these sensors compute quantities frequently not compatible with the algorithms developed by researcher for their applications.

Conscious of this problem, we tried the opposite approach. By looking at what the potential users want and need, we designed a series of computational sensors that would provide a real advantage over the standard combination of CCD camera and dedicated digital image processor.

The selection and tracking of features in an image stream is a fundamental problem in computer vision. In principle, from the stream of image frames produced by a moving camera, it is possible to recover the shape of objects in the field of view and the motion of the camera iteself. Information on both the structure of the environment and the motion of the viewer can be recovered from the displacement of key features in the image sequence. The selection of features is also a computationally intensive task and usually is accomplished off-line on a sequence of images recorded from a camera. Real-time systems, of which very few exist [43, 2], are usually bulky, expensive, require cameras and frame grabbers and have a very high power consumption.

Starting from a well-known algorithm for feature detection and tracking, we derived two algorithms that allow a substantial reduction in complexity to the point where an integration in a computational sensor with circuits working in weak inversion becomes possible. By carefully designing the circuits, we were able to implement the two algorithms without approximations, as results from accurate simulations will show. We present also results from the first attempt to include adaptation in CMOS visual sensor as a way to remove offsets mismatches that very often are the reason

behind inaccuracies in CMOS visual sensors operating in weak inversion.

The thesis is organized as follows: in chapter 2 we review the known algorithms for feature detection and tracking and present the two low-complexity feature selection algorithms. In chapter 3 we present the first computational sensor to implement the feature detection algorithm. In chapter 4 we present the final sensor design along with accurate simulations and experimental results from a fabricated chip. In chapter 5 we introduce the problem and present results from the first attempt to use floating-gate devices to reduce offsets mismatches in CMOS visual sensors. And, finally, in chapter 6 we summarize our contributions and present ideas for possible improvements and possible use of the sensors. Appendix A presents the family of wide linear range four quadrant multipliers that were the byproduct of the research on the computational sensors.

Chapter 2 The Feature Selection Problem

In principle, from the stream of image frames produced by a moving camera, it is possible to recover both the shape of objects in the field of view and the motion of the camera. Information on both the structure of the environment and the motion of the viewer can be recovered from the displacement of key features in the image sequence. The selection and tracking of features in an image stream is therefore a fundamental problem in computer vision.

In general two basic questions must be answered: how to select the features, and how to track them from frame to frame. One of the best know and most frequently used algorithm to track features was proposed by Lucas and Kanade in 1981 [29].

Their approach is to minimize the sum of squared intensity differences between past and current windows. Because of the small inter-frame motion, the current window can be approximated by a translation of the old one. Furthermore, for the same reason, the image intensities in the translated window can be written as those in the original window plus a residue term that depends almost linearly on the translation vector. As a result of these approximations, one can write a linear 2×2 system whose unknown is the displacement vector between the two windows.

The first question posed above, however, was left unanswered in [29]: how to select the windows that are suitable for accurate tracking. In the literature, several definitions of a "good feature" have been proposed, based on an *a priori* notion of what constitutes "interesting" windows. For example, Moravec and Thorpe proposed to use windows with high standard deviations in the spatial intensity profile [35, 45], Marr, Poggio, and Ullman prefer zero crossing of the Laplacian of the image intensity [30], and others define corner features based on first and second derivatives of the image intensity function [20, 12].

In contrast with these selection criteria, which are defined independently of the tracking algorithm, Tomasi and Kanade [46] presented a criterion that explicitly optimize the tracking performance. In other words, they define a feature to be good if it can be tracked well.

This feature selection method, while providing an elegant solution to the problem, is still computationally intensive, and an implementation in a CMOS sensor would be impossible without a substantial reduction in complexity.

For this purpose, we derived two successive simplifications to the original algorithm that allowed the design of two compact CMOS sensors that implement them. With the first simplification step, we obtained a low complexity algorithm that is equivalent to the original one proposed by Tomasi and Kanade, and, with the second step, an even further reduction in complexity was achieved at the cost of a lost equivalence with the original method.

In section 2.1 we will review the tracking method. In section 2.2 we will present the original selection method proposed by Tomasi and Kanade. In section 2.3 we present the simplified versions of the algorithm. In section 2.4 we present results from a comparison of the two selection criteria, and finally, section 2.5, summarize the contributions.

2.1 Feature Tracking Algorithm

As the camera moves, the patterns of image intensity change in a complex way. In general, any function of three variables $I(x, y, t)$, where the space variables x and y as well as the time variable t are discrete and suitably bonded, can represent an image sequence. However, images taken at near time instants are usually strongly related to each other, because they refer to the same scene taken from only slightly different viewpoints.

This correlation can be expressed by saying that there are patterns that move in the image stream. Formally, this means that the function $I(x, y, t)$ is not arbitrary,

but satisfies the following property:

$$I(x, y, t + \tau) = I(x - \xi, y - \eta, t) \; . \tag{2.1}$$

That is, a later image taken at the time $t + \tau$ can be obtained by moving every point in the current image, taken at time t, by a suitable amount. The amount $\mathbf{d} = (\xi, \eta)$ is called the *displacement* of the point at $\mathbf{x} = (x, y)$, between time instants t and $t + \tau$, and is in general a function of x, y, t, and τ.

Even in a static environment under constant lighting, the property described by equation (2.1) is violated in many situations. For instance, at occluding boundaries, points do not just move within the image, but appear and disappear. Furthermore, the photometric appearance of a region on a visible surface changes when reflectivity is a function of the viewpoint.

However, equation (2.1) is by and large satisfied at surface markings and away from occluding contours. At locations where the image intensity changes abruptly with x and y, the point of change remains well defined even in spite of small variations of overall brightness around it. Surface markings abound in images of natural scenes, and are not infrequent in man-made environments.

An important problem in finding the displacement $\mathbf{d}$ of a point from one frame to the next is that a sigle pixel cannot be tracked unless it has a very distinctive brightness with respect to all of its neighbours. In fact, the value of the pixel can change due to noise and be confused with adjacent pixels. As a consequence, it is often hard or impossible to determine where the pixel went in the subsequent frame, based only on local information.

Because of these problems, we do not track pixel but *windows* of pixels, and, as we will explain later on, we look for windows that contain sufficient texture.

Unfortunately, different points within a window may behave differently. The corresponding three-dimensional surface may be very slanted, and the intensity pattern in it can become warped from one frame to the next. Or the window may lie along an occluding boundary, so that points move at different velocities, and may even

disappear or appear anew.

This can be a problem in two ways. First, how do we know that we are following the same window, if its contents change over time? Second, if we measure "the" displacement of the window, how are the different velocities combined to give the one resulting vector?

One solution of the first problem is to keep checking that the appearance of a window has not changed too much. If it has then the window is discarded.

The second problem can be solved by utilizing a more complex transformation than a simple translation to describe the window changes. One such transformation is an affine map that allows to associate different velocities to different points of the window. Examples of these more complex transformations can be found already in [29, 37, 41].

On the one hand, when the world is known to be rigid, the danger of over-parametrizing the system outweighs the advantages of a richer model. More parameters to estimate require the use of larger windows to constrain the parameters sufficiently. On the other hand, using small windows implies that only few parameters can be estimated reliably, but also alleviates the problems mentioned above. The simplest choice therefore is to estimate only two parameters (the displacement vector) for small windows. Any discrepancy between successive windows that cannot be explained by translation is considered to be error, and the displacement vector is chosen so as to minimize the residue error.

Formally, if we define $J(\mathbf{x}) = I(x, y, t+\tau)$, and $I(\mathbf{x} - \mathbf{d}) = I(x-\xi, y-\eta, t)$, where the time variable has been dropped for brevity, our local image model is

$$J(\mathbf{x}) = I(\mathbf{x} - \mathbf{d}) + n(\mathbf{x}) \ ,$$

where n is noise.

The displacement vector $\mathbf{d}$ is then chosen so as to minimize the residue error

defined by the following double integral over the given window $\mathcal{W}$:

$$\epsilon = \int_{\mathcal{W}} [I(\mathbf{x} - \mathbf{d}) - J(\mathbf{x})]^2 \, w(\mathbf{x}) \, d\mathbf{x} \, . \tag{2.2}$$

In this expression, $w(\mathbf{x})$ is a weighting or window function. In the simplest case, w could be set to 1. Alternatively, $w(\mathbf{x})$ could be a Gaussian-like function or other functions that emphasize the central area of the window.

If the inter-frame displacement is sufficiently small with respect to the texture fluctuations within the window, the displacement vector itself can be written approximately as the solution to a 2×2 linear system of equations.

In particular, when the displacement vector is small, the intensity function can be approximated by its Taylor series truncated to the linear term:

$$I(\mathbf{x} - \mathbf{d}) = I(\mathbf{x}) - \mathbf{g}(\mathbf{x}) \cdot \mathbf{d} \, ,$$

where $\mathbf{g}(\mathbf{x})$ is the image gradient at location $\mathbf{x} = (x, y)$

$$\mathbf{g}(\mathbf{x}) = \left(\frac{\partial I(\mathbf{x})}{\partial x} \, , \, \frac{\partial I(\mathbf{x})}{\partial y} \right)^T$$

We can now write the residue defined in equation (2.2) as

$$\epsilon = \int_{\mathcal{W}} [I(\mathbf{x}) - \mathbf{g}(\mathbf{x}) \cdot \mathbf{d} - J(\mathbf{x})]^2 \, w(\mathbf{x}) \, d\mathbf{x} = \int_{\mathcal{W}} [h - \mathbf{g}(\mathbf{x}) \cdot \mathbf{d}]^2 \, w(\mathbf{x}) \, d\mathbf{x} \, , \tag{2.3}$$

where $h = I(\mathbf{x}) - J(\mathbf{x})$.

This residue is a quadratic function of the displacement $\mathbf{d}$. As a consequence, the minimization can be done in closed form. Differentiating the last expression of the residue ϵ in equation (2.3) with respect to $\mathbf{d}$, we have

$$\frac{\partial \epsilon}{\partial \mathbf{d}} = 2 \int_{\mathcal{W}} [h - \mathbf{g}(\mathbf{x}) \cdot \mathbf{d}] \, \mathbf{g}(\mathbf{x}) \, w(\mathbf{x}) \, d\mathbf{x} \, .$$

To find the displacement $\mathbf{d}$, we set the derivative to zero:

$$\frac{\partial \epsilon}{\partial \mathbf{d}} = 2 \int_{\mathcal{W}} [h - \mathbf{g}(\mathbf{x}) \cdot \mathbf{d}] \, \mathbf{g}(\mathbf{x}) \, w(\mathbf{x}) \, d\mathbf{x} = 0 \, .$$

Rearranging the terms, we obtain

$$\left[\int_{\mathcal{W}} \mathbf{g}(\mathbf{x}) \mathbf{g}^T(\mathbf{x}) \, w(\mathbf{x}) \, d\mathbf{x} \right] \mathbf{d} = \int_{\mathcal{W}} h \, \mathbf{g}(\mathbf{x}) \, w(\mathbf{x}) \, d\mathbf{x} \, ,$$

where we used the fact that

$$(\mathbf{g}(\mathbf{x}) \cdot \mathbf{d}) \, \mathbf{g}(\mathbf{x}) = (\mathbf{g}(\mathbf{x}) \, \mathbf{g}^T(\mathbf{x})) \, \mathbf{d} \, ,$$

and $\mathbf{d}$ is assumed constant within the window $\mathcal{W}$.

This is a system of two scalar equations in two unknowns. It can be rewritten as

$$\mathbf{G} \, \mathbf{d} = \mathbf{e} \, . \tag{2.4}$$

The coefficient matrix is the symmetric, 2×2 matrix

$$\mathbf{G} = \int_{\mathcal{W}} \mathbf{g}(\mathbf{x}) \, \mathbf{g}^T(\mathbf{x}) \, w(\mathbf{x}) \, d\mathbf{x} \, ,$$

that can be rewritten as

$$\mathbf{G} = \int_{\mathcal{W}} \mathbf{g}(\mathbf{x}) \, \mathbf{g}^T(\mathbf{x}) \, w(\mathbf{x}) \, d\mathbf{x} = \begin{bmatrix} \sum_{k=1}^{N} w(k) \, \left(I_x^k\right)^2 & \sum_{k=1}^{N} w(k) \, I_x^k I_y^k \\ \sum_{k=1}^{N} w(k) \, I_x^k I_y^k & \sum_{k=1}^{N} w(k) \, \left(I_y^k\right)^2 \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \, , \tag{2.5}$$

where the partial derivatives of $I(x, y, t)$ with respect of x and y are denoted by I_x, and I_y and the number of pixels in the window $\mathcal{W}$ is N.

The right side of the system of equation (2.4) is the two-dimensional vector

$$\mathbf{e} = \int_{\mathcal{W}} h \, \mathbf{g}(\mathbf{x}) \, w(\mathbf{x}) \, d\mathbf{x} = \int_{\mathcal{W}} [I - J] \, \mathbf{g}(\mathbf{x}) \, w(\mathbf{x}) \, d\mathbf{x}$$

where h is explicitly written as the difference between the two frames I and J.

Equation (2.4) is the basic step of the tracking procedure. For every pair of adjacent frames, the matrix $\mathbf{G}$ can be computed from one frame by estimating gradients and computing their second order moments. The vector $\mathbf{e}$, on the other hand, can be computed from the difference between the two frames, along with the gradient computed above. The displacement $\mathbf{d}$ is then the solution to the system (2.4).

It is possible to obtain a corresponding expression for the continuous time case that is more closely related to the computations performed in visual sensors where the computation is not clocked.

If $I(x, y, t)$ is the image brightness, under the assumption of constant brightness, we can write

$$\frac{dI(x,y,t)}{dt} = I_x v_x + I_y v_y + I_t = [\, I_x \;,\; I_y \,]\, \mathbf{v} + I_t = 0$$

where dI/dt is the total temporal derivative of the image intensity, $\mathbf{v} = [v_x \; v_y]^{\mathrm{T}} = [dx/dt \; dy/dt]^{\mathrm{T}}$ and the partial derivatives of $I(x, y, t)$ with respect of x, y and t are denoted by I_x, I_y and I_t respectively.

If we make the assumption that all the N points in the region or window of interest are moving at the same speed, which is reasonable for small displacements and a small neighborhood, we can formulate the linear problem

$$\begin{bmatrix} I_x^1 & I_y^1 \\ \vdots & \vdots \\ I_x^N & I_y^N \end{bmatrix} \mathbf{v} = - \begin{bmatrix} I_t^1 \\ \vdots \\ I_t^N \end{bmatrix} \quad \text{or} \quad \mathbf{A}\mathbf{v} = \mathbf{b}\,.$$

The velocity vector $\mathbf{v}$ can be computed as the least squares solution to

$$\mathbf{A}\mathbf{v} = \mathbf{b}, \qquad \text{i.e.} \qquad \mathbf{v} = \mathbf{G}^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}, \tag{2.6}$$

where

$$\mathbf{G} = \mathbf{A}^{\mathrm{T}}\mathbf{A} = \begin{bmatrix} \sum_{k=1}^{N} \left(I_x^k\right)^2 & \sum_{k=1}^{N} I_x^k I_y^k \\ \sum_{k=1}^{N} I_x^k I_y^k & \sum_{k=1}^{N} \left(I_y^k\right)^2 \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} .$$

Once again the velocity $\mathbf{v}$ is related to the partial derivative of the image intensity with respect of time $\mathbf{b}$ (see equation (2.6)) in much the same way the displacement $\mathbf{d}$ was related to the interframe difference $\mathbf{e}$ in the system of equation (2.4).

2.2 Feature Selection Algorithm

Not all the regions in an image contain motion information. Some researchers propose to track corners or windows with high spatial frequency content or region where some mix of second-order derivatives is sufficiently high. It is possible to use a different approach that does not require to define *a priori* the characteristics of the image features to track. We can select a region in an image only if that region can be tracked well and omit a region if it is not good for the purpose. In this way the selection criterion is optimal by construction.

With the formulation introduced in the previous section, this concept is easy to formalize. In fact, we can track a window from frame to frame if the systems (2.4) represents good measurements, and if it can be solved reliably.

This means that $\mathbf{G}$ must be both above noise and well-conditioned. The noise requirement implies that both eigenvalues λ_1 and λ_2 (with $\lambda_1 \leq \lambda_2$) of $\mathbf{G}$ must be large, while the conditioning requirement means that they cannot differ by several orders of magnitude. This corresponds to enforcing an upper bound on the condition number:

$$\mathrm{cond}(\mathbf{G}) = \frac{\lambda_2}{\lambda_1} < c_{th} .$$

Two small eigenvalues mean a roughly constant intensity profile within a window. A large and a small eigenvalue correspond to unidirectional pattern. Note that when one eigenvalue is zero (for instance $\mathbf{G}$ is rank deficient), there is an unreachable

subspace in the solution for $\mathbf{v}$ that corresponds to the velocity component parallel to the direction of the edge. This situation is often referred to as the Aperture Problem. Two large eigenvalues are found for windows with corners, salt-and-pepper textures, or any other pattern that can be tracked reliably.

For all practical purposes, when the smaller eigenvalue is sufficiently large to meet the noise criterion, the matrix $\mathbf{G}$ is usually well conditioned. This is due to the fact that the intensity variations in a window are bounded by the maximum allowable pixel value, so the greater eigenvalue, λ_2, cannot be arbitrarily large.

This observation simplifies the selection of the trackable windows as to the ones for which

$$\lambda_1 > \lambda_t \tag{2.7}$$

where λ_1 is the minimum eigenvalue and λ_t is a predefined threshold value. This is the method to select image features proposed by Tomasi-Kanade [46]. In real implementations it is usually performed off-line with a computer working on an image stream produced by a camera and stored in the disk of a computer.

There are several ways to determine the appropriate value for λ_t for tracking purposes. A common approach is to measure the minimum eigenvalues λ_1 for images of regions with approximately uniform brightness. The average of these values is a good estimate of a lower bound for λ_t. Again, taking an average of λ_1 for highly textured regions or windows with various features yields an estimate for an upper bound to λ_t. Usually the two bounds are well separated and a choice of λ_t halfway between is usually not critical.

In Fig. 2.1 we show a picture taken in the corridor just outside our lab. It is a typical scene from a man-made environment, but it was carefully chosen as it contains several examples of the kind of features we are interested in as we will see later. In Fig. 2.2 we re-plot the same image where the white boxes indicates the features that were selected applying the Tomasi-Kanade algorithm. For this example we used a window of size 3×3 pixels, and an operation of local maximization was performed in order to isolate the pixel with the highest minimum eigenvalue if there were a cluster

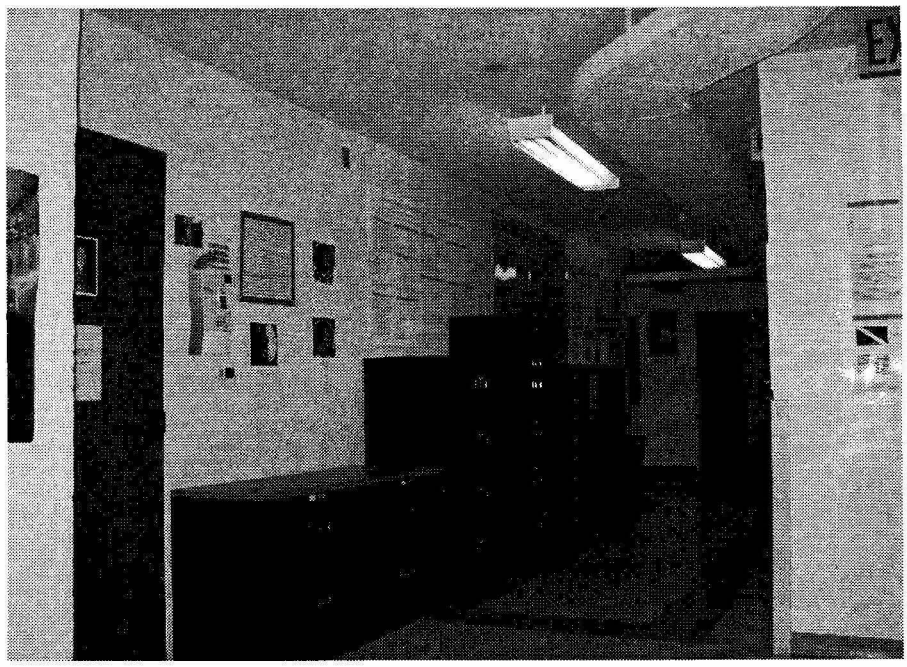

Figure 2.1: A picture taken in the corridor just outside our lab. It is a typical scene from a man-made environment, but it was carefully chosen as it contains several examples of the kind of features we are interested in.

of adjacent pixels with minimum eigenvalue above threshold.

Fig. 2.3 shows a detailed view of some of the features detected in Fig. 2.2. All feature windows have substantial variation of intensity and can be characterized as having a spatial gradient significantly different from zero in both the horizontal and vertical directions even if most of them can hardly be classified as "corners."

Fig. 2.4 shows the histogram of the number of pixels in Fig. 2.2 satisfying equation (2.7) for different values of the minimum eigenvalue λ_1. For the example of Fig. 2.2, a threshold value of 10000 was chosen yelding few hundreds pixel locations. Varying the windows size, of course, the magnitude of the coefficients a, b, and c of the matrix $\mathbf{G}$ increases since the summations have to be carried out over a larger number of pixels. An increase in the coefficients leads to an increase in the minimum eigenvalue λ_1 for every pixel, as we can see from Fig. 2.5. The histograms, for increasing windows sizes, are, in fact, stretched towards higher λ_1 values.

Yet, even a region that is very rich in texture can be very poor for tracking

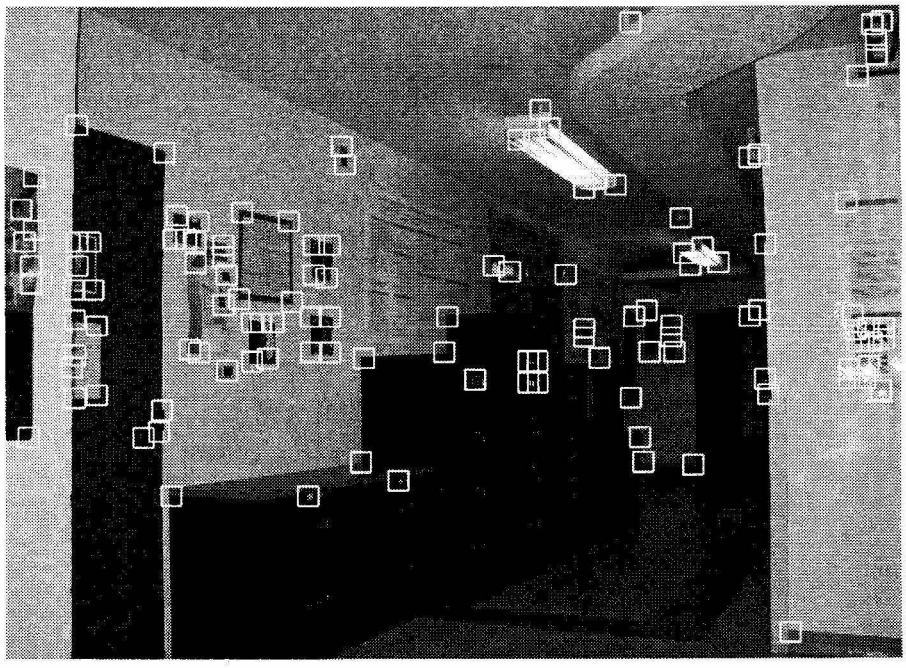

Figure 2.2: The same image of Fig. 2.1 with the white boxes indicating the features selected by the Tomasi-Kanade algorithm. For this example we used a window of size 3×3 pixels, and an operation of local maximization was performed in order to isolate the pixel with the highest minimum eigenvalue if there were a cluster of adjacent pixels all above the threshold.

purposes. Two of the most common cases of *bad* features that are useless or even harmful for most applications of the feature-tracking algorithm are reported in the two detailed view of Fig. 2.6. As we can see the presence of a depth discontinuity, like the corners in both the left and right view, or the boundary of a reflection highlight on a glossy surface, like the reflection on the poster on the right view, can lead the algorithm to believe that there is a *good* feature while in reality the feature is just an artifact due to that particular view or due to the lightning and is not attached to a fixed point in the environment. It is possible to monitor the quality of the image features during tracking to avoid this problem. Since this is beyond the scope of this thesis, we prefer to refer the reader, for example, to the work of Shi and Tomasi [41].

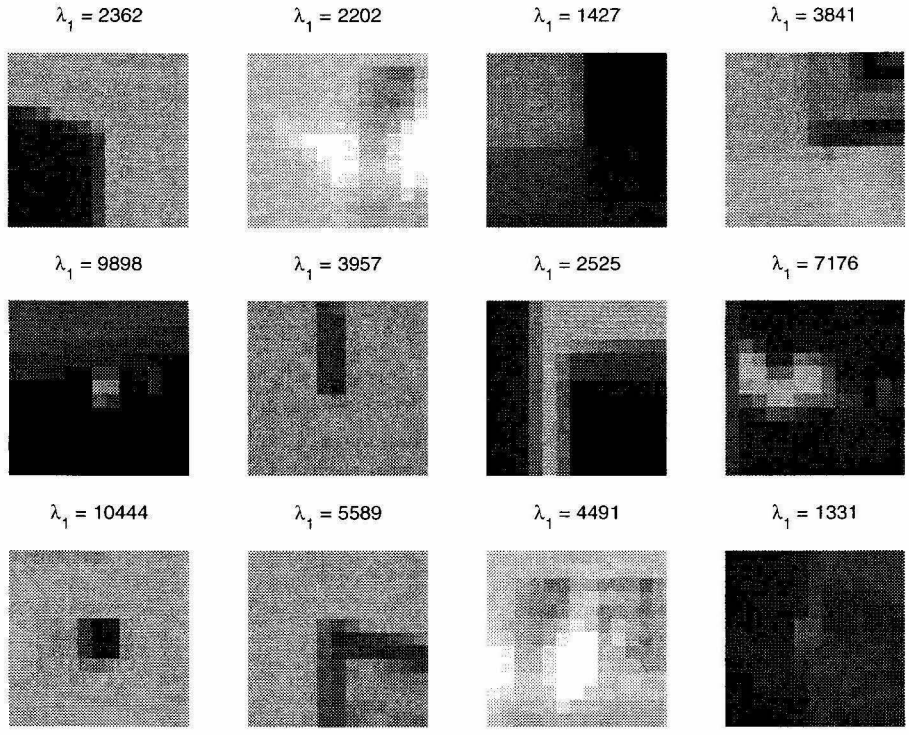

Figure 2.3: Detailed view of some of the features detected in Fig. 2.2. All feature windows have substantial variation of intensity and can be characterized as having a spatial gradient significantly different from zero in both the horizontal and vertical directions even if most of them can hardly be classified as "corners." The value of the minimum eigenvalue λ_1 for every feature is reported above every image patch.

2.3 Complexity Reduction

In order to design a computational sensor implementing the algorithm, it is necessary to have a better understanding of the complexity of the algorithm step represented by equation (2.7).

The eigenvalues of the matrix $\mathbf{G}$ are the roots of its characteristic polynomial

$$P(\lambda) = (a - \lambda)(c - \lambda) - b^2 \tag{2.8}$$

where a, b and c are the coefficients of $\mathbf{G}$; see equation (2.5). Equating the expression of $P(\lambda)$ to zero

$$P(\lambda) = (a - \lambda)(c - \lambda) - b^2 = \lambda^2 - (a + c)\,\lambda + (ac - b^2) = 0 \tag{2.9}$$

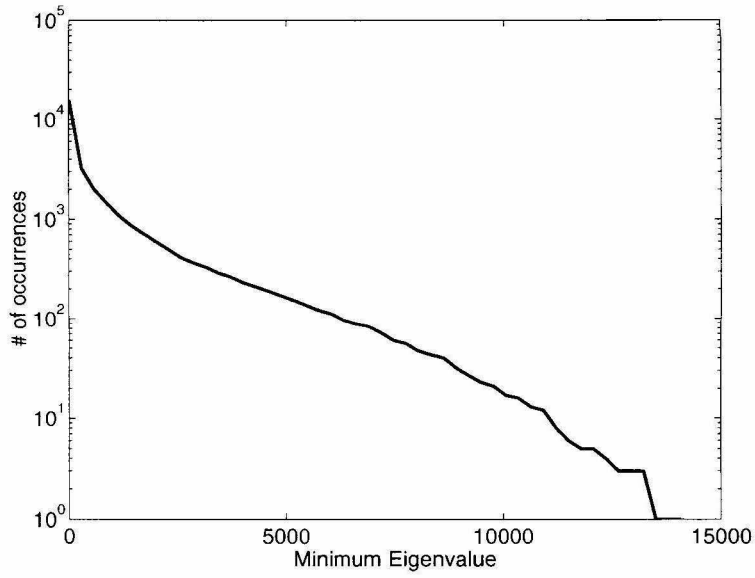

Figure 2.4: Histogram of the number of pixels of Fig. 2.1 satisfying equation (2.7) for different values of the minimum eigenvalue λ_1. For the example of Fig. 2.2, a threshold value of 10000 was chosen yelding few hundreds pixels location.

and solving it for the smaller of the two solutions, we find that the minimum eigenvalue λ_1 can be computed as

$$\lambda_1 = \frac{a+c}{2} - \sqrt{\left(\frac{a+c}{2}\right)^2 - (ac - b^2)} \,. \tag{2.10}$$

Since a sensor implementing the algorithm would have to compute λ_1 at every pixel to check for the condition (2.7), it is necessary to find a simplified expression for λ_1 because the computation represented by equation (2.10) is too complex to be implemented at every pixel of the sensor.

It is important to point out that even if in the following pages we focus attention on how to reduce the complexity of the computation for λ_1, it should be remembered that the computation of equation (2.10) is just the final step of the algorithm at (2.7). A significant amount of computation is hidden underneath the coefficients a, b, and c as we can see in equation (2.5).

The first thing we need to show is that the minimum eigenvalue λ_1 is always more or equal to zero (i.e., $\lambda_1 \geq 0$). Looking at equation (2.10) and remembering that a

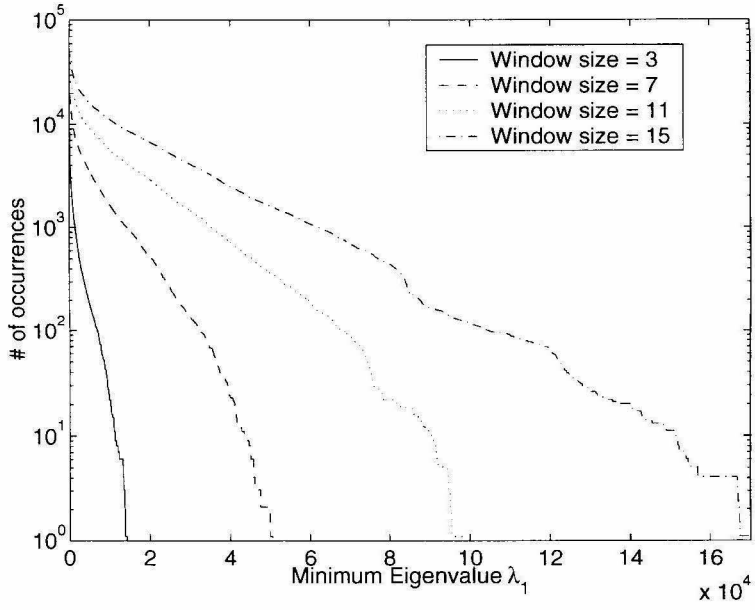

Figure 2.5: Histogram of the number of pixels satisfying equation (2.7) for different values of the minimum eigenvalue λ_1 and different values of window size.

and c are positive by construction (sums of second order moments), it is clear that $\lambda_1 \geq 0$ if and only if

$$\frac{a+c}{2} \geq \sqrt{\left(\frac{a+c}{2}\right)^2 - (ac - b^2)}\,,$$

and this is true only if $ac - b^2 \geq 0$.

Now, remembering equation (2.5), if we define the vectors

$$\mathbf{x} = \left[\sqrt{w(1)}I_x^1, \cdots, \sqrt{w(N)}I_x^N\right]^T \quad \text{and} \quad \mathbf{y} = \left[\sqrt{w(1)}I_y^1, \cdots, \sqrt{w(N)}I_y^N\right]^T$$

it is clear that

$$\begin{aligned} a &= \mathbf{x}^T\mathbf{x} = \mathbf{x}\cdot\mathbf{x} = ||\mathbf{x}||^2 \\ b &= \mathbf{x}^T\mathbf{y} = |\mathbf{x}\cdot\mathbf{y}|^2 \\ c &= \mathbf{y}^T\mathbf{y} = \mathbf{y}\cdot\mathbf{y} = ||\mathbf{y}||^2\,, \end{aligned}$$

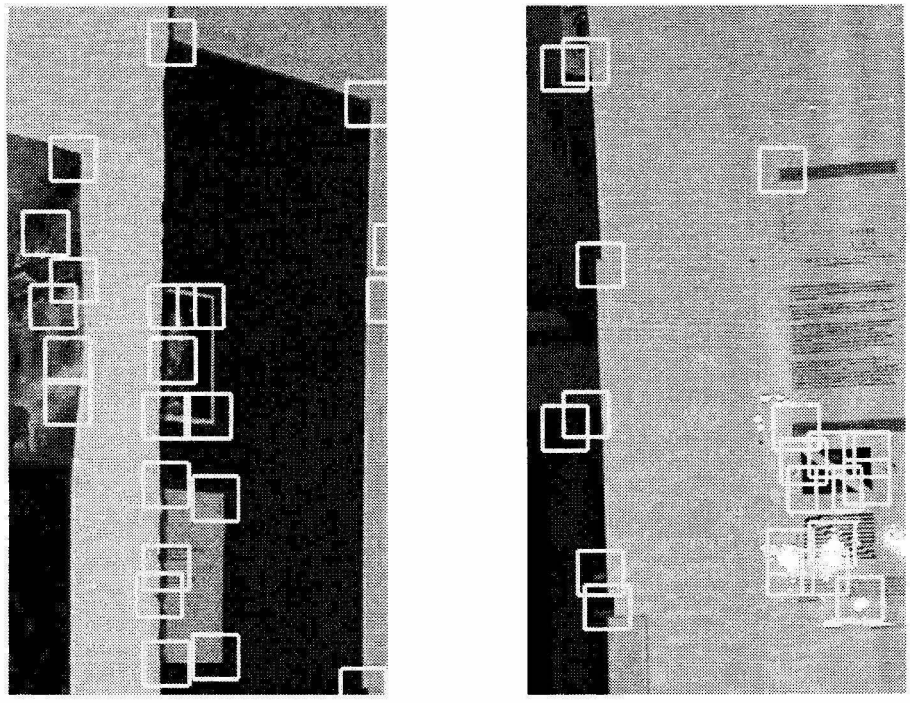

Figure 2.6: The presence of a depth discontinuity, like the corners in both the left and right view, or the boundary of a reflection highlight on a glossy surface, like the reflection on the poster in the right view, can lead the algorithm to believe that there is a *good* feature while in reality the feature is just an artifact due to that particular view or due to the lightning and is not attached to a fixed point in the environment.

and, finally, remembering the Cauchy-Schwartz inequality, $||\mathbf{x}||||\mathbf{y}|| \geq |\mathbf{x} \cdot \mathbf{y}|$, we have

$$\begin{aligned} ||\mathbf{x}||||\mathbf{y}|| &\geq |\mathbf{x} \cdot \mathbf{y}| \\ ||\mathbf{x}||^2||\mathbf{y}||^2 &\geq |\mathbf{x} \cdot \mathbf{y}|^2 \\ (\mathbf{x} \cdot \mathbf{x})(\mathbf{y} \cdot \mathbf{y}) &\geq |\mathbf{x} \cdot \mathbf{y}|^2 \\ (\mathbf{x}^T\mathbf{x})(\mathbf{y}^T\mathbf{y}) &\geq \mathbf{x}^T\mathbf{y} \\ a\,c &\geq b^2 \\ a\,c - b^2 &\geq 0\,. \end{aligned}$$

Once we established that the minimum eigenvalue λ_1 is non negative, we can plot, see Fig. 2.7, the parabolic function $P(\lambda)$ of the characteristic polynomial of equations (2.8) and (2.9).

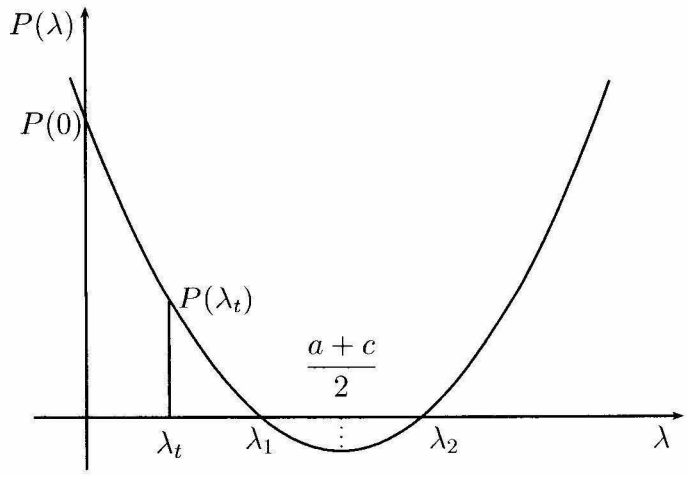

Figure 2.7: Requiring $\lambda_1 > \lambda_t$ is equivalent to the conditions $P(\lambda_t) > 0$ and $\frac{a+c}{2} > \lambda_t$, as shown in this picture. Furthermore, by requiring that $P(0) > p_t$ we are imposing a restriction on λ_1 to be sufficiently large.

From Fig. 2.7 we can see that, for a given target λ_t, if $\frac{a+c}{2} > \lambda_t$ then λ_t is constrained to lie between zero and the minimum of the parabolic function (i.e., $\frac{a+c}{2}$). Furthermore, if $P(\lambda_t) > 0$ then λ_t has to be smaller than the minimum eigenvalue λ_1 that is the lowest root of the function $P(\lambda)$.

Summarizing, the condition $\lambda_1 > \lambda_t$ is equivalent to

$$P(\lambda_t) > 0 \quad \text{and} \quad \frac{a+c}{2} > \lambda_t \,. \tag{2.11}$$

Remembering that the coefficients a and c are positive by construction, the previous condition is equivalent to

$$P(\lambda_t) > 0 \quad \text{and} \quad a > \lambda_t \quad \text{and} \quad c > \lambda_t \tag{2.12}$$

and this is because if both $a > \lambda_t$ and $c > \lambda_t$, then it is obvious that $\frac{a+c}{2} > \lambda_t$. If either only $a > \lambda_t$ or only $c > \lambda_t$ it is still possible that $\frac{a+c}{2} > \lambda_t$ but the condition $P(\lambda_t) > 0$ would be not satisfied. Finally, if both $a < \lambda_t$ and $c < \lambda_t$, both conditions (2.11) and (2.12) are not satisfied.

Finally, we can observe that it is not necessary to test that both $a > \lambda_t$ *and* $c > \lambda_t$

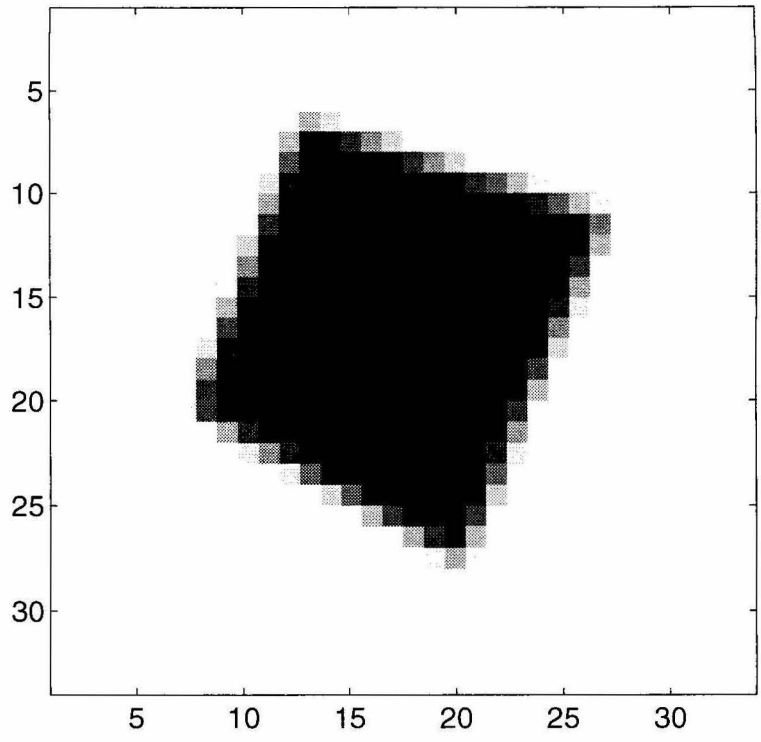

Figure 2.8: Synthetic image used as example.

as it is stated in (2.12). If, for example, the condition $a > \lambda_t$ is verified, the fact that $P(\lambda_t) > 0$ automatically imply that $c > \lambda_t$. Consequently, the conditions originally expressed by the equations (2.11) are equivalent to either one of the following two conditions:

$$P(\lambda_t) = (a - \lambda_t)(c - \lambda_t) - b^2 > 0 \quad \text{and} \quad a > \lambda_t \tag{2.13}$$

or

$$P(\lambda_t) = (a - \lambda_t)(c - \lambda_t) - b^2 > 0 \quad \text{and} \quad c > \lambda_t \ . \tag{2.14}$$

Looking at equations (2.13-2.14) and equation (2.10), it is evident how these simple observations allow a significant complexity reduction. Equation (2.10) requires five multiplications/divisions, five additions/subtraction and one square root while equations (2.13-2.14) require just three subtractions and two multiplications.

Since the original method proposed by Tomasi and Kanade and the one just derived are completely equivalent, it is not necessary to show an example of its application on the image proposed before because the features detected would be exactly

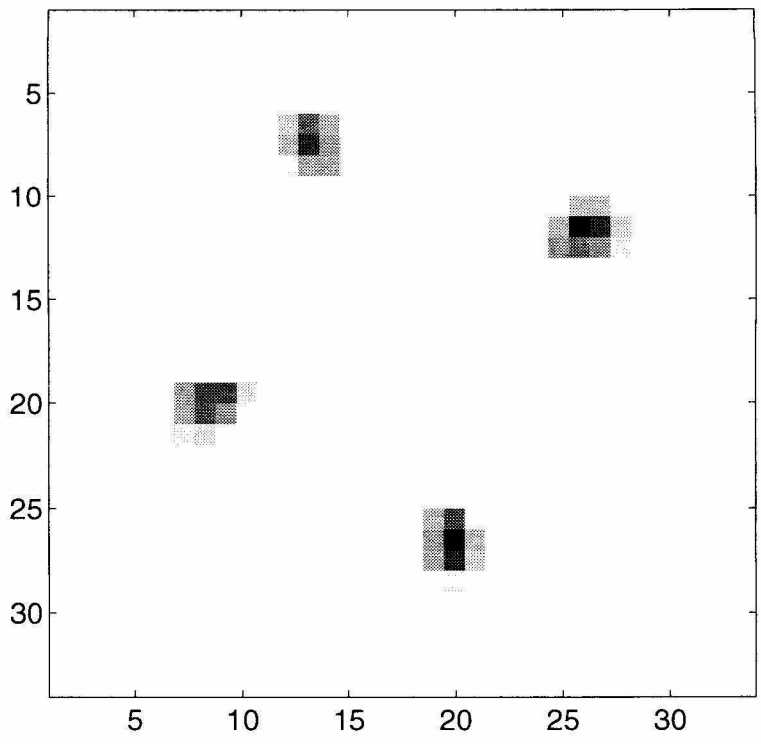

Figure 2.9: Plot of the minimum eigenvalue λ_1 for the synthetic of Fig. 2.8.

the same as in Fig. 2.2.

On the other hand, it is possible to gain some insight by looking at some of the quantities like λ_1 and $P(\lambda_t)$ that play an important role in the computation. In this case it is better if the synthetic image of Fig. 2.8 is used in place of the image in Fig. 2.1. In Fig. 2.9 and Fig. 2.10 we plot the minimum eigenvalue λ_1. It is clear that λ_1 is higher in the proximity of the four corners of the synthetic image. In Fig. 2.11 and Fig. 2.12 we present $P(\lambda_t)$ for a particular choice of λ_1. Even in this case, $P(\lambda_t)$ is maximum for the pixels that correspond to the four corners of the synthetic image of Fig. 2.8.

It is possible to decrease the computation complexity even further if we make the following observation. As we can see from Fig. 2.7, the higher the intercept $P(0)$ of $P(\lambda)$ with the ordinate axis, the larger the minimum eigenvalue λ_1 we should expect to be. Therefore, by imposing a threshold on $P(0)$, a restriction on the minimum eigenvalue λ_1 is indirectly imposed.

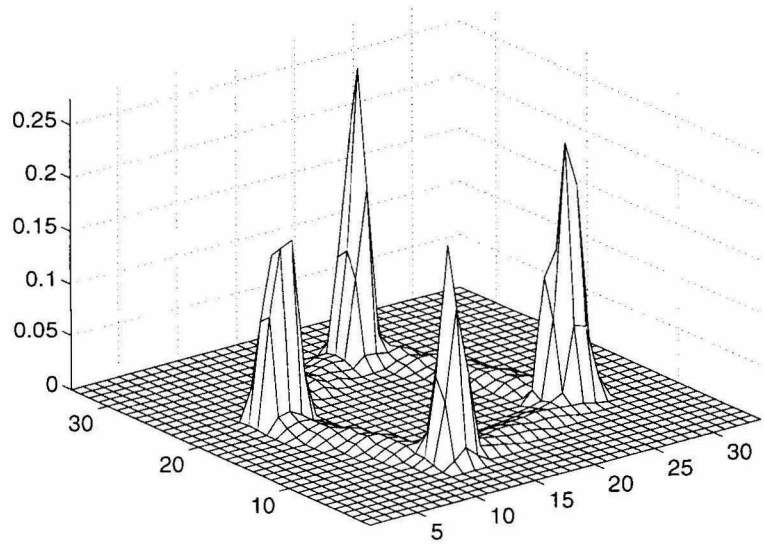

Figure 2.10: Another view of the minimum eigenvalue λ_1 for the synthetic of Fig. 2.8.

By forcing $P(0)$ to be greater than a certain treshold value, for example,

$$P(0) > p_t \ , \tag{2.15}$$

we can make sure that λ_1 is sufficiently large and therefore the window centered on the pixel most likely contains a feature.

The complexity reduction arises from the fact that the computation of $P(0) = ac - b^2$ is simpler that the computation of $P(\lambda_t) = (a - \lambda_t)(c - \lambda_t) - b^2$. Furthermore, with this simplified version, only one test is necessary compared to the three necessary before.

In Fig. 2.13 we show the features selected in the real image by the approximate algorithm. As we can see, comparing this figure with Fig. 2.2, the two algorithms select roughly the same features. Finally in Fig. 2.15 and Fig. 2.16 we plot the quantity $P(0) = ac - b^2$. We can notice how the minimum eigenvalue λ_1 of Fig. 2.9 and Fig. 2.10 and the plots of $P(0) = ac - b^2$ are very similar and again $P(0) = ac - b^2$ is maximum for the pixels corresponding to the four corners of the synthetic image. Fig. 2.14 shows the histogram of the number of pixels in Fig. 2.2 satisfying equation (2.15) for different values of the threshold p_t and different windows sizes. The same

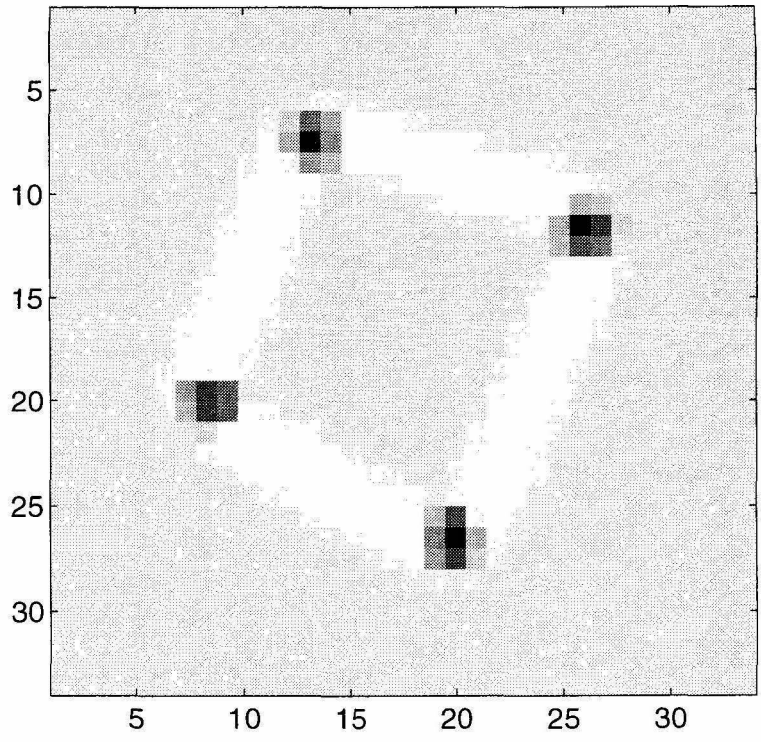

Figure 2.11: $P(\lambda_t)$ for a particular choice of λ_1.

considerations about the increase of magnitude for the coefficients a, b, and c applies here, as in fact, the histograms are stretched towards higher values of p_t for increasing window sizes.

We can summarize the two algorithms as follows:

a) Compute I_x and I_y at every pixel location.

b) For the window centered at (x, y), compute a, b and c as defined by equation (2.5).

c) Compute the polynomial $P(\lambda_t) = (a - \lambda_t)(c - \lambda_t) - b^2$ or $P(0) = ac - b^2$ depending on which algorithm is used.

The window contains a trackable feature if

$$P(\lambda_t) > 0 \quad \text{and} \quad a > \lambda_t$$

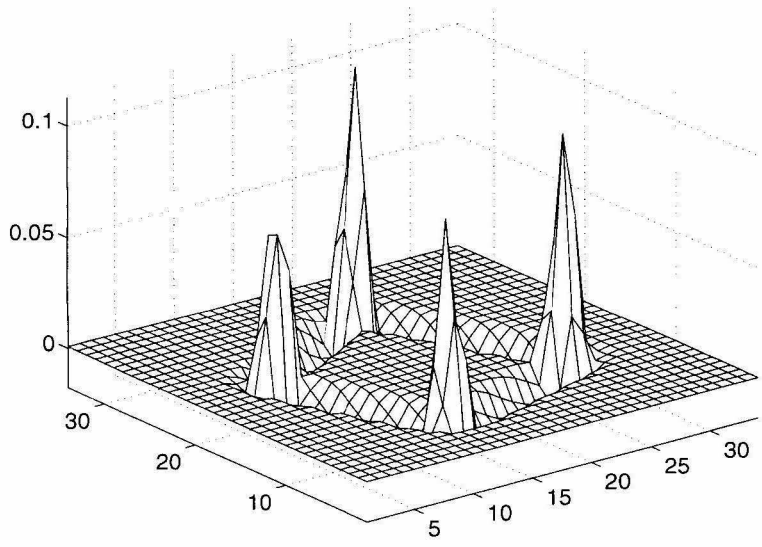

Figure 2.12: Another view of $P(\lambda_t)$.

or, if the approximated algorithm is used,

$$P(0) > p_t \ .$$

Sometime it may be necessary to modify the condition $P(\lambda_t) > 0$ with the more stringent one $P(\lambda_t) > p_t$ where p_t is a second threshold introduced to eliminate any false positive given by image noise.

The selection of the two threshold values λ_t and p_t depends on the contrast of the image and the desired density of features.

2.4 Analysis of the Two Algorithms

The two algorithms, the Tomasi-Kanade of equation (2.7) (or its equivalent simplified version of equations (2.13-2.14)) and the approximated algorithm of equation (2.15) are similar but not exactly equivalent. Even looking at Fig. 2.2 and Fig. 2.13, we can see that most of the apparent features of the images are detected by both algorithms, but that some are missed by one or the other.

One observation we can make for sure is that all the pixels that satisfy equation

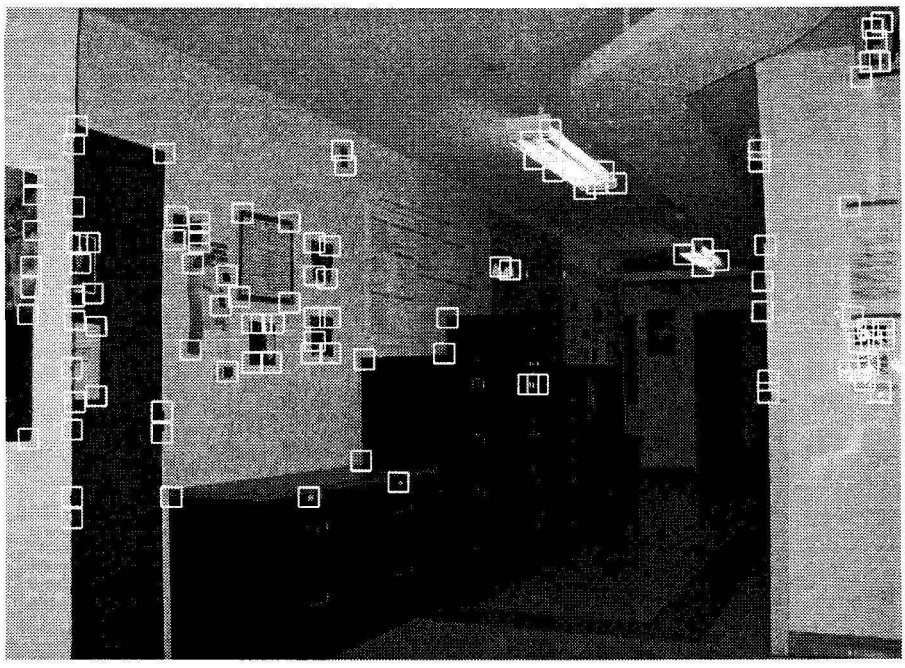

Figure 2.13: Features selected in the image by the approximate algorithm expressed by equation (2.15). As we can see, comparing this figure with Fig. 2.2, the two algorithms select roughly the same features.

(2.7) for some λ_t also satisfy the approximated algorithm for some value of p_t. And vice versa, all the pixels that satisfy the approximated algorithm for some value of p_t also satisfy the original algorithm for some other value of λ_t. This is because, for all pixels, the minimum eigenvalue λ_t and the intersection of the parabolic function $P(\lambda)$ with the ordinate axis $P(0) = ac - b^2$ is always positive.

The two algorithms would be exactly equivalent if and only if for every choice of λ_t there exists a corresponding choice for p_t such that the test $P(0) > p_t$ selects exactly the same pixels (in location and number). This is only possible if, for all the pixels, the parabolic functions of equation (2.8) have the same curvature. This is generally not the case and depends on the statistics of the images under consideration.

In Fig. 2.17, for example, we report the plot of the parabolic function of equation (2.8) for all the pixels of the synthetic image of Fig. 2.8 satisfying either one of the algorithms for a particular choice of λ_t and p_t. More specifically, we chose λ_t and p_t so that the two algorithms would be satisfied by a specific number (in this case 7) of pixels. It turns out that 6 out of the 7 pixels in the two groups are the same. At

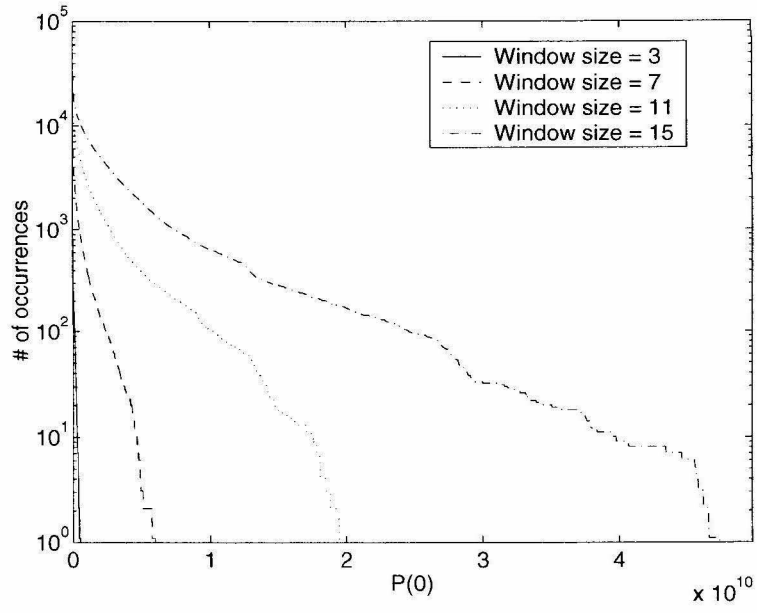

Figure 2.14: Histogram of the number of pixels satisfying equation (2.15) for different values of the threshold p_t and different values of window size.

least in this case, therefore, the two algorithms select almost the same pixels. We can in fact see how in Fig. 2.5 one of the parabolic functions crosses the ordinate axis below the threshold value p_t and another one (barely visible in the plot) crosses the abscissa axis to the left of the chosen threshold value λ_t.

It is interesting to have an idea of the percentage of the number of pixels that satisfy both algorithms for a given number of pixels satisfying each one. In other words, after choosing the value for the threshold λ_t that selects, for example, 1000 pixels and choosing the value of p_t also giving 1000 pixels, we are interested in knowing how many pixels are in common in the two groups. In Fig. 2.18 we plot that percentage, for different window sizes. As we would expect the percentage increases with the group size and, as the number increases toward the total number of pixels in the image, the percentage tends to 100%. In Fig. 2.19 we plot a zoom of Fig. 2.18 for very small group sizes. As we can see, even for very small group sizes the percentage quickly reaches a plateau between 60% and 80%.

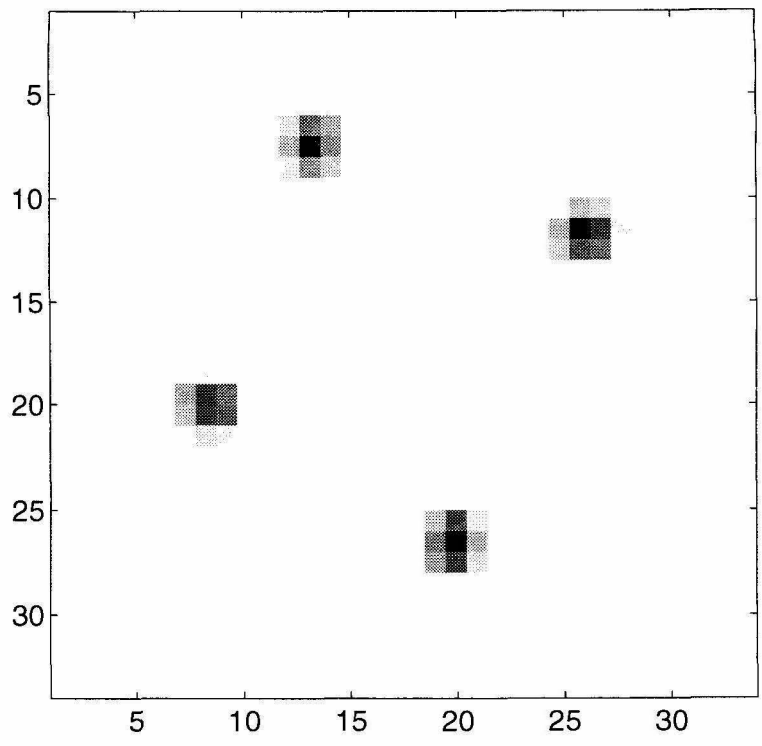

Figure 2.15: Plot $P(0) = ac - b^2$ for the synthetic image.

2.5 Conclusion

In this chapter we presented the feature selection and tracking problem and we derived the feature selection equations for both the discrete and continuous time cases. We presented the Tomasi and Kanade algorithm and the simplified equivalent version of conditions (2.11) and (2.12). This simplified version was first used by Benedetti and Perona [2] in their real-time implementation of the Tomasi and Kanade algorithm using a FPGA based custom board. We then presented an approximated version of the same algorithm that allow an even further reduction of the computation at the expense of the lost of the equivalence with the original algorithm. We also presented a more in depth study of the two different algorithms in order to better understand similarities and differences of the two. As we will see in the next chapter, the design of the computational sensors implementing the two algorithms are extremely similar and there is a minimal gain in choosing to just implement the simpler one.

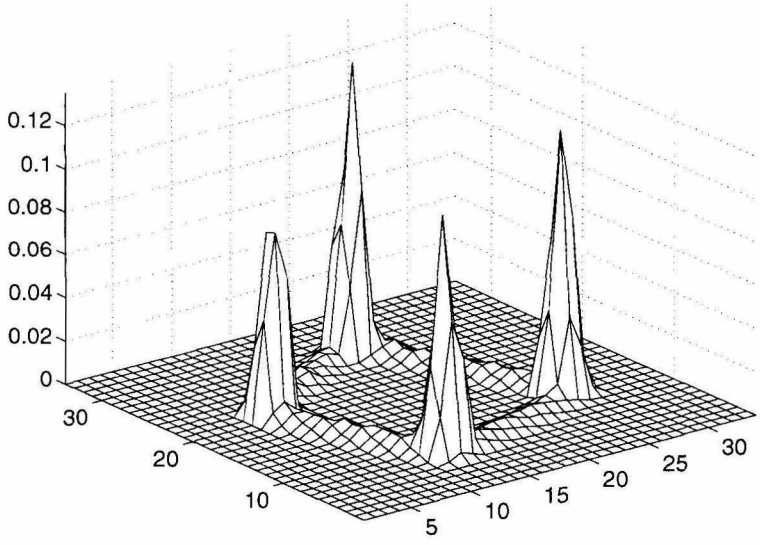

Figure 2.16: Another view of $P(0) = ac - b^2$ for the synthetic image.

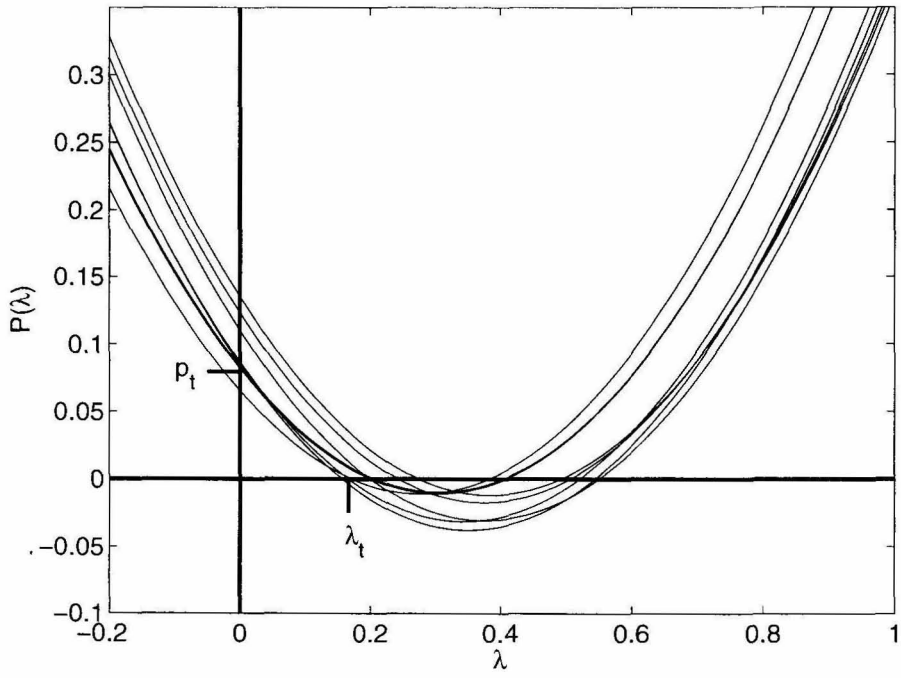

Figure 2.17: Plot of the characteristic functions for the two goups of seven pixels.

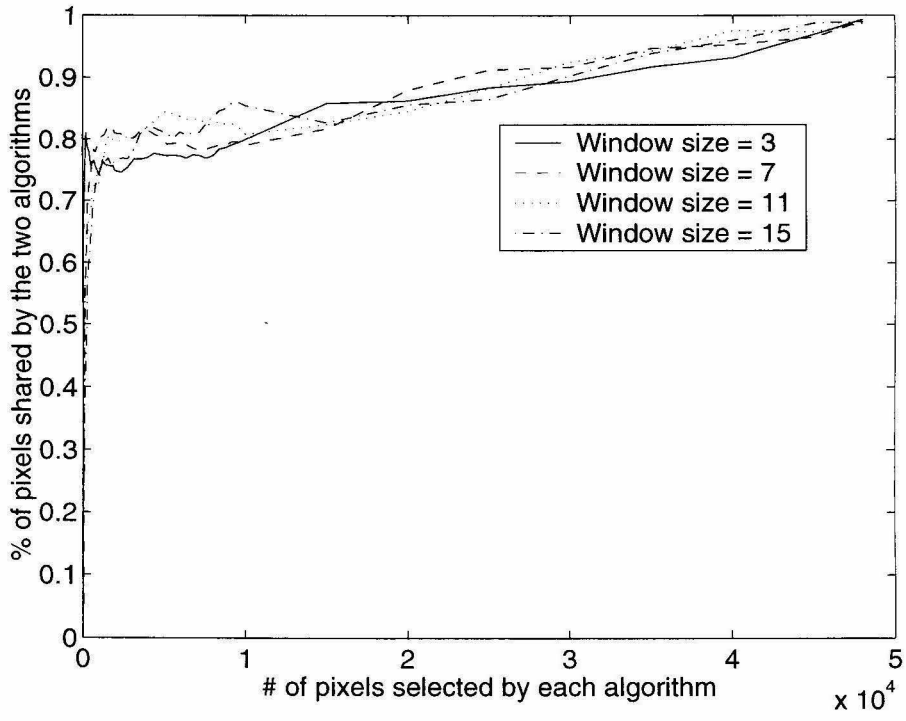

Figure 2.18: Percentage of the number of pixels that satisfy both the simplified and the approximated algorithm for a given number of pixels satisfying each one.

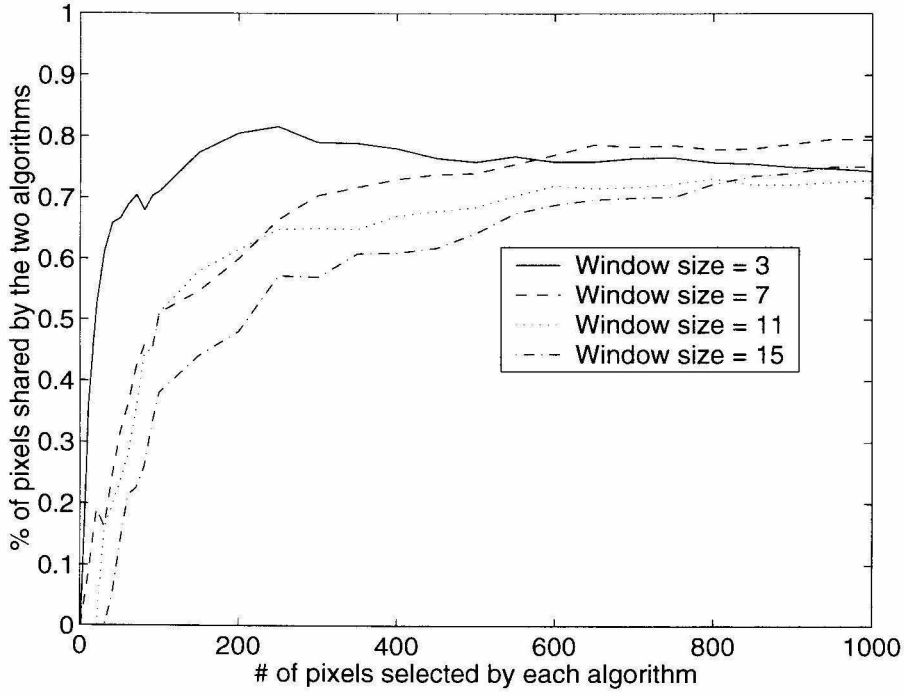

Figure 2.19: Zoom in on the plot of Fig. 2.18.

Chapter 3 First CMOS Implementation

In this chapter we present the description of *Detector1*, the first computational sensor, implementing the algorithms presented in the previous chapter, that was designed, fabricated and tested by us.

In order to better understand the various design issues that we confronted and the choices made in the design, it is useful to represent the algorithms under a different prospective. We can, in fact, conceptually separate the computation performed by the CMOS sensor in four separate layers as it is illustrated in Fig. 3.1.

1) First light is detected and converted to a voltage value by a photoreceptor. In all our implementations we used the logarithmic photoreceptor proposed by Delbrück and Mead [7].

2) In the second layer, the operations of differentiation and multiplication are performed using the photoreceptor values of the adjacent pixels to obtain the quantities $(I_x^i)^2$, $(I_y^i)^2$ and $I_x^i I_y^i$.

3) In the third layer the terms I_a, I_b, and I_c are generated by combining the quantities $(I_x^i)^2$, $(I_y^i)^2$, and $I_x^i I_y^i$ with the corresponding terms coming from the neighboring pixels.

4) In the fourth and final layer, the quantities $P(\lambda_t) = (a - \lambda_t)(c - \lambda_t) - b^2$ or $P(0) = ac - b^2$ are computed and the test for $P(\lambda_t) > p_n$, *and* $a > \lambda_t$ or $P(0) > p_n$ are performed to verify the presence of a feature.

In section 3.1 we will present the description of the structure of the visual sensor fabricated. In section 3.2 we will decribe the circuit choices for the sensor. In section 3.3 the results from the testing of the chip will be presented and section 3.4 summarizes the contributions.

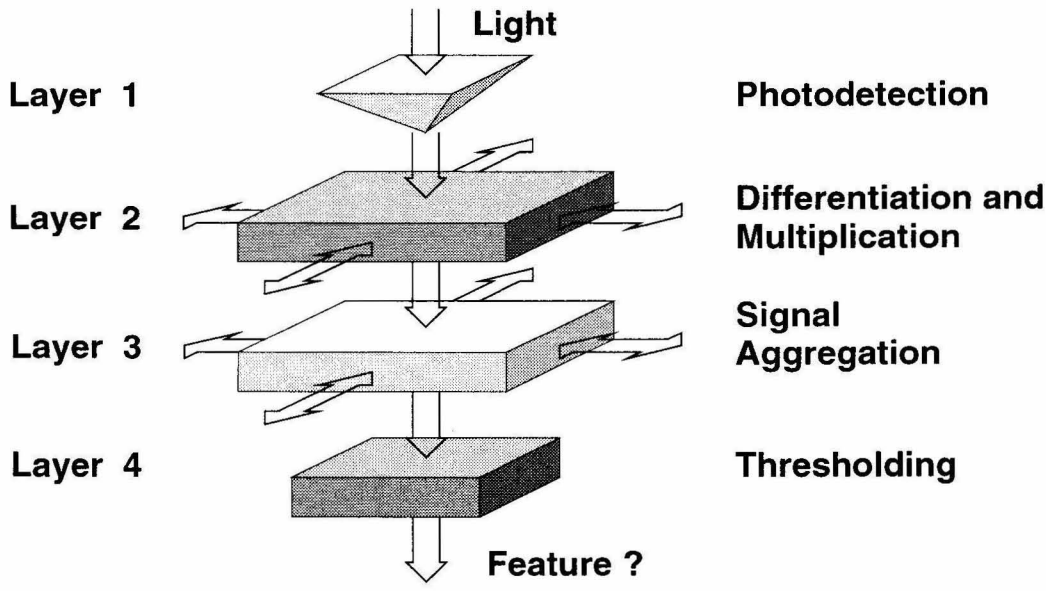

Figure 3.1: We can conceptually group the computation performed by the CMOS sensor in four separate layers of computations. First light is detected and converted to a voltage value by a continuous time photoreceptor. Then the operations of differentiation and multiplication are performed using the photoreceptor values of the adjacent pixels to obtain the quantities $(I_x^i)^2$, $(I_y^i)^2$, and $I_x^i I_y^i$. The third layer combines the terms $(I_x^i)^2$, $(I_y^i)^2$, and $I_x^i I_y^i$ with the corresponding terms coming from the neighboring pixels to form the quantities I_a, I_b, and I_c. In the fourth layer the quantities $P(\lambda_t) = (I_a - I_t)(I_c - I_t) - I_b^2$ $P(0) = I_a I_c - I_b^2$ or are computed and the test for $P(\lambda_t) > I_{p_n}$, *and* $I_a > I_t$ or $P(0) > I_{p_n}$ are performed.

3.1 The Design of *Detector1*

The structure of the chip *Detector1* is depicted in Fig. 3.2.

In this first implementation of the algorithm we chose a window of size 3×3 pixels ($N = 9$). The chip computes the algorithm in four 3×3 pixels windows. Since every pixel at the edge of the window computes the spatial-derivative both along the x axis and along the y axis from the adjacent pixels, the actual number of photoreceptors involved in the four computation is 21 and the patch has the shape depicted in Fig. 3.2. It is worth noticing that, even if the 3×3 pixels windows are non-overlapping, the actual 21-pixel patches used in the algorithm are overlapping and only the photoreceptor of the central pixel of the window contributes to only one patch. While the pixels in the four windows contain the additional circuitry to perform the first three layers of the computation (i.e., photodetection, differentiation

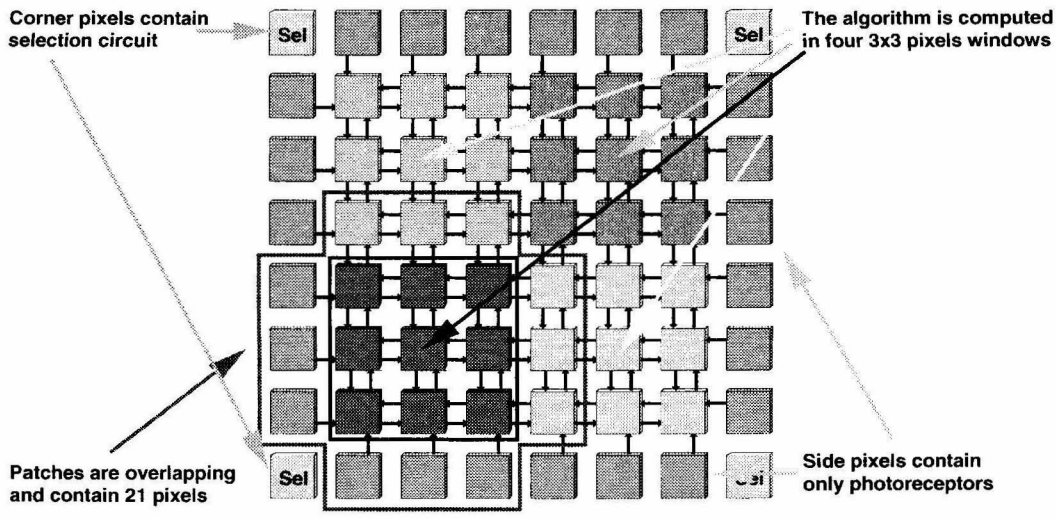

Figure 3.2: In this first implementation of the algorithm 3×3 pixels window was selected. The algorithm is computed in four 3×3 pixels windows. The actual number of photoreceptors involved in the four computation is 21 and the patch has the shape depicted. The 21-pixels patches used in the algorithm are overlapping and only the photoreceptor of the central pixel of the window contributes to only one patch. The pixels on the border of the array contain only the photoreceptors. The four special pixels at the four corners of the array contain the selection circuits.

and multiplication, and signal aggregation), the pixels on the border of the array contain only the photoreceptors. Four special pixels at the corners of the array finally contain the selection circuitry to perform the thresholding operation. Not shown in Fig. 3.2 are the scanners that allows the read out of the value of the outputs of all photoreceptors and the outputs of the four selection circuits.

Going into detail to explain more accurately how the algorithm is implemented in every window, we can refer to Fig. 3.3. Since every pixel has to perform the operations of differentiation and multiplication to obtain the quantities $(I_x^i)^2$, $(I_y^i)^2$ and $I_x^i I_y^i$, the photoreceptor output value is passed on to all four neighboring pixels, and, vice versa, the photoreceptor output values of the four adjacent pixels is received by every pixel; see Fig. 3.3a.

If the photoreceptor output value is represented by a voltage, the operation of differentiation can be approximated, in the first order, by the difference of the two values. This means, for example, that if $I_x^i = V_{left}^i - V_{right}^i$ is the first order approximation for I_x for the i-th pixel. V_{left}^i and V_{right}^i are, in this case, the photoreceptors

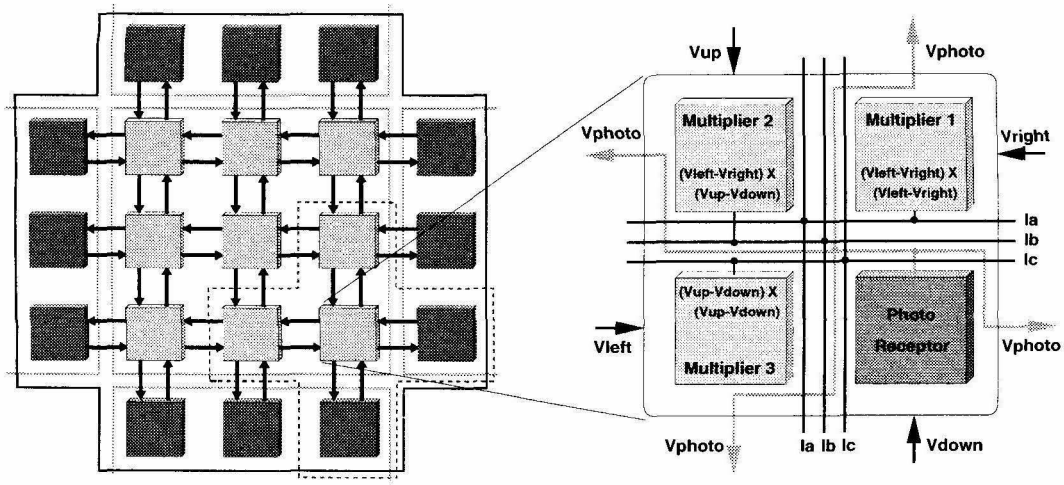

Figure 3.3: **a)** The algorithm is evaluated on a 3×3 pixels windows (solid gray lines). Every pixel computes the spatial derivatives I_x and I_y along the x and y directions with the photoreceptor voltage of the four adjacent pixels (dashed line). The number of photoreceptors involved in the computation is 21 (solid black line). **b)** The pixel computes the quantities $(I_x^i)^2$, $(I_y^i)^2$ and $I_x^i I_y^i$ with three four-quadrant multipliers and add the resulting currents to the global I_a, I_b and I_c wires that are shared by the nine pixels in the 3×3 windows. The photoreceptor feeds its output value to the four adjacent pixels.

output voltages of the two pixels at the left and right of the i-th pixel. In this way the term $(I_x^i)^2$ is equivalent to $(I_x^i)^2 = I_x^i I_x^i = \left(V_{left}^i - V_{right}^i\right)\left(V_{left}^i - V_{right}^i\right)$. We can now be easily convinced that the second layer operations of differentiation and multiplication can be easily and elegantly implemented by three four-quadrant multipliers that at every pixel i performs the operations of

$$\begin{aligned}
(I_x^i)^2 = I_x^i I_x^i &= \left(V_{left}^i - V_{right}^i\right)\left(V_{left}^i - V_{right}^i\right) \\
(I_y^i)^2 = I_y^i I_y^i &= \left(V_{up}^i - V_{down}^i\right)\left(V_{up}^i - V_{down}^i\right) \\
I_x^i I_y^i &= \left(V_{left}^i - V_{right}^i\right)\left(V_{up}^i - V_{down}^i\right) ,
\end{aligned}$$

where the quantities V_{left}^i, V_{right}^i, V_{up}^i and V_{down}^i are photoreceptors voltages of the four neighboring pixels; see Fig. 3.3b.

For this first design, the algorithm was implemented in a 3×3 pixels window and the simplest of the weighting or window function was chosen. With the weighting function $w(\mathbf{x}) = 1$ for every location, the third layer operation of signal aggregation

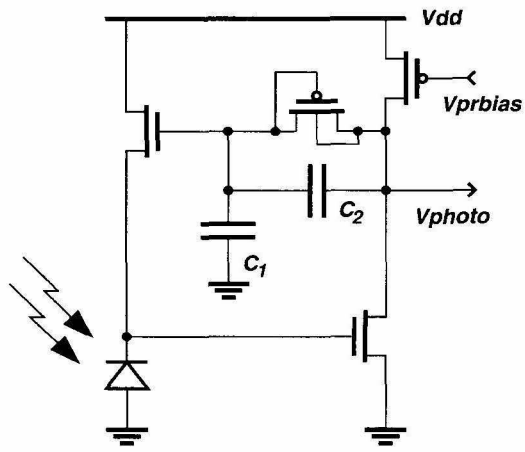

Figure 3.4: The logarithmic photoreceptor proposed by Delbrück and Mead used in the chip. The main characteristic of this photoreceptor is that, at least in first approximation, its output is proportional to the contrast of the image and invariant to the absolute value of the illumination.

is elegantly accomplished by the summation of the quantities $(I_x^i)^2$, $(I_y^i)^2$ and $I_x^i I_y^i$ of the $N = 9$ pixels of the window. As we will see in the next section, if the output of the three four-quadrant multipliers is a current, then the sum of those quantities can be easily accomplished by connecting each output node of the multipliers to the corresponding wires that carry the currents I_a, I_b and I_c representing the quantities a, b and c; see Fig. 3.3b.

3.2 CMOS Implementation

The photoreceptor used for this chip is the logarithmic photoreceptor proposed by Delbrück and Mead reported in Fig. 3.4. The main characteristic of this photoreceptor is that, at least in first approximation, its output is proportional to the contrast of the image and invariant to the absolute value for the illumination

$$\frac{dV_{photo}}{dt} = \frac{C_1 + C_2}{C_2}\left(\frac{U_t}{\kappa}\right)\frac{1}{I}\frac{dI}{dt} \tag{3.1}$$

where in this case the contrast is defined as the ratio of dI, the varying small signal component of the intensity and the intensity I. As we can see from equation (3.1)

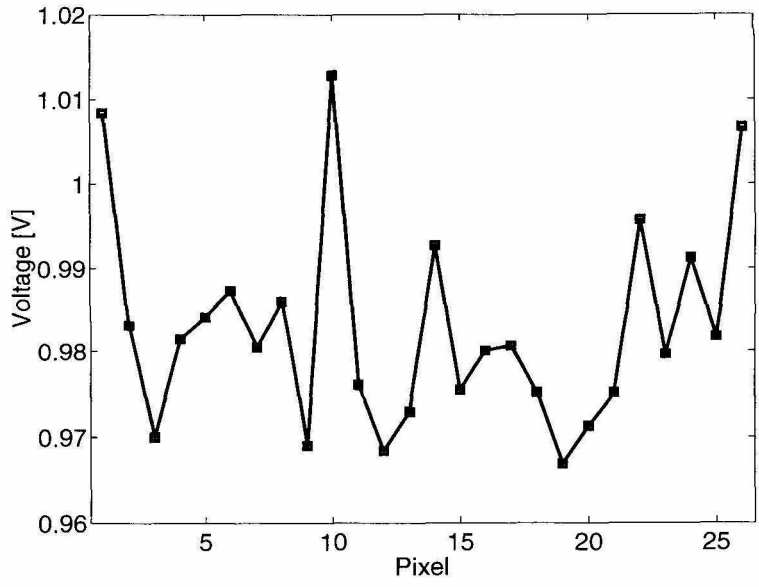

Figure 3.5: Plot the output voltages, under same intensity illumination, of an array of 26 photoreceptors like the one in Fig. 3.4. The offset mismatches in output voltages, even for adjacent photoreceptors, can be as high as 40mV. The standard deviation in this example was 13mV.

the gain can be set by choosing the appropriate value of the capacitive-divider ratio $(C_1 + C_2)/C_2$. The choice for the ratio depends on many factors. In our case, due to the fairly accurate computations expected from the sensor, we are interested in obtaining the highest signal-to-noise ratio possible. The noise component that affects the computation the most is the fixed-pattern noise due to output voltage offsets of the array of photoreceptors. This kind of noise is due to transistor mismatches in the fabrication of the chips. A wrong choice in the capacitive-divider ratio would cause the differences between supposedly identical photoreceptor outputs to be dominant over the typical signal variations produced by real scenes.

In Fig. 3.5 we plot the output voltage, under same intensity illumination, of an array of 26 such photoreceptors. The offset mismatches in output voltages even for adjacent photoreceptor can be as high as 40mV. We found that a choice of $C_1 = 10C_2$ (i.e., $(C_1+C_2)/C_2 = 11$) would give us a peak-to-peak output voltage of about 500mV in response of a 100% contrast stimuli. In Fig. 3.6 we plot the output of one of the photoreceptors in response of a 100% contrast drifting sinusoid. A higher choice for

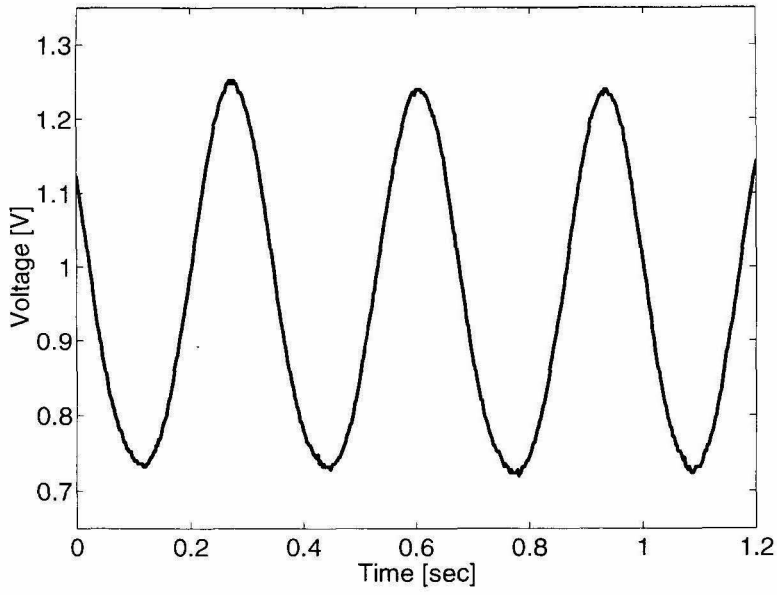

Figure 3.6: A choice of $C_1 = 10C_2$ (i.e., $(C_1 + C_2)/C_2 = 11$) for the capacitive-divider ratio gives a peak-to-peak output voltage of about 500mV in response of a 100% contrast difference. The plot is the output of one of the photoreceptors in response of a 100% contrast drifting sinusoid.

the capacitive-divider ratio often cause unwanted oscillations of the photoreceptor for some biasing choices and therefore was avoided.

With a peak-to-peak output voltage of about 500mV, we were facing the problem of finding a four quadrant multiplier with a matching linear range. If images with high contrast were focused onto the chip, a standard subthreshold Gilbert four-quadrant multiplier [32] with an input linear range of about 100mV would saturate, degrading the precision of the computation. There are many designs of four quadrant multipliers using the MOS transistor above threshold with a linear range up to a few volts. It is fairly easy to obtain wide linear range above threshold due to the square law voltage-current characteristic of the MOS device. In the subthreshold region the characteristic is exponential and therefore it is more difficult to obtain a comparable linear range.

We therefore were forced to design a four-quadrant multiplier, working in the subthreshold region, with a linear range matching the peak-to-peak voltage expected from the photoreceptors. The schematic of the new multiplier is shown in Fig. 3.7. The multiplier has a linear range of about ± 1V as we can see from Fig. 3.8, a factor

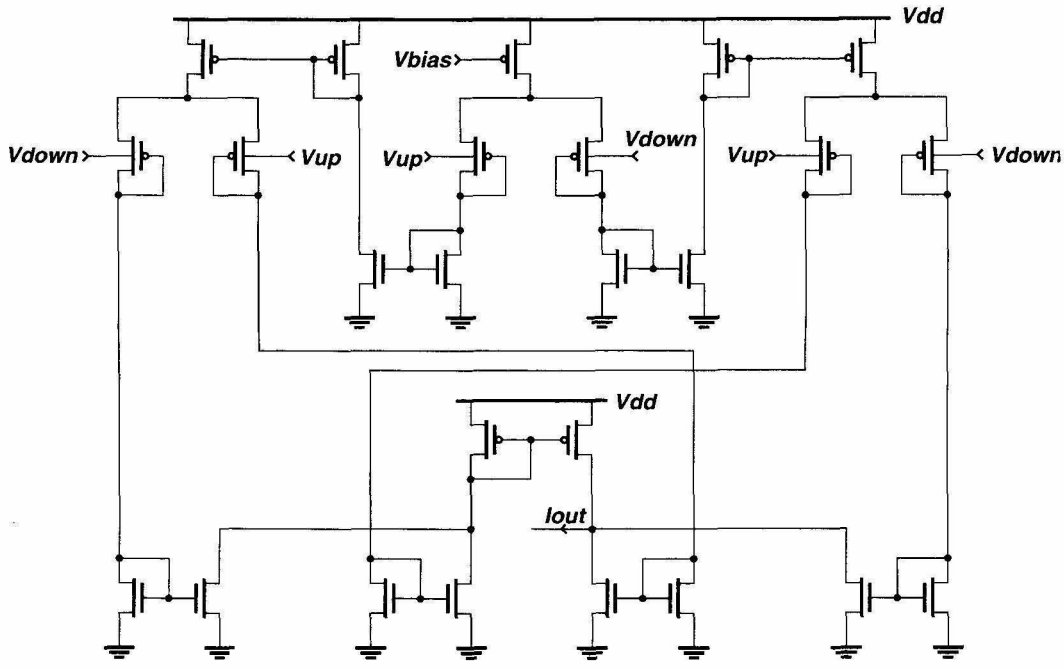

Figure 3.7: The four-quadrant multiplier used in the pixels has a linear range of about $\pm$1V with power consumption below 1μW. Here shown is the multiplier that computes the term $(I_y^i)^2$ and feeds the resulting current to the wire carrying the current I_c.

of 20 increase with respect to the normal 100mV, with less than double the number of transistors with respect to a standard implementation of the Gilbert multiplier. The multiplier achieves a wider input linear range by using the well terminals of the input transistors as low transconductance inputs and by using the feedback technique know as "gate degeneration" to extend the linear range even further. It is possible to show that the overall transfer characteristic of the multiplier is

$$I_{out} = I_b \frac{\sinh(\Delta_{12})}{1+\cosh(\Delta_{12})} \frac{\sinh(\Delta_{34})}{1+\cosh(\Delta_{34})} \tag{3.2}$$

where

$$\Delta_{12} = \frac{\kappa_{eff}(V_1 - V_2)}{V_T} \quad \text{and} \quad \Delta_{34} = \frac{\kappa_{eff}(V_3 - V_4)}{V_T} \quad \text{and} \quad \kappa_{eff} = \frac{1-\kappa}{1+\frac{\kappa}{\kappa_n}}$$

where I_b is the biasing current set by the gate voltage V_{bias}. V_1, V_2, V_3, and V_4 are the four input voltages and κ and κ_n measure the effectiveness of the gate potential in

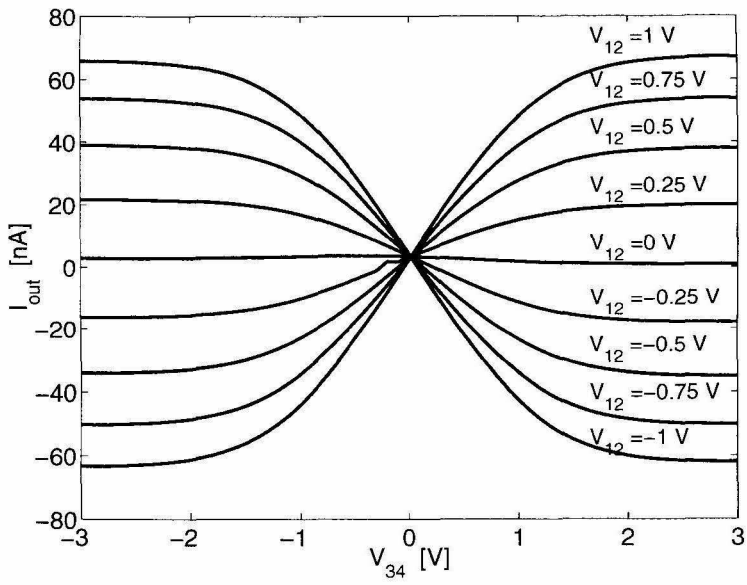

Figure 3.8: DC transfer characteristics of the multiplier used in the chip ($V_{34} = V_3 - V_4$ sweeping in the range -3.0 − 3.0 V and $V_{12} = V_1 - V_2$ at fixed values, common mode voltage for both differential input V_{CM}=2.5 V).

controlling the channel currents for the well input transistors and the nFET transistors respectively.

The power consumption of the multiplier under normal bias condition is well below 1μW, allowing a total power consumption for the pixel of about 1μW. One last note about this multiplier is that if the input voltage of one input terminal or the other falls below about 1V, the well-to-source junction of the input transistors becomes forward biased, and the parasitic bipolar transistor, which is part of every well transistor, shunts the current of the amplifier to ground, severely degrading the accuracy of the four quadrant multiplication. To reduce the risk of that happening, a pair of pFET source followers was included at the output of every photoreceptors to raise its value by a few hundred millivolts.

This multiplier is just one representative of a bigger family of wide linear range four-quadrant multipliers that we designed. For a complete description and characterization of the family of multipliers that we designed and tested, the reader can see appendix A.

Every 3×3 pixel window computes the currents I_a, I_b and I_c that represent the

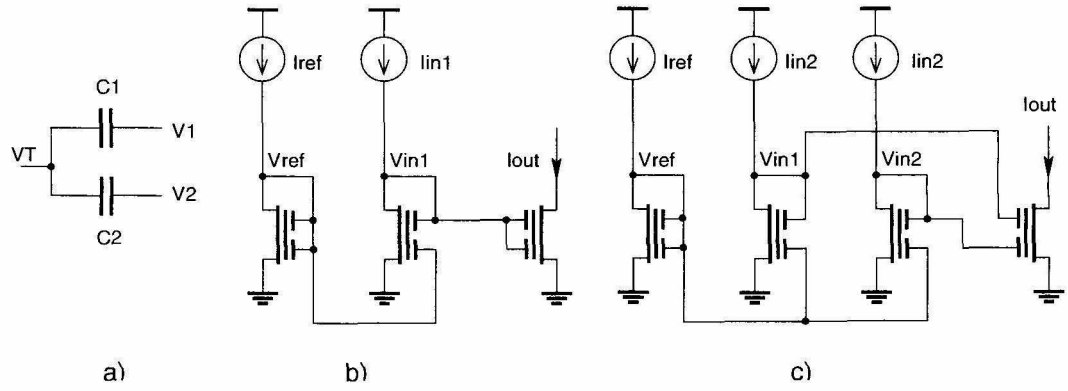

Figure 3.9: **a)** The capacitive voltage divider is the fundamental building block of any network of Multi Input Translinear Elements (MITEs). **b)** Network of that implements the computation $I_{out} = I_{in1}^2/I_{ref}$. **c)** Network of that implements the computation $I_{out} = I_{in1}I_{in2}/I_{ref}$.

quantities a, b, and c. At this stage of the computation all the quantities involved are represented by currents and, therefore the expressions that follow will reflect this fact.

The last step of the algorithm is to compute the quantities $P(\lambda_t) = (I_a - I_t)(I_c - I_t) - I_b^2$ or $P(0) = I_a I_c - I_b^2$ and test for $P(\lambda_t) > I_{p_t}$, *and* $I_a > I_t$ or $P(0) > I_{p_t}$.

While subtraction and addition of current are easily implemented with a minimum cost, a little more effort is required to implement multiplications and square laws. There are several circuit solutions that perform multiplications of currents. For this sensor we choose to use simple networks of Multi Input Translinear Elements (MITEs) introduced in [34].

To understand how these MITEs works we can refer to Fig. 3.9 where the two basic MITEs configurations used in the selection circuit are shown.

For the capacitive voltage divider in Fig. 3.9a, if all the voltages are initially set to zero and then V_1 and V_2 are applied, the voltage at the node V_T becomes

$$V_T = \frac{C_1 V_1 + C_2 V_2}{C_1 + C_2} .$$

Now, remembering that in a saturated (n-type) MOS transistor working in subthresh-

old the current is given by

$$I_{ds} = I_0 \, e^{-\frac{\kappa V_{gs}}{V_T}} \tag{3.3}$$

and assuming that all transistors and capacitors are identical, we can easily derive the expression of the current output for the remaining two circuits in Fig. 3.9.

For the squaring circuit of Fig. 3.9b, we have that

$$\frac{V_T}{\kappa} \ln\left(\frac{I_{out}}{I_0}\right) = V_{in1} \quad \text{and} \quad \frac{V_T}{\kappa} \ln\left(\frac{I_{ref}}{I_0}\right) = V_{ref} \, ,$$

and, therefore,

$$\begin{aligned} \frac{V_T}{\kappa} \ln\left(\frac{I_{in1}}{I_0}\right) &= \frac{V_{in1}}{2} + \frac{V_{ref}}{2} \\ \frac{V_T}{\kappa} \ln\left(\frac{I_{in1}}{I_0}\right) &= \frac{V_T}{\kappa} \ln\left(\frac{I_{out}}{I_0}\right)^{\frac{1}{2}} + \frac{V_T}{\kappa} \ln\left(\frac{I_{ref}}{I_0}\right)^{\frac{1}{2}} \\ I_{out} &= I_0 \left(\frac{I_{in1}/I_0}{(I_{ref}/I_0)^{\frac{1}{2}}}\right)^2 = \frac{I_{in1}^2}{I_{ref}} \, . \end{aligned}$$

In a very similar way, we can show that the circuit of Fig. 3.9c computes that function

$$I_{out} = \frac{I_{in1} I_{in2}}{I_{ref}} \, .$$

At this point it is straightforward to understand the behavior of the selection circuit of Fig. 3.10 performing the thresholding operation on the quantities I_a, I_b and I_c. The two nFETs M_7 and M_8 subtract the current I_t, representative of λ_t, to the two input currents I_a and I_c. The four MITEs $M1$ and $M4$-$M6$ output the current $(I_a - I_t)(I_c - I_t)/I_{ref}$ (i.e., $(a - \lambda_t)(c - \lambda_t)$) that is then mirrored to the node A. It is interesting to notice that this circuit, at the same time, test for the two conditions $I_a > I_t$ and $I_c > I_t$ (corresponding to the conditions $a > \lambda_t$ and $c > \lambda_t$) because if one of the two input currents I_a and I_c is less than the threshold current I_t, the current output of the network of four MITEs is zero. On the right side of the figure, first the absolute value of the current I_b is computed, then, using MITEs $M1$-$M3$, the current

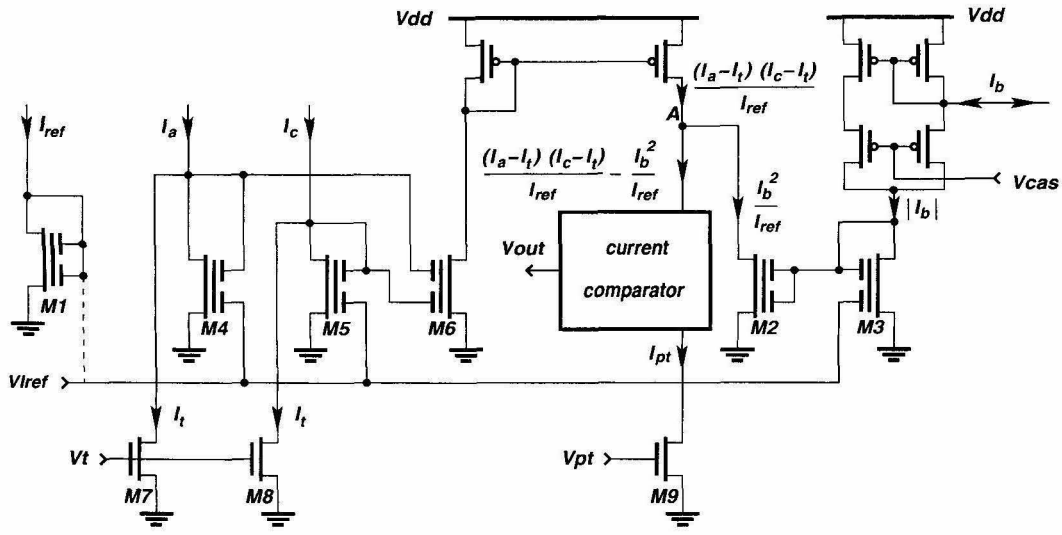

Figure 3.10: Network of MITEs that implements the final layer of the computation. If $(I_a - I_t)(I_c - I_t)/I_{ref} - I_b^2/I_{ref} > I_n$ the output of the current comparator goes low to indicate the presence of a feature. It stays high otherwise. The voltage VI_{ref} could be either supplied as an external bias or generated injecting a current in a diode-connected MITE as illustrated. The former solution was used in this case as a mean to reduce to a minimum every potential source of mismatches.

is squared to obtain I_b^2/I_{ref}. Finally the current comparator connected to node A and the drain of the transistor $M9$ performs the final comparison: its output is low if $(I_a - I_t)(I_c - I_t)/I_{ref} - I_b^2/I_{ref} > I_{pt}$ (condition $P(\lambda_t) = (a - \lambda_t)(c - \lambda_t) - b^2 > p_t$) and high otherwise. In this first implementation, since we were not concerned with speed or power issues, a simple CMOS inverter was used as current comparator. In case of more extended arrays, where power consumption becomes an issue, more sophisticated choices of current comparator are available. The voltage VI_{ref} could be either supplied as an external bias to all the selection circuits or generated at every location by injecting a current in a diode-connected MITE as illustrated Fig. 3.10. The former solution was used in this case as a mean to reduce to a minimum every potential source of mismatches.

From Fig. 3.10 we can also see how the circuit easily allows the approximated version of the algorithm (i.e., test only for $P(0) = (I_a I_c - I_b^2)/I_{ref} > I_{p_n}$) to be implemented by simply turning off (i.e., current $I_t = 0$) the two transistors $M7$ and $M8$. On the other hand, this also means that, if we were to implement only

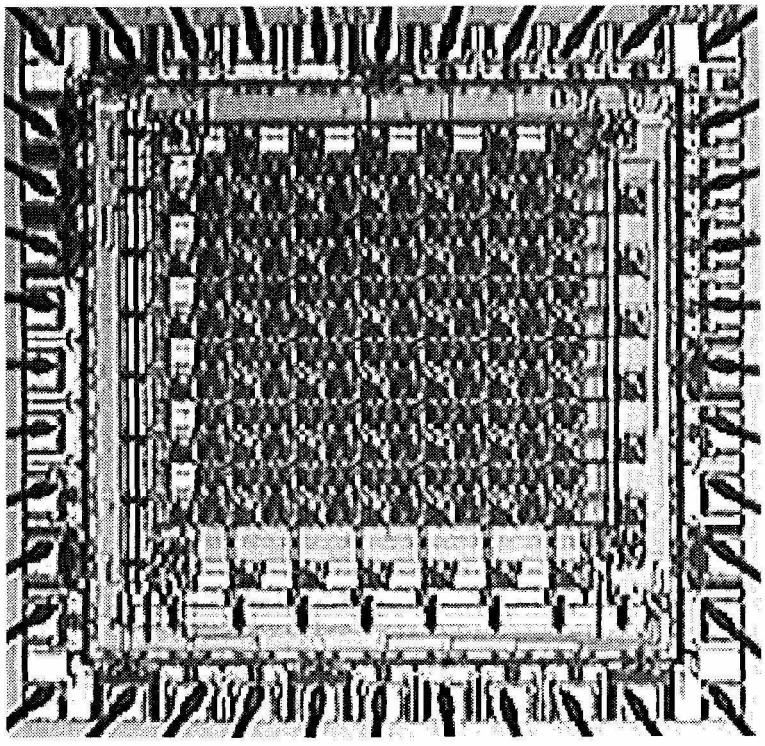

Figure 3.11: Micro-graph of *Detector1*. The layout mirrors the structure of Fig. 3.2. The chip was fabricated in a Tiny-Chip die (i.e., 2.1 × 2.1mm) in a 1.2μm double-poly double-metal process available through the MOSIS fabrication service. The actual area occupied by the design is 1.6 × 1.5mm.

this version of the algorithm, the gain, in terms of reduced circuit size, is only two transistors.

For this reason, in our design, we decided to allow the freedom to implement both versions of the algorithm at the cost of just two transistors. In the case of an implementation of the simpler version of the algorithm in a digital ASIC or with a FPGA platform, one can expect the advantages in terms of reduced die size (or gates used) or reduced circuit complexity to be far greater than the one resulting from an analog VLSI one.

3.3 Experimental Results

We designed, fabricated and tested a chip with a 8 × 8 array of pixels that allows the detection of features in four windows of 3 × 3 pixels. The chip was fabricated

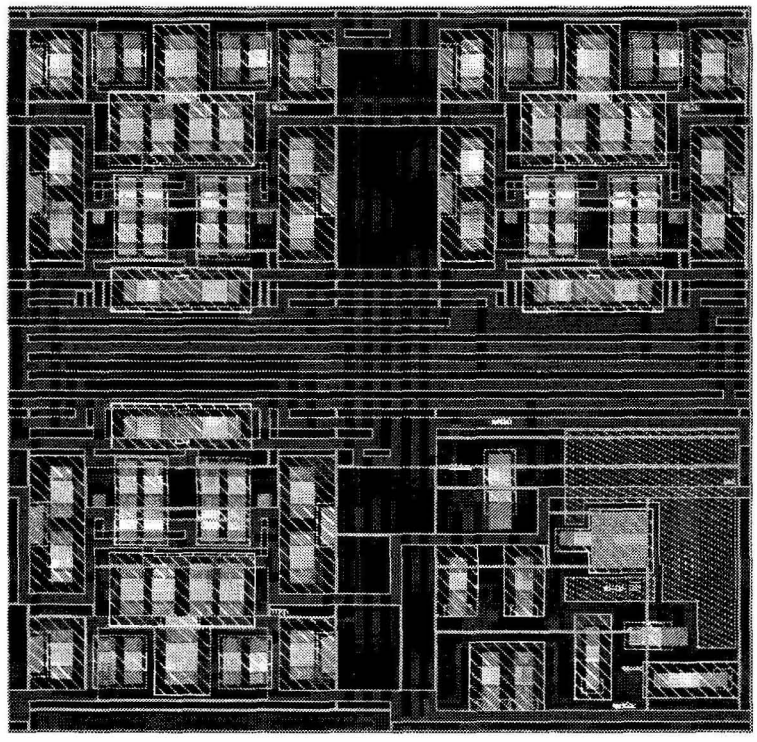

Figure 3.12: The layout of the pixel. The layout mirrors the pixel shown in Fig. 3.3b with the multipliers in the top-right, top-left and bottom-left corners, the photoreceptor in the bottom-right corner and the wires for I_a, I_b and I_c in between. In the double-metal technology used, most of the area is lost to wiring that carries the currents I_a, I_b and I_c to all the pixels in the 3×3 pixel windows.

in a Tiny-Chip die (i.e., 2.2×2.2mm) in a $1.2\mu m$ double-poly double-metal process available through the MOSIS fabrication service.

A micro-graph of the chip is shown in Fig. 3.11. The actual area occupied by the design is 1.6×1.5mm. In Fig. 3.12 we show a picture of the layout of the pixel. In Fig. 3.13 we show a micro-graph of the pixel. The layout mirrors the pixel shown in Fig. 3.3 with the multipliers in the top-right, top-left and bottom-left corners, the photoreceptor in the bottom-right corner and the wires for I_a, I_b and I_c in between. The pixel size is $189 \times 189\mu m^2$ in a $1.2\mu m$ CMOS technology. In the double-metal technology used, most of the area is lost to wiring that carries the currents I_a, I_b and I_c to all the pixels in the 3×3 pixels windows. The total number of transistors in the pixel is 80 and the fill factor of the sensor is about 1%. This figure is considerably lower that the 20% to 40% fill factor figures that are commonly found in today's

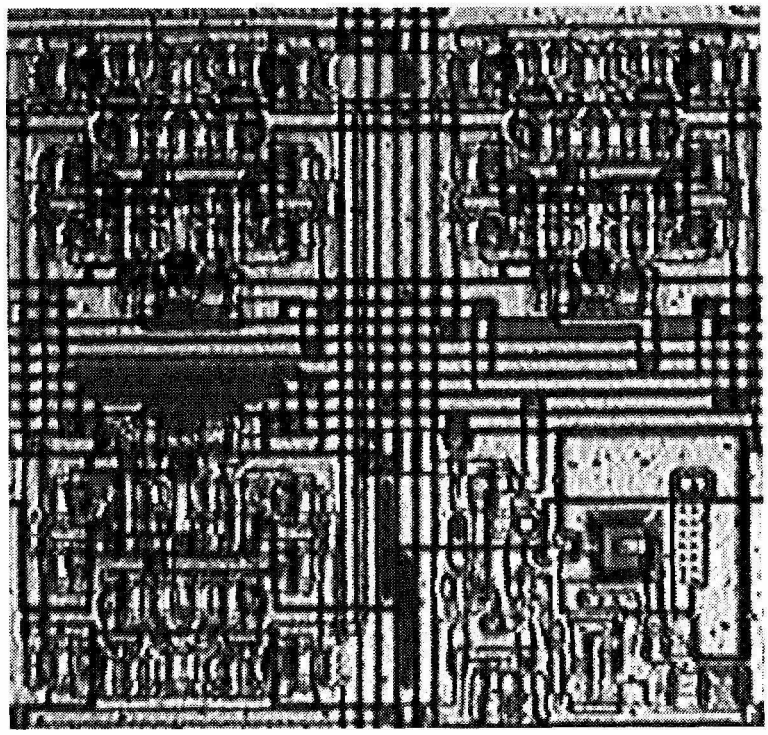

Figure 3.13: Micro-graph of the pixel of the CMOS sensor. The layout mirrors the pixel shown in Fig. 3.3b with the multipliers in the top-right, top-left and bottom-left corners, the photoreceptor in the bottom-right corner and the wires for I_a, I_b and I_c in between. The pixel size is $189 \times 189 \mu m^2$ in a $1.2 \mu m$ CMOS technology.

APS CMOS imager, but we have to remember that the main purpose of these sensors is to perform a computation, in this case detect features in the image, and not to obtain the best possible image. Furthermore, since this sensor was designed to work primarily in a man-made environment where we expect the features to be sparse and well defined, a high resolution is not required.

We tested the chip with different stimuli, and it performed reliably in the tests we conducted. In Fig. 3.14a, the image of a pen was projected onto the chip; as expected, none of the four patches reported a feature. In Fig. 3.14b, the tip of the pen was just on the top-right patch and the feature was correctly detected by the corresponding thresholding circuit.

For a more rigourous characterization of the sensor, we used the setup presented in Fig. 3.15. We mounted a lens directly over the chip to focus an image onto the pixel array. Random patterns were generated by a computer and presented on a TFT

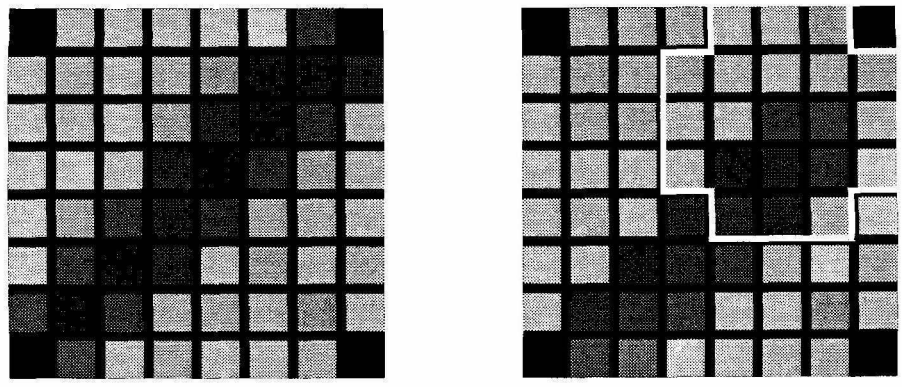

Figure 3.14: **a)** The image of a pen is projected onto the chip. The tip is not in the field of view and the chip correctly reports that there are no features present. **b)** When the tip of the pen is in the field of view of the top-right patch, the chip correctly signals the presence of the feature.

display. The same computer recorded the output of the selection circuits. We used a TFT display because the photoreceptors are able to detect the flicker of normal monitors, preventing a reliable characterization of the sensors. A 64-value gray scale was used to generate the stimuli on the display.

It is not trivial to generate random patterns with a chosen value of minimum eigenvalue λ_1. In particular, as we noted in the previous chapter, the eigenvalues of the matrix $\mathbf{G}$ are bounded and their maximum value is a function of the image contrast. In order to test the chip with patterns with the largest possible range of minimum eigenvalue λ_1, we randomly selected patterns from a pool of three different type of images. Examples of these three types of images used are reported in Fig. 3.16. For low λ_1 images the patterns were selected from the leftmost image in Fig. 3.16. That image uses all 64 values of intensity and therefore the maximum eigenvalues achievable is lower than what can be found in the two other random images that use less intesity values and have higher contrasts. Various examples of different patches, with different minimum eigenvalue λ_1, are presented on Fig. 3.17.

The testing methodology used is very simple. We presented to the sensor different patterns like the ones in Fig. 3.17, and for every presentation we recorded whether or not the selection circuit signaled the presence of a feature.

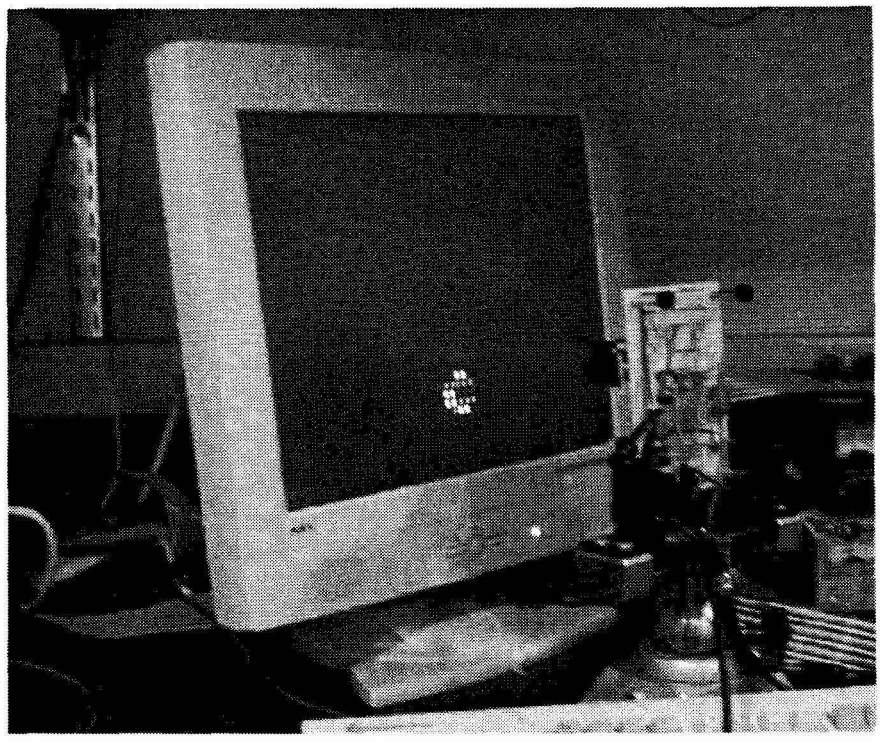

Figure 3.15: Chip testing methodology. A lens was mounted directly over the chip to focus an image on the pixel array. Patterns were generated randomly by the computer and presented on a TFT disply. The same computer recorded the output of the selection circuits.

In a first series of experiments, we tested the implementation of the simplified algorithm of conditions (2.13-2.14). The current I_p was set to zero and the value of the current I_t was varied to chracterize the implementation of the condition $(I_a - I_t)(I_c - I_t)/I_{ref} - I_b^2/I_{ref} > 0$ (i.e., $P(\lambda_t) = (a - \lambda_t)(c - \lambda_t) - b^2 > 0$). Fig. 3.18 shows the results of the experiments. Every curve in Fig. 3.18 is obtained by presenting to the sensor over 3300 different patterns and then clustering the recorded responses in 15 different bins according to their minimum eigenvalue λ_1. For every bin the percentage of positive answers was calculated. The abscissa of the point representing every bin is the mean of the λ_1 values of the 220 patterns in the bin. Every point, therefore, can be thought of as the likelihood of a positive response, by the selection circuit, for patterns with a certain close range of minimum eigenvalue λ_1. By varying the current I_t, we vary the threshold of the selection circuit. The higher the current I_t the higher the minimum eigenvalue λ_1 of the pattern has to be to trigger a positive

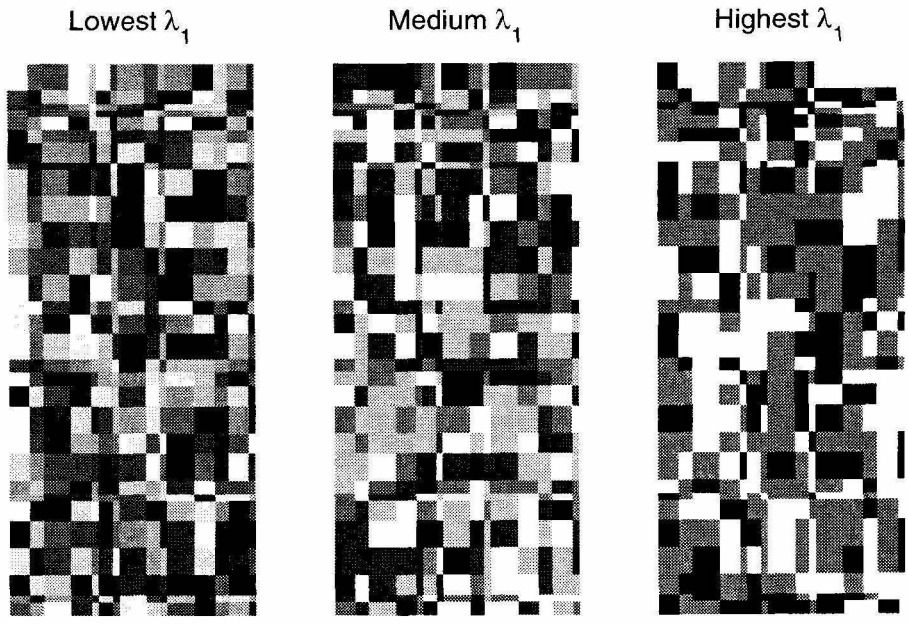

Figure 3.16: An example of three different images from which the random patterns were selected. Lower contrasts images (i.e., maximum number of gray values) provided lower λ_1 patterns while high contrast images provided patterns with the highest λ_1s.

response. The slope of the various lines in the transition region between 0% and 100% is an indication of the accuracy of the calculations performed by the sensor. Perfect calculations would be represented by an almost vertical line with just one point between 0% and 100%. In this case the curves have a very high slope, but there is more than one point in the transition region between 0% and 100%. We can also see how, for higher λ_1, the slope decreases slightly. There are several reasons why we should not be surprised by a behaviur like that.

First of all, even if we tried our best, we cannot be sure that the patterns were perfectly focused onto the chip. The photoreceptors' array is not positioned in the perfect center of the silicon die, and the die itself is often not placed in the center of the package.

Secondarily, the relationship between the gray scale light intensity of the patterns prsented on the display and the output response of the photoreceptors is not linear. In Fig. 3.19 we plot the measured transfer characteristic between gray scale values and peak to peak output voltage of one of the photoreceptors. We could compensate

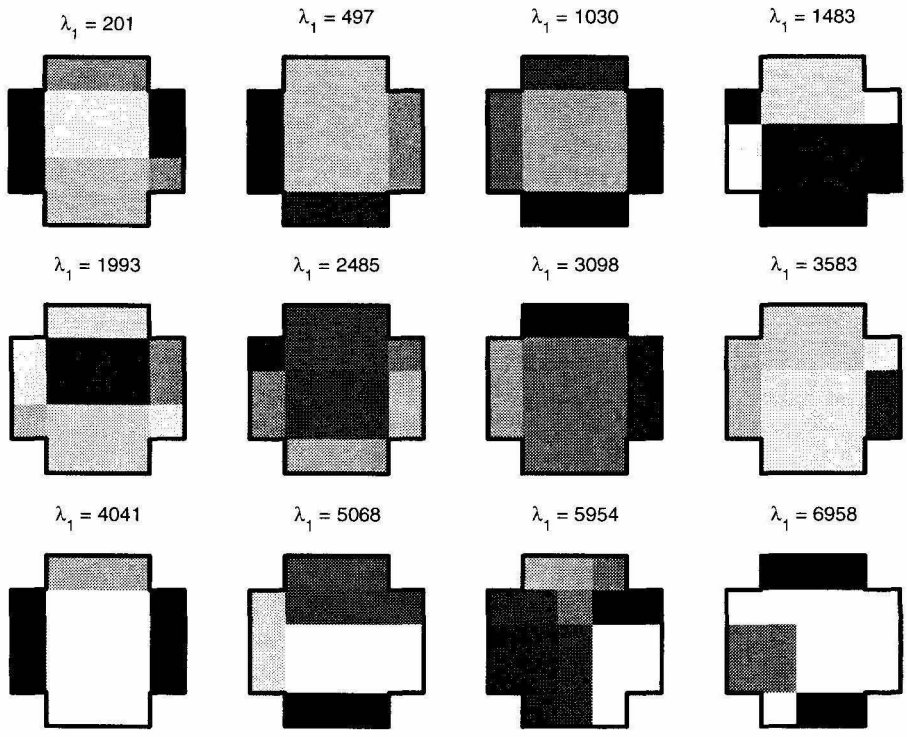

Figure 3.17: Examples of different patterns used for the characterization of the sensors. We obtained patterns with λ_1 values varying from 0 to about 8000.

this by inverting the function and adjusting the gray values of the various patterns, but we chose not too. If we started going in that direction, we could have ended up trying to compensate every little distortion in order to obtain better and better results.

The third and, most probably, main reason why the transition portion of the curves is not vertical is transistor mismatches. We discussed earlier that starting from the photoreceptors array, every circuit in the sensor presents some offest. The overall effect of all mismatches in the circuits manifest itself by decreasing the slopes of the curves in the transition region. For example, for the photoreceptors array of the sensor for which we report the results, we measured an average peak-to-peak output of 426mV with a standard deviation of 32mV, for a 100% contrast stimuli. The DC operating point of every photoreceptor was also measured. The mean was 1.232V with a standard deviation of 24mV. Just to have a feeling of how these transistor mismatches can affect the calculations, we performed a simple numerical simulation

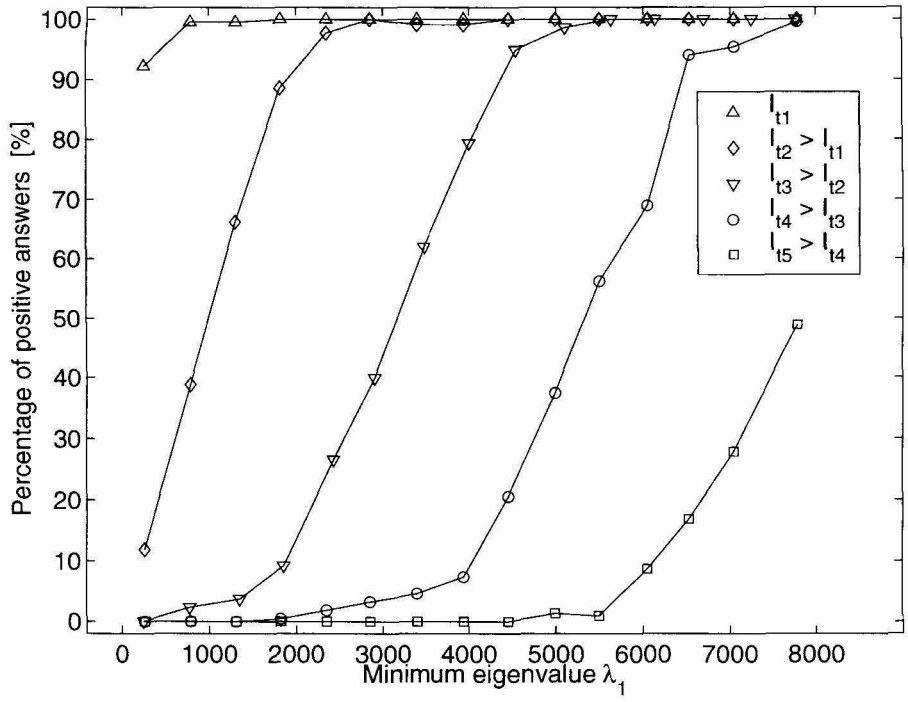

Figure 3.18: By varying the current I_t, we vary the threshold of the selection circuit. The higher the current I_t, the higher the minimum eigenvalue λ_1 of the pattern has to be to trigger a positive response. The slope of the various curves in the transition region between 0% and 100% is an indication of the accuracy of the calculations performed by the sensor.

of the effect of photoreceptors' mistatches.

The 24mV standard deviation DC offset is about 5% of the average peak-to-peak value of 426mV. This corresponds to an offset of 3.2 gray values for a 64 colors range. If we run a numerical simulation of these DC offsets, without even considering the offsets for peak-to-peak values, using the same 3300 patterns used before, we obtain the results of Fig. 3.20. Just with a systematic random offset with standard deviation of 3.2 gray values, the transition regions of the curves are no longer close to vertical but have a lower slope. It is interesting to see that, even for these simulation, the slopes decrease with higher values of lambda. We can now understand how transistor mismatches in the photoreceptors and all the others present in the various circuits can directly affect the precision of the analog circuits used in these kind of sensors.

In the second series of experiments, the implementation of the approximated algorithm of the condition (2.15) was tested. After setting the current I_t to zero, we

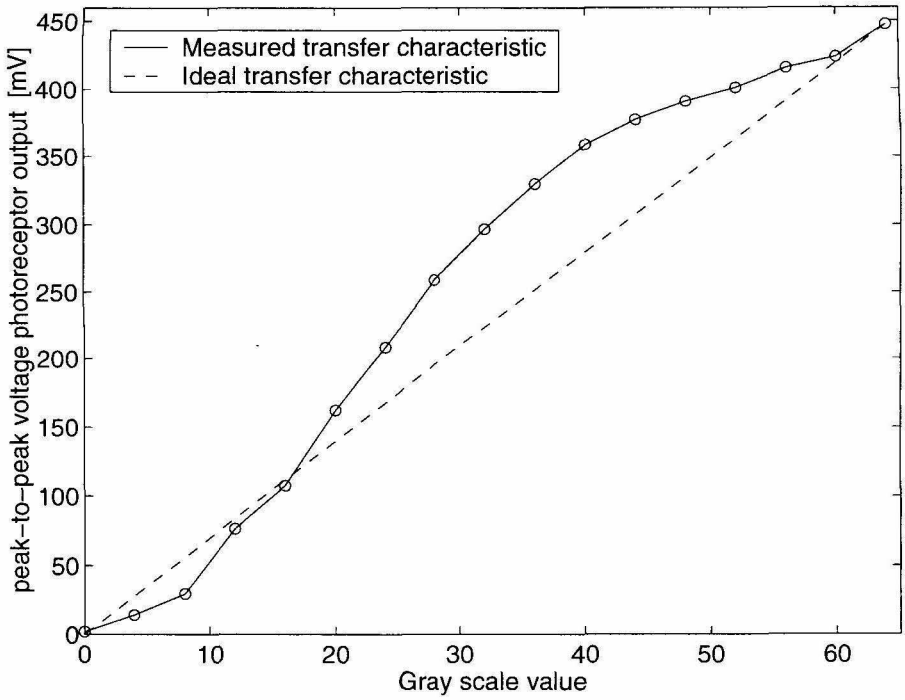

Figure 3.19: Measured transfer characteristic between gray scale values and peak-to-peak output voltage of one of the photoreceptors.

varied the value of the current I_{pt} to characterize the implementation of the condition $(I_a)(I_c)/I_{ref} - I_b^2/I_{ref} > I_{pt}$ (i.e., $P(0) = ac - b^2 > p_t$). In Fig. 3.21 we report the results of the experiments. Again, every curve is obtained by presenting to the sensor over 3300 different patterns and clustering the recorded responses in 15 different bins. Even in this case, by varying the current I_{pt}, the thresholding circuits become more selective and only patterns with higher $P(0)$ value are detected. As in the previous experiment, the slopes of the transition regions increase with the value of the threshold current.

Finally, we tried to go "inside" the curves of Fig. 3.18 and Fig. 3.21 to arrive at an idea of the cross section behaviour of the various patches clustered in one of the 15 points of the figures. We took ten patterns, see Fig. 3.22, with minimum eigenvalue λ_1 very close to 4000, and, for five different settings of the threshold current I_t, we presented each pattern for 200 times and recorded the percentage of positive responses for every pattern. In Fig. 3.23 we report the five different curves that the five different settings of I_t produced. The five points representing the clusters of over 220 patterns

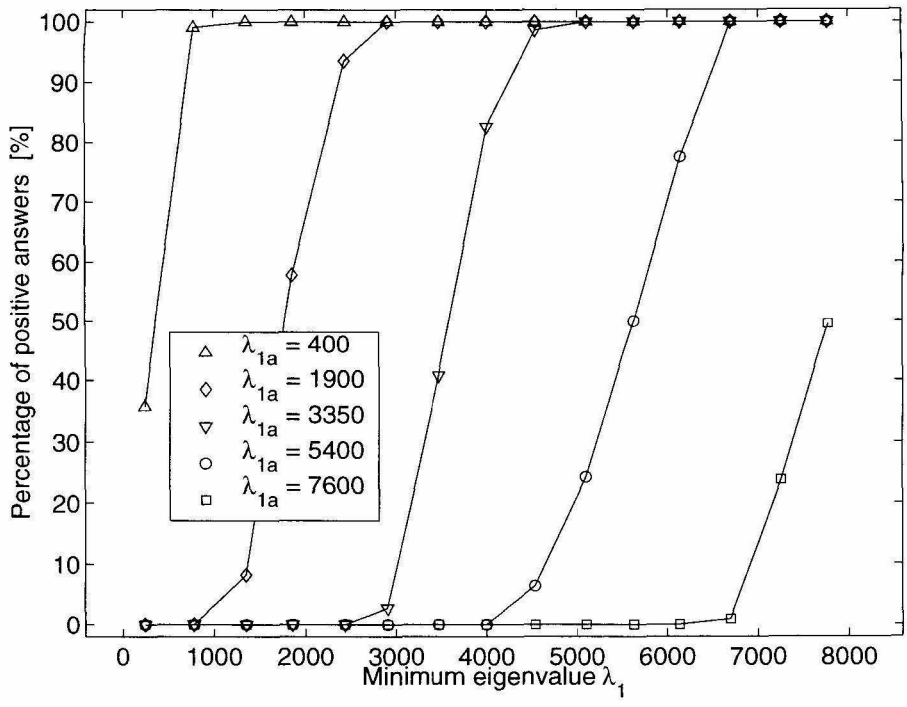

Figure 3.20: Result of the numerical simulation of the DC offsets not considering the offsets for peak-to-peak values. The same 3300 patterns utilized to obtain Fig. 3.18 were used.

centered around $\lambda_1 = 4000$ have aggregate positive response percentages of 96%, 87%, 54%, 20% and 3% respectively. The two vertical lines around $\lambda_1 = 4000$ represent the set of ten patterns tested. In Fig. 3.24 we report the results of these measurements. While the cross section averages for every one of the five settings, 98%, 87%, 52%, 9% and 1%, is very close to the aggregate values of Fig. 3.23, we observe a large variability between the ten different patterns. This is especially significant for the central curve, which, even if the average of 52% is very close, the expected value of 54% of Fig. 3.23, has a standard deviation of 28%.

As a final test we presented to the sensor a reduced-size version of the image of Fig. 2.1. The threshold was set very high in order to try to avoid false positive responses at the cost of missing some significant features. Most of the obvious features were selected along with some false positive resposes, like points 1 to 4. Other evident features were missed like the one numbered 5 to 8.

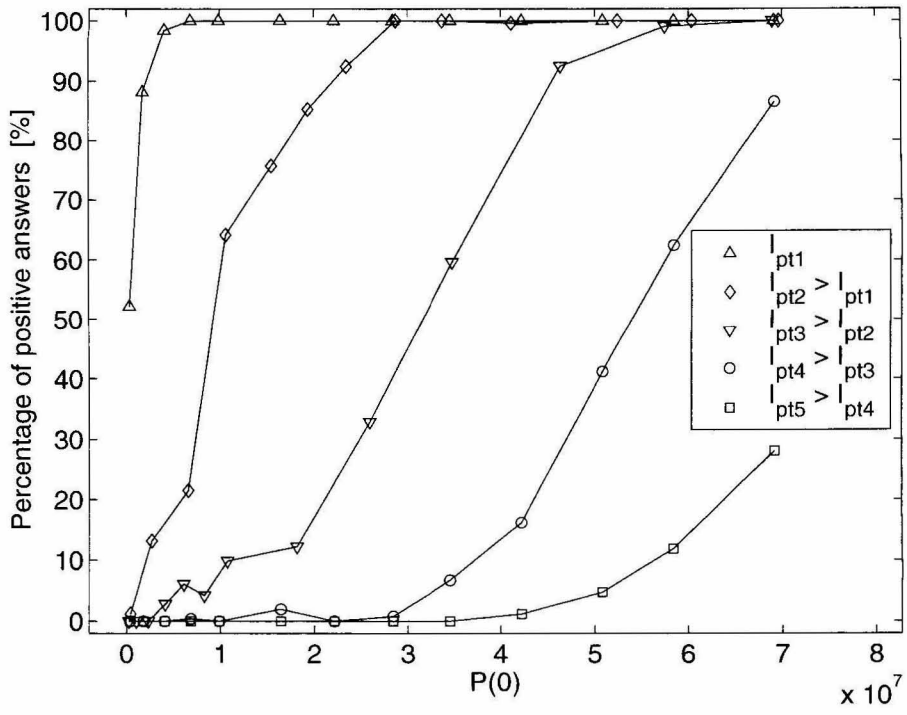

Figure 3.21: Result of the experiment testing the implementation of the approximated algorithm of the condition (2.15). Again, every curve is obtained by presenting to the sensor over 3300 different patterns and clustering the recorded respones in 15 different bins. Even in this case, by varying the current I_{pt} the thresholding circuits becomes more selective and only patterns with higher $P(0)$ value are detected.

3.4 Conclusion

In this chapter we presented the first implementation of a CMOS visual sensor with focal plane computation for feature detection. From the two algorithms derived in chapter 2, and, using the abstract representation of Fig. 3.1, we were able to design a CMOS visual sensor that implements either one of these complex algorithms. It is interesting to notice how the algorithms are elegantly translated in silicon with simple and well characterized circuits and yet every step of the algorithms is implemented without approximations. To the best of our knowledge, this work represents the first successful example of a CMOS sensor with focal-plane computation for continuous time feature detection.

Even if this first implementation worked well on "first silicon", there were still some issues that we were interested to see resolved beside a better accuracy of the

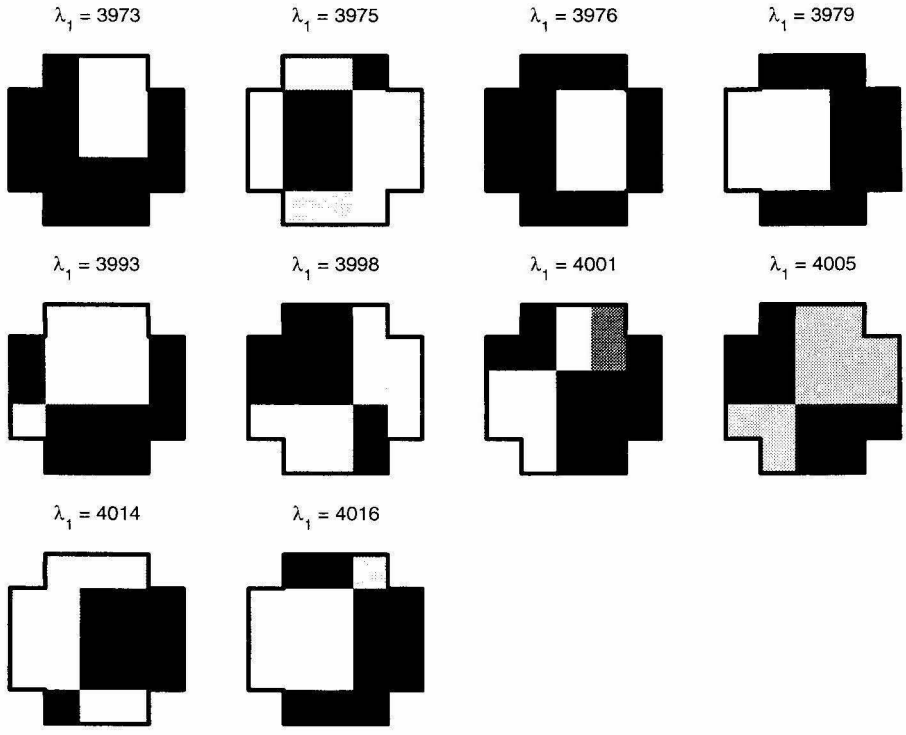

Figure 3.22: The ten different patterns with minimum eigenvalue λ_1 very close to 4000 that were used in the cross section experiment.

computations. The most important is that since the algorithm is computed on 3×3 pixels windows, the actual resolution of the feature detection algorithm is 1 to 9. That is, the sensor detects features centered on pixels that are located three pixels away both horizontally and vertically. In order to compute the algorithm at *every* pixel, it is necessary to include the selection circuit at every pixel and, most importantly, find a better way to implement the signal aggregation layer of the computation. Once the currents $(I_x^i)^2$, $(I_y^i)^2$ and $I_x^i I_y^i$ are combined with the corresponding currents coming from adjacent pixels to form I_a, I_b and I_c, there is no clear way to provide the same $(I_x^i)^2$, $(I_y^i)^2$ and $I_x^i I_y^i$ currents to all the other eight adjacent pixels. One solution would be to duplicate (nine copies) all the mirrors at the outputs of the three multipliers and then afford the cost of running eight additional wire lines for every one of the three wires carrying I_a, I_b and I_c. This solution, beside being not elegant, is extremely area consuming and therefore would prevent solving the secondary issue of improving the fill factor of the sensor. Another issue that a second design should address is whether

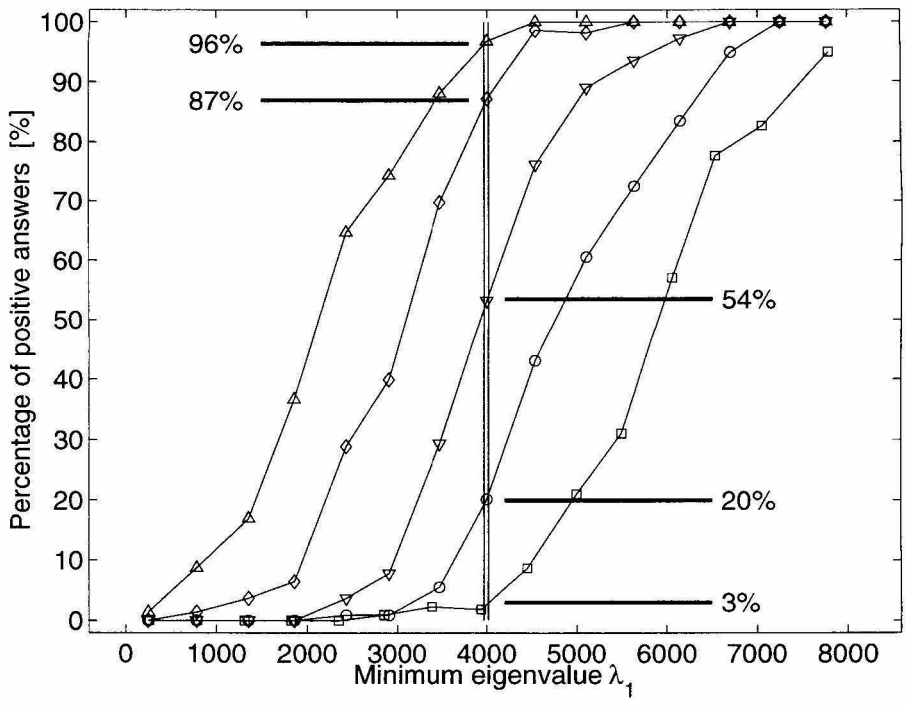

Figure 3.23: Five different curves obtained with five different values of I_t. The points representing the cluster of over 220 patterns centered around $\lambda_1 = 4000$ have percentages of 96%, 87%, 54%, 20% and 3%. The two vertical lines around $\lambda_1 = 4000$ represent the set of ten patterns tested.

it is possible to find an alternative design to implement the selection circuit that does not use floating-gate transistors. These devices, in fact, require the use of UV eraser to equalize the charges on the floating-gates before the circuit can be put in use, and are not area efficient due to the large number of capacitors required.

In the next chapter we will show how we were able to address these issues in the final design of the visual sensor.

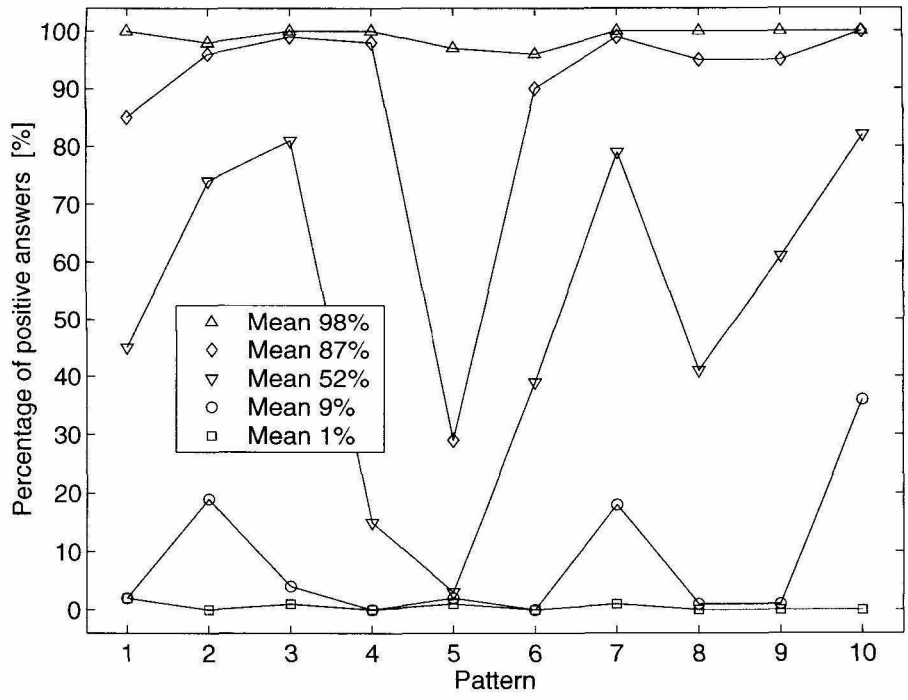

Figure 3.24: Results of the cross section measurements. While the cross section averages for every one of the five settings, 98%, 87%, 52%, 9% and 1%, are very close to the values of Fig. 3.23, we observe a large variability between the ten different pattern percentages.

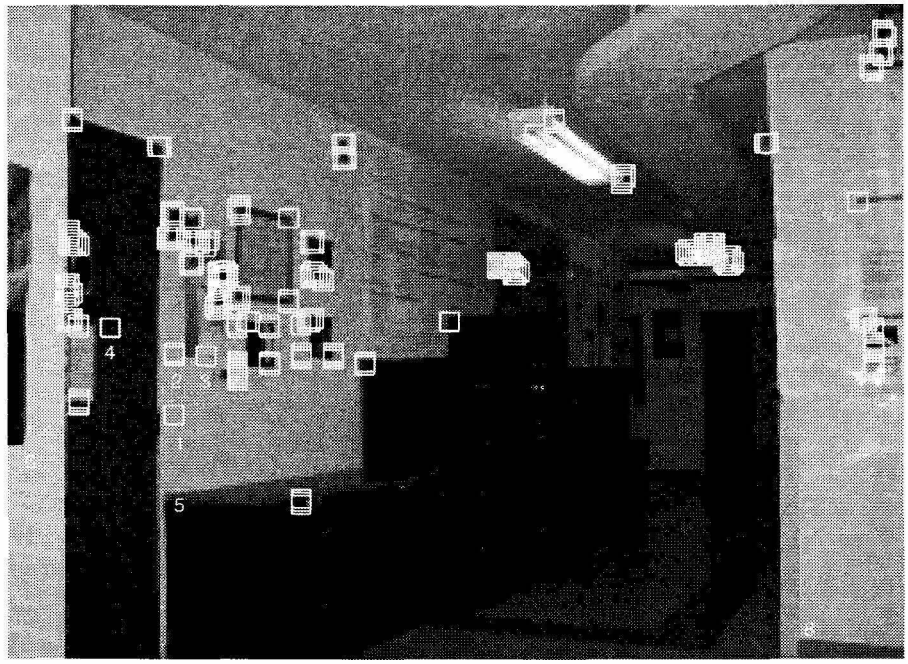

Figure 3.25: Features selected in the reduced-size version of the image of Fig. 2.1. Most of the obvious features are selected along with some false positive responses, like points 1 to 4. Other evident features were missed like the ones numbered 5 to 8.

Chapter 4 Final Design of the Sensor

As was noted at the end of the previous chapter, one of the main issues with the previous sensor design was the fact that it could only detect features that are located three pixels away both horizontally and vertically from each other. In order to increase the resolution of the computation, for example, to be able to perform the algorithm computations at every pixel, it is necessary to include the selection circuits at every pixel and, most importantly, find an efficient way to implement the signal aggregation layer.

Other concerns were the fact that the use of floating-gate transistors in the network of MITEs that implements the selection circuit require the use of UV eraser to equalize the charges on the floating gates before the circuit can be put in operation. Another issue with the use of floating-gate devices was the difficulty of performing accurate simulations of the sensor before fabrication. As we will describe in the following sections, most of those issues were addressed by the design of the final sensor that we will call *Detector2*.

In section 4.1 we will describe the new implementation of the signal aggregation layer that represents the single major improvement of this new design compared to the early one. In section 4.2 the design choices of *Detector2* will be presented. In section 4.3 we will describe the circuits used. Section 4.4 presents results from both the extensive simulations and the experimental testing of the sensor and, finally, section 4.5 summarizes the contribution of this work.

4.1 The New Signal Aggregation Layer

The problem of designing a more efficient signal aggregation layer lies in the fact that once the currents $(I_x^i)^2$, $(I_y^i)^2$ and $I_x^i I_y^i$, outputs of the three multipliers in the pixel, are combined with the corresponding currents coming from adjacent pixels to form

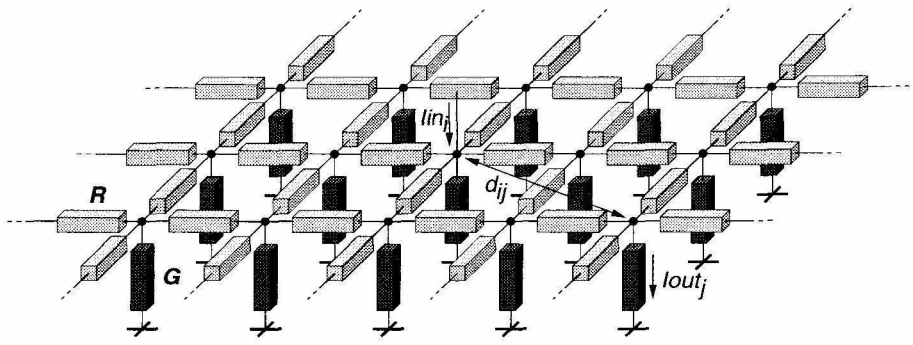

Figure 4.1: A two-dimensional diffusion network is an array of lateral resistors R and vertical grounded conductances G. The current I_{outj} flowing to ground at node j as a result of a current I_{ini} injected into node i at a distance d_{ij} (measured in pitch) exponentially decays with the distance.

I_a, I_b and I_c, the same currents cannot be used again to form the currents I_a, I_b and I_c for the eight other adjacent pixels (if a 3×3 pixels window configuration is used). The idea of generating nine actual copies of the current from the three multipliers, as we discussed in the previous chapter, is not practical in terms of area efficiency.

One alternative solution is to split the currents $(I_x^i)^2$, $(I_y^i)^2$ and $I_x^i I_y^i$ in various fractions to be provided to the neighboring pixels. In this way the expensive current mirror structures used to generate the copies of the currents are no longer necessary. Furthermore, it is possible to accomplish this without the need to have a dedicated wire to carry every fraction of the current to every corresponding pixel. All of this can be, in fact, realized using a resistive diffusion network.

A two-dimensional diffusion network is shown in Fig. 4.1. It is an array of lateral resistors R and vertical grounded conductances G. The current I_{outj} flowing to ground at node j as a result of a current I_{ini} injected into node i at a distance d_{ij} (measured in pitch) decays with the distance according to the approximate law [32]

$$\frac{I_{outj}}{I_{ini}} \sim \frac{\exp(-d_{ij}/L)}{\sqrt{d_{ij}/L}} \quad \text{for} \quad d_{ij} \geq L \,, \tag{4.1}$$

with $L = 1/\sqrt{RG}$.

Since the network is linear, the effects of currents injected into all nodes are superimposed and the network behaves essentially as a low-pass spatial filter that is

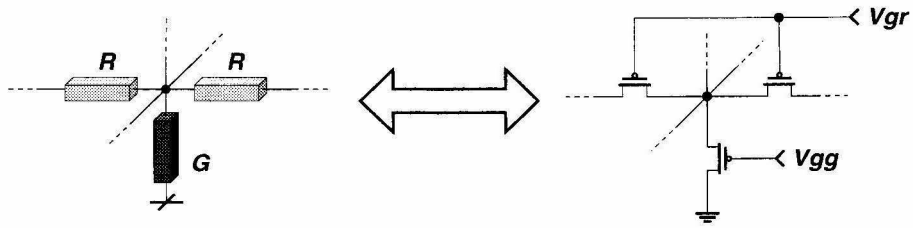

Figure 4.2: A diffusion network like the one in Fig. 4.1 can be implemented using standard MOS transistors used as pseudoconductance.

exactly what was required to implement the signal aggregation layer.

A diffusion network like the one in Fig. 4.1 can be implemented using standard MOS transistors used as pseudoconductance as it is explained in Fig. 4.2. It is possible to implement any network of linear resistors by means of only transistors and to control the value of each one of these pseudo-resistors by a voltage or a current.

The property that allows to use standard transistor as pseudo-resistors is only understandable if the MOS device is adequately modeled. In section 4.1.1 we will briefly recall this model and explain the general principle.

4.1.1 MOS Transistors as Pseudo Resistors

As shown by the schematic representation Fig. 4.3, the MOS transistor is fundamentally a symmetrical device. The source and drain ends are in principle not distinguishable and are here labeled as the two terminal A and B of the channel, with potential V_A and V_B with respect to the local substrate (general substrate of the chip, or local well). V is the local value of the channel "potential." It is not the electrostatic potential but a measure of disequilibrium in the distribution of electrons or holes in the channel that is produced by the application of voltages V_A and/or V_B. It has the values V_A and V_B at the two respective ends of the channel.

For a given value of gate voltage, the local sheet conductivity g_s of the channel is a decreasing function of the channel potential [47].

This function decreases approximately linearly with V for large values of g_s (strong inversion of the channel) and exponentially for low values of g_s (weak inversion). It can be shown [47] that the current flowing through the transistor is simply given by

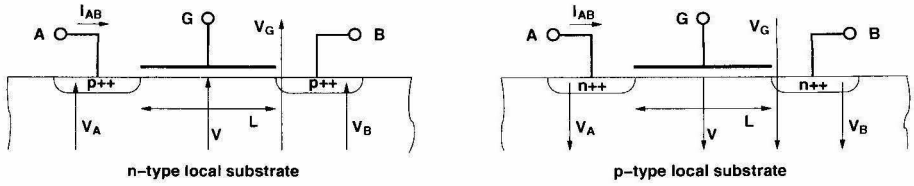

Figure 4.3: The MOS transistor is fundamentally a symmetrical device. The source and drain ends are in principle not distinguishable and are here labeled as the two terminals A and B of the channel, with potential V_A and V_B with respect to the local substrate (general substrate of the chip, or local well). V is the local value of the channel "potential."

$$I_{AB} = \frac{W}{L}\int_{V_B}^{V_A} g_s\,(V_G, V)\,dV = \frac{W}{L}\left[\int_{V_B}^{\infty} g_s\,(V_G, V)\,dV - \int_{V_A}^{\infty} g_s\,(V_G, V)\,dV\right] . \quad (4.2)$$

Due to the particular definition of V, this equation includes the two possible mechanisms of current transport: conduction (which dominates in strong inversion) and diffusion (which dominates in weak inversion). It is therefore valid for any value of V_A and V_B.

The decomposition in two terms is possible because g_s tends to 0 for large V. It provides a symmetrical expression with respect to V_A and V_B, which can be written

$$I_{AB} = I_S[f(V_G, V_B) - f(V_G, VA)] \quad (4.3)$$

where I_S is a specific current proportional to the width-to-length ratio W/L of the transistor. The transistor is said to be saturated when the smaller of the two terms of equation (4.3) becomes negligible. Channel shortening degrades the precision of equation (4.3) by making I_S itself slightly dependent on V_A and/or V_B. The relation is no more applicable if the channel length is reduced below the short channel limit.

By defining a *pseudo-voltage* V^* given by

$$V^* = \pm V_0\, f(V_G, V) \quad (+ \text{ for } p\text{–ch}, - \text{ for } n\text{–ch})$$

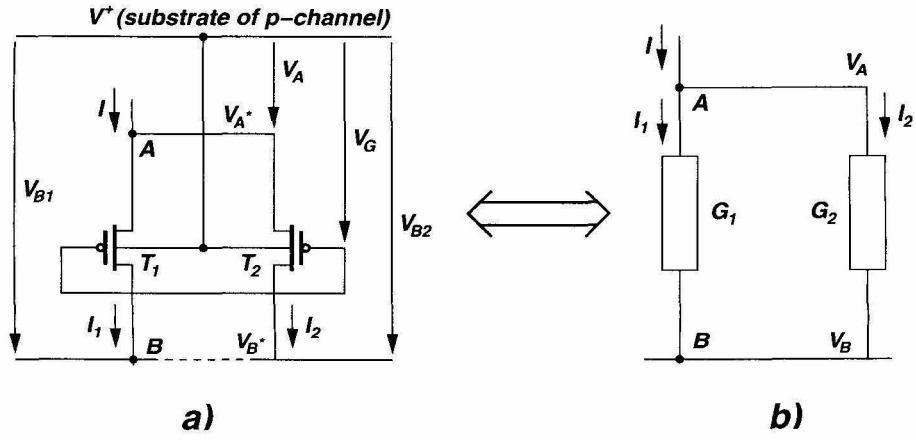

Figure 4.4: A current I imposed through two transistors T_1 and T_2 connected in parallel splits linearly into two components I_1 and I_2 respectively proportional to G_1^* and G_2^*. This is identical to the current splitting through two linear conductances G_1 and G_2.

where V_0 is an arbitrary scaling voltage, equation (4.3) can be rewritten as

$$I_{AB} = G^*(V_A^* - V_B^*) \tag{4.4}$$

which corresponds to a linear *pseudo-Ohm's* law, with constant *pseudo-conductance* $G^* = I_S/V_0$, proportional to W/L through I_S. The pseudo-voltage V^* is always positive for a p-channel transistor (negative for a n-channel). Moreover, it tends to 0 for V large. Thus the *pseudo-ground* 0^* (0-reference for the pseudo voltage V^*) is obtained by imposing $V = V_{pg}$ large enough to make $f(V_G, V_{pg})$ negligible.

As a consequence of equation (4.4), a current I imposed through two transistors T_1 and T_2 connected in parallel splits linearly into two components I_1 and I_2, respectively, proportional to G_1^* and G_2^*; see Fig. 4.4. This is identical to the current splitting through two linear conductances G_1 and G_2.

If $V_B = 0$ (ground potential) in the conductance network, V_B^* in the pseudo-conductance network should be at pseudo-ground 0^*. Voltages V_{B1} and V_{B2} need not be fixed or equal anymore; they must be sufficiently large to make $f(V_G, V_{Bi})$ negligible. This means the transistors must be saturated.

For a given value of V_G, a transistor is in weak inversion if both V_A and V_B are large enough to obtain $f(V_G, V) \ll 1$ at both ends of the channel. This corresponds

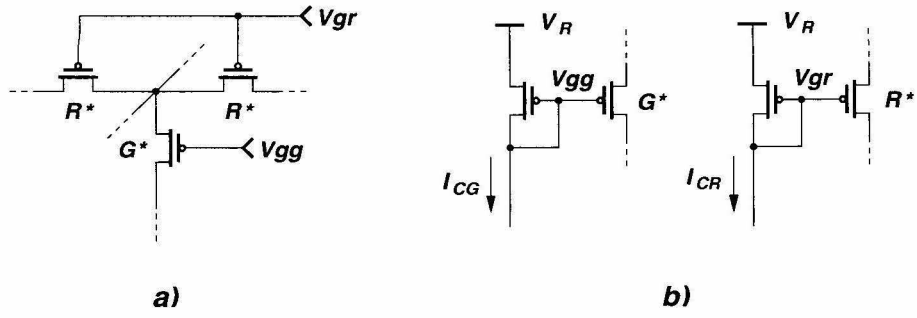

Figure 4.5: The transistors implementing R^* and G^* can be controlled by different gate voltages V_{gg} and V_{gr} and the diffusion length of the diffusion network can be electrically adjusted through the control currents I_{CR} and I_{CG}.

to ensuring a value of its saturation current much smaller than the specific current I_S.

In this case it can be shown that $f(V_G, V)$ reduces to

$$f(V_G, V) = \exp\left(\frac{\kappa(V_G - V_T)}{V_T}\right) \exp\left(-\frac{V}{V_T}\right) \ll 1$$

and is separable in exponential functions of V_G and V. Pseudo-voltage and pseudo conductance may then be redefined as

$$V^* = \pm V_0 \exp\left(-\frac{V}{V_T}\right) \quad (+ \text{ for } p\text{–ch}, - \text{ for } n\text{–ch})$$

and

$$G^* = \frac{I_S}{V_0} \exp\left(\frac{\kappa(V_G - V_T)}{V_T}\right) .$$

The linear pseudo-Ohm's law of equation (4.4) is still valid, but the pseudo-conductance G^* of each transistor is now controllable independently by the value of its gate voltage V_G. The linearity of currents is available in the whole range of weak inversion, which may correspond to 3 to 6 orders of magnitude.

If operation is maintained in weak inversion, the transistors implementing R^* and G^* can be controlled by different gate voltages V_{gg} and V_{gr} and the diffusion length of the diffusion network of Fig. 4.1 (realized with MOS transistors as pseudoconductances) can be electrically adjusted according to

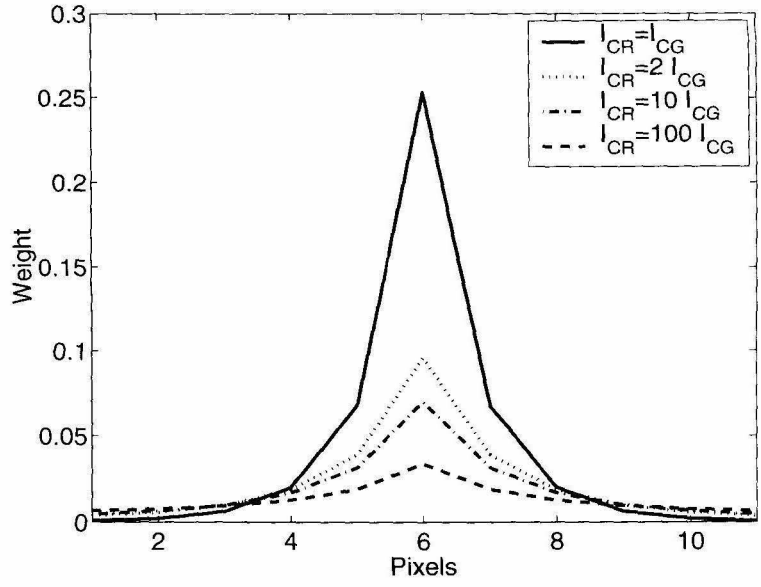

Figure 4.6: Cross section of some of the different weighting functions $w(\mathbf{x})$ that can be obtained by changing the ratio of the two biasing currents I_{CR} and I_{CG}.

$$L = 1/\sqrt{R^* G^*} = \sqrt{\exp\left(\frac{\kappa(V_{gr} - V_{gg})}{V_T}\right)} = \sqrt{I_{CR}/I_{CG}}$$

where I_{CR} and I_{CG} are the control currents of R^* and G^* according to Fig. 4.5. All transistors are in the same substrate and the reference voltage V_R is common to all the control transistors of the network.

Going back to the initial problem of designing a new signal aggregation layer, the possibility of controlling the diffusion length of the pseudo-resistive network allows us the freedom of changing the shape of the weighting function $w(\mathbf{x})$ discussed in chapter 2. In Fig. 4.6 and Fig. 4.7 we present different views of some of the different shapes of the weighting function $w(\mathbf{x})$ that can be obtained by changing the ratio of the two biasing currents I_{CR} and I_{CG}. The plots in Fig. 4.6 and Fig. 4.7 are the results of accurate subthreshold simulations obtained with the circuit simulator Spectre provided with the Cadence environment.

As we can see from the figures, when $I_{CR} = I_{CG}$ most of the weight of the function $w(\mathbf{x})$ is on the central pixel. This corresponds to implementing a feature selection algorithm that emphasizes local singularity of the image. If $I_{CR} \gg I_{CG}$ the weighting

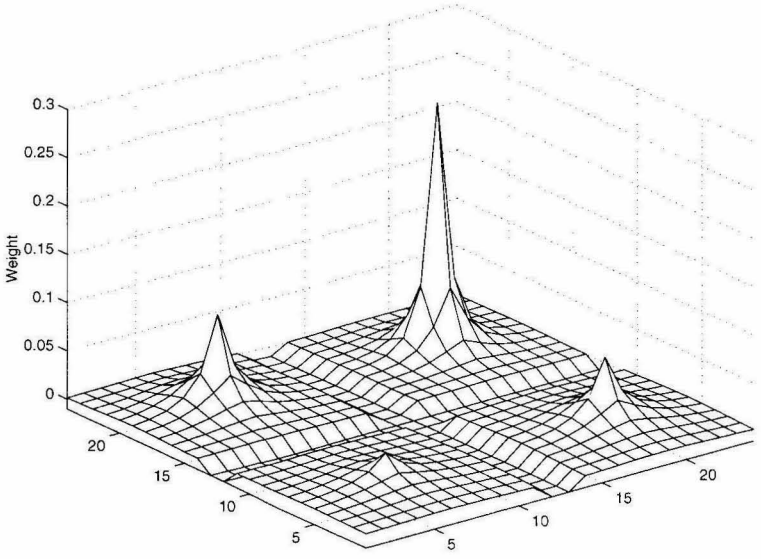

Figure 4.7: Different views of some of the different shapes of the weighting function $w(\mathbf{x})$ that can be obtained by changing the ratio of the two biasing currents I_{CR} and I_{CG}.

function $w(\mathbf{x})$ still has the exponential shape of of equation (4.1) but is flatter than before. Local singularities have less effect on the outcome of the feature selection process. It is interesting to notice that the sum of all the weights for all the different weighting functions is 1 as one should expect. The current I_{ini} injected at the node i has to be equal to the sum of all the currents I_{outj} flowing to the ground for all j. Visually, this is not apparent from Fig. 4.6 and Fig. 4.7, but it is verified by the data.

4.2 Design of *Detector2*

Having found the circuital choice that allows an efficient implementation of the signal aggregation layer, we were able to design a sensor performing the algorithm computations at every pixel. The structure of the sensor *Detecotor2* is reported in Fig. 4.8.

The chips contains an array of 15×15 pixels. The pixels at the border of the array contain only the photoreceptors and the termination transistors of the diffusion networks for the quantities I_a, I_b and I_c. All the other pixels contain the circuitry necessary to perform the computations: the photoreceptors, three multipliers, the

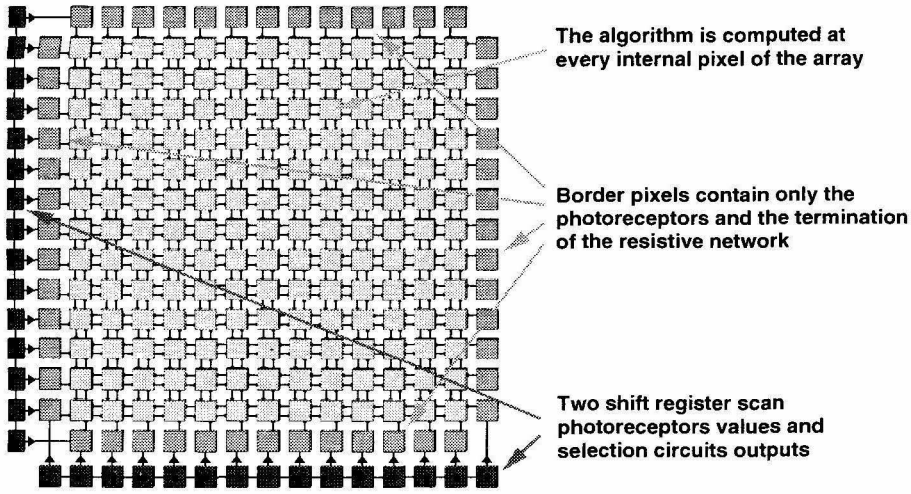

Figure 4.8: The chips contains an array of 15 × 15 pixels. The pixels at the border of the array contain only the photoreceptors and the termination transistors of the diffusion networks for the quantities I_a, I_b and I_c. All the other pixels contain the circuitry necessary to perform all the computations: the photoreceptors, three multipliers, the diffusion network, and the selection circuit. The vertical and horizontal scanner/shift-register allows the read-out of the photoreceptors' values and the result of the selection circuits.

diffusion network, and the selection circuit. The vertical and horizontal scanner/shift-register allows the read-out of the photoreceptors' values and the result of the selection circuits. The biasing circuits are not shown in Fig. 4.8.

4.3 CMOS Implementation

For this design we used a $0.35\mu m$ double-poly quad-metal technology provided by Taiwan Semiconductor Manufacturing Corporation through the MOSIS fabrication service. The choice of a smaller scale technology was primarily due to the necessity to have available more metal layers for wiring the complicated pixel-to-pixel interconnections. As we can recall from Fig. 3.12 of the previous chapter, almost 30% of the area of the pixel was wasted to wiring due to the availability of only two metal layers.

We tested different versions of the same photoreceptor used for *Detector1*, and we chose to use the version using a photodiode made with n-well substrate. In a comparison with another version using n-diffusion photodiode, it showed a higher

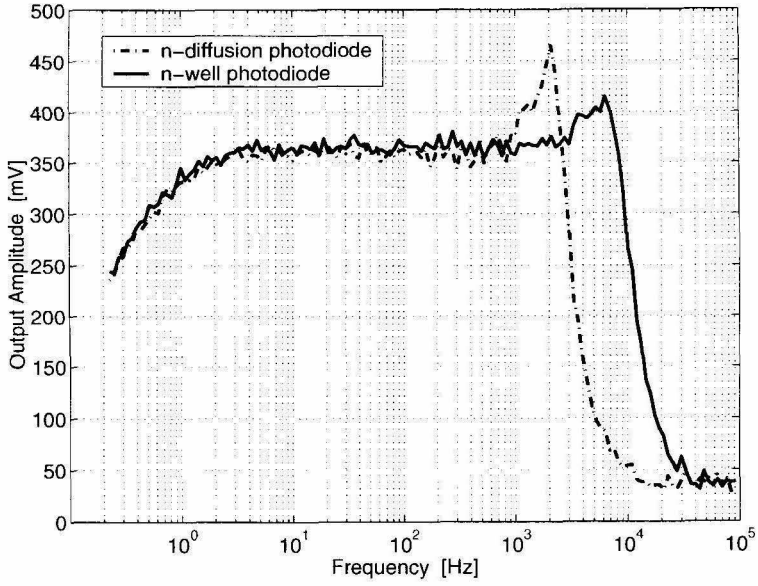

Figure 4.9: Measurements of the bandwidth of photoreceptors using n-well substrate and n-diffusion photodiodes.

bandwidth, as we can see from Fig. 4.9. The higher bandwidth is due to a higher value of photocurrent, as we can see from the measurements of Fig. 4.10, and lower parasitic capacitance. A third implementation using a parasitic bipolar transistor as photodiode was tested and discarded due to instability problems.

In the plot of Fig. 4.9, we can see the effect of the adaptation due to the so-called "tobi-element" (i.e., the central pFET that acts like a pair of diodes in parallel with opposite polarity) of Fig. 3.4. At very low frequencies the response of the photoreceptor goes to zero. In other words, it is able to adapt (i.e., filter out) huge DC variations of incident light intensity.

The pixel still uses three multipliers to compute the quantities $(I_x^i)^2$, $(I_y^i)^2$ and $I_x^i I_y^i$ like in *Detector1* with two minor differences.

First, since in the diffusion network the currents are injected at the source of the pFETs acting as conductances G and extracted from their drain, it is necessary to work with unidirectional currents. The cross term $I_x^i I_y^i$, is therefore encoded as difference between two unidirectional currents that we call I_p and I_n; see Fig. 4.11. This just implies that it is necessary to double the output mirror of the wide linear

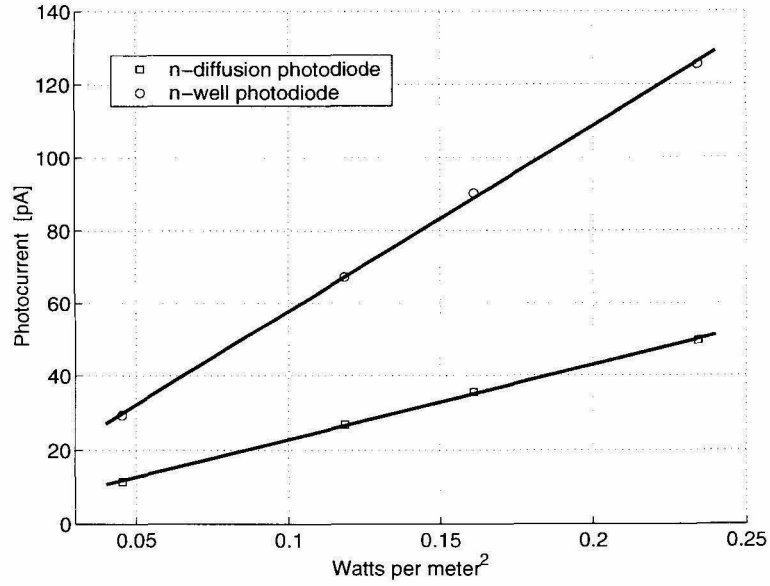

Figure 4.10: Measurements of photocurrent for photodiodes made with n-well substrate and n-diffusion.

range four quadrant multiplier computing the cross term $I_x^i I_y^i$.

The second difference with the previous design of the chip is that some area was saved in the pixel by not including the input differential pair for the multiplier computing the two unidirectional currents I_p and I_n. It is clear, in fact, that if the inputs of the input differential pair are the two voltages V_{left}^i and V_{right}^i, for example, it is possible to obtain the currents that are input of the two output differential pairs from the multiplier computing $(I_x^i)^2 = (V_{left}^i - V_{right})^2$. In this way, it was possible to save the area that before was used for 9 FET transistors, two of which were in their separate wells. The full schematic of the three multipliers is reported at the end of the chapter in Fig. 4.32.

In the signal aggregation layer the terms I_a, I_{bp}, I_{bn} and I_c are generated by combining the quantities $(I_x^i)^2$, $(I_y^i)^2$, I_p^i and I_n^i with the corresponding terms coming from the neighboring pixels using the linear networks based on transistors depicted in Fig. 4.12. Only three transistors per pixel are necessary for every one of the four diffusion networks. Only one lateral transistor is, in fact, necessary between pixels. Not shown in Fig. 4.12 is the simple current mirror that recombines the two currents I_{bp} and I_{bn} into I_b. It is worthwhile to remember that, as long the "vertical"

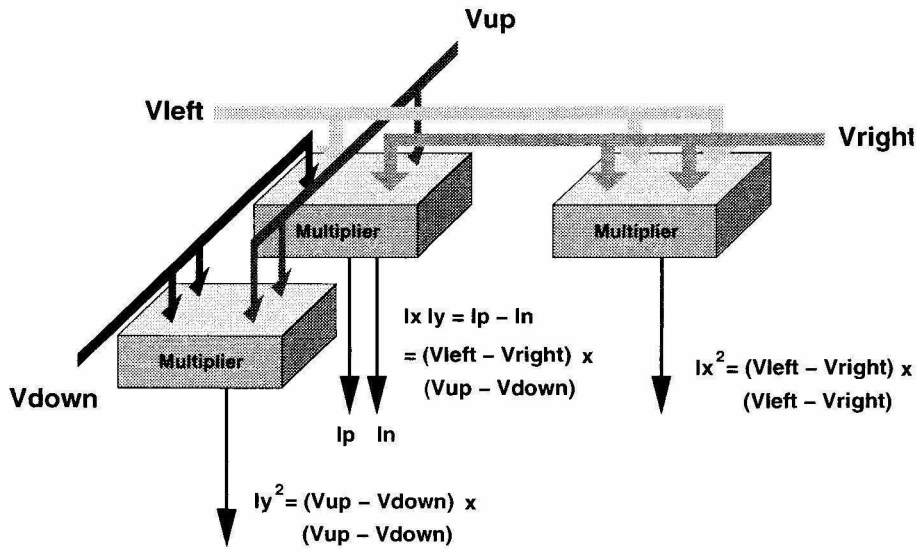

Figure 4.11: The three four quadrant multipliers in the pixel compute the quantities $(I_x^i)^2$, $(I_y^i)^2$ and $I_x^i I_y^i$. The cross term $I_x^i I_y^i$, is encoded as difference between two unidirectional currents I_p and I_n.

transistors remain in saturation, the behavior of the diffusion network is not affected if instead of connecting their drains to ground we insert current mirrors to copy the output currents for further computations.

Due to the behavior of the linear network discussed in section 4.1, the terms I_a, I_b and I_c are now weighted summations of all the $(I_x^i)^2$, $(I_y^i)^2$, I_p^i and I_m^i quantities, i.e.,

$$
\begin{aligned}
I_a &= \sum_{k=1}^{N} w(k) \left(I_x^k\right)^2 \\
I_b &= I_{bp} - I_{bm} = \sum_{k=1}^{N} w(k) \left(I_p^k\right)^2 - \sum_{k=1}^{N} w(k) \left(I_m^k\right)^2 = \sum_{k=1}^{N} w(k) I_x^k I_y^k \\
I_c &= \sum_{k=1}^{N} w(k) \left(I_y^k\right)^2,
\end{aligned}
$$

where N is the total number of pixels in the array and the terms $w(k)$ are the weight of the windowing function, centered on the ith-pixel.

The necessity to perform accurate simulations along with the fact that floating-gate circuits require the use of UV light to equalize the changes forced us to find an

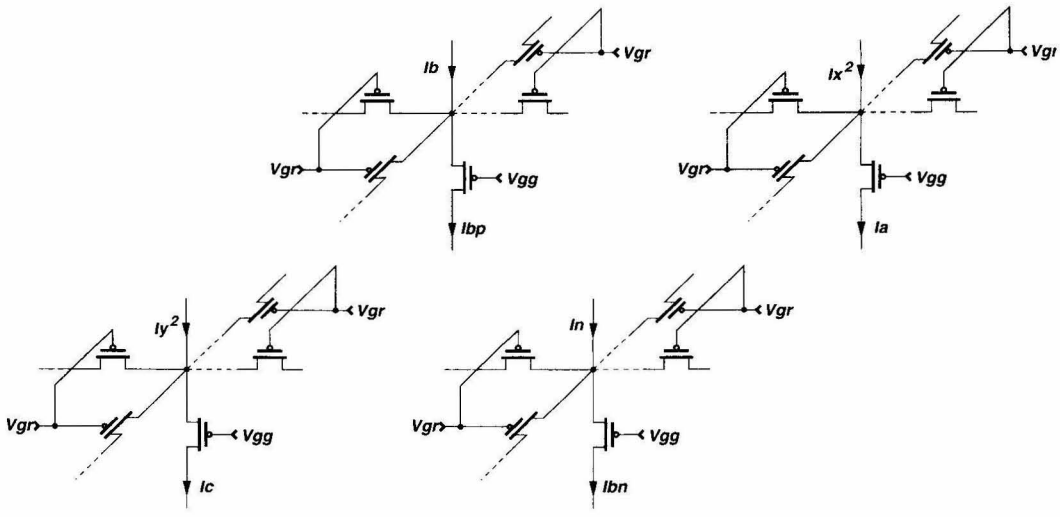

Figure 4.12: The linear networks based on transistors that generate the terms I_a, I_b and I_c combining the quantities $(I_x^i)^2$, $(I_y^i)^2$, I_p and I_n with the same terms coming from the neighboring pixels. Only three transistors per pixel are necessary for every one of the four diffusion networks. Only one lateral transistor is, in fact, necessary between pixels. Not shown in the picture is the simple current mirror that recombines the two currents I_p and I_n into the current I_b.

alternative solution to the use of MITEs for the selection circuit.

The core element of the new selection circuit is current multiplier of Fig. 4.13. To analyze the circuit we can use the Translinear Principle that can be stated as follow: In a closed loop containing an equal number of appositely connected translinear elements, the product of the current densities in the elements connected in one direction is equal to the corresponding product for elements connected in the opposite direction. A translinear element is simply a physical device with a linear relationship between transconductance and current. A FET transistor working in subthreshold can be described as the ideal translinear element thanks to the absence of gate currents.

Analyzing the circuit of Fig. 4.13 and applying the translinear principle around the loop V_{dd}-A-B-C-V_{dd}, we can write

$$I_1 I_2 = I_{ref} I_{out} \ ,$$

and therefore we have

$$I_{out} = \frac{I_1 I_2}{I_{out}} \ .$$

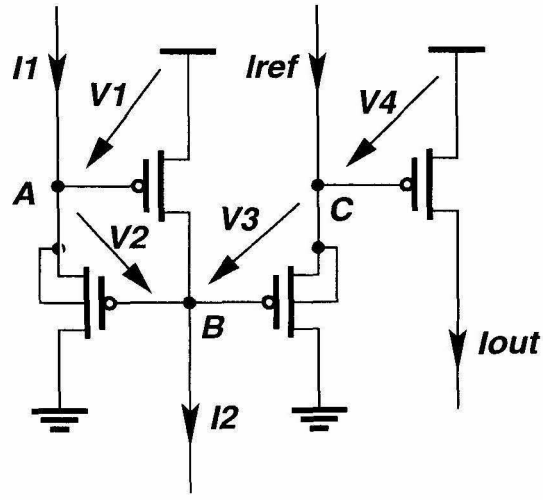

Figure 4.13: The current multiplier used in the new selction circuit. Applying the translinear principle around the loop V_{dd}-A-B-C-V_{dd}, we have $I_1 I_2 = I_{ref} I_{out}$, and therefore $I_{out} = I_1 I_2 / I_{out}$.

Without using the translinear element we can derive the same equation by first summing the voltages around the loop

$$V_1 + V_2 = V_3 = V_4 \ ,$$

and substituting from equation (3.3) from the previous chapter, we have

$$\frac{V_T}{\kappa} \ln\left(\frac{I_1}{I_0}\right) \frac{V_T}{\kappa} \ln\left(\frac{I_2}{I_0}\right) = \frac{V_T}{\kappa} \ln\left(\frac{I_{ref}}{I_0}\right) \frac{V_T}{\kappa} \ln\left(\frac{I_{out}}{I_0}\right) \ ,$$

from which we can obtain again $I_{out} = I_1 I_2 / I_{out}$.

The current multiplier of Fig. 4.13 can be easily simulated with standard simulation tools, as will see in the next section. On the other hand it is difficult to say whether something was gained in terms of area efficiency. Before, a considerable amount of area has to be allocated for the capacitors of the floating-gate transistors; in this case, some area has to be left unused to allow two pFETs to sits in their separate well. Moreover, the circuits require to be operated in subthreshold and therefore the four pFET transistors have to be drawn with a high W/L ratio.

A simplified version of the selection circuit is reported in Fig. 4.14. The two

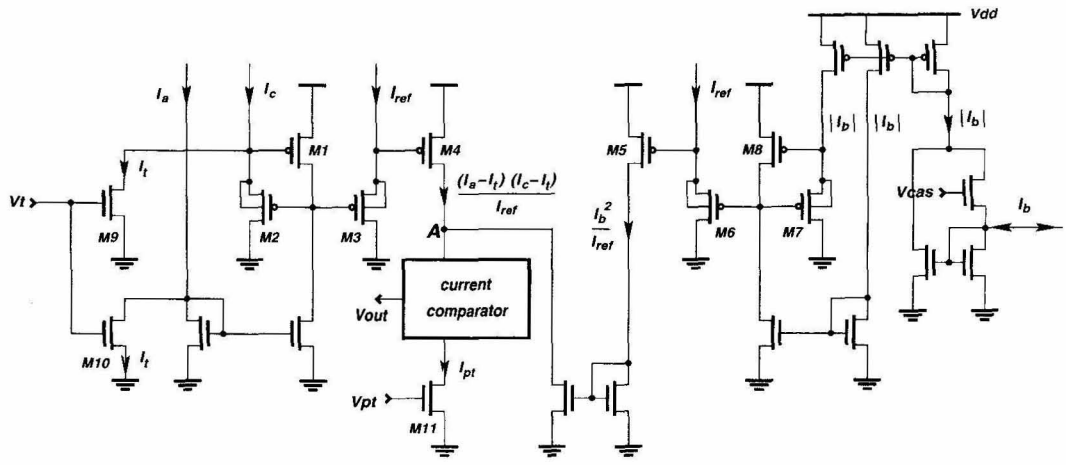

Figure 4.14: The selection circuit for the sensor Detector2. The two current multipliers are the two groups of four pFET transistors $M1$-$M4$ and $M5$-$M6$. The multiplier comprising of transistors $M1$-$M4$ outputs the term $(I_a - I_t)(I_c - I_t)/I_{ref}$ while the other computes the term I_b^2/I_{ref}. The two currents are then subtracted at node A that is the input of the current comparator performing the final comparison: its output is, in this case, high if $(I_a - I_t)(I_c - I_t)/I_{ref} - I_b^2/I_{ref} > I_{pt}$ (condition $P(\lambda_t) = (a - \lambda_t)(c - \lambda_t) - b^2 > p_t$) and low otherwise.

current multipliers are the two groups of four pFET transistors $M1$-$M4$ and $M5$-$M6$. The multiplier comprising of transistors $M1$-$M4$ outputs the term $(I_a - I_t)(I_c - I_t)/I_{ref}$ while the other computes the term I_b^2/I_{ref}. Once again the two currents are subtracted at node A that is the input of the current comparator performing the final comparison: its output is, in this case, high if $(I_a - I_t)(I_c - I_t)/I_{ref} - I_b^2/I_{ref} > I_{pt}$ (condition $P(\lambda_t) = (a - \lambda_t)(c - \lambda_t) - b^2 > p_n$) and low otherwise. Once again, we can easily see how the circuit of Fig. 4.14 allows the approximated version of the algorithm (i.e. test only for $P(0) = I_a I_c - I_b^2 > I_{pt}$) to be implemented by simply turning off (i.e., current $I_t = 0$) the two transistors $M9$ and $M10$.

In the case of an implementation of the second algorithm, some area can be saved by the elimination of the two transistors $M9$ and $M10$ that subtract I_t from the two currents I_a and I_c.

Performing extensive simulations of the circuit, we noticed that the accuracy of the computation could be increased significantly if most of the mirrors were replaced by cascoded mirros. The final version of the selection circuit that was included in the chip is reported in Fig. 4.33. The other difference with the circuit of Fig. 4.14 is that

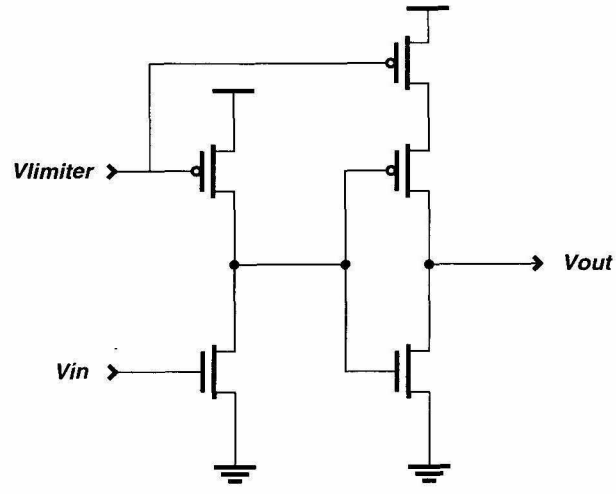

Figure 4.15: The very simple current comparator used in the chip. The two current starved inverters allow some power saving in case of borderline cases.

the current $I_c - I_t$ is not injected directly into the source of the transistor $M2$ from the diffusion network, and double mirrors had to be used instead. The reason for those extra transistors is that if $I_a - I_t = 0$, then the transitor $M1$ is off causing the transistor $M2$ to be off also. In that case there is no path to ground for the current $I_c - I_t$ coming from the diffusion network. That current, consequently, is routed to other pixels altering the precision of the calculation.

The very simple current comparator is reported in Fig. 4.15. Its input is connected to node A of Fig. 4.14 and it consists of just a series of two current starved inverters with a fairly high gain output stage in order to properly drive the digital circuitry of the scanning architecture. The only reason to include current starved inverters was to decrease the power wasted in case the currents $(I_a - I_t)(I_c - I_t)/I_{ref} - I_b^2/I_{ref}$ and I_{pt} were close in magnitude. In that case it is reasonable to expect the simple inverter used in *Detector1* to be in conduction consuming, therefore, a significant amount of power.

The scanning architecture is reported is Fig. 4.16. The shift-register on the side of the array select only one row at the time (i.e., *RowSelect* high); if there is a feature, the output of the current comparator is high and therefore the node A is low due the nand circuit. If node A is low, the transistor $M1$ pulls the $ColumnOutputWire$

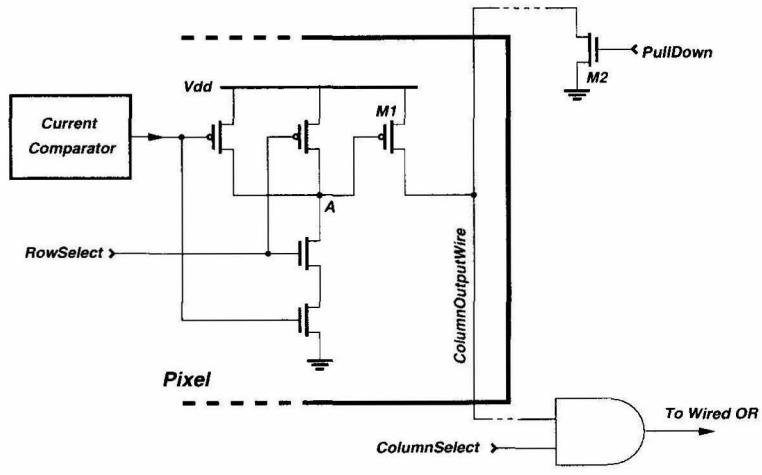

Figure 4.16: The scanning architecture of the chip. The vertical shift-register selects a row, the wired OR, made by transistors $M1$ in the pixels and transistor $M2$, outputs the result of every column to the bottom of the chip where the horizontal shift-register and another wired OR conveys the results to the output pad.

high. The $ColumnOutputWire$ and the pull down transistor $M2$ at the top of every column work as a wired OR. Only one of the transistors $M1$ of the column can be on at every time because of the selection of the shift-register. If none of the pixels in that column has a feature, then none of the transistors $M1$ is active and the line is pulled low by the transistor $M2$. At the output of the column the horizontal shift-register selects only one column at a time and another wired OR conveys the value to the output pad of the chip.

4.4 Simulation Results

One advantage of this second thresholding circuit over the one integrated used for Detector1 is the possibility of performing accurate simulations without any of the problems associated with having to set the initial conditions on the floating-gates of the MITEs. The photoreceptor circuits were not included in the netlist used for the simulations, and therefore a static set of voltages was applied at every pixel to the inputs of the multipliers to simulate the effect of focusing an image onto the chip with a lens. For all the results that we will present, the set of voltages used corresponds

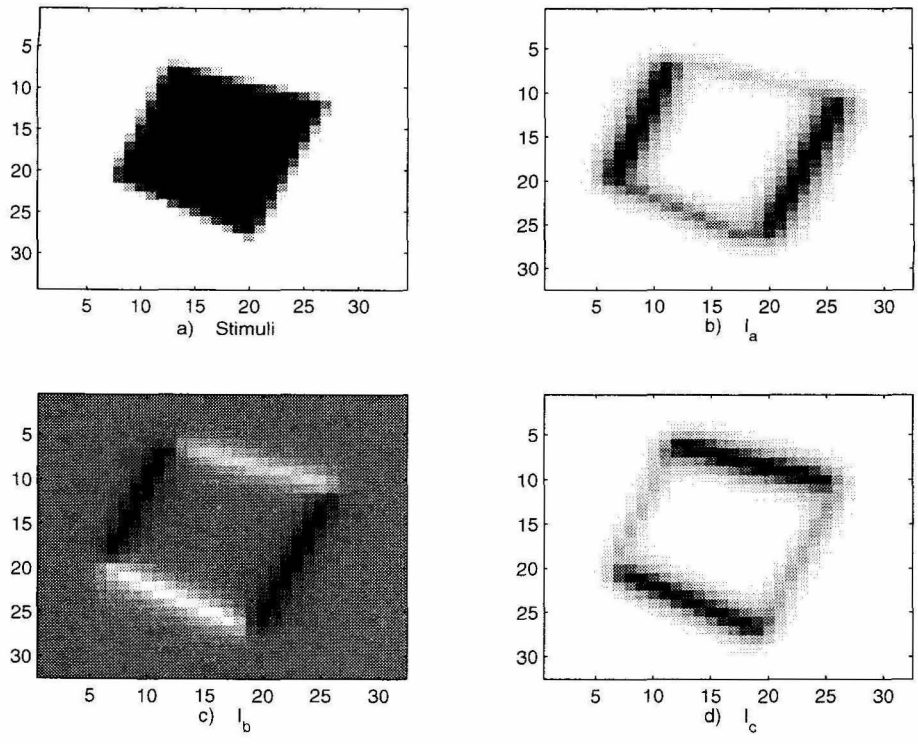

Figure 4.17: SPICE simulation results: **a)** Input image. **b)** Current I_a. **c)** Current I_b. **d)** Current I_c.

to the synthetic image of Fig. 2.8.

In Fig. 4.17a the plot of the stimuli used again displayed to ease the comparison with the adjacent plots. In Fig. 4.17b, c and d we plot the simulation results for the three currents I_a, I_b and I_c respectively. As was expected I_a is higher for those pixels that correspond to the two almost vertical edges of the stimuli while I_c clearly detects the two horizontal ones.

In Fig. 4.18 and Fig. 4.19, we plot the current $(I_a - I_t)(I_c - I_t)/I_{ref} - I_b^2/I_{ref}$ that is subsequently compared to I_{pt} to determine the presence of a feature. It is evident how the circuit is able to select the four corners of the input image that are the only regions not affected by the aperture problem. This current should be proportional to the function representing the minimum eigenvalue λ_1. In Fig. 4.20 we plot that function computed with MATLAB starting from the synthetic image. As we can see it is almost identical to the result of the circuit simulation reported on Fig. 4.19.

In Fig. 4.21, Fig. 4.22, Fig. 4.23 and Fig. 4.24, we perform the same compar-

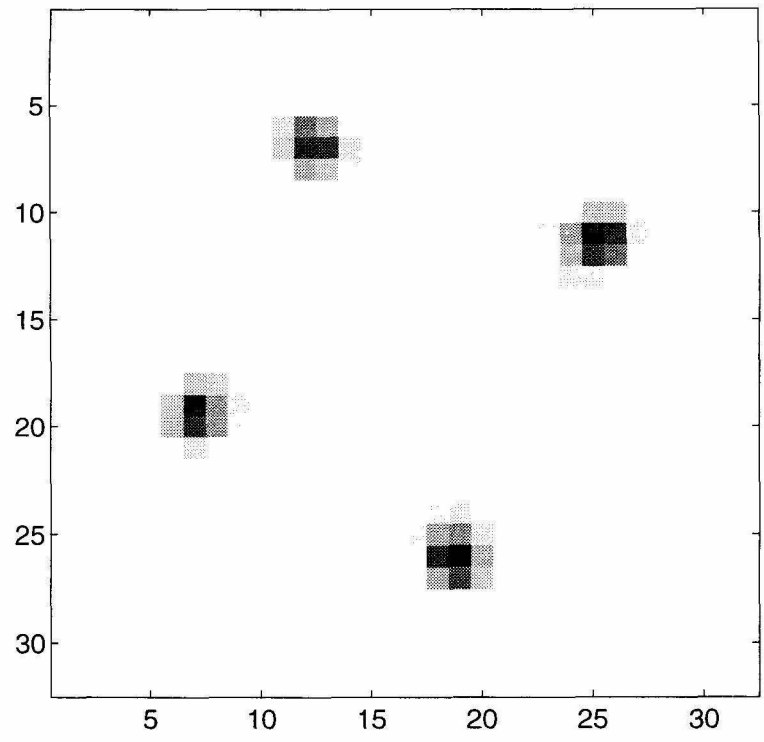

Figure 4.18: The simulation results for the current $(I_a - I_t)(I_c - I_t)/I_{ref} - I_b^2/I_{ref}$. It is evident how the circuit is able to find the four corners of the input image in Fig. 4.17a.

ison between circuits simulations and MATLAB results for four of the fundamental computation modules of the sensor. In Fig. 4.21 we compare the simulation result of the multiplier computing I_a with the ideal result of I_x^2 computed from the synthetic image. In Fig. 4.22 we compare the accuracy of the multiplier that computes the quantity $(I_a - I_t)(I_c - I_t)/I_{ref}$, and in Fig. 4.23 we look at the operation of absolute value on the current I_b, and finally in Fig. 4.24 is reported the comparison between the simulation results for I_b^2/I_{ref} and its ideal counterpart. As we can see for all the four comparisons, the simulation results closely match the expected ideal results.

We designed, fabricated and tested the sensor with a 15×15 array of pixels. As we mentioned before the chip was fabricated in a $0.35\mu m$ double-poly quad-metal technology provided by Taiwan Semiconductor Manufacturing Corporation through the MOSIS fabrication service Tiny-Chip. The die size was 2×2mm.

A micro-graph of the chip is shown in Fig. 4.25. The actual area occupied by the design is 1.6×1.6mm. In Fig. 3.12 we show a picture of the layout of the pixel. In

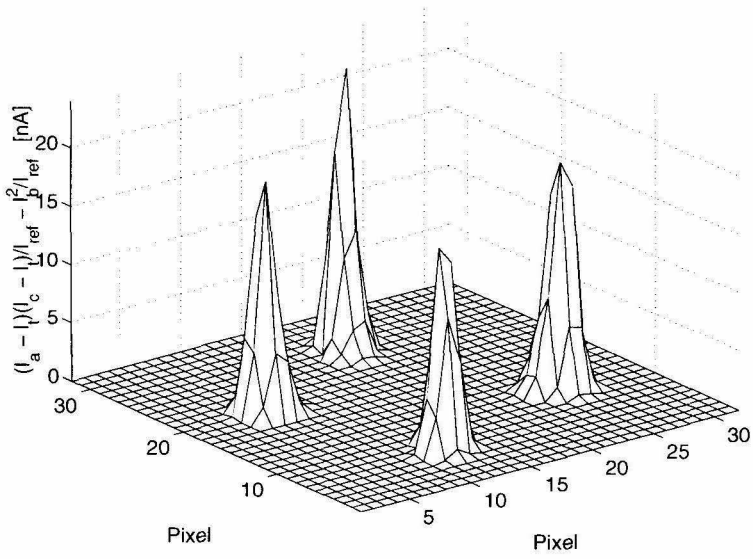

Figure 4.19: The simulation results for the current $(I_a - I_t)(I_c - I_t)/I_{ref} - I_b^2/I_{ref}$ that is subsequently compared to I_{pt}.

Fig. 3.13 we show a micro-graph of the pixel. The layout is reported in Fig. 4.26 with a close-up micro-graph of the pixel in Fig. 4.27. The pixel size is $100 \times 100\mu m^2$ in this technology. The total number of transistors in the pixel is 140 and the fill factor of the sensor is about 6%.

The sensor was characterized with the same testing methodology used for *Detector1.*

In Fig. 4.28 we present the results of the characterization of one pixel of the sensor implementing the simplified algorithm of conditions (2.13-2.14). The current I_p was again set to zero and the value of the current I_t was varied to chracterize the implementation of the condition $(I_a - I_t)(I_c - I_t)/I_{ref} - I_b^2/I_{ref} > 0$. As we can see, even if two sensors are quite different in design, the curves are very similar to the curves obtained for the sensor Detector1 in Fig. 3.18. The slopes of the different transition regions are lower than before, hinting a lower selectivity of the new selection circuit with respect to the one used in Detector1. Another possible explanation of the decreasing slopes can be found in the higher standard deviation of the output voltages of the array of photoreceptors. For this design we measured an average peak-to-peak output of 496mV with a standard deviation of 81mV, for

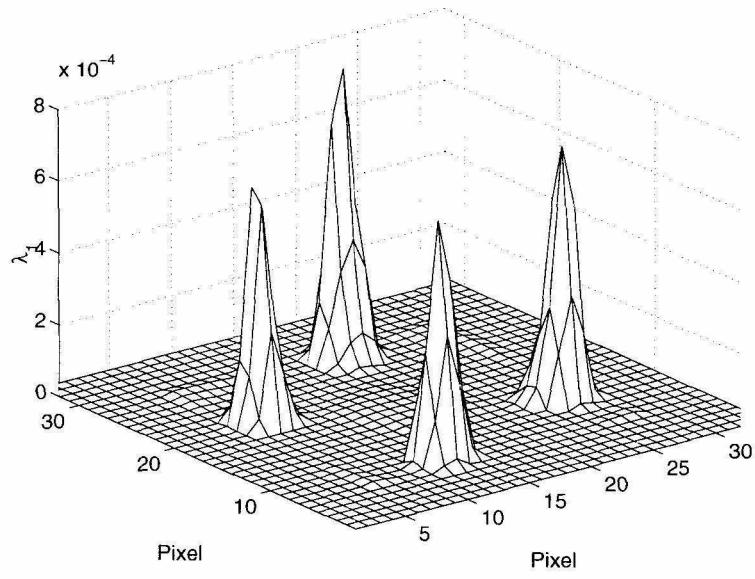

Figure 4.20: MATLAB calculation of the minimum eigenvalue λ_1 of the synthetic image used.

a 100% contrast stimuli. The DC operating point of every photoreceptor was also measured. The mean was 1.198V with a standard deviation of 49mV. Comparing these results with the corresponding values measured for *Detector1*, we see that the standard deviation of the peak-to-peak value is almost three times as big, and the one associated with the DC operating point is more than double than before. We learned in the previous chapter that with these increases in standard deviation we should expect some degree a decrease in the slopes of the curves.

The only difference in interpreting the data of Fig. 4.28 and comparing them with the data obtained for Detector1 is that the signal aggregation layer is now implemented with a diffusion network instead of hard wiring the currents of the 9 pixels to the selection circuit as it was for Detector1. The implication of this fact is that when the results of Fig. 4.28 are plotted, the minimum eigenvalue of every test pattern should be recomputed to correctly reflect the effect on the overall computation of the diffusion network.

For all the measurements here presented, the two biasing currents of the diffusion network were kept equal. Using the simulation results of the weighting functions, we obtained a rough estimate of the weighting function that correspond to that biasing

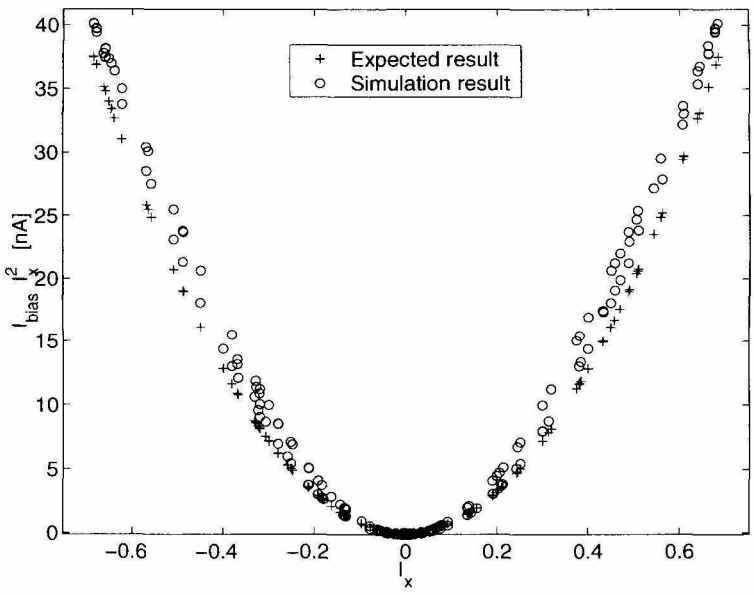

Figure 4.21: Comparison between the simulation result of the multiplier computing I_a with the ideal result of I_x^2 computed from the synthetic image.

setting. Using that information to plot once again the same data of Fig. 4.28, we obtain Fig. 4.29 that represents a better picture of the overall circuit behavior. The range of the minimum eigenvalue of the test patterns is now greatly reduced, but the overall shape of the four curves is not drastically altered.

Fig. 4.30, present the results of the characterization of the implementation of the approximated algorithm of condition (2.15). Even in this case the curves are not significantly different from the corresponding curves obtained during the testing of *Detector1*.

Since *Detector2* has an array of 15×15 pixels, we were able to perfom some test to better characterize the effect of transistor mismatches. The most significant of these tests is the characterization of the variability of the curves between pixel and pixel keeping the biasing and thresholding currents constant.

In Fig. 4.31 we plot the curves obtained from the 11 pixels on one of the two main diagonals at the same biasing conditions. The two pixels at the two ends of the diagonal were not tested to avoid edge effects in the results. As we can see even if more that half of the pixels have similar curves, there are at least four of them that are

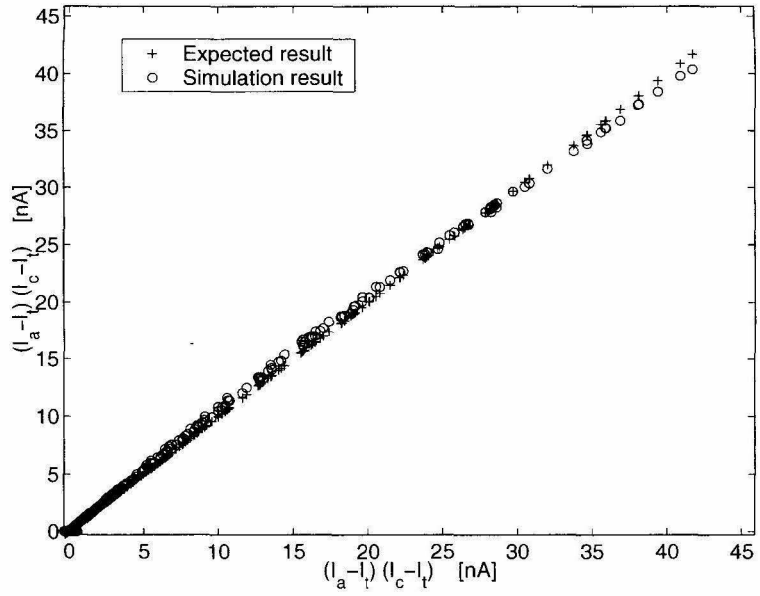

Figure 4.22: Comparison of the accuracy of the multiplier that computes the quantity $(I_a - I_t)(I_c - I_t)/I_{ref}$.

completely outliers. One pixel, for example, always reports the presence of features even for very low minimum eigenvalue levels, while three others hardly reported any feature at all even for the patterns with the highest λ_1. This fact in itself limits the accuracy of the overall computation of the sensors since we should expect a much more uniform behavior of the various pixels across the chip.

4.5 Conclusion

In this chapter we presented the final design of the computational sensor for focal plane computation of image features. This final design addressed all the issues raised by the initial design. The algorithm is now computed at every pixel thanks to the elegant use of the diffusion network. The change in selection circuit allowed accurate simulation to be performed on the design without any evident degradation of selectivity of the computation itself. The fill factor was also increased by more than a factor of five in this new design.

The testing of the sensor highlighted the only major issue of this new design that

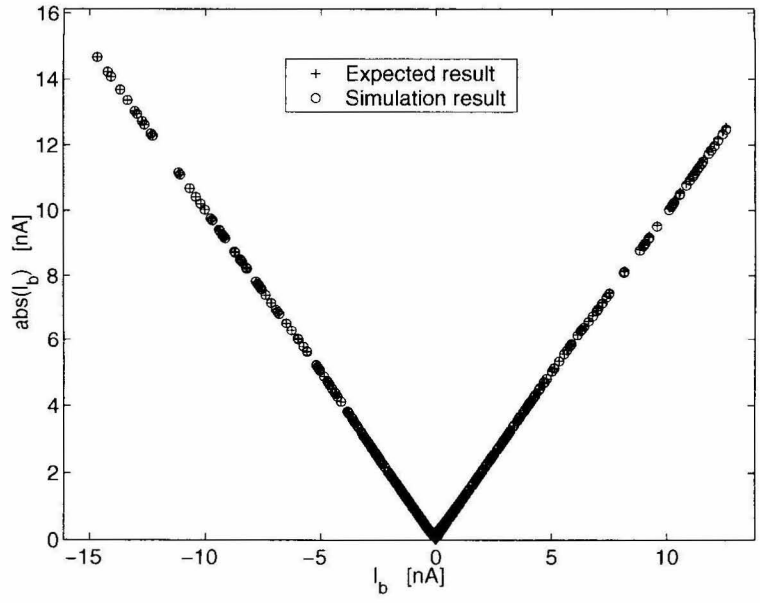

Figure 4.23: Comparison of the accuracy of the circuit that computes the absolute value of the current I_b.

is the not perfect consistency of the computation across all the pixels of the sensor. The cause of this non uniformity has to be found in voltage and current offsets due to transistor mismatches. The next chapter presents a possible way to address this problem in a fundamental new way.

This feature detection scheme in association with a simple token-based velocity algorithm [22] can be used to build a motion sensor not affected by the aperture problem. Another possible use of this implementation of the algorithm is to replace the intensity-based saliency map with a feature-based one in visual attention or oculomotor systems [17, 24].

If fabricated in a denser array, either using a smaller technology or a large die, a feature detection chip can be used as a front-end to those navigation systems that detect and track features to navigate in the environment.

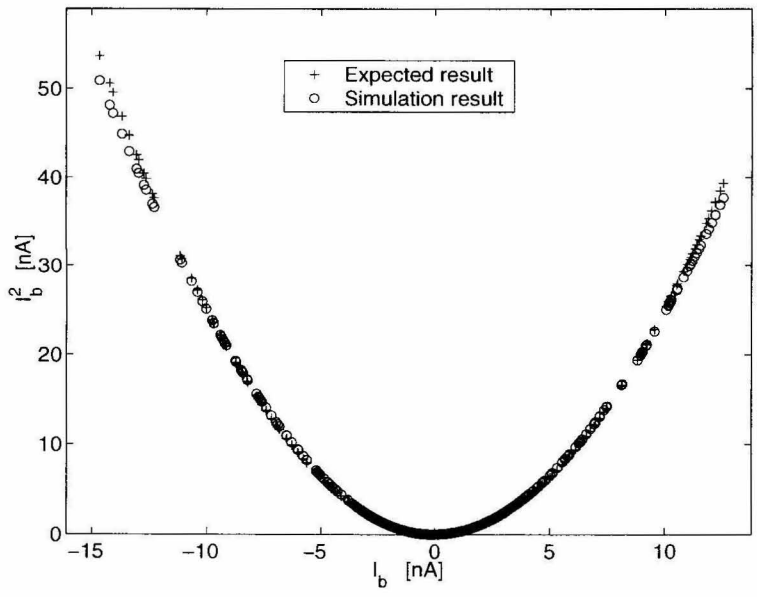

Figure 4.24: Comparison of the simulation results of I_b^2/I_{ref} and its ideal counterpart.

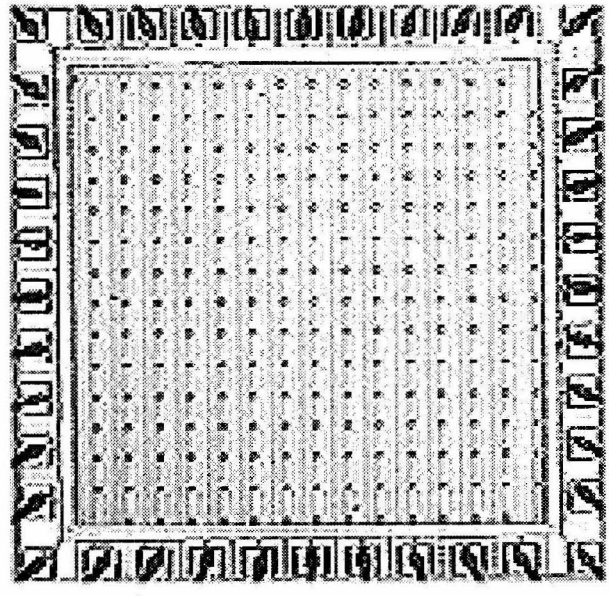

Figure 4.25: Micro-graph of CMOS sensor. The actual area occupied by the design is 1.6×1.6mm.

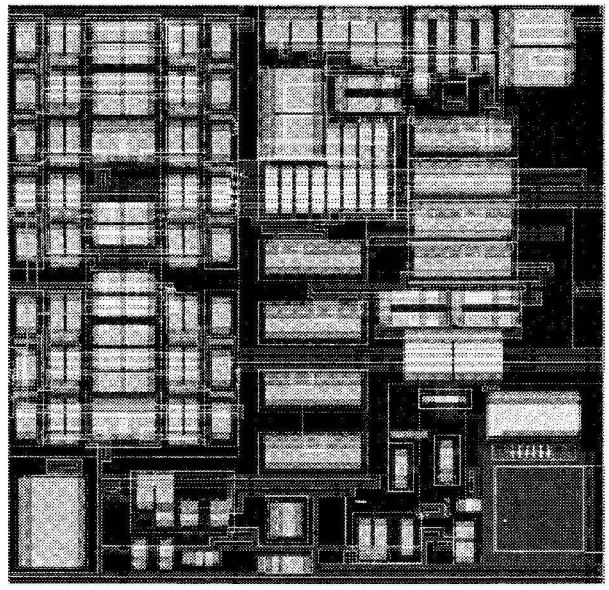

Figure 4.26: The layout of the pixel. The pixel size is $100 \times 100 \mu m^2$ in this technology. The total number of transistors in the pixel is 140 and the fill factor of the sensor is about 6%.

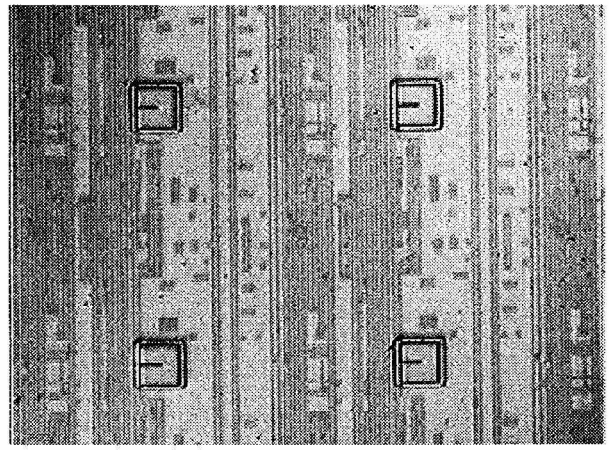

Figure 4.27: Close-up micro-graph of the area around one pixel. The third metal layer cover most of the area and only the photodiode is visible. The fourth layer of metal was not used in the design.

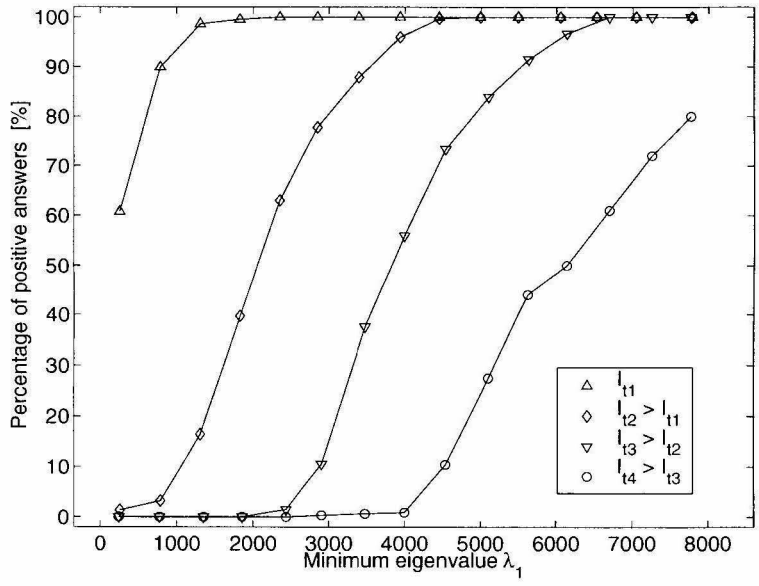

Figure 4.28: Results of the characterization of one pixel of the sensor implementing the simplified algorithm of conditions (2.13-2.14). The current I_p was set to zero and the value of the current I_t was varied to chracterize the implementation of the condition $(I_a - I_t)(I_c - I_t)/I_{ref} - I_b^2/I_{ref} > 0$.

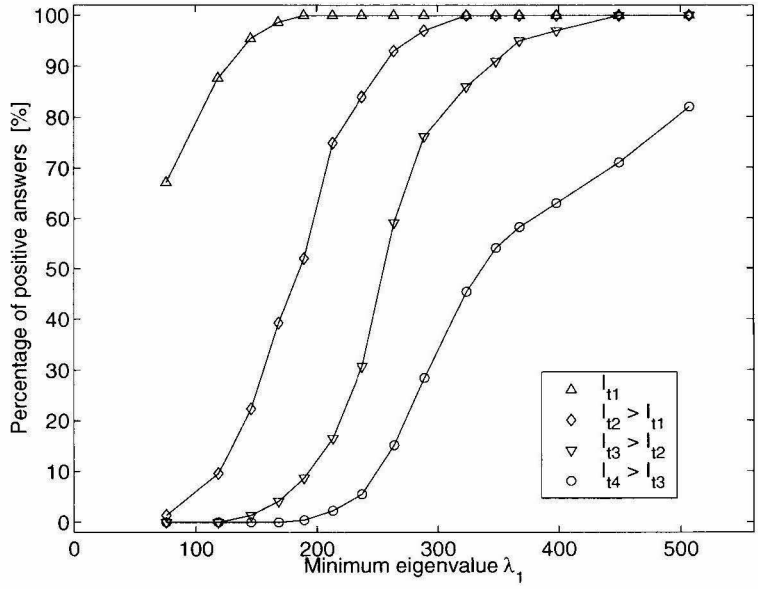

Figure 4.29: Plot of the same curves of Fig. 4.28 taking into account the effect of the implementation using a diffusion network of the signal aggregation layer.

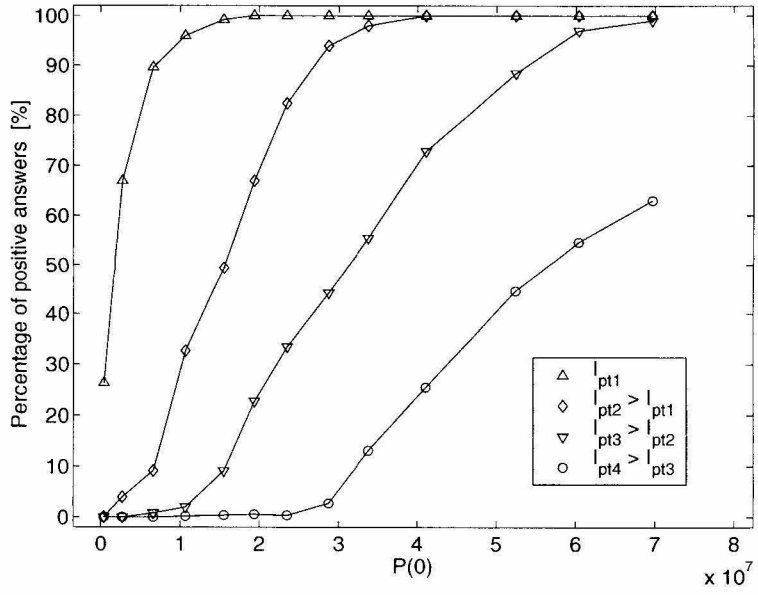

Figure 4.30: Results of the characterization of the implementation of the approximated algorithm of condition (2.15). The curves are not significantly different from the corresponding curves obtained during the testing of Detector1.

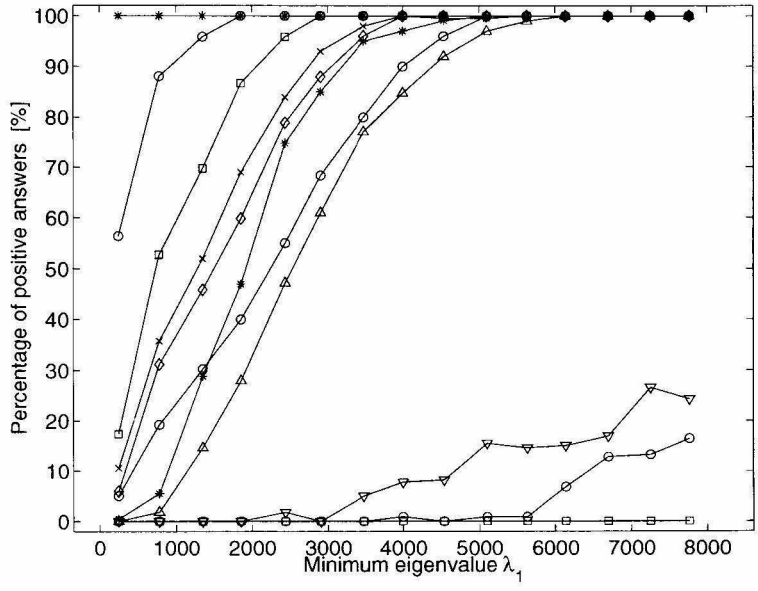

Figure 4.31: Plot of the curves of the implementation of the simplified algorithm for 11 pixels of one of the diagonals for the same biasing settings.

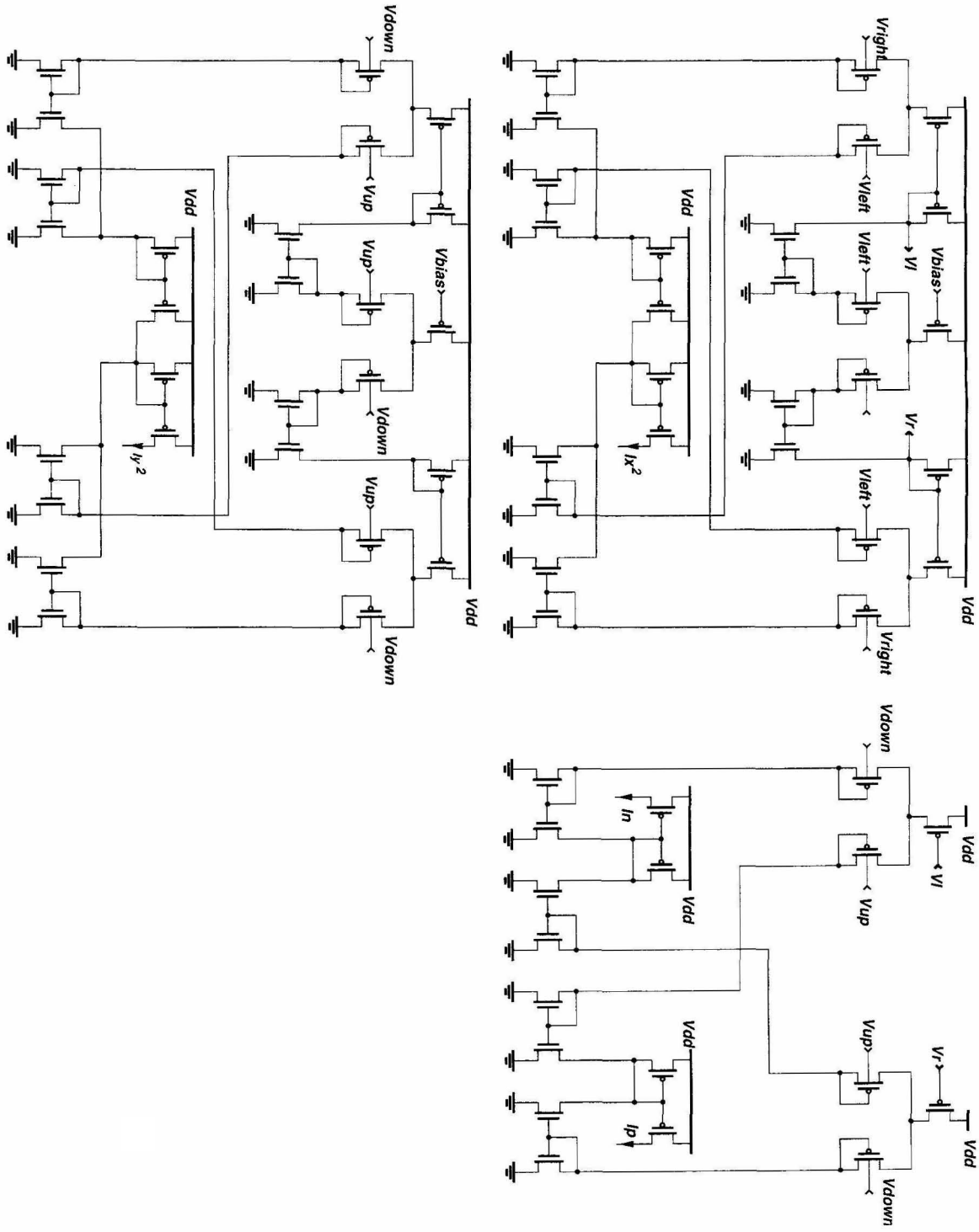

Figure 4.32: The complete schematic of the three multipliers.

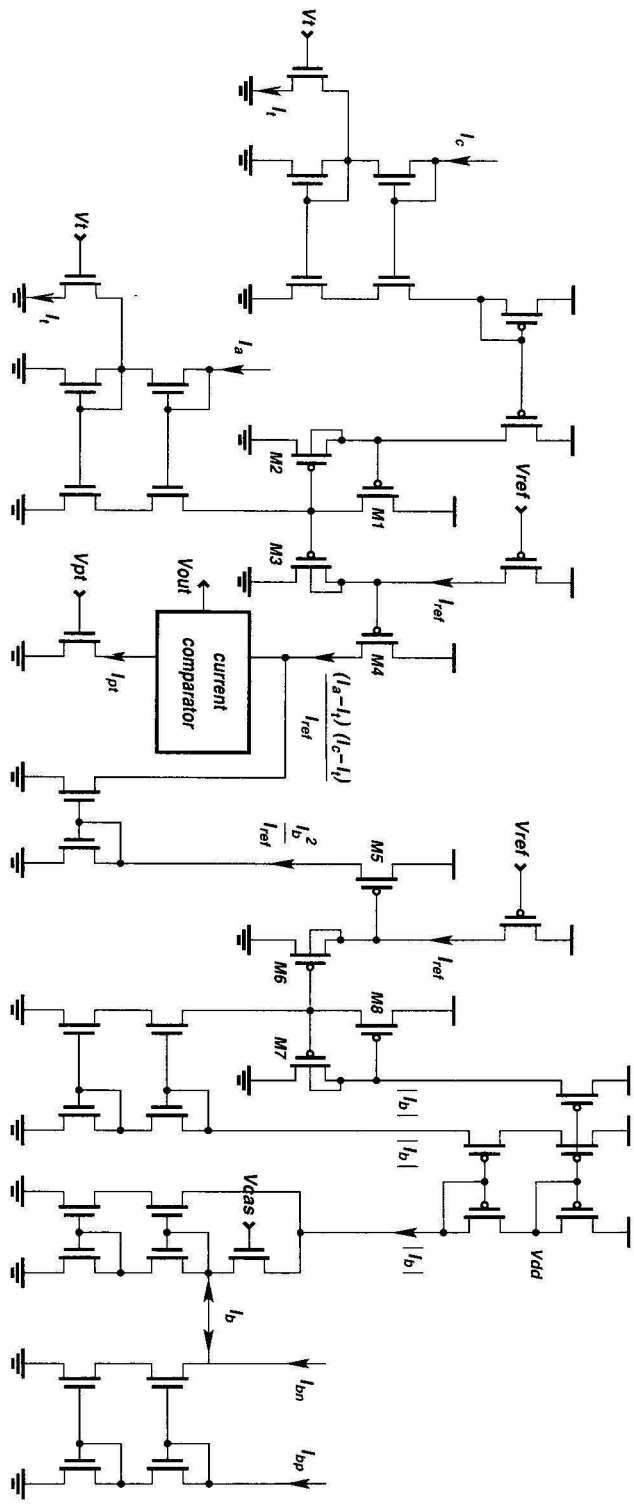

Figure 4.33: The complete schematic of the selection circuit used in the chip.

Chapter 5 Foating-Gate Transistors and Focal Plane Arrays

The design of large arrays of analog circuits in VLSI is constrained by the inherent mismatch of transistors from the fabrication process. In photoreceptor arrays, the mismatch can appear as gain and offset errors. Under uniform intensity, such pixels will report slightly different values, producing a "fixed-pattern noise" image. While the removal of fixed-pattern noise is often performed by the subtraction of a calibration image stored on a downstream digital computer, the desire to combine both sensing and processing on the same chip ("smart sensors") has precipitated the need for a more integrated solution. A common solution to this problem is to measure and store a correction value locally at each pixel which is subtracted before the output.

Although short-term storage can be performed on integrated capacitors, junction leakage from the connected circuitry limits its retention time to seconds, particularly for analog parameters. Floating-gate MOSFETs (MOS transistors with their gate completely surrounded by silicon dioxide), however, can provide an extremely effective charge-storage technique with its retention measured in years. Charge modification techniques using ultra-violet (UV) radiation [5, 14, 31, 19] and bidirectional Fowler-Nordheim tunneling (e.g., [26, 38, 3, 23]) have both been successfully tested; there are, however, some drawbacks to these techniques such as the need for a UV light source, multiple high-voltage supplies, or special fabrication processes. Recently the combination of tunneling and hot-electron injection [10] has emerged as a promising new charge-modification technique that requires only one high-voltage supply and standard CMOS fabrication processes. These types of structures are now being used for many different applications such as on-chip parameter storage (e.g., [5, 15, 25]) and neural networks (e.g., [42, 4, 36]).

As a result of these developments in technology, the use of floating-gate structures

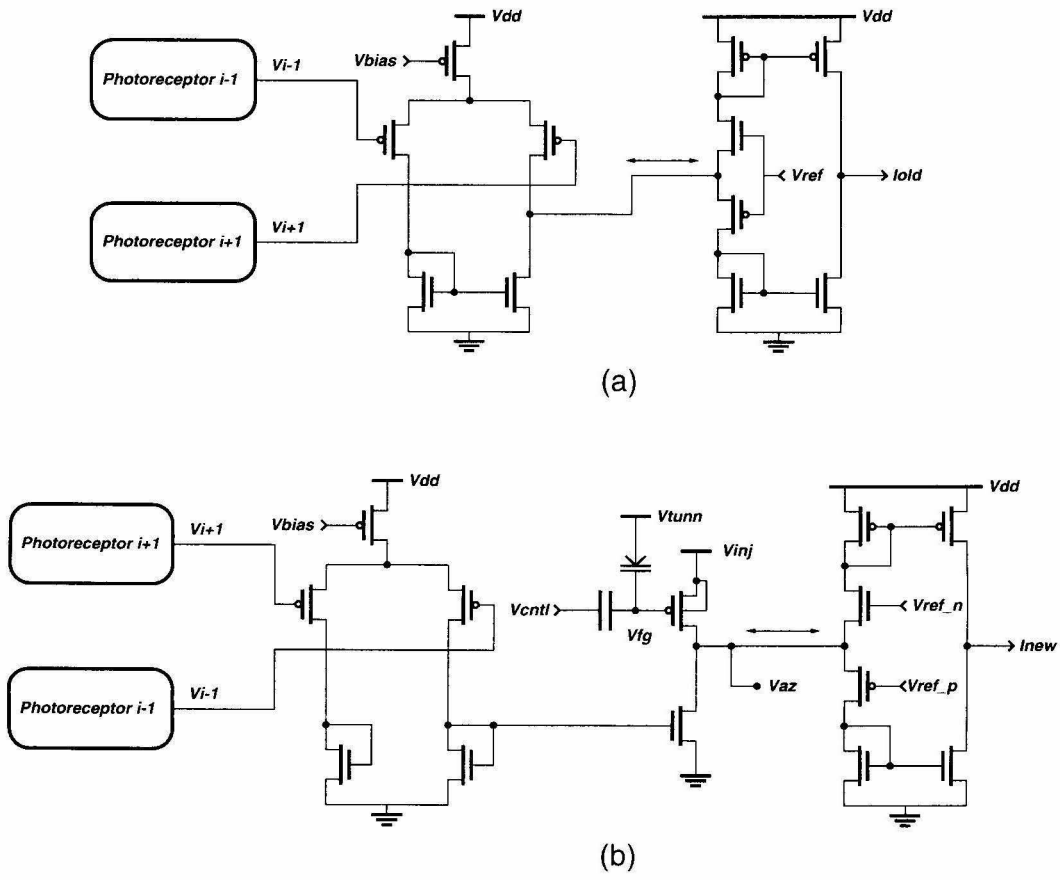

Figure 5.1: (a) Circuit diagram of the old circuit used to compute the spatial-derivative in analog VLSI imagers. (b) Circuit diagram of the adaptive spatial-derivative circuit. Adding the floating-gate amplifier and one bias line to the pair of stacked mirrors allows the circuit to remove offsets continuously.

for fixed-pattern noise removal in images has been growing steadily in recent years. While earlier work used the UV technique to null offsets in a silicon retina [31], recent approaches have used bidirectional Fowler-Nordheim tunneling [8, 1] for storing image offset values.

In applications where the absolute image intensity is not preserved and the local spatial-derivative information is used, spatial-derivative is an appropriate signal to calibrate. This is the situation for many neuromorphic circuits [11] that adapt signals both spatially and temporally.

In this paper we present a new approach to fixed-pattern noise removal, by very slowly adapting the output of the spatial-derivative computation to zero rather than

matching the photoreceptor outputs and balancing the spatial derivative circuit. Using the floating-gate auto-zeroing amplifier described by Hasler et al. [16], a spatial-derivative circuit currently being used in other projects [21, 18] was modified to dramatically reduce offset errors, or equivalently, to increase the dynamic range. This circuit uses a combination of electron tunneling to reduce charge and hot-electron injection to increase charge on the transistor gate.

5.1 Circuit Description

The circuit previously used to compute the spatial-derivative is shown in Fig. 5.1a. A transconductance amplifier receives input from two photoreceptors (P_{i+1} and P_{i-1}) and provides an output current that is a sigmoidal function of its differential input and therefore a second-order approximation of the spatial-derivative of the input image. Positive current sourced from the amplifier is pushed into the n-type current mirror (in the bottom part of the pair of stacked mirrors) and negative current is drawn out of the p-type current mirror (in the top part of the circuit). The output arms of the two current mirrors are connected together to provide a bidirectional output current. The two transistors connected to V_{ref} perform a thresholding operation, preventing very small spatial-derivative currents from appearing in the final output. While this can be desirable to reduce the effects of circuit offsets, it manifests itself as a "dead-zone" in the spatial-derivative transfer characteristics. Other configurations of the pair of stacked mirrors to compute the polarity or the absolute value of the bipolar current have been presented in the literature [21, 18].

The new circuit that we present is shown in Fig. 5.1b. The pFET differential pair is, in this case, terminated through a pair of diode-connected transistors. The diode-connected transistor on the right is the input to a current mirror which constitutes the input to the inverting auto-zeroing amplifier. This is the new output stage of the differential amplifier. As in the previous case, the output current is either drawn out of, or pushed into, the current mirrors on the top and bottom. The pair of stacked mirrors is a slightly modified version of the previous one to provide more control over

the dead-zone created by the threshold voltages of the nFET and pFET transistors in the center.

Before considering the adaptive behavior of the circuit when tunneling and injection are present, let us first describe the principle of operation without considering the auto-zeroing properties of the amplifier.

If the two input voltages V_{i+1} and V_{i-1} are equal, the current provided by the bias transistor is divided equally between the two arms of the differential pair. Let us assume now that in this condition the voltage V_{az} sits at $V_{dd}/2$. When the output voltage of the photoreceptor P_{i+1} increases with respect to P_{i-1}, more current starts flowing through the right-hand arm of the differential-pair and into the nFET current mirror. This increased current then pulls V_{az} down. There is a voltage level V_{down} at which the nFET controlled by V_{ref_n} turns on and the pFET mirror starts conducting, thus clamping the V_{az} voltage near V_{down}. The value of V_{down} is set by the threshold voltage of the nFET and by the bias voltage V_{ref_n}. If the difference between V_{i+1} and V_{i-1} keeps increasing, then the current flowing through the pFET mirror will increase and V_{az} will remain very close to the same value. Conversely, if V_{i+1} is less than V_{i-1}, the current through the diode connected nFET will decrease causing V_{az} to increase until the pFET controlled by V_{ref_p} turns on and the nFET mirror starts conducting. The voltage value at which V_{az} is clamped in the upswing, V_{up}, is set by the threshold voltage of the pFET and by V_{ref_p}. When V_{az} is between the two clamping voltages, the final output of the spatial-derivative is zero; thus, for differential input voltages $\Delta V = V_{i+1} - V_{i-1}$ such that $V_{down} < V_{az} < V_{up}$, the circuit fails to compute the correct spatial-derivative. To avoid this dead-zone in the transfer characteristics, it is necessary to set the bias voltages V_{ref_n} and V_{ref_p} so that the dead-zone is at a minimum. In Fig. 5.2 we plot different transfer characteristics obtained from the circuit as a function of the value of $\Delta V_{AZ} = V_{up} - V_{down}$.

Let us now consider the behavior of the circuit with the auto-zeroing, floating-gate amplifier [16]. The auto-zeroing, floating-gate amplifier is a simple two transistor amplifier stage (transistors M1 and M2 in Fig. 5.1) that has the ability to adapt its steady state output voltage to lie at a value determined largely by fabrication pa-

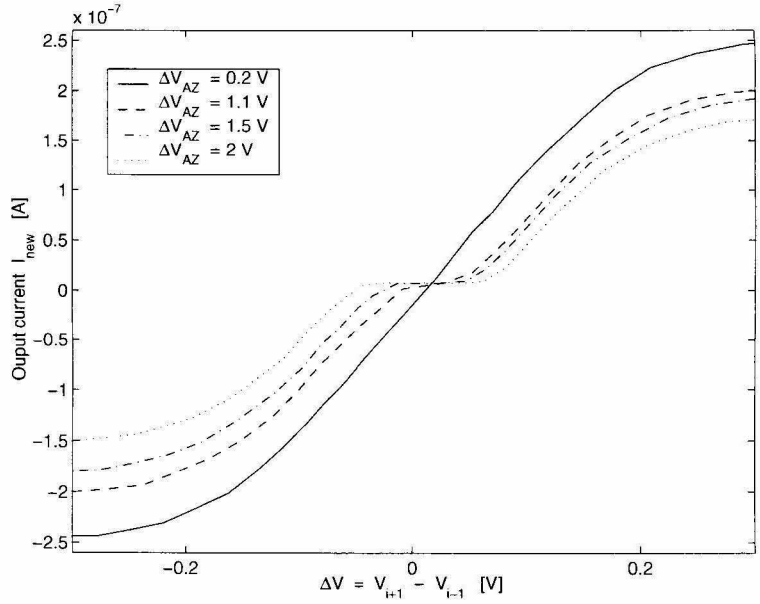

Figure 5.2: Plot of four different transfer characteristics obtained from the new spatial-derivative circuit for different values of $\Delta V_{AZ} = V_{up} - V_{down}$. A careful selection of the bias voltages V_{ref_n} and V_{ref_p} permits a transfer characteristic without a "dead-zone".

rameters and global circuit variables and minimally by the individual signal levels. This adaptation is performed by modifying the charge on the floating-gate of the pFET transistor. Electrons are removed from the floating-gate by means of Fowler-Nordheim tunneling and added by pFET hot-electron injection [10]. The steady-state output voltage is kept nearly constant by changing the charge on the floating-gate. The amplifier reaches equilibrium when the tunneling current equals the injection current. The hot-electron injection current increases linearly with the source current in the pFET and exponentially with V_{ds}, while the tunneling current increases exponentially with the voltage across the gate oxide ($V_{tunn} - V_{fg}$). The tunneling process tends to turn the pFET transistor off and the injection process tends to turn the transistor on. Because the output of the amplifier directly controls V_{ds}, the amplifier provides a high-gain, negative feedback signal which drives the system to equilibrium. By modifying V_{tunn} and V_{inj}, the steady-state output voltage and the rate of adaptation can be controlled.

Fabrication mismatch is present in any CMOS processes and this usually translates

into reduced precision for the circuits that are affected. In the case of the spatial-derivative circuit, if the chip is placed under uniform illumination and the steady-state values of two photoreceptors V_{i-1} and V_{i+1} differ by just a few thermal voltage units ($V_T = kT/q = 25$ mV at room temperature), the output of the amplifier V_{az} will be forced to one of the two clamping voltages. This causes a non-zero value for the spatial-derivative current I_{new} when the desired output value is zero. Using the auto-zeroing floating-gate amplifier, we cancel much of the error caused by fabrication mismatch because the amplifier will counter offsets and drive the output to a known voltage level. Then, in order to obtain a balanced output transfer characteristic of the spatial-derivative circuit, we just need to choose appropriate values for V_{ref_n} and V_{ref_p} such that V_{down} and V_{up} are symmetric with respect to its steady-state value. When the spatial-derivative circuit is used in arrays, it is also necessary that the difference $\Delta V_{AZ} = V_{up} - V_{down}$ matches the amount of variation expected from the statistics of the auto-zeroing amplifiers' equilibrium points. It is worth noticing that in the case of the spatial-derivative circuit, the auto-zeroing amplifier cancels the effects of offset mismatch from both photoreceptors and the spatial-derivative circuit.

5.2 Test Results

To test the new circuit, we fabricated a chip with an array of 26 photoreceptors connected to 25 new, auto-zeroing spatial-derivative circuits and 25 old, non-adaptive spatial-derivative circuits. We used the same photoreceptor used in the sensors *Detector1* and *Detector2*. To perform a fair comparison, all the transistor sizes of the differential-pair circuits and the stacked current mirrors were kept the same in the two designs. The circuit was fabricated in a 1.2 μm double-poly, double-metal, n-well CMOS process.

We performed the experiments by focusing a uniform stimulus onto the chip (i.e., a white screen illuminated by diffuse light). We measured the output of the array of photoreceptors over three orders of magnitude of uniform light conditions. Fig. 5.3 shows that the photoreceptor offsets are perfectly conserved across three orders of

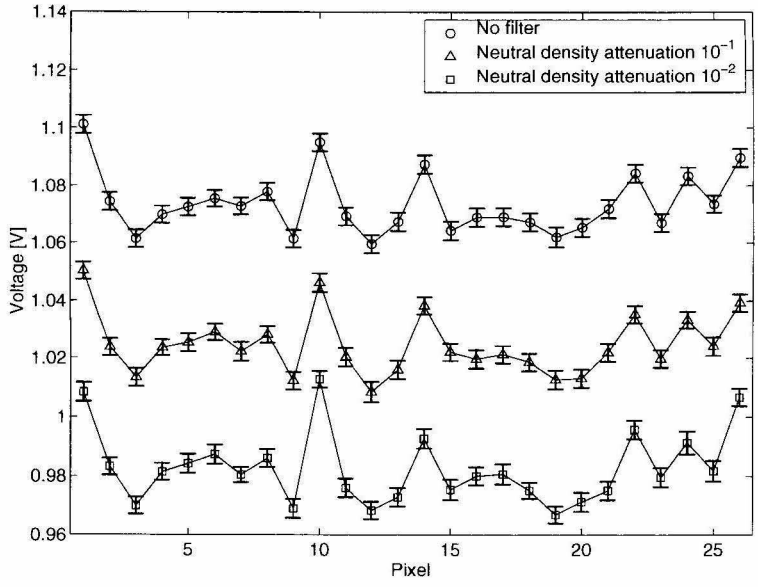

Figure 5.3: Voltage output of the array of photoreceptors under uniform diffuse illumination at different light intensities.

magnitude of light intensities and that there are cases where pairs of adjacent photoreceptors have a difference greater than a thermal voltage V_T. We then measured the ability of the auto-zeroing spatial-derivative circuit to adapt and remove the offsets.

In Fig. 5.4 we report the measurements of the non-adaptive spatial-derivative circuit. The output is not perfectly flat as we might expect for a uniform white stimulus. All the current measurements were performed using a current-sense amplifier (i.e., recorded as voltage) and the reported values were obtained by numerically converting voltage back to current. The calculated standard deviation of the current offset for this circuit was 7.8 nA which, compared to a dynamic range of 340 nA, gives a "resolution" of 4.4 bits (for dynamic range we intend the difference between the positive and negative saturation values of the transfer characteristic).

In our first series of tests of the auto-zeroing spatial-derivative circuit, we raised the tunneling and injection voltages and we let the array of auto-zeroing amplifiers adapt to their equilibrium point. By varying both the tunneling and injection voltages, the time-constant of floating-gate circuits like the one in Fig. 5.1 can be adjusted from

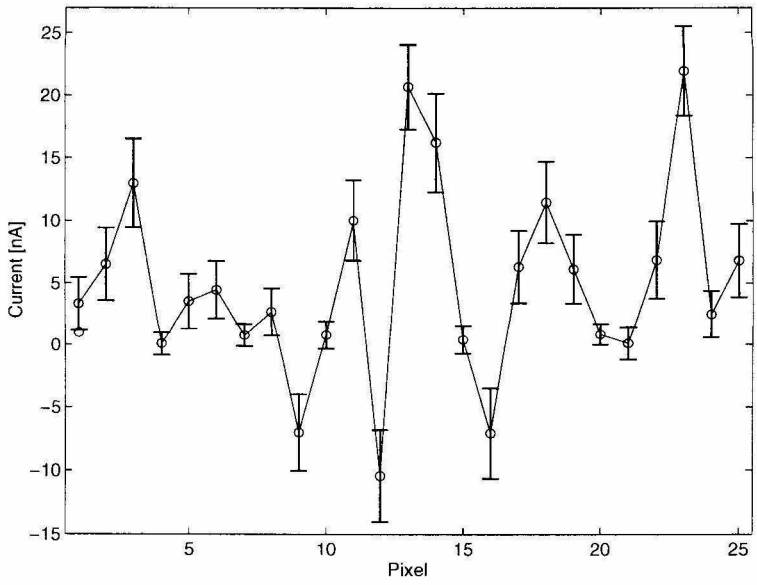

Figure 5.4: Output current of the old spatial-derivative circuit under uniform diffuse illumination.

values of seconds to thousands of seconds [9]. In our experiments the voltage range for the tunneling voltage, V_{tunn}, was between 25 and 30 V and the range for the injection voltage, V_{inj}, was from 7.5 V to 8.5 V. All the results reported here were obtained using a tunneling voltage of 26 V and an injection voltage of 7.7 V. The tunneling and injection voltages are much lower in modern submicron processes. In Fig. 5.5 we report the output of the auto-zeroing spatial-derivative circuit both before we started the adaptation process and after the equilibrium was reached. The effect of the auto-zeroing amplifier is a dramatic reduction of the peak value of the offset by a factor of 20. The calculated standard deviation of the offset after adaptation is about 1.2 nA. We can now compare in Fig. 5.6 the offset after the adaptation process with the constant offset of the non-adaptive circuit. The current offset of the auto-zeroing circuit is sensibly lower than the one of the non-adaptive circuit, the peak of the new circuit is about one order of magnitude lower than the peak of the old circuit and the standard deviation ratio is about 6 to 1 in favor of the new scheme. Considering that the dynamic range of the auto-zeroing circuit was 235 nA, the corresponding resolution was about 6.4 bits compared to the 4.4 bits of the non-adaptive circuit.

It is possible to obtain even better results if we use only the injection mechanism

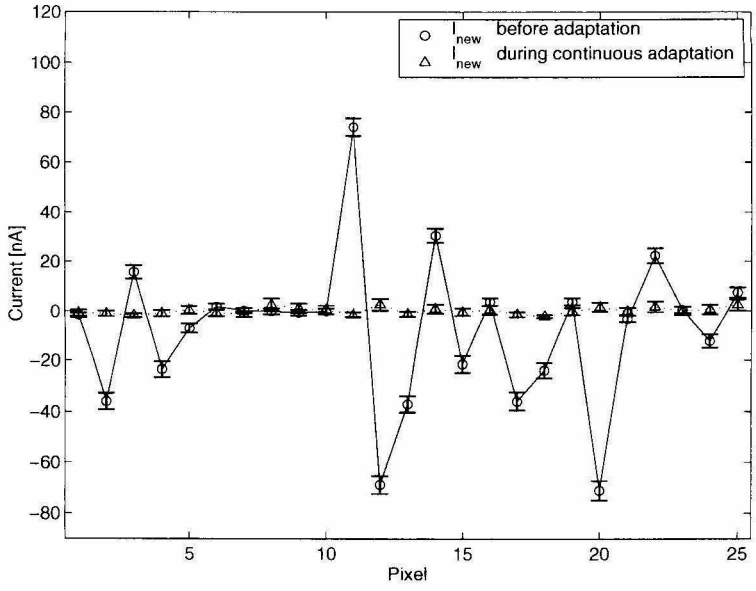

Figure 5.5: Comparison of the output current of the new spatial-derivative circuit before and during continuous time adaptation.

in the auto-zeroing amplifier. We will call this procedure "one-time" adaptation because, contrary to the continuous time adaptation described before, the adaptation is performed only once for every array of spatial-derivative circuits. In this case, after setting the injection voltage to the appropriate value, the voltage V_{cntl} is increased (thus raising the floating-gate and reducing the current in the pFET transistor) so the output of the floating-gate amplifiers drops to a lower voltage. In this situation, the injection process becomes active and adds electrons onto the floating-gate until the pFET transistor drives the output of the amplifier high enough to turn the injection process off. With this procedure it was possible to further reduce the offsets of the spatial-derivative circuit, as shown in Fig. 5.7 where we compare the final offsets obtained with the two different methods. Even more significant are the benefits of this procedure if we compare the remaining offsets after the one-time adaptation procedure, with the offset of the non-adaptive circuit in Fig. 5.8. The calculated standard deviation of the offset noise obtained with the one-time adaptation was 0.3 nA, a factor of 26 smaller than the offset noise in the non-adaptive circuit. The computed resolution for this case was 8.5 bits with a gain of four bits with respect

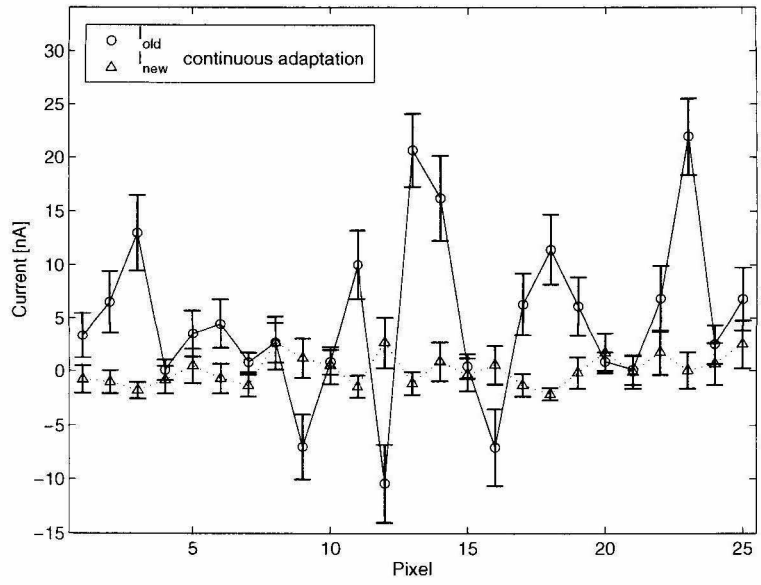

Figure 5.6: Comparison of the output current of the old spatial-derivative circuit and the new circuit during continuous time adaptation.

to the non-adaptive circuit. The results for the auto-zeroing circuit were obtained by setting the biases voltages V_{ref_n} and V_{ref_p} in such a way that the dead-zone of the transfer characteristic was negligible with respect its linear range. Finally, in Fig. 5.9 we compare the output of the floating-gate amplifiers before and after the one-time adaptation. One drawback of this procedure is that the adaptation is performed only once and then unless tunneling is resumed the electrons are permanently stored on the floating-gates. Another important point is that, in the one-time adaptation case, the biasing of the pFET transistor of the floating-gate amplifier becomes critical to the correct functioning of the circuit. Temperature shifts could change the bias condition and therefore change the equilibrium point of the amplifier output. Temperature compensation could be possible by controlling the temperature dependence of the bias current in the differential-pair to match the dependence in the pFET transistor current.

Indirect evidence suggests that the reason the one-time adaptation is more accurate than the continuous adaptation is because the injection process is better matched across the chip than is the tunneling process. Consequently, turning off the tunneling reduces the errors.

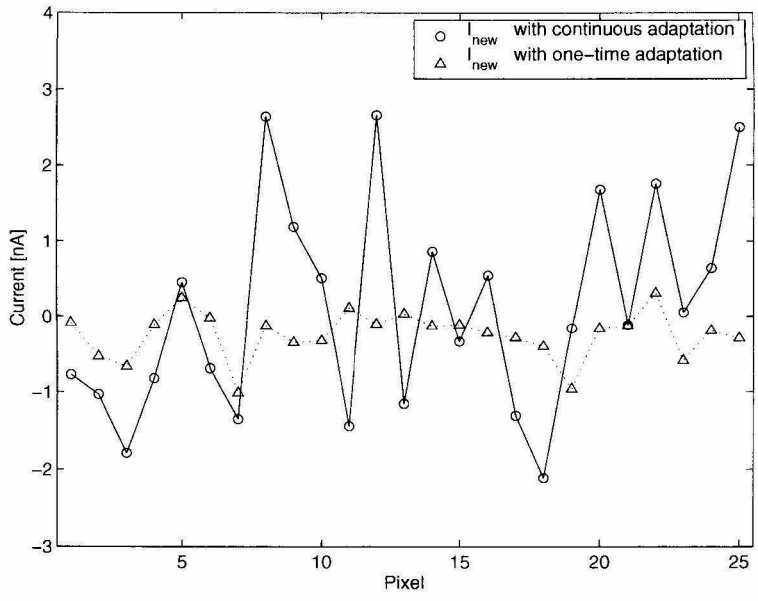

Figure 5.7: Comparison of the output current of the new spatial-derivative circuit during continuous time adaptation and after one-time adaptation.

In the injection-only case, once the pFET has largely balanced the input current, V_{ds} is reduced (output rises) until the injection shuts itself off. Since there is no tunneling current to balance, the equilibrium voltage only depends on the transistor matching. The actual gain of the injection process (which *is* a function of input current) only affects the rate at which the equilibrium is approached.

5.3 Conclusion

In this paper we have presented a circuit for auto-zeroing a current signal as applied to a visual processing task. First, the array of offset-ridden current signals was balanced by a floating-gate auto-zeroing amplifier. Second, adjustable thresholds were introduced to prevent any remaining offsets from appearing at the output. The adaptation was achieved by adding a floating-gate amplifier and one extra circuit parameter. In comparison to previous designs, offset "noise" was reduced by more than an order of magnitude. It should be noted that the technique of offset correction *after* the differential-pair, while zeroing the final output, does not correct the imbal-

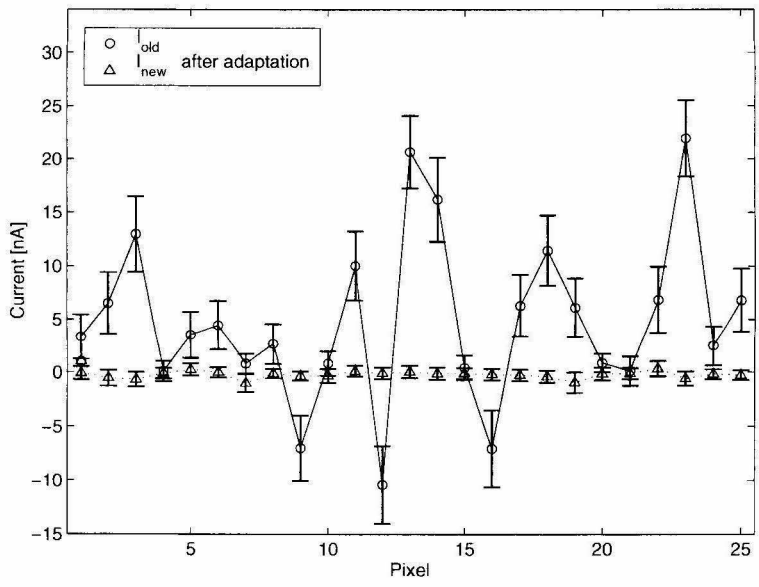

Figure 5.8: Comparison of the output current of the old spatial-derivative circuit and the new circuit after one-time time adaptation.

ance produced by the offsets. Such an imbalance results in an asymmetrical output characteristic.

We also demonstrated two different strategies for adaptation, a one-time (injection-only) calibration routine and a continuously-adapting strategy (tunneling and injection). While the one-time calibration strategy provides a lower offset error after adaptation, we are most interested in the use of continuous calibration for systems that require very long periods of operation without intervention. As imaging systems are in operation over long periods of time and are exposed to the environment, persistent offsets from dirty optics or circuit failure can increasingly impair performance.

While one technique for reducing the effect of fabrication offsets is to increase the size of all of the transistors or improve the fabrication process, another is to make the circuit layout very small and utilize an adaptive system. In the chip we presented, all of the transistors were $6\lambda \times 6\lambda$, leaving room for further reduction in layout area in future designs. While both approaches to offset reduction are valid, the adaptive approach is attractive due to the potential for ignoring bad pixels and its ability to compensate for unforeseen changes in the system over time.

It should be noted that the work presented here is different from other work in the

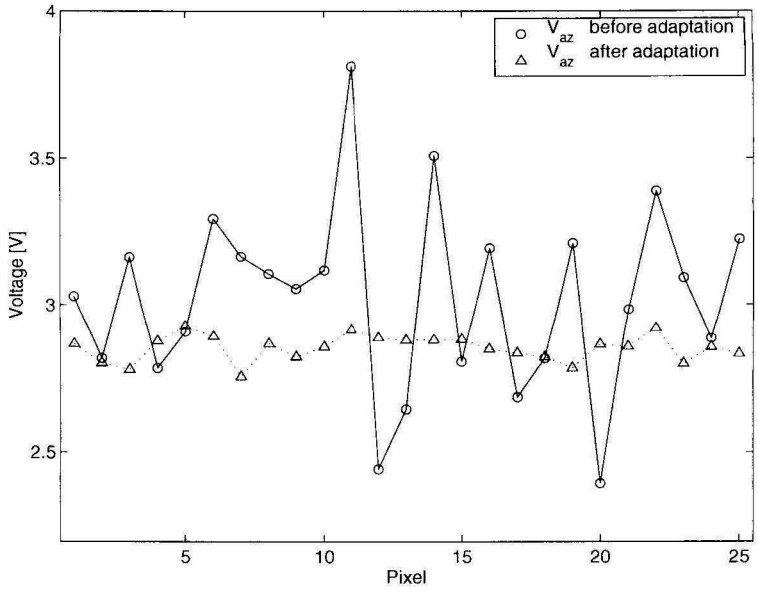

Figure 5.9: Comparison of the output of the amplifiers' V_{az} before and after one-time adaptation.

literature [8, 1], in that the sensor's output is not an image to be used by a downstream computer, rather it is intended to be used in a fully-integrated computational sensor [18] or in a larger system that requires pre-processed data. It is for this reason that adaptive photoreceptors are used and the spatial-derivative is used to calibrate the system rather than the image intensity. This adaptation strategy, however, does make the assumption that the visual world the sensor experiences has zero-mean spatial-derivative statistics over a time-interval comparable to the adaptation time-constant. For an autonomous, mobile visual system viewing natural scenes, the zero-mean assumption of the continuous adaptation approach is likely to be reasonable.

The brain has long been an inspiration to engineers for reasons of both computational ability and adaptability; however, attempts to mimic even the smallest portions of it have fallen surprisingly short. While early attempts to build neural circuits used small numbers of discrete components, recent approaches have utilized VLSI technology. Neuromorphic analog VLSI chips [11], while space and power efficient, have often been criticized for their lack of precision and lack of realistic memory structures. The recent surge in development of non-volatile analog parameter storage on silicon and

the rapid growth of knowledge in neuroscience (where memory and computation are inextricably intermingled), however, have made neuromorphic analog VLSI systems a viable technology for designing tomorrow's extremely-low-power, smart sensors and systems.

Chapter 6 Conclusions

In this dissertation, we presented a series of computational sensors that perform focal-plane computation for feature detection.

The main contributions of this dissertation may be the fact that we were able to identify a fundamental task for the computer vision or image processing communities that, if successfully implemented in a computational sensor, could provide a real advantage over the standard combination of CCD camera and dedicated digital image processor.

The choice of the feature selection task was particularly fortunate because, at the same time, it is both a fundamental topic for the field and offers a considerable challenge to the designer.

The fact that feature selection is an important and established research topic in the computer vision and image processing community relieved ourselves from the burden of actually trying to explain and justify why we were interested in building such a sensor. We feel that many of the previous sensor designs, unfortunately, were not in the same fortunate situation.

The fact that it presented a real challenge for a successful implementation in a CMOS sensor provided the opportunity to show that, with deep understanding of the problems, careful design, fortunate intuitions, and, sometimes, just sheer luck in finding the right circuits at the right time, it is still possible to implement very complicated algorithms using CMOS circuits, working in subthreshold, without having to *go digital* as soon as possible.

While the previous two considerations were, more or less, on a personal level, we can evaluate, in a engineering sense, what we set of to accomplish and what we were able to show as a result of four years of work.

Indeed we showed that, thanks to a careful design, it is possible to implement a very complex feature detection algorithm in a CMOS visual sensor with focal plane

computation. We showed that, at least in simulations, the algorithm can be implemented without approximations thanks to the very elegant circuit choices adopted. We realized that, for sensor utilizing transistors working in subthreshold, mismatches play a significant role in the accuracy that can be achieved. To counter that, we presented one of the first implementations, and the first implementation to focal-plane arrays, of floating-gate technology to lessen the effects of mismatches in analog CMOS circuits. We are confident that, with the progress of silicon technology, both in terms of smaller transistor sizes and better matching, and a more frequent use of floating-gate technology, it will be possible, in the near future, to redesign one of the sensors presented and obtain far better results in terms of accuracy and consistency of the computations across the pixels array.

Looking at the bigger picture, we have to realize that the claim to fame of these computational sensors is that they can perform very complex computations with a significant reduction of power consumption with respect to their digital implementation counterparts. While this thesis can be considered a step in that direction, we probably need to consider if, with all the added design and testing time and the increasing efficiency of low power digital design, all this is really worth it. It is debatable, in fact, if the small gain in power savings of an analog sensor that guide the navigation along a building corridor of a 20 Watts roving robot or aids the parking of a 250 hp (186 KWatts) Mercedes can be really considered a determining factor. Of course, there is the very large market of portable/wearable battery-powered devices (PDA, hearing aids, implantable electronic devices, etc.) that, by nature, is very power sensitive. Even here, though, if we look carefully, the low power digital implementations are gaining ground with respect to the analog ones. If we are very pessimistic (or very optimistic), we can still find the once-in-a-lifetime super-low-power mission to Mars or the occasional micro-fly project were the integralistic subthreshold CMOS view can find receptive hears, but we probably would not want to try to make a living out of those opportunities.

We personally think the truth is, as often, somewhere in the middle. Presented with a new problem, the engineer should weigh all the alternatives and settle for

the solution that optimally matches the required specifications in terms of power consumption, design time, accuracy, cost of manufacturing, etc. We do not have any problem if, more often that not, this means that a digital implementation, obtained automatically from some VHDL code, wins over the analog one. We personally think that CMOS analog design is not a dead art; it just needs to be used in the right context.

The world out there is still analog after all, and every machine that interacts with it will always have to deal with analog quantities.

Appendix A A Family of Wide Linear Range Multipliers

Four quadrant analog multipliers are important building blocks for a large number of signal processing applications such as correlators, convolvers, adaptive filters, frequency doublers and modulators, etc. Many CMOS analog multipliers which exploited the MOS transistor operating in the above-threshold region have been presented in the literature. However, most of them are not suitable for the application in Neuromorphic aVLSI chips [33, 32], Artificial Neural Network application and for low power analog signal processing for portable applications where a very low power consumption is required. In these cases it is desirable to develop MOS building blocks that operate the transistors in the subthreshold region (weak inversion). Subthreshold operation has the advantage that the current levels are typically orders of magnitude lower than devices biased above threshold (strong inversion).

There are many different approaches for implementing four quadrant analog multiplier in CMOS technology: some of them, including the one here presented, use modified versions of the Gilbert cell [13] that was originally implemented with bipolar transistors. Others are based on the quarter-square algebraic identity that can be realized easily using the square law characteristics of the MOS device working above threshold. Then other multipliers based on the pulse-width modulation technique, switched capacitors technique, etc, have been reported.

Many of these designs using the MOS transistor above threshold report linear range up to few volts. It is fairly easy to obtain wide linear range above threshold due to the square law voltage-current characteristic of the MOS device. In the subthreshold region the characteristic is exponential; therefore, it is more difficult to obtain a comparable linear range.

To the best of our knowledge all the reported designs of four quadrant multiplier

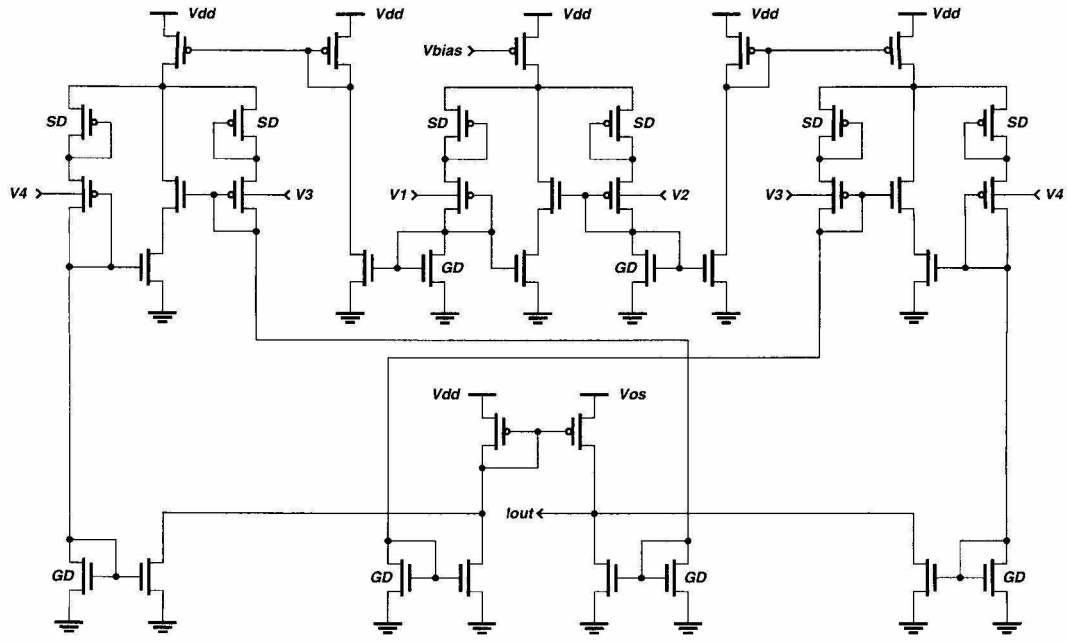

Figure A.1: The wide linear range four quadrant multiplier. The inputs are through the well of the transistors WI, V_1 and V_2 in the middle differential pair and V_3 and V_4 in the two outer differential pairs. The transistors SD reduce the transconductance of the differential pairs through source degeneration. The transistors GDM further reduce the transconductance through gate degeneration and mirror the current to the other differential pairs or to the output. The transistors M are used in the current mirrors of the circuit. The transistors B further linearize the transfer function through bump linearization. The current output is I_{out}, the voltage V_{bias} set the bias current of the multiplier and V_{os} allows fine adjustment of the its offset if necessary.

working in subthreshold have a linear range that does not exceed ± 100 mV [32, 27, 28, 6] while our design exhibits a linear range of ± 2 V. To achieve this value a combination of four different techniques is used. First the well terminals of the input transistors are used as low transconductance inputs. Then, feedback techniques known as source degeneration and gate degeneration provide further reduction of the transconductance. Finally a technique known as bump-linearization extends the linear range even further. The circuit that incorporates all four techniques is shown in Figure A.1.

In section 2 we describe the principle of operation and the design of the multiplier. Section 3 shows the results of the measurements, and in section 4 we conclude by summarizing our contributions.

A.1 Principle of Operation

To understand the principles of operation of the multiplier, we first analyze the characteristics of the single leg of the three differential pairs present in the multiplier. The circuit is shown in Figure A.2(a).

In a subthreshold (p-type) MOS transistor, the current is given by [33]

$$I_{ds} = I_0\, e^{-\frac{\kappa V_{gs}}{V_T} - \frac{(1-\kappa)V_{ws}}{V_T}} (1 - e^{\frac{V_{ds}}{V_T}} - V_{ds}/V_0)\ , \tag{A.1}$$

where V_{gs}, V_{ds} and V_{ws} are the gate-to-source, drain-to-source and the well-to-source potential respectively, I_0 is the zero-bias current for the given device, $V_T = kT/q$ is the thermal voltage, V_0 is the Early voltage and κ measure the effectiveness of the gate potential in controlling the channel current.

A.1.1 The Well Inputs

For a device in saturation ($V_{ds} \geq 4V_T$) and neglecting the Early effect, equation (A.1) simplifies to

$$I_{ds} = I_0\, e^{-\frac{\kappa V_{gs}}{V_T} - \frac{(1-\kappa)V_{ws}}{V_T}}\ . \tag{A.2}$$

By differentiating equation (A.2) we obtain the gate and well transconductances:

$$g_g = \frac{\partial I_{ds}}{\partial V_g} = -\kappa \frac{I_{ds}}{V_T}, \qquad g_w = \frac{\partial I_{ds}}{\partial V_w} = -(1-\kappa)\frac{I_{ds}}{V_T}\ . \tag{A.3}$$

From equation (A.3) it is clear that if $\kappa > 0.5$, which is almost always the case, the well transconductance has a lower value than the gate transconductance and thus is preferable over the gate as a low-transconductance input.

A.1.2 Source and Gate Degeneration

The technique known as source degeneration was first used in vacuum-tube design, where is was called chatode degeneration, as well as in bipolar design, where it is

referred to as emitter degeneration. The principle is to convert the current of the MOS transistor into a voltage using a resistor, a diode or a diode-connected transistor, and then feed this voltage back to the emitter or source of the transistor to decrease its current.

The technique of gate degeneration was first used by Sarpeshkar et al. [40]. It is based on the same principle of converting the current passing through the transistor into a voltage that is then fed back to the gate of the transistor decreasing in this way its current.

It is straightforward to quantify the reduction of transconductance obtained through the source and gate degeneration. The current through the transistor SD can be expressed as

$$I = I_0\, e^{-\frac{\kappa_p V_s}{V_T} + \frac{V}{V_T}} \quad \text{or} \quad e^{\frac{V_s}{V_T}} = e^{\frac{V}{\kappa_p V_T}} \left(\frac{I}{I_0}\right)^{-\frac{1}{\kappa_p}} . \tag{A.4}$$

The current of transistor GDM is

$$I = I_0\, e^{\frac{\kappa_n V_g}{V_T}} \quad \text{or} \quad e^{-\frac{\kappa V_g}{V_T}} = \left(\frac{I}{I_0}\right)^{-\frac{\kappa}{\kappa_n}} . \tag{A.5}$$

Now rewriting equation (A.2) for the input transistor WI, we have

$$I = I_0\, e^{-\frac{\kappa V_g}{V_T} - \frac{(1-\kappa)V_w}{V_T} + \frac{V_s}{V_T}} . \tag{A.6}$$

Using equations (A.4)-(A.5) we can rewrite equation (A.6) as

$$\frac{I}{I_0} = \left(\frac{I}{I_0}\right)^{-\frac{\kappa}{\kappa_n}} e^{-\frac{(1-\kappa)V_w}{V_T}} \left(\frac{I}{I_0}\right)^{-\frac{1}{\kappa_p}} e^{\frac{V}{\kappa_p V_T}} ,$$

and solving for I we finally obtain

$$I = I_0\, e^{-\frac{\kappa_{eff}}{V_T}\left(V_w - \frac{V}{\kappa_p(1-\kappa)}\right)} \tag{A.7}$$

where

$$\kappa_{eff} = \frac{1-\kappa}{1 + \frac{1}{\kappa_p} + \frac{\kappa}{\kappa_n}} . \tag{A.8}$$

That is to say, due to the source and gate degeneration, the three transistors of Figure

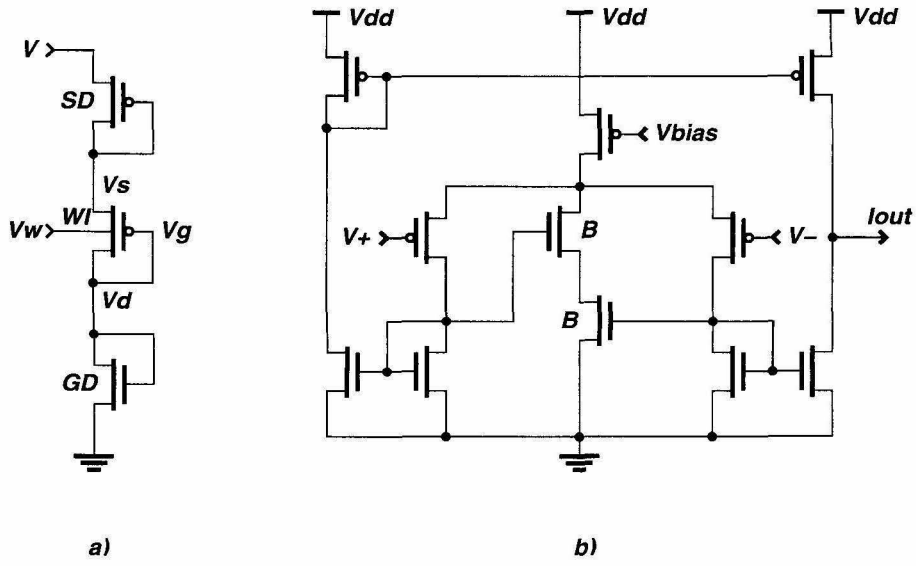

Figure A.2: **(a)** One leg of the differential pair used in the multiplier of Figure A.1. The transistor WI use the well terminal as input to reduce the transconductance. The diode-connected transistor SD is responsible for the source degeneration while the transistor GDM further reduce the transconductance through gate degeneration and also mirror the current. **(b)** The simple bump transconductance amplifier. For a low value of the differential input voltage $V = V_+ - V_-$ the bump transistors B steal current to the other legs of the differential pair linearizing in this way the transfer function.

A.2(a) are equivalent to a single transistor with well transconductance of

$$\hat{g}_w = \frac{\partial I}{\partial V_w} = -\kappa_{eff} \frac{I}{V_T}$$

that is a lower value with respect to equation (A.3).

A.1.3 Bump Linearization

Bump linearization is a technique to extend the linear range of a subthreshold differential pair [7].

A bump differential pair, Figure A.2(b), has two series-connected transistors in a middle leg in addition to its outer two legs. These transistors are called *bump* because their I-V characteristics is a bump-shaped function of the differential voltage $\Delta V = V_+ - V_-$.

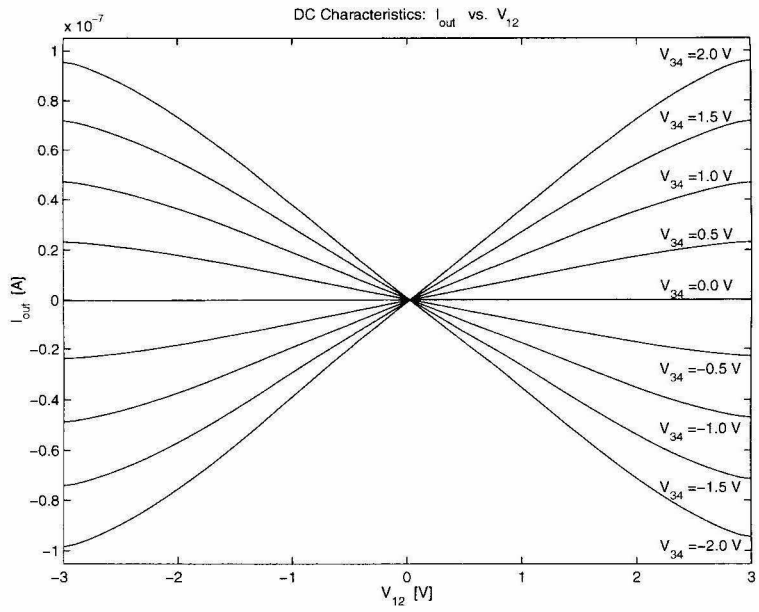

Figure A.3: DC transfer characteristics ($V_{12} = V_1 - V_2$ sweeping in the range -3.0 – 3.0 V and $V_{34} = V_3 - V_4$ at fixed values, common mode voltage for both differential input V_{CM}=2.5 V).

The overall I-V curve of the differential pair is the usual tanh-shaped function of ΔV except for a region near the origin where the bump transistors are in conduction. If we refer to the W/L ratio for a transistor as the *strength* of the transistor, then we can control the I-V curve of the differential pair by changing the strength *ratio* S between the strength of the transistors in the middle leg to the strength of the transistors in the outer legs. A large value of S will cause a flat zone near the origin where a small S will not affect much the tanh-shaped transfer function.

For the simple subthreshold bump amplifier with strength ratio S, the output current can be shown to be

$$I_{out} = I_b \frac{\sinh(\kappa \Delta V / V_T)}{1 + S/2 + \cosh(\kappa \Delta V / V_T)} \,. \tag{A.9}$$

Moreover Sarpeshkar et al. [40] demonstrated that if $S \leq 2$, the I-V curve of the amplifier has no point of inflection except at the origin and that for $S = 2$ the Taylor expansion of equation (A.9) doesn't contain a cubic distortion term unlike the Taylor expansion of tanh.

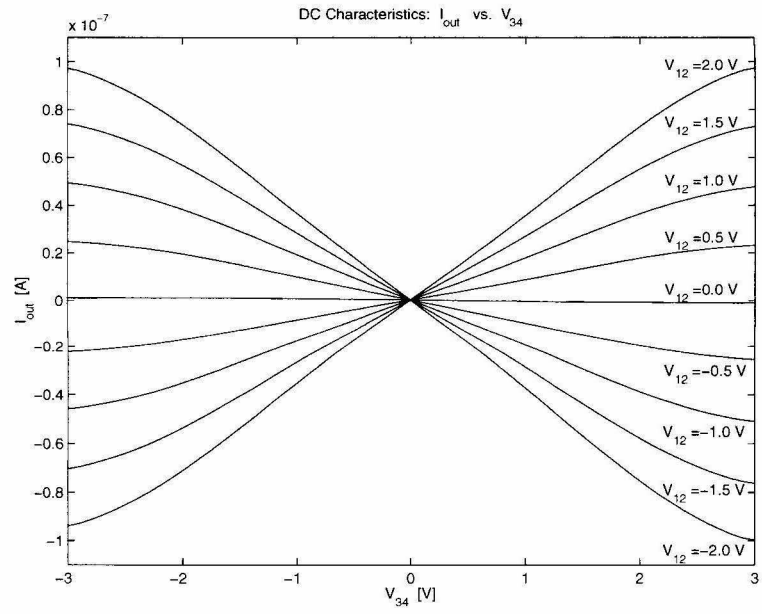

Figure A.4: DC transfer characteristics ($V_{34} = V_3 - V_4$ sweeping in the range -3.0 – 3.0 V and $V_{12} = V_1 - V_2$ at fixed values, common mode voltage for both differential input V_{CM}=2.5 V).

To obtain the output current of the subthreshold bump amplifier with well inputs and source and gate degeneration, it is sufficient to replace κ of equation (A.9) with κ_{eff} of equation (A.8) as demonstrated in section A.1.2

A.2 Experimental Results

Assuming as ideal all the current mirrors and using the results of equation (A.8)-(A.9) it is easy to show that the overall transfer function of the multiplier is

$$I_{out} = I_b \frac{\sinh(\Delta_{12})}{1 + S/2 + \cosh(\Delta_{12})} \frac{\sinh(\Delta_{34})}{1 + S/2 + \cosh(\Delta_{34})} \tag{A.10}$$

where

$$\Delta_{12} = \frac{\kappa_{eff}(V_1 - V_2)}{V_T} \quad \text{and} \quad \Delta_{34} = \frac{\kappa_{eff}(V_3 - V_4)}{V_T} .$$

In equation (A.10), I_b is the bias current set with the voltage V_{bias} and S is the strength ratio between the transistors of the inner and outer legs respectively.

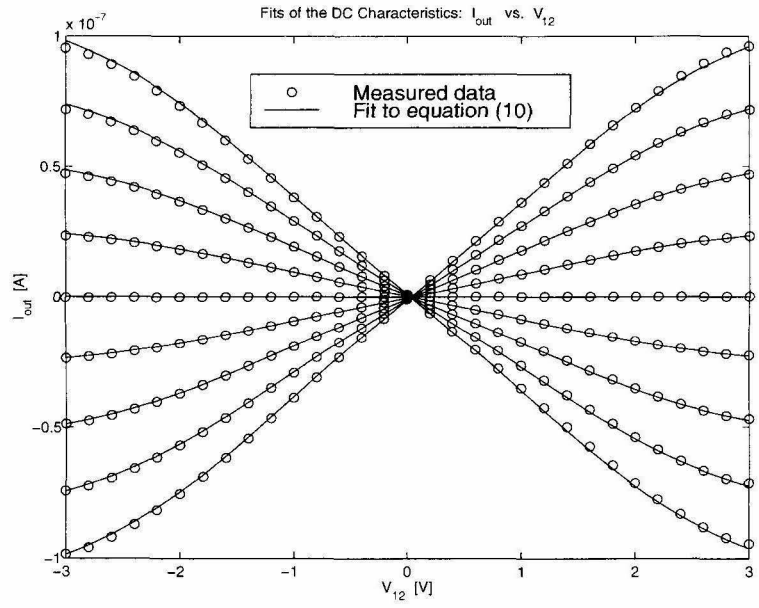

Figure A.5: Fit of the experimental data of the DC characteristics I_{out} vs. V_{12} to equation (A.10).

We designed the multiplier with a strength ratio $S = 2$ to verify the correctness of the theoretical analysis. We used a 1.2 μm double-poly double-metal n-well CMOS process available through the MOSIS fabrication service, and we obtained data from it. Figures A.3 and A.4 show the DC characteristics of the multipliers. As we can see the linear range extends between ± 2 V. Figure A.5 shows the fit of equation (A.10) performed over the data of Figure A.3. As we can verify from the figure, equation (A.10) perfectly describes the DC characteristics of the multiplier.

We also measured the Total Harmonic Distortion (THD) with respect to both inputs: we imposed a 2 V_{pp} sinusoid (Freq. 1 Khz) at one differential input and 2 V DC at the other input. With the sinusoidal input applied to V_{12} and $V_{34} = 2.0$ V, the distortion was 2.8%, and when the sinusoidal input was applied to V_{34} with $V_{12} = 2.0$ V we measured a distortion of 3.0%. Figure A.6 shows the classic example of a frequency doubling operation with the multiplier.

The power consumption is very low and depends on the bias current set by the voltage V_{bias} and on the input values. It is possible to derive an analytical expression for the power consumption as a function of the inputs. Here we just say that the

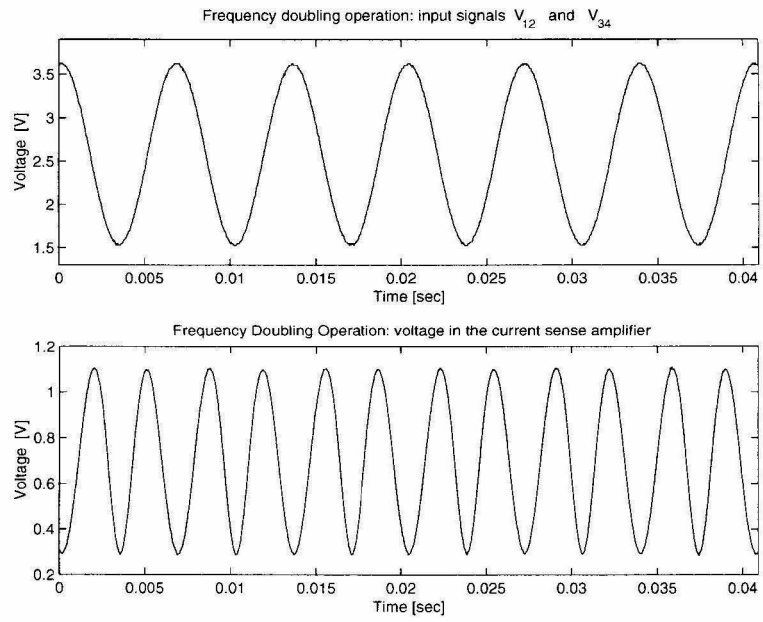

Figure A.6: Example of frequency doubling. $V_{12} = V_{34}$ =2.0 V_{pp}, Freq. 100Hz.

power consumption P is always P< $5\,V_{dd}\,I_{bias}$, therefore, for example, for a value of $I_{bias} = 40$ nA and with $V_{dd} = 5$ V, the power is less than 1 μW.

The bandwidth of the multiplier depends on the bias current and the amplitude of the two differential inputs. This multiplier is not suitable for medium or high speed systems due to the extremely low value of the currents involved. With a bias current of 80 nA, we measured a -3dB bandwidth of about 10 KHz with a peak-to-peak input of 2 V.

A.3 Conclusion

We presented a novel design of an analog four quadrant multiplier in subthreshold CMOS. Special techniques allow a linear range of up to ± 2 V. The power consumption depends on the bias current, but with normal subthreshold current levels, a power consumption of less than 1 μW is possible. This circuit is suitable for low power analog signal processing applications and for Neuromorphic aVLSI chips [33, 32] where the 100 mV linear range of the usual subthreshold four quadrant multipliers is too small.

Bibliography

[1] A. Aslam-Siddiqi, W. Brockherde, M. Schanz, and B. J. Hosticka. A 128-pixel CMOS image sensor with integrated analog nonvolatile memory. *IEEE Journal of Solid-State Circuits*, 33(10):1497–1501, 1998.

[2] A. Benedetti and P. Perona. Real-time 2-d feature detection on a reconfigurable computer. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '98)*, 1998.

[3] L. R. Carley. Trimming analog circuits using floating-gate analog MOS memory. *IEEE J. Solid State Circ.*, 24(6):1569–1575, 1989.

[4] H. Castro, S. Tam, and M. Holler. Implementation and performance of an analog non-volatile neural-network. *Analog Integrated Circuit and Signal Processing*, 4(2):97–113, 1993.

[5] G. Cauwenberghs, C. F. Neugebauer, and A. Yariv. Analysis and verification of an analog vlsi incremental outer-product learning system. *IEEE Trans. on Neural Networks*, 3(3):488–497, 1992.

[6] D. Coué and G. Wilson. A four-quadrant subthreshold mode multiplier for analog neural-network applications. *IEEE Trans. Neural Networks*, 7:1212–1219, 1996.

[7] T. Delbrück and C. A. Mead. Bump circuits for computing similarity and dissimilarity of analog voltages. CNS Memo 26, California Institute of Technology, 1993.

[8] F. Devos, M. Zhang, Y. Ni, and J. F. Pone. Trimming CMOS smart imager with tunnel-effect nonvolatile analog memory. *Electronic Letters*, 29(20):1766–1767, 1993.

[9] C. Diorio. *Neurally Inspired Silicon Learning: From Synapse Transistors to Learning Arrays.* PhD thesis, Electrical Engineering, California Institute of Technology, 1997.

[10] C. Diorio, P. Hasler, B. Minch, and C. Mead. A complementary pair of four-terminal silicon synapses. *Analog Integrated Circuits and Signal Processing*, 13(1/2):153–166, 1997.

[11] R. Douglas, M. Mahowald, and C. Mead. Neuromorphic analogue VLSI. In W. M. Cowan, E. M. Shooter, C. F. Stevens, and R. F. Thompson, editors, *Annual Reviews in Neuroscience, Vol. 18*, pages 255–281. Annual Reviews Inc., Palo Alto, CA, 1995.

[12] L. Dreschler and H. H. Nagel. Volumetric model and 3d trajectory of a moving car derived from monocular tv frame sequences of a street scene. In *Proceedings of the International Joint Conference on Artifical Intelligence*, Vancouver, Canada, 1981.

[13] B. Gilbert. A precise four-quadrant multiplier with subnanosecond response. *IEEE J. Solid-State Circuits*, 3:365–373, 1968.

[14] L. A. Glasser. A UV write-enabled PROM. In H. Fuchs, editor, *Chapel Hill Conference on VLSI (1985)*, pages 61–65. Computer Science Press, Rockville, MD, 1985.

[15] R. R. Harrison, P. Hasler, and B. A. Minch. Floating-gate CMOS analog memory cell array. In *Proceedings of the 1998 IEEE ISCAS Meeting*, pages 204–207, 1998. Monterey, CA.

[16] P. Hasler, B. A. Minch, C. Diorio, and C. A. Mead. An autozeroing amplifier using pfet hot-electron injection. In *Proceedings of the 1996 IEEE ISCAS Meeting*, pages 325–328, 1996. Atlanta.

[17] T. G. Morrisand T. K. Horiuchi and S. P. DeWeerth. Object-based selection within an analog vlsi visual attention system. *IEEE Trans. on Circuits and Systems II*, 45:1564–1572, 1998.

[18] T. K. Horiuchi, T. G. Morris, C. Koch, and S. P. DeWeerth. Analog VLSI circuits for attention-based visual tracking. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Processing Systems 9*, pages 706–712. MIT Press, 1997.

[19] D. A. Kerns. *Experiments in Very Large-Scale Analog Computation.* PhD thesis, Electrical Engineering, California Institute of Technology, 1993.

[20] L. Kitchen and A. Rosenfeld. Gray-level corner detection. Computer science center 887, University of Maryland, 1980.

[21] J. Kramer and G. Indiveri. Neuromorphic vision sensors and preprocessors in system applications. In *Advanced Focal Plane Arrays and Electronic Cameras (AFPAEC'98)*, Zürich, Switzerland, May 1998.

[22] J. Kramer, R. Sarpeshkar, and C. Koch. Pulse-based analog vlsi velocity sensors. *IEEE Trans. Circuits and Systems II*, 44:86–101, 1997.

[23] T. S. Lande, H. Ranjbar, M. Ismail, and Y. Berg. An analog floating-gate memory in a standard digital technology. In *Proc. 5th Intl. Conf. on Microelectronics for Neural Networks and Fuzzy Systems - MicroNeuro96*, pages 271–276, 1996. Feb. 12-14, 1996, Lausanne, Switzerland, IEEE Computer Society Press, Los Alamitos, CA.

[24] O. Landolt. *Place coding in analog VLSI - A neuromorphic approach to computation.* Kluwer Academic Publishers, 1998.

[25] J. Lazzaro, J. Wawrzynek, and A. Kramer. Systems technologies for silicon auditory models. *IEEE Micro*, 14(3):7–15, 1994.

[26] M. Lenzlinger and E. H. Snow. Fowler-nordheim tunneling into thermally grown sio_2. *Journal of Applied Physics*, 40(1):278–283, 1969.

[27] S. I. Liu and C. C. Chang. Cmos subthreshold four-quadrant multiplier based on unbalanced source-coupled pairs. *Int. J. Electronics*, 78:327–332, 1995.

[28] S. I. Liu and P. K. Chen. Low voltage cmos subthreshold four-quadrant tripler. *Analog Integrated Circuits and Signal Processing*, 11:303–307, 1996.

[29] B. D. Lucas and T. Kanade. An iterative image registration technique with application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artifical Intelligence*, Vancouver, Canada, 1981.

[30] D. Marr, T. Poggio, and S. Ullman. bandpass channels, zero-crossing, and early visual information processing. *Journal of the Optical Society of America*, 69:914–916, 1976.

[31] C. Mead. Adaptive retina. In C. Mead and M. Ismail, editors, *Analog VLSI Implementation of Neural Systems*, pages 239–246. Kluwer Academic Publishers, Boston, MA, 1989.

[32] C. A. Mead. *Analog VLSI and Neural Systems.* Addison-Wesley Publishing Co., Reading, MA, 1989.

[33] C. A. Mead. Neuromorphic electronic systems. *Proc. IEEE*, 78:1629–1636, 1990.

[34] B. A. Minch, P. Hasler, and C. Diorio. The multiple-input translinear element: A versatile circuit element. In *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, Monterey, CA, 1998.

[35] H. Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover.* PhD thesis, Stanford University, 1980.

[36] A. F. Murray, S. Churcher, A. Hamilton, A. J. Holmes, G. B. Jackson, H. M. Reekie, and R. J. Woodburn. Pulse stream VLSI neural networks. *IEEE Micro*, 14(3):29–39, 1994.

[37] J. Rehg and A. Witkin. Visual tracking with deformation models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, CA, 1991.

[38] E. Sackinger and W. Guggenbuhl. An analog trimming circuit based on a floating-gate device. *IEEE Journal of Solid State Circuits*, SC-23:1437–1440, 1988.

[39] R. Sarpeshkar, J. Kramer, G. Indiveri, and C. Koch. Analog vlsi architectures for motion processing. *Proc. IEEE*, 84:969–987, 1996.

[40] R. Sarpeshkar, R. F. Lyon, and C. A. Mead. A low-power wide-linear-range transconductance amplifier. *Analog Integrated Circuits and Signal Processing*, 13:123–151, 1997.

[41] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR '94)*, Seattle, WA, 1994.

[42] C. K. Sin, A. Kramer, V. Hu, R. R. Chu, and P. K. Ko. EEPROM as an analog storage device, with particular applications in neural networks. *IEEE Trans. on Electron Devices*, 39(6):1410–1419, 1992.

[43] S. Smith and J. Brady. Asset-2: Real-time motion segmentation and shape tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 8(17):814–820, 1995.

[44] J. Tanner and C. Mead. An integrated analog optical motion sensor. In *VLSI Signal Processing II*, pages 59–76. IEEE Press, 1986.

[45] C. E. Thorpe. *Vision navigation for robot rover.* PhD thesis, Carnegie Mellon University, 1984.

[46] C. Tomasi and T. Kanade. Detection and tracking of point features. Tech. Rep. CMU-CS-91-132 30, Carnegie Mellon University, 1991.

[47] E. A. Vittoz and X. Arrenguit. Linear-networks based on transistors. *Electronics Letters*, 35:297–299, 1994.