An Analog VLSI Motion Energy Sensor and its Applications in System Level Robotic Design

by
Sudhir Korrapati

A Thesis Submitted to the Faculty of the

Electrical and Computer Engineering Department

In Partial Fulfillment of the Requirements
For the Degree of

Master of Science

In the Graduate College

The University of Arizona

2 0 0 1

Statement by Author

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the copyright holder.

Signed: _____

Sudhir Korrapati

Approval by Thesis Director

This thesis has been approved on the date shown below:

_____ _____

Charles M. Higgins Date

Assistant Professor of Electrical and Computer Engineering

Acknowledgements

I am greatly indebted to my parents for their love and support through all my endeavors in life.
I am grateful to my advisor Chuck Higgins for his guidance, advice and encouragement throughout my work.
I am thankful to Prof. Strausfeld and Dr. John Douglass for their help in the neural modeling project.
I am thankful to Prof. Harold Parks and Prof. Jeffrey Rodriguez for serving on my thesis defense committee.
I am thankful to my lab mates: Michael Schwager for his thought provoking discussions; Sam Hill for his good company; Shaikh, Robert and Raj for creating a pleasant and friendly atmosphere in the lab to work in.

Table of Contents

TABLE OF CONTENTS—*Continued*

List of Figures

LIST OF FIGURES—*Continued*

List of Tables

Abstract

Motion detection is a very important elementary task performed on the visual input received from eyes in both vertebrates and invertebrates like insects. In this work we describe a VLSI implementation of a biologically inspired elementary motion detector. This sensor is based on the Adelson-Bergen algorithm, designed to model the response of a primate cortical complex cell. We first describe the model in detail and then explain the circuit level details of the implementation of the model. Results from the characterization of the chip are presented. Next we describe two applications based on this motion sensor. The first application is an active tracking system using the sensor. The second application is the design of a chip, RoaCh (Robot on a Chip). RoaCh is a monolithic implementation of the motion detector along with a control system to navigate a robot whose objective is to run away from moving targets surrounding it. We also describe the details of the modeling of an early visual pathway of the fly, which is thought to be involved in motion computation.

Chapter 1

Introduction

The pursuit of making an intelligent machine able to mimic a biological system has been with us since we started building engineering systems. There has been a tremendous amount of research during the past few decades to build a system as intelligent and agile as its biological counterpart. During the same time there have been a lot of advances in the field of neuroscience, leading to new insights into the way biological systems process sensory information.

Most biological systems have a brain, which is the central complex computational structure. It has a huge number of locally connected neurons performing massive parallel computations on sensory information it receives to control the entire system efficiently. The microprocessors available today perform very complex tasks such as number crunching problems, search and logic problems with great precision at remarkable speeds. However, they fail to impress us when they try performing even the simplest of the activities the brain does with ease, such as controlling navigation in a cluttered environment. Their performance only becomes worse when the complexity of the task increases, such as object tracking or recognition. Clearly there is a fundamental difference of processing between the brain and a microprocessor. There is a need for us to understand how the brain does things to build such an efficient system. During the late 1980's Carver Mead started a new paradigm of realizing neural systems in silicon integrated circuits. These chips could "see" or "hear" (Mead, 1989). This new paradigm called Neuromorphic Engineering has grown over the past few years in building analog VLSI circuits that can perform more complex tasks ranging from sensory information processing for autonomous robotics to learning.

Vision provides some of the most important sensory information on which biological systems rely heavily. The human brain has about 10^{11} neurons (Koch, 1999), and it has been observed that more of the brain is devoted to vision than to any other sensory function (Zigmond *et al.*, 1999). Vision is a very complex sensory function and is used in a variety of tasks such as motion detection, focus of expansion estimation, stereo disparity measurement, color estimation, object tracking, recognition and even more high-level tasks. It plays a crucial role not only in primates, but also in invertebrates like flies, which have only about 340,000 neurons (Strausfeld, 1976). Visual processing needed for complex tasks does not happen all at one place. As visual input is a fairly extensive amount of data, transferring it all the way up through the ascending pathways in the brain would require a lot of neurons, leading to increased size and power consumption. To deal with this, the brain breaks down the complex tasks into more elementary tasks which are performed at lower levels in the ascending pathways, and the results are passed on to the higher levels. This is an appropriate approach for engineers trying to implement neural systems on silicon. We can make integrated circuits that compute such elementary tasks and combine these for obtaining different complex behaviors.

One such elementary task performed on visual input is motion computation. Motion information is used extensively to perceive the environment around us. Some of the functions that use motion information include: (1) tracking location of moving objects; (2) egomotion (determining one's own movement); (3) warning danger of other moving things; (4) determining what the scene in front of us is like (e.g., for figure-ground detection).

Motion forms a key component even in insects. Flies heavily rely on motion information for various behaviors like gaze control, flight stabilization, deceleration, tracking (Egelhaaf *et al.*, 1988), approach or landing (Borst, 1990), and others. This is the reason that motion computation becomes one of the elegant visual tasks that can be used independently in realizing various functions. It can also be used in conjunction with other tasks like disparity in realizing systems which can be very autonomous and more like their biological counterparts.

Motion computation has been done before both in software and hardware, and there are many ways to detect visual motion. Broadly these methods can be classified into feature-tracking or token-based algorithms and intensity-based algorithms. The sections below explain the general principles of motion computation with emphasis on hardware implementations of motion computation.

1.1 Feature-Tracking Algorithms

These kind of algorithms can again be classified into two kinds based on the token or the feature they try to track.

The first kind are the spatial feature tracking algorithms. These algorithms are especially popular in software-based methods. In these algorithms, spatial features like an edge or a particular region in an image sequence are first identified. Then the immediate sequence of the image is checked for the previously identified spatial feature. This essentially is a correspondence problem-i.e., to match the previously identified spatial feature in the second image sequence. Once the correspondence is obtained, the velocity can be computed in various ways, as shown in (Barnard and Thomson, 1980; Anandan, 1989; Little *et al.*, 1988) and others. Though this method is popular in software based methods because of the discrete nature involved in processing, it has also been implemented in hardware. Etienne-Cummings *et al.* demonstrate such a sensor (Etienne-Cummings *et al.*, 1997) where they compute optic flow based on the disappearance of an edge at a pixel and its reappearance at a neighboring pixel. Similarly, Barrows describes a two dimensional optical flow measurement sensor (Barrows, 1998) based on timing the movement of a feature across the visual field.

The second kind of feature-tracking algorithms use temporal features for tracking. These algorithms look for change in intensity of the image at each pixel to compute optic flow. Hardware implementations of these algorithms typically have temporal edge detectors at their first stage which respond to an abrupt change in light intensity at that pixel with a spike/pulse. Kramer demonstrates the use of a FTI (facilitate, trigger and inhibit) algorithm using three adjacent pixels to calculate the time of travel (Kramer, 1996). Similarly, there has been a lot of work using the FS (facilitate and sample) algorithm for computing velocity (Kramer *et al.*, 1995; Higgins and Koch, 1997; Kramer *et al.*, 1997; Sarpeshkar *et al.*, 1996). Higgins *et al.* demonstrated a hardware implementation of two algorithms, ITI (inhibit, trigger and inhibit) and FTC (facilitate, trigger and compare) for computing two-dimensional local direction of motion (Higgins *et al.*, 1999).

1.2 Intensity-Based Algorithms

Intensity based algorithms are again divided as gradient based and correlation based algorithms.

Gradient based methods compute velocity from the spatial and temporal derivatives (gradients) of the image intensity. This approach for the two dimensional case was proposed by Horn and Schunck (Horn and Schunck, 1981). There have been at least two hardware realizations of this model (Tanner and Mead, 1986; Deutschmann and Koch, 1998). However, these models are very sensitive to noise.

Correlation based algorithms are by far the most successful methods realized in hardware for motion computation. In these methods, motion is computed from correlating the response of a pixel and the delayed response of its neighbor. The popular correlation based algorithms are the ones proposed by Hassenstein and Reichardt in 1956 for explaining the optomotor response in flies, by Barlow and Levick to explain direction selectivity in rabbit's retina (Barlow and Levick, 1965), and the Adelson-Bergen algorithm (Adelson and Bergen, 1985). The Adelson-Bergen model is often cited as the underlying model of a primate cortical cell (Qian *et al.*, 1994; Nowlan and Sejnowski, 1994; Heeger *et al.*, 1996). Van Santen and Sperling proposed an elaborated Reichardt detector (Van Santen and Sperling, 1985) and show that the Adelson-Bergen model is equivalent to an elaborated Reichardt model. There have been many hardware realizations of these correlation based algorithms. In (Delbrück, 1993), the author realizes correlation based motion computation using delay lines. The Barlow-Levick algorithm has also been realized in hardware (Benson and Delbrück, 1991; Horiuchi *et al.*, 1991). The Reichardt detector has been implemented in silicon in different ways. First was an implementation using translinear current mode circuits (Andreou and Strohbehn, 1990; Harrison and Koch, 1998). Similarly Harrison showed two VLSI implementations of the Reichardt model, one based on a current mode design and the other on a voltage mode design (Harrison, 2000). The Adelson-Bergen algorithm has also been implemented in hardware (Higgins and Korrapati, 2000) and is described in more detail later in this thesis. A large scale version of the AB-model has been implemented on a general-purpose analog neural computer (Etienne-Cummings *et al.*, 1999). In this work the Adelson-Bergen algorithm was chosen as the motion algorithm since it is more efficient

to implement in VLSI as it can be realized using fewer circuits when compared with other motion algorithms. Hence it is more efficient in terms of layout area.

1.3 Applications of Motion Computation

Motion computation is used extensively in real time machine-vision application tasks. Traditional methods of real-time machine vision applications use a CCD camera as the front end and a processor in the back end. These are quite popular and are used in a lot of applications. However, they are power intensive and require many resources. An attractive solution for this problem is to use parallel image-processing architectures on silicon. That is, these chips would combine the photo detection and processing capabilities on the same chip, making them more efficient in terms of power, space and cost. We now look at some of the work done in past to realize such applications related to our current work.

Indiveri demonstrated a vision chip which selectively detects and tracks the position of the feature with highest spatial contrast in the visual scene (Indiveri, 1999). Motion and velocity measurement has been used in smooth pursuit tracking (Cummings *et al.*, 1996). Velocity sensors have also been used to estimate the heading direction and to compute the time of contact (Indiveri *et al.*, 1996). Higgins and Koch describe a sensor (Higgins and Koch, 1999), in which they show how the direction of local motion, along with the location of singular points in the visual flow field, can be used for ego-motion. Cummings *et al.* demonstrate a navigation application in which the robot avoids obstacles during line following (Cummings *et al.*, 1998). However, this is a hybrid system in which they use a vision chip that detects edges and a micro controller for implementing the actual navigation algorithms. Barrows *et al.* demonstrate an application in which they compute optic flow from motion detectors and use it to steer a glider away from walls to avoid collisions (Barrows *et al.*, 1999). The author of this thesis was also involved in two projects based on visual input. The first application was to detect a high-contrast portion of a scene and track it using visual motion information. In this application the sensor was mounted on a pan-tilt head, and the motion information from the sensor was used to control the pan-tilt head for tracking. The second application was based on a vision sensor which computed the position of a target in the scene. The sensor was mounted on a small robot, Khepera (K-Team Inc Online, 2001), and the sensor controlled the robot for tracking the target. More details about these two projects can be found in the report of the 2000 Workshop of Neuromorphic Engineering (Cohen *et al.*, 2000). The use of visual motion detection is not just limited in tracking and other machine vision applications. Motion computation is also heavily used in MPEG coding, intelligent transportation systems (ITS) and others. A traditional method of motion computation in camera and processor based methods is the block matching technique. In this technique, each new frame of data is partitioned into several blocks to detect motion vectors. These blocks are then matched with a reference block from the previous frame. Once a best match is found, the position displacement between the current block and the reference block gives the motion vector, from which the velocity can be estimated. However, this algorithm is computationally intensive. There have been several dedicated custom hardware implementations of this traditional block matching algorithm and other variants of it (Moshnyaga and Tamaru, 1997; Fang *et al.*, 2000; Wang *et al.*, 1994; Zhang and Chi-Ying, 1997).

Chapter 2

Biological Motion Algorithms

In this chapter we explain biological methods of motion computation. First we discuss the Reichardt detector based on flies (Hassenstein and Reichardt, 1956). Then we describe the Adelson-Bergen algorithm (Adelson and Bergen, 1985), which is the underlying algorithm of our motion energy sensor (to be described in more detail later in Chapter 4). The Adelson-Bergen algorithm is meant to model motion computation in the primate cortical cell. Both Reichardt detector and the Adelson-Bergen algorithm fall under the class of correlation based algorithms for motion computation.

One of the first models of motion computation was proposed by Hassenstein and Reichardt in 1956 to explain the optomotor response in flies. The Reichardt detector is shown in Figure 2.1(a). It has two subunits in it. Each subunit as shown in Figure 2.1(b) correlates the input from a photoreceptor with a delayed input from it's neighboring photoreceptor, separated by a distance $\Delta\phi$. Each subunit can be thought of as if it were tuned to motion in a particular direction (left or right). The Reichardt detector takes the difference of the subunits to get the opponent motion output.

We now explain the Reichardt detector in more detail. Let the input signal be a sinusoidal grating in one-dimension. Let ω_t be it's temporal frequency and ω_x be it's spatial frequency. If the mean luminance of the signal is I, then we can write the signals from the two photoreceptors as follows:

$$\begin{aligned} A &= I + \Delta I \sin(\omega_t \cdot t + \omega_x \cdot x) \\ B &= I + \Delta I \sin(\omega_t \cdot t + \omega_x \cdot x \pm \omega_x \Delta\phi) \end{aligned}$$

Where $\omega_x \Delta\phi$ corresponds to the phase difference introduced because of the separation, $\Delta\phi$ between the two photoreceptors. The sign of the phase delay introduced depends on the direction of the stimulus, it is positive if moving in one direction and negative if moving in the opposite direction. The signals are then taken through the temporal filters. The outputs from the temporal filters are given by:

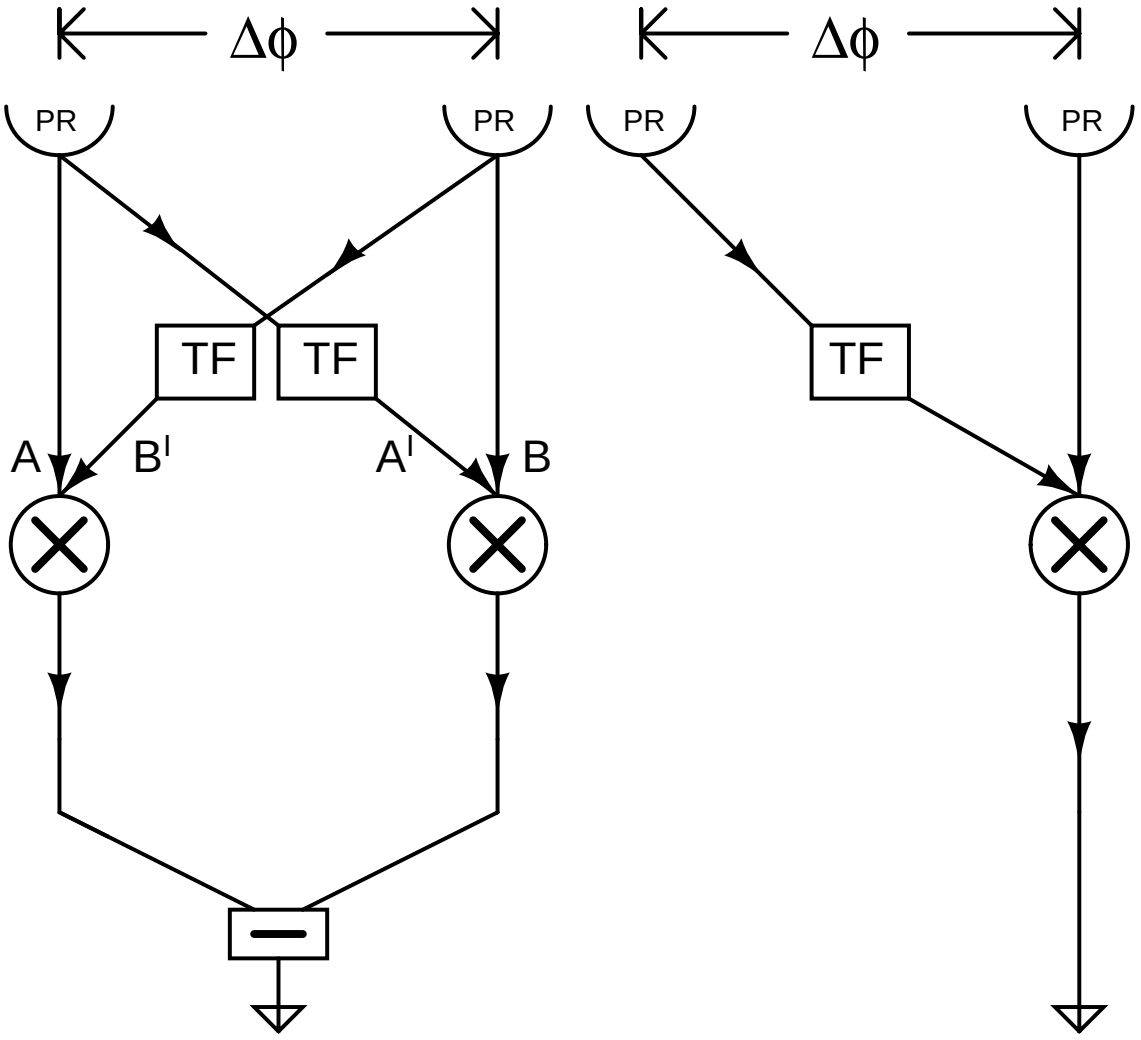
$$\begin{aligned} A' &= I + F(\omega_t)\Delta I \sin(\omega_t \cdot t + \omega_x \cdot x + \phi(\omega_t)) \\ B' &= I + F(\omega_t)\Delta I \sin(\omega_t \cdot t + \omega_x \cdot x + \phi(\omega_t) \pm \omega_x \Delta\phi) \end{aligned}$$

Where $F(\omega_t)$ is the amplitude and $\phi(\omega_t)$ is the phase of the temporal filters. For simplicity we consider here the case when these two filters are identical. In hardware implementations, the input image to the photoreceptors, A and B are high pass filtered before they go into their next stages. This is also the case in flies. The photoreceptor's outputs are high pass filtered to remove the mean luminance value. They adapt to the background luminance and thus report only changes in luminance to the next stage in the visual pathway. Taking this into consideration, before going into the correlation stage where we compute $A'B$ and AB', we remove the DC term (mean luminance term), I from the above four signals. The output from the correlation stage is given by:

$$\begin{aligned} A'B &= (\Delta I)^2 F(\omega_t) \sin(\omega_t \cdot t + \omega_x \cdot x + \phi(\omega_t)) \sin(\omega_t \cdot t + \omega_x \cdot x \pm \omega_x \Delta\phi) \\ AB' &= (\Delta I)^2 F(\omega_t) \sin(\omega_t \cdot t + \omega_x \cdot x) \sin(\omega_t \cdot t + \omega_x \cdot x + \phi(\omega_t) \pm \omega_x \Delta\phi) \end{aligned}$$

We can now write the opponent motion output of the Reichardt detector, $A'B - AB'$. After simplifying the difference, $A'B - AB'$ using the trigonometric identity $\cos(A-B) - \cos(A+B) = 2 \sin A \sin B$, the final opponent motion output is given as follows:

$$O = (\Delta I)^2 F(\omega_t) \sin(\phi(\omega_t)) \sin(\pm \omega_x \Delta\phi) \tag{2.1}$$



(a) Reichardt Detector (b) Subunit of a Reichardt detector

FIGURE 2.1. (a). The Reichardt detector. The input image pattern moves across the photoreceptors denoted as PR, separated by $\Delta\phi$. The direct input from a photoreceptor is correlated with the temporally filtered input from the adjacent photoreceptor (shown as TF). This temporal filtering introduces a delay in the photoreceptor output. The outputs from these correlation stages are subtracted to get an opponent motion output. Temporal averaging can be done after the correlation stage, before the subtraction to get a mean output. (b) The right subunit of the detector. The Reichardt detector has two component subunits, tuned to motion in opposite directions (right and left) and the outputs of these subunits are subtracted to get the motion output.

We can see that the above opponent motion output can distinguish stimuli moving in its preferred direction from stimuli moving in its null direction. In case of a one dimensional sensor, the preferred direction is the direction of the stimulus when it moves across the pixels in the positive direction and the null direction is the direction of the stimulus when it moves in the negative direction across the pixels. From Equation 2.1, we can see that if the sign of the phase delay introduced is positive, then the opponent motion output is positive and if the sign of the phase delay is negative, the opponent motion output is negative. A more elaborate derivation of the Reichardt detector, which includes the cases when the subtraction stage is unbalanced (i.e., the case when the opponent motion output is given by $A'B - gAB'$, where $g \neq 1$), and for non identical temporal filters in the two sub units is given in (Egelhaaf *et al.*, 1989).

We now discuss the Adelson-Bergen model. Before we explain the Adelson-Bergen algorithm, let us look at motion in space-time and see how we can interpret it. Figure 2(a) shows a bar in two dimensions, x and y moving along the x direction in time. Since the bar is constant in the y direction, let us consider the same bar in the x-t space as shown in Figure 2(b). We can see that as time proceeds, the bar drifts along the x-axis as shown. Now, consider the bar moving in the x-t space as shown in Figure 2(c). In this figure we plot the bar moving with five different velocities. The extreme left plot shows the bar moving with a velocity of -2 and the extreme right plot shows it move with a velocity of $+2$. We can see that motion can be thought as orientation in space and time and so if we can find the orientation in space-time, we can find the velocity of the image pattern. Thus the problem of motion computation can be transformed into a problem of orientation detection in space and time. For computing orientation, we can use oriented filters in space and time. This is the premise of the Adelson-Bergen model, and they propose the use of such oriented filters in quadrature phase to compute phase independent motion energy.

The model of the detector is shown in Figure 2.3. The input image is fed into the detector. The two receptive fields are displaced in position. $f_1(x)$ and $f_2(x)$ are two spatial filters. The outputs of these are passed through two different temporal filters $h_1(t)$ and $h_2(t)$. One of these filters delays (or low passes) the input signal more than the other. By combining the signals as shown in the model, we obtain four separable outputs, A, B, A', B'. Once we obtain the four separable responses, they are combined as shown to obtain oriented linear responses, $(A - B'), (A' + B), (A + B'), (A' - B)$. Each of these is then squared as shown. Then they are summed to obtain the oriented energy, $(A - B')^2 + (A' + B)^2$ and $(A + B')^2 + (A' - B)^2$. These two are subtracted to obtain the opponent motion energy $4(AB' - A'B)$ as shown in the model. From here on we call this model the Elementary Motion Detector, EMD.

Adelson-Bergen propose the use of linear spatial and temporal filters which are in quadrature for the EMD. They suggest the use of gabor filters in quadrature as the spatial filters. These are plotted in Figure 2.4(a), and can be expressed mathematically as the following:

$$\begin{aligned} f_{s1}(x) &= e^{(\frac{-x^2}{2\cdot\sigma^2})} \cdot \cos(\omega_x \cdot x) \\ f_{s2}(x) &= e^{(\frac{-x^2}{2\cdot\sigma^2})} \cdot \sin(\omega_x \cdot x) \end{aligned}$$

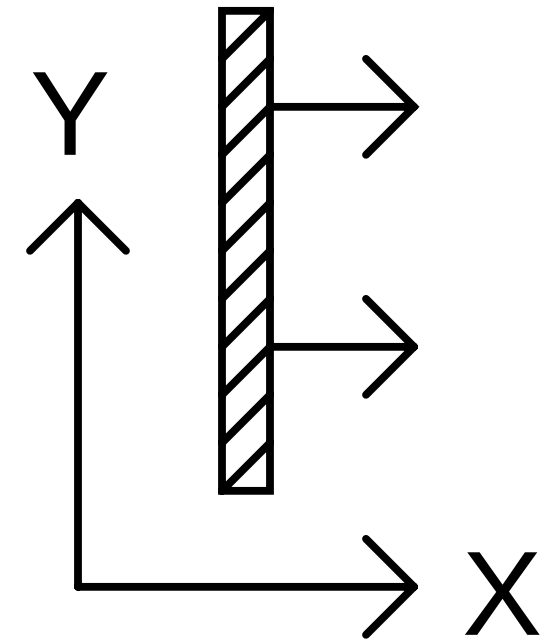
And they suggest the use of the second and third derivatives of gaussians as temporal filters. These are plotted in Figure 2.4(b) and are of the form:

$$\begin{aligned} f_{t1}(t) &= (kt)^3 \cdot e^{-kt^2} \cdot \left[\frac{1}{3!} - \frac{(kt)^2}{(3+2)!}\right] \\ f_{t2}(t) &= (kt)^5 \cdot e^{-kt^2} \cdot \left[\frac{1}{5!} - \frac{(kt)^2}{(5+2)!}\right] \end{aligned}$$

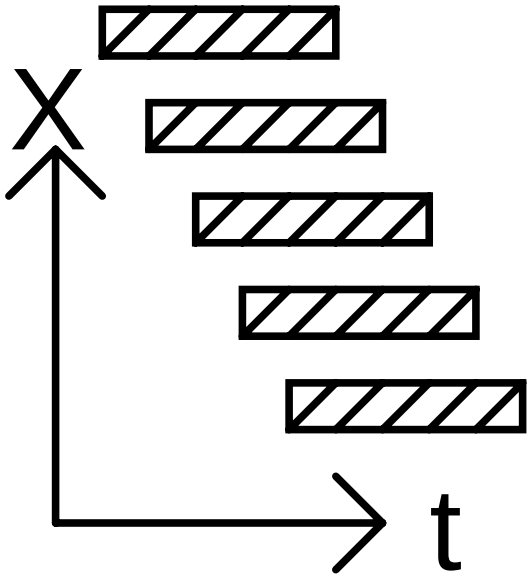
Figure 2.4(c) shows a spatio-temporal plot of the model. It plots the opponent energy from the model at various spatial and temporal frequencies.

To appreciate the working of the EMD, let us consider the case of a pure sinusoidal grating pattern in one dimension as input to the EMD. So, the input stimulus can be written as:

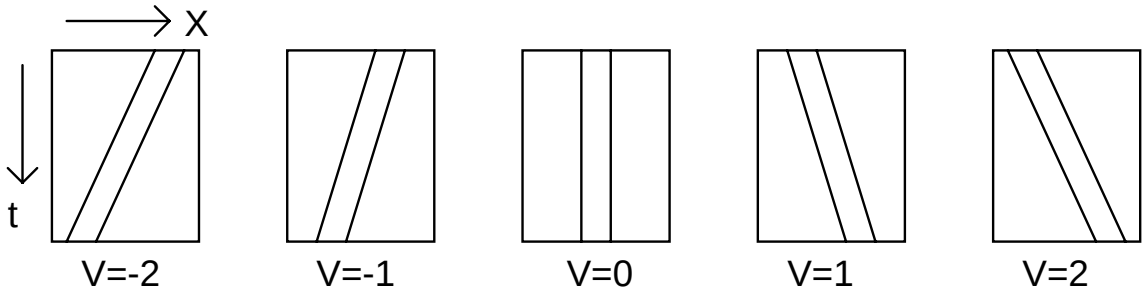
$$I(x,t) = I \cdot \sin(\omega_t \cdot t + \omega_x \cdot x)$$



(a) Bar in X-Y space



(b) Bar in X-t space



(c) Motion as orientation in X-T

FIGURE 2.2. Interpreting motion: (a) A bar in X-Y space, which drifts along the X-axis in time. (b) The same bar plotted in the X-t space, we can see the bar progressing in time. (c) A space-time plot of the bar moving with different velocities. We can see that each velocity can be thought of as a particular orientation in space-time. Reproduced without permission from (Adelson and Bergen, 1985).

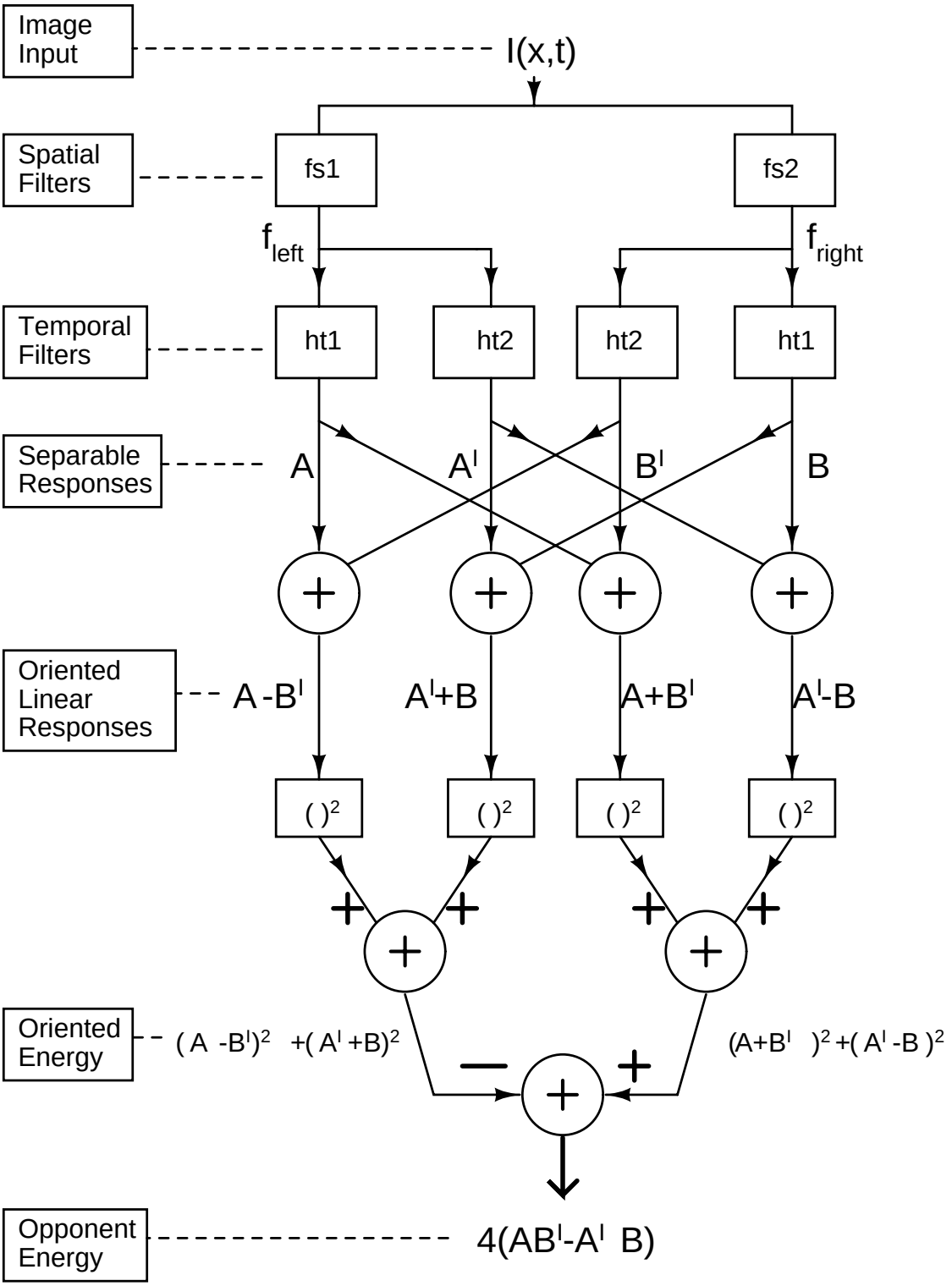
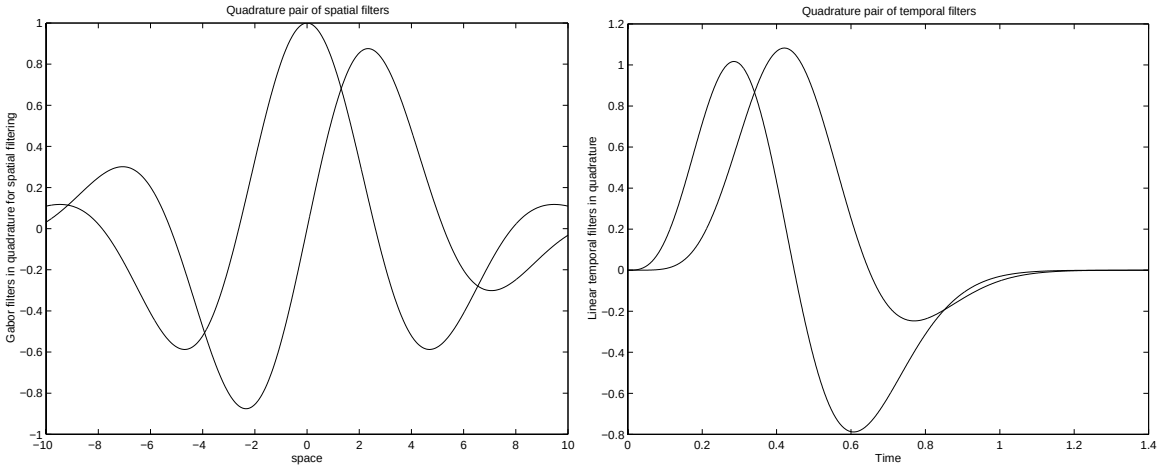
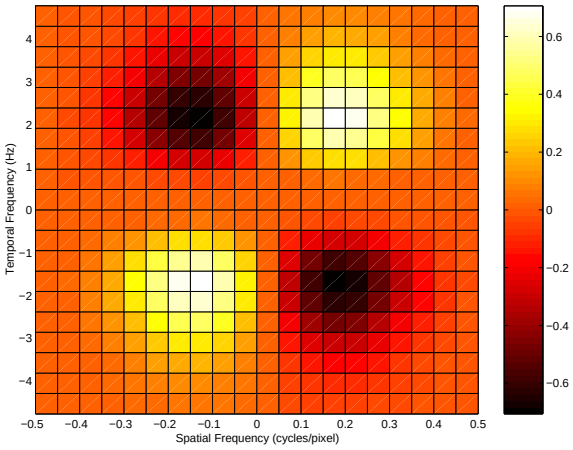


FIGURE 2.3. The Adelson-Bergen motion detector. Reproduced without permission from (Adelson and Bergen, 1985).



(a) Spatial Filters

(b) Temporal Filters



(c) Opponent energy

FIGURE 2.4. Spatial and Temporal filters in the Adelson-Bergen model. (a) Spatial Gabor filters in quadrature. (b) Temporal filters in quadrature. (c) Spatio-Temporal plot of the final opponent energy of the model. The opponent energy is plotted for spatial frequencies on the X-axis versus temporal frequencies on the Y-axis. We can see that the the model responds best to a particular spatio-temporal frequency to which it is tuned for and the response decreases at other frequencies. The simulations use $\sigma = 4.8$, $\omega_x = 0.6$ and $k = 9$.

where ω_t is the temporal frequency of the grating and ω_x is the spatial frequency of the grating. After the image passes through the spatial filters, since the input is a pure sinusoid, we get the following (Haykin, 1996):

$$\begin{aligned} f_{left}(t) &= |fs1| \cdot I \cdot \sin(\omega_t \cdot t + \omega_x \cdot x + \phi_{s1}(\omega_x)) \\ f_{right}(t) &= |fs2| \cdot I \cdot \sin(\omega_t \cdot t + \omega_x \cdot x + \phi_{s2}(\omega_x)) \end{aligned}$$

Where $|fs1|$ and $|fs2|$ are the magnitudes of the two spatial filters and $\phi_{s1}(\omega_x)$ and $\phi_{s2}(\omega_x)$, are the phases of the spatial filters and come as a result of applying the impulse response of the two filters on the input. From here on we use ϕ_{s1} and ϕ_{s2} for representing $\phi_{s1}(\omega_x)$ and $\phi_{s2}(\omega_x)$ respectively. f_{left} and f_{right} are now passed through the two temporal filters, $ht1$ and $ht2$ respectively. We get the following four separable responses:

$$\begin{aligned} A(t) &= |ht1| \cdot |fs1| \cdot I \cdot \sin(\omega_t \cdot t + \omega_x \cdot x + \phi_{s1} + \phi_{t1}(\omega_t)) \\ A'(t) &= |ht2| \cdot |fs1| \cdot I \cdot \sin(\omega_t \cdot t + \omega_x \cdot x + \phi_{s1} + \phi_{t2}(\omega_t)) \\ B'(t) &= |ht2| \cdot |fs2| \cdot I \cdot \sin(\omega_t \cdot t + \omega_x \cdot x + \phi_{s2} + \phi_{t2}(\omega_t)) \\ B(t) &= |ht1| \cdot |fs2| \cdot I \cdot \sin(\omega_t \cdot t + \omega_x \cdot x + \phi_{s2} + \phi_{t1}(\omega_t)) \end{aligned}$$

Where $|ht1|$ and $|ht2|$ are the magnitudes of the two temporal filters and $\phi_{t1}(\omega_t)$, $\phi_{t2}(\omega_t)$ come as a result of applying the impulse response of the two filters on the input signals, $f_{left}(t)$ and $f_{right}(t)$. From here on we represent $\phi_{t1}(\omega_t)$ as ϕ_{t1} and $\phi_{t2}(\omega_t)$ as ϕ_{t2}. From the model we see that the final opponent motion energy is $4(A \cdot B' - A' \cdot B)$. Substituting for A, B, A' and B', we can write the final opponent motion energy as follows:

$$\mathbf{O} = 4 \cdot \left[\begin{array}{l} |fs1| \cdot |fs2| \cdot |ht1| \cdot |ht2| \cdot I^2 \cdot \left(\begin{array}{l} \sin(\omega_t \cdot t + \omega_x \cdot x + \phi_{s1} + \phi_{t1}) \times \\ \sin(\omega_t \cdot t + \omega_x \cdot x + \phi_{s2} + \phi_{t2}) \end{array} \right) - \\ |fs1| \cdot |fs2| \cdot |ht1| \cdot |ht2| \cdot I^2 \cdot \left(\begin{array}{l} \sin(\omega_t \cdot t + \omega_x \cdot x + \phi_{s2} + \phi_{t1}) \times \\ \sin(\omega_t \cdot t + \omega_x \cdot x + \phi_{s1} + \phi_{t2}) \end{array} \right) \end{array} \right]$$

Using the identity $2 \cdot \sin(A) \cdot \sin(B) = \cos(A - B) - \cos(A + B)$, we can write the above equation as follows:

$$\mathbf{O} = 2 \cdot |fs1| \cdot |fs2| \cdot |ht1| \cdot |ht2| \cdot I^2 \cdot \left[\begin{array}{l} \cos(\phi_{s1} - \phi_{s2} + \phi_{t1} - \phi_{t2}) - \\ cos(2 \cdot \omega_t \cdot t + 2 \cdot \omega_x \cdot x + \phi_{s1} + \phi_{s2} + \phi_{t1} + \phi_{t2}) - \\ \cos(\phi_{s2} - \phi_{s1} + \phi_{t1} - \phi_{t2}) + \\ cos(2 \cdot \omega_t \cdot t + 2 \cdot \omega_x \cdot x + \phi_{s1} + \phi_{s2} + \phi_{t1} + \phi_{t2}) \end{array} \right]$$

$$\Rightarrow \mathbf{O} = 2 \cdot |fs1| \cdot |fs2| \cdot |ht1| \cdot |ht2| \cdot I^2 \cdot \left[\begin{array}{l} \cos((\phi_{s1} - \phi_{s2}) + (\phi_{t1} - \phi_{t2})) - \\ \cos((\phi_{s1} - \phi_{s2}) - (\phi_{t1} - \phi_{t2})) \end{array} \right]$$

Again using the same identity $2 \cdot \sin(A) \cdot \sin(B) = \cos(A - B) - \cos(A + B)$,we can write the above equation as follows:

$$\mathbf{O} = 4 \cdot I^2 \cdot \underbrace{(|fs1| \cdot |fs2| \sin(\phi_{s1} - \phi_{s2}))}_{Spatialcomponent} \cdot \underbrace{(|ft1| \cdot |ft2| \sin(\phi_{t1} - \phi_{t2}))}_{Temporalcomponent}$$

If the spatial and temporal filters are quadrature pairs for the EMD tuned for that particular spatial frequency ω_x and temporal frequency ω_t , ie.,

$$\begin{aligned} \phi_{s1}(\omega_x) - \phi_{s2}(\omega_x) &= \frac{\pi}{2} \\ \phi_{t1}(\omega_t) - \phi_{t2}(\omega_t) &= \frac{\pi}{2} \end{aligned}$$

Then we can see that the Adelson-Bergen model achieves phase independency. Since $sin(\pi/2) = 1$, the sinusoidal terms become unity and the final opponent motion energy is given by:

$$\mathbf{O} = 4 \cdot |fs1| \cdot |fs2| \cdot |ft1| \cdot |ft2| \cdot I^2$$

It is thought that the primate cortical cell has a bank of such EMD's, each tuned to a particular spatial and temporal frequency. We now consider the case when two different sinusoidal gratings (i.e, with different spatial and temporal frequencies), are given to such single EMD unit. This case resembles more closely to the real world visual input. The image input to the EMD is,

$$I(x,t) = I_1 \cdot \sin(\omega_{t1} \cdot t + \omega_{x1} \cdot x) + I_2 \cdot \sin(\omega_{t2} \cdot t + \omega_{x2} \cdot x)$$

After the image passes through the spatial filters, we get the following, by using the definition of linear filters:

$$\begin{aligned} f_{left}(t) &= \left[\begin{array}{l} |fs1| \cdot I_1 \cdot \sin\left(\omega_{t1} \cdot t + \omega_{x1} \cdot x + \phi_{s1}(\omega_{x1})\right) + \\ |fs1| \cdot I_2 \cdot \sin\left(\omega_{t2} \cdot t + \omega_{x2} \cdot x + \phi_{s2}(\omega_{x2})\right) \end{array} \right] \\ f_{right}(t) &= \left[\begin{array}{l} |fs2| \cdot I_1 \cdot \sin\left(\omega_{t1} \cdot t + \omega_{x1} \cdot x + \phi_{s3}(\omega_{x1})\right) + \\ |fs2| \cdot I_2 \cdot \sin\left(\omega_{t2} \cdot t + \omega_{x2} \cdot x + \phi_{s4}(\omega_{x2})\right) \end{array} \right] \end{aligned}$$

From here on, we use ϕ_{s1} for $\phi_{s1}(\omega_{x1})$, ϕ_{s2} for $\phi_{s2}(\omega_{x2})$, ϕ_{s3} for $\phi_{s3}(\omega_{x1})$ and ϕ_{s4} for $\phi_{s4}(\omega_{x2})$. |fs1| and |fs2| are the magnitudes of the two spatial filters and ϕ_{s1}, ϕ_{s2}, ϕ_{s3}, ϕ_{s4} are phases which come as a result of applying the impulse response of the two spatial filters on the input. These are now taken through the temporal filters to obtain the four separable responses shown below:

$$\begin{aligned} A(t) &= \left[\begin{array}{l} |h_{11}| \cdot |fs1| \cdot I_1 \cdot \sin\left(\omega_{t1} \cdot t + \omega_{x1} \cdot x + \phi_{11} + \phi_{s1}\right) + \\ |h_{12}| \cdot |fs1| \cdot I_2 \cdot \sin\left(\omega_{t2} \cdot t + \omega_{x2} \cdot x + \phi_{12} + \phi_{s2}\right) \end{array} \right] \\ A'(t) &= \left[\begin{array}{l} |h_{21}| \cdot |fs1| \cdot I_1 \cdot \sin\left(\omega_{t1} \cdot t + \omega_{x1} \cdot x + \phi_{21} + \phi_{s1}\right) + \\ |h_{22}| \cdot |fs1| \cdot I_2 \cdot \sin\left(\omega_{t2} \cdot t + \omega_{x2} \cdot x + \phi_{22} + \phi_{s2}\right) \end{array} \right] \\ B(t) &= \left[\begin{array}{l} |h_{11}| \cdot |fs2| \cdot I_1 \cdot \sin\left(\omega_{t1} \cdot t + \omega_{x1} \cdot x + \phi_{11} + \phi_{s3}\right) + \\ |h_{12}| \cdot |fs2| \cdot I_2 \cdot \sin\left(\omega_{t2} \cdot t + \omega_{x2} \cdot x + \phi_{12} + \phi_{s4}\right) \end{array} \right] \\ B'(t) &= \left[\begin{array}{l} |h_{21}| \cdot |fs2| \cdot I_1 \cdot \sin\left(\omega_{t1} \cdot t + \omega_{x1} \cdot x + \phi_{21} + \phi_{s3}\right) + \\ |h_{22}| \cdot |fs2| \cdot I_2 \cdot \sin\left(\omega_{t2} \cdot t + \omega_{x2} \cdot x + \phi_{22} + \phi_{s4}\right) \end{array} \right] \end{aligned}$$

ϕ_{11}, ϕ_{12}, ϕ_{21} and ϕ_{22} come as a result of applying the impulse response of the temporal filters on the input. From the model, the final opponent motion energy is $4(A \cdot B' - A' \cdot B')$. Substituting for A, B, A' and B', and after a lot of simplification using the trigonometric identity described previously, the final result for the opponent motion energy is given by:

$$\mathbf{O} = \left[\begin{array}{l} \frac{I_1 \cdot I_2 \cdot |fs1| \cdot |fs2| \cdot h_{11} \cdot h_{21}}{2} \cdot \left(\begin{array}{l} \cos((\phi_{11} - \phi_{21}) + (\phi_{s1} - \phi_{s3})) - \\ \cos((\phi_{21} - \phi_{11}) + (\phi_{s1} - \phi_{s3})) \end{array} \right) + \\ \frac{I_1 \cdot I_2 \cdot |fs1| \cdot |fs2| \cdot h_{12} \cdot h_{21}}{2} \cdot \left(\begin{array}{l} \cos(\omega_{dt} \cdot t + \omega_{dx} \cdot x + (\phi_{12} - \phi_{21}) + (\phi_{s2} - \phi_{s3})) - \\ \cos(\omega_{st} \cdot t + \omega_{sx} \cdot x + (\phi_{12} + \phi_{21}) + (\phi_{s2} + \phi_{s3})) + \\ \cos(\omega_{st} \cdot t + \omega_{sx} \cdot x + (\phi_{12} + \phi_{21}) + (\phi_{s1} + \phi_{s4})) - \\ \cos(\omega_{dt} \cdot t + \omega_{dx} \cdot x + (\phi_{12} - \phi_{21}) + (\phi_{s4} - \phi_{s1})) \end{array} \right) + \\ \frac{I_1 \cdot I_2 \cdot |fs1| \cdot |fs2| \cdot h_{22} \cdot h_{12}}{2} \cdot \left(\begin{array}{l} \cos((\phi_{12} - \phi_{22}) + (\phi_{s2} - \phi_{s4})) - \\ \cos((\phi_{22} - \phi_{12}) + (\phi_{s2} - \phi_{s4})) \end{array} \right) + \\ \frac{I_1 \cdot I_2 \cdot |fs1| \cdot |fs2| \cdot h_{22} \cdot h_{11}}{2} \cdot \left(\begin{array}{l} \cos(\omega_{dt} \cdot t + \omega_{dx} \cdot x + (\phi_{22} - \phi_{11}) + (\phi_{s4} - \phi_{s1})) - \\ \cos(\omega_{st} \cdot t + \omega_{sx} \cdot x + (\phi_{11} + \phi_{22}) + (\phi_{s1} + \phi_{s4})) + \\ \cos(\omega_{st} \cdot t + \omega_{sx} \cdot x + (\phi_{22} + \phi_{11}) + (\phi_{s2} + \phi_{s3})) - \\ \cos(\omega_{dt} \cdot t + \omega_{dx} \cdot x + (\phi_{11} - \phi_{22}) + (\phi_{s3} - \phi_{s2})) \end{array} \right) \end{array} \right]$$

Where we use $(\omega_{t2} - \omega_{t1}) = \omega_{dt}$; $(\omega_{t2} + \omega_{t1}) = \omega_{st}$; $(\omega_{x2} - \omega_{x1}) = \omega_{dx}$; $(\omega_{x2} + \omega_{x1}) = \omega_{sx}$. If this EMD has a quadrature pair of spatial and temporal filters tuned for the frequencies of ω_{x1} and ω_{t1} we can write the following:

$$\begin{aligned} \phi_{s1} - \phi_{s3} &= \frac{\pi}{2} \\ \phi_{11} - \phi_{21} &= \frac{\pi}{2} \end{aligned}$$

Substituting these in the above equation, we obtain the following:

$$\mathbf{O} = \left[\begin{array}{l} \frac{I_1 \cdot I_2 \cdot |fs1| \cdot |fs2| \cdot h_{11} \cdot h_{21}}{2} \cdot \left(\begin{array}{l} \cos(\frac{\pi}{2} + \frac{\pi}{2}) - \\ \cos(-\frac{\pi}{2} + \frac{\pi}{2}) \end{array} \right) + \\ \frac{I_1 \cdot I_2 \cdot |fs1| \cdot |fs2| \cdot h_{12} \cdot h_{21}}{2} \cdot \left(\begin{array}{l} \cos(\omega_{dt} \cdot t + \omega_{dx} \cdot x + (\phi_{12} - \phi_{21}) + (\phi_{s2} - \phi_{s3})) - \\ \cos(\omega_{st} \cdot t + \omega_{sx} \cdot x + (\phi_{12} + \phi_{21}) + (\phi_{s2} + \phi_{s3})) + \\ \cos(\omega_{st} \cdot t + \omega_{sx} \cdot x + (\phi_{12} + \phi_{21}) + (\phi_{s1} + \phi_{s4})) - \\ \cos(\omega_{dt} \cdot t + \omega_{dx} \cdot x + (\phi_{12} - \phi_{21}) + (\phi_{s4} - \phi_{s1})) \end{array} \right) + \\ \frac{I_1 \cdot I_2 \cdot |fs1| \cdot |fs2| \cdot h_{22} \cdot h_{12}}{2} \cdot \left(\begin{array}{l} \cos((\phi_{12} - \phi_{22}) + (\phi_{s2} - \phi_{s4})) - \\ \cos((\phi_{22} - \phi_{12}) + (\phi_{s2} - \phi_{s4})) \end{array} \right) + \\ \frac{I_1 \cdot I_2 \cdot |fs1| \cdot |fs2| \cdot h_{22} \cdot h_{11}}{2} \cdot \left(\begin{array}{l} \cos(\omega_{dt} \cdot t + \omega_{dx} \cdot x + (\phi_{22} - \phi_{11}) + (\phi_{s4} - \phi_{s1})) - \\ \cos(\omega_{st} \cdot t + \omega_{sx} \cdot x + (\phi_{11} + \phi_{22}) + (\phi_{s1} + \phi_{s4})) + \\ \cos(\omega_{st} \cdot t + \omega_{sx} \cdot x + (\phi_{22} + \phi_{11}) + (\phi_{s2} + \phi_{s3})) - \\ \cos(\omega_{dt} \cdot t + \omega_{dx} \cdot x + (\phi_{11} - \phi_{22}) + (\phi_{s3} - \phi_{s2})) \end{array} \right) \end{array} \right]$$

From the above equation, we see that the EMD output has a phase independent component with a fixed mean value (first term, $-I_1 \cdot I_2 \cdot |fs1| \cdot |fs2| \cdot h11 \cdot h21$), a component which arises because of the EMD being tuned to only one particular spatio-temporal frequency (third term), and two other components at the sum and difference of temporal and spatial frequencies. So, when a real world stimulus is given to an EMD tuned for a particular spatio-temporal frequency, there would be a ripple riding on a mean value. Though we derived the case for a stimulus with two different temporal and spatial frequencies, we can extend this derivation for the case of stimuli with many different spatial and temporal frequencies in it. This is the reason why the EMD output would give a ripple riding on a mean value when a single bar is given as the stimulus. as an edge can be thought as a sum of pure sinusoid inputs with different frequencies in it. Also, we derive the results for a one dimensional case, this derivation can be easily extended for 2-dimensional stimulus (in both x and y directions) which is omitted here, as it is exactly a similar derivation but with an added component in the y direction.

In this chapter we described two biologically inspired motion detection algorithms, the Reichardt detector and the Adelson-Bergen algorithm. Though these algorithms realize motion detection through different computations, these two detectors are mathematically equivalent as shown in (Van Santen and Sperling, 1985). The Reichardt detector is well suited to implement in VLSI (Harrison, 2000), as it doesn't have too many computations in it. But, the hardware implementation of the Reichardt detector needs two four quadrant multiplier circuits for computing the multiplication. The four quadrant multiplier circuit has a large transistor count. The Adelson-Bergen algorithm has more computations in it than the Reichardt detector. However, hardware realization of the Adelson-Bergen algorithm is not very complicated if we keep the signals in the current mode instead of voltage mode. We explain in more detail in Chapter 4 as to how we can make other approximations in the Adelson-Bergen model to achieve an efficient VLSI realization of the Adelson-Bergen algorithm for motion detection.

Chapter 3

Modeling of Visual Motion Detection Circuits in Flies

In the previous chapter we talked about an Elementary Motion detector and discussed two well-known EMDs, the Reichardt detector and the Adelson-Bergen detector. EMDs derived from insect visual system are based on observations from the giant motion-sensitive tangential neurons in the lobula plate of a fly. These neurons correspond to the final opponent motion output stage discussed in the EMDs of previous chapter. Until recently there have been no recordings from neurons projecting onto the motion-sensitive tangential neurons. The EMDs proposed thus far are only from theory and are not based on anatomical observations. In this chapter we describe some of our modeling efforts based on the recordings from neurons afferent to these tangential neurons.

Before going into the details of modeling, let us first examine some essential features an EMD should possess for it to be direction selective. Direction selectivity implies that an EMD can distinguish motion in the direction it is tuned for (preferred direction) from the motion in the opposite direction (null direction). There are some general requirements for a directional selective motion detector (Franceschini *et al.*, 1989; Borst and Egelhaaf, 1989). These are:

- Two inputs are needed for motion detection. In order to determine motion there have to be different points in space which sample the visual input. With just one sampling point (or photoreceptor) we cannot distinguish an edge passing from left to right from an edge moving from right to left.
- The signals from the sampling points should undergo asymmetric linear filtering. That is, one of the signals should be low pass filtered (or delayed) more than the other.
- A non linear interaction between the two signals is needed, i.e., the two signals should be combined in a non linear fashion (like multiplication) before we can identify motion.
- Time averaging of the resulting signals from the non linear interaction is performed in neurons, though this might not be a necessary requirement in models.

Figure 3.1 shows the visual system of a fly, which contains the EMD circuit in it. The figure shows the main areas in the nervous system of the fly relevant to motion computation. In flies motion computation happens at a very early stage in the visual pathway. The various stages past retina are the lamina, medulla and then the lobula and lobula plate. The motion-sensitive tangential cells are in the lobula plate. Motion computation is thought to happen before it, in the previous stages. Figure 3.2 shows the wiring diagram that contains the neurons in the early visual pathway of the fly that are thought to participate in motion detection (N. Strausfeld, personal correspondence). Intracellular recordings from neurons early in the wiring diagram have only been reported recently (Douglass and Strausfeld, 1995; Douglass and Strausfeld, 1996; Douglass and Strausfeld, 1998). Our modeling is based on these recordings.

Let us first look at the HS cells which are at the bottom of the wiring diagram. HS (horizontally selective) cells are in the lobula plate and they pool inputs from the dendrites of the bushy T-cells, T4 and T5. HS neurons are spiking. They have a steady firing rate and depolarize with stimulus moving in its preferred direction and hyperpolarize with stimulus moving in the null direction (Franceschini *et al.*, 1989). As shown in Figure 3.2, these HS neurons pool input signals from a large number of T5 neurons and are sensitive to motion. We can conclude that these are computing a global sum of motion, which is computed earlier by individual EMDs.

The T5 neurons whose dendrites originate in the outermost stratum of the lobula provide the input to the HS neurons. The recordings from the T5 neurons (Douglass and Strausfeld, 1995) show that the response of T5 neurons resembles the response of the HS neurons (i.e., they depolarize to stimulus in its preferred direction and hyperpolarize to stimulus in its null direction). This

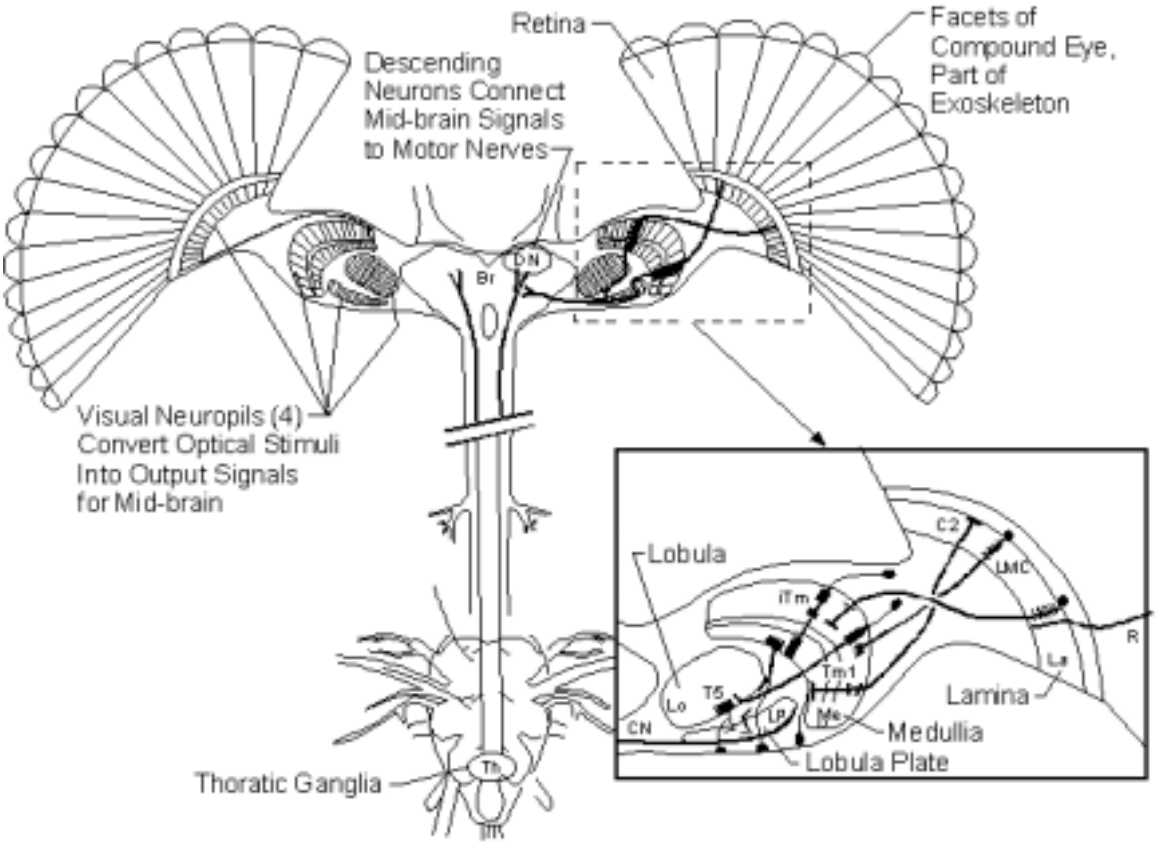


FIGURE 3.1. Schematic of the visual system of dipteran insects.

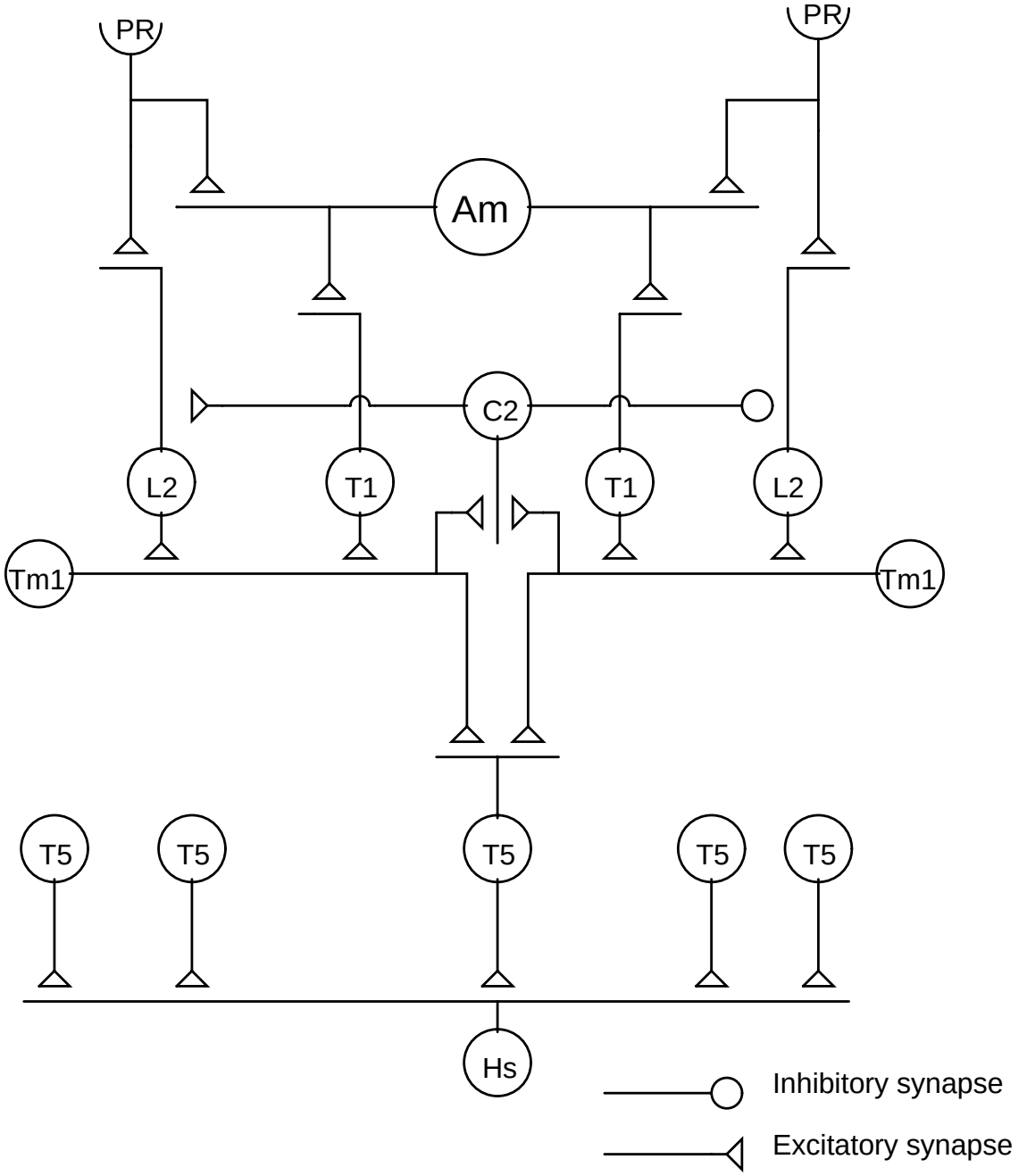


FIGURE 3.2. Proposed anatomical model for an EMD in the early visual pathway of flies. PR is a photoreceptor. The various neurons are, amacrine neuron, AM; type 2 lamina centrifugal cell, C2; large monopolar cell, L2; centripetal neuron, T1; transmedullary cell Tm1; bushy T cell, T5; horizontally selective cell, HS.

fully opponent response to stimulus in its preferred and null direction indicates that T5 acts as a "subtraction" stage.

The wiring diagram shows the main input to the T5 cell as the Tm1 neuron. The recordings of Tm1 neurons (Douglass and Strausfeld, 1995) show that Tm1 does show direction selectivity, but not in level shifts, instead it shows variations in its frequency response. Response to a stimulus in the null direction was at slightly higher peak to peak amplitude than the response to the stimulus in its preferred direction and the frequency of the response was twice the frequency of the stimulus. The response to a stimulus in the preferred direction had the same frequency as that of the stimulus and a slightly smaller overall amplitude. As Tm1 shows a change in frequency, we can conclude that Tm1 is acting as the "non-linear" stage in the EMD.

Going up the model we see two neurons, T1 and L2, making excitatory synapses onto the Tm1 neuron. T1 neurons are not post synaptic to the photoreceptors. But, they obtain their inputs in lamina from intermediate neurons. They receive input from the amacrine cells, which are post synaptic to the photoreecptors. The recordings from these T1 neurons (Douglass and Strausfeld, 1995), show that they do not distinguish motion direction, but show hyperpolarizing fluctuations at the frequency of the input stimulus for both the preferred and null directions. So, T1 can act as an intermediate stage in the EMD.

The large monopolar cell, L2, has been studied extensively before (Laughlin, 1989). L2 is directly post synaptic to the photoreceptor, and is thought to perform three different transformations on the incoming visual input. They are inversion, amplification and high pass filtering. The input light intensity can vary about five orders in magnitude from bright to dark but the L2 neurons have only a restricted range of voltage (about 60mv) to encode this intensity of light. L2 uses the trick of neural adaptation to cope with this. By this adaptation mechanism it encodes only the changes in illumination from the background illumination. It high pass filters all the background intensity and amplifies just the change in intensities. Thus when the photoreceptors are initially adapted to darkness and when a small light is presented to the photoreceptors as stimulus for a certain time and then removed, L2 responds with an initial "hyperpolarizing on transient" and then a "depolarizing off transient". It responds the other way for a dark bar over an illuminated background. This characteristic response has been very well modeled in hardware in (Liu, 1998).

The other two neurons in the wiring diagram are the amacrine neuron and the C2 neuron. Recordings from the amacrine cell (Douglass and Strausfeld, 1996) show that these neurons exhibit transient depolarizations at the temporal frequency of the grating. Also, these responses exhibit direction-dependent phase shifts. These neurons can thus be thought to perform some kind of delaying, as these receive synaptic inputs from adjacent photoreceptors. The last neuron in the model, the type 2 lamina centrifugal cell, C2, is different from all the previous neurons in the sense that it has an excitatory synapse onto the L2 neuron in lamina and an inhibitory synapse to the L2 neuron of the neighboring column. The recordings from C2 neuron (Douglass and Strausfeld, 1995) show that it exhibits hyperpolarization for motion and has small fluctuations at the contrast frequency of the grating. Thus these two neurons, L2 and C2 can be thought to play the vital role of linking the two adjacent visual columns to perform motion computation.

Based on all these observations, Figure 3.3 shows an elementary motion detector based on the motion detection circuits in flies. We can observe that the model has all the salient features necessary for motion computation as described earlier in the chapter. The photoreceptors are denoted as PR1 and PR2 in the model. The response from the photoreceptors is delayed through the temporal filters denoted as TF1 and TF2 in the model. These delay stages can be thought to occur through the amacrine and T1 cells as explained previously. Next are the six summation stages, which can be thought to occur at the synapses of L2, T1 and C2. Following these, there are four non-linear stages. These four stages can be though to occur at the four synapses between the L2 and Tm1, T1 and Tm1 cells. Tm1 cells compute the non-linearity as explained previously when describing the response of Tm1 cells. The final opponent motion output is obtained though a subtraction which could be computed by the T5 cell.

Simulation results

We show results from simulating the proposed model in figures 3.4(a) and 3.4(b). Figure 3.4(a) corresponds to the case when the stimulus is given in the preferred direction and Figure 3.4(b) shows the results when the stimulus is given in the null direction. In both figures, the first plot

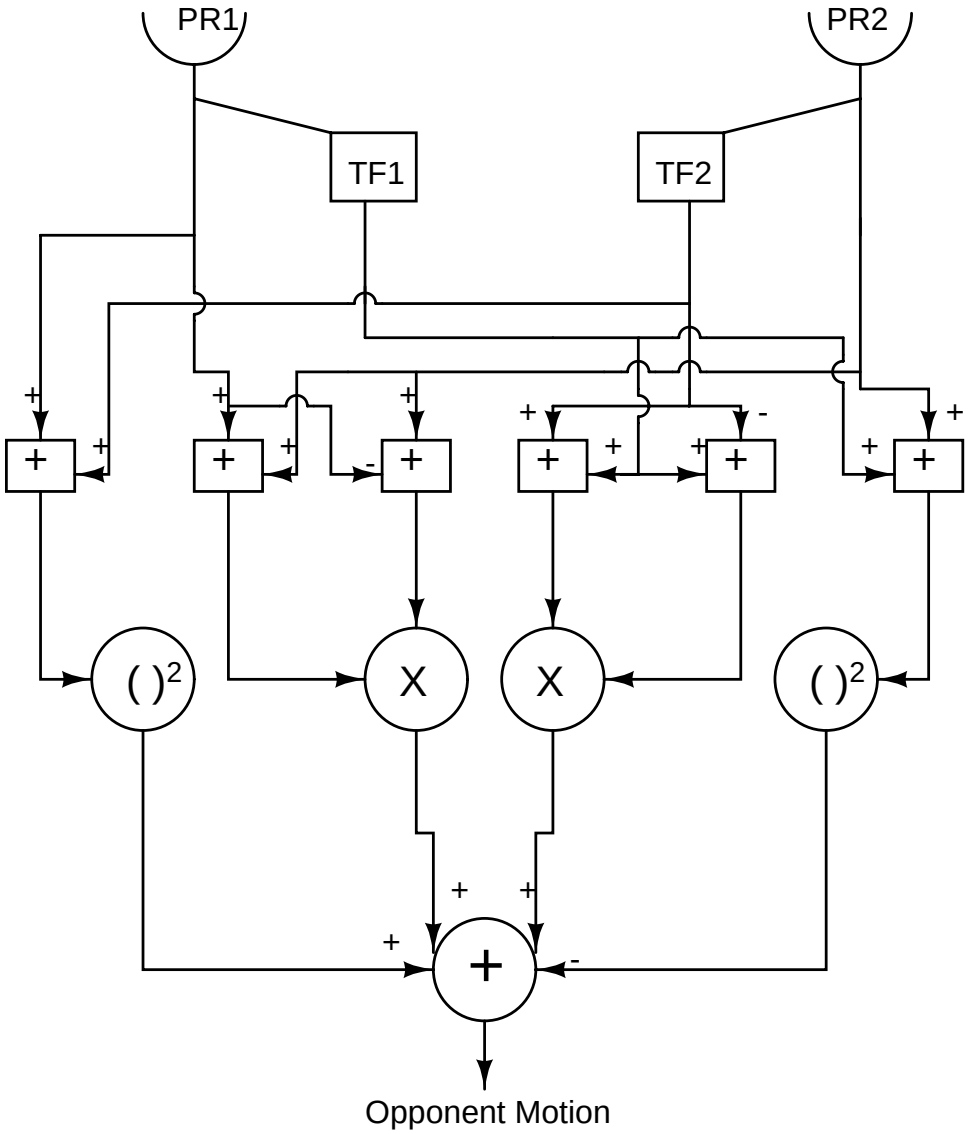
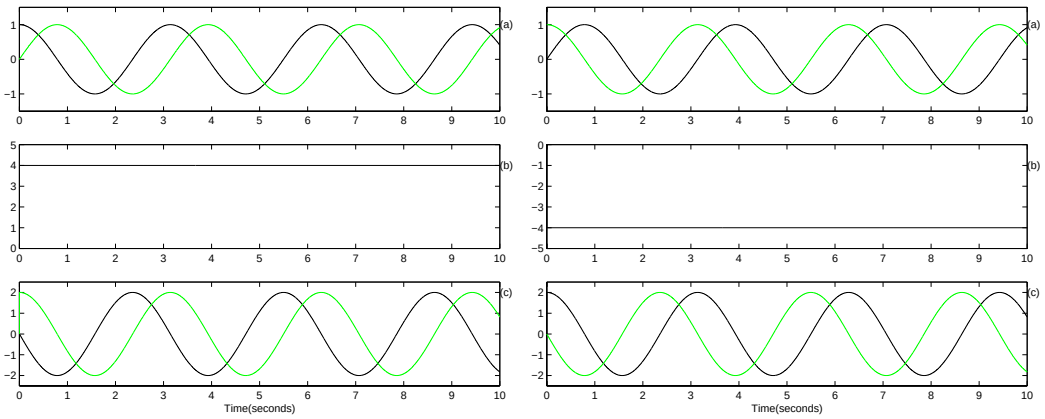


FIGURE 3.3. Proposed model for an EMD based on motion detection circuits in flies.

(a), shows the response of the two photoreceptors versus time. The darker trace corresponds to the response of PR1 and the lighter trace corresponds to the response of PR2. We can see that in the preferred direction, the stimulus first reaches PR1 and then reaches PR2 after a delay. And in the null direction it reaches PR2 before it reaches PR1. The third plot (c), in both figures shows the response after the temporal filtering stage. The darker trace corresponds to the response of TF1 and the lighter trace corresponds to the response of TF2. Similarly in both figures, the plots (b) correspond to the final opponent energy. We can see that the model is clearly direction selective. The response for stimulus in the preferred direction is positive and the response for stimulus in the null direction is negative.



(a) Preferred direction

(b) Null direction

FIGURE 3.4. In both figures, the top plot (a) shows the response of the photoreceptors. The darker trace shows the response of PR1 and the lighter trace corresponds to the response of PR2. The bottom plot (c) shows the response of the temporal filtering stage, again the darker trace corresponds to the response of TF1 and the lighter trace corresponds to the response of TF2. The middle plot (b) shows the response of the final opponent motion output. We can see that the model is clearly directional selective.

This work done by us has been pursued further and the elementary neuronal model has been modified to explain motion detection in flies. The proposed new model can be found in (Higgins *et al.*, 2001).

Chapter 4

VLSI Implementation of the Adelson-Bergen Algorithm

In this chapter we explain the hardware architecture and circuitry for a VLSI implementation of the Adelson-Bergen algorithm. The basic architecture we use can be understood from Figure 4.1. The first stage is the photodetection stage. In this stage the input from the image is projected onto an adaptive photodetector circuit. This adaptive photodetector circuit detects only changes in the image intensities and works for a very wide input range of intensities. The spatial filtering is done using a diffuser network which approximates a Gabor like spatial filtering. Next stage is the temporal filtering stage as shown. We use a voltage mode low pass filter for computing the delayed version of the input signals. The four separable signals obtained are combined as proposed by Adelson and Bergen in the original model. The next stage is the non-linearity stage. Adelson and Bergen propose the use of squaring the input signals at this stage. Computing the square of signals that can go both positive and negative involves more circuitry. So, instead of computing the square directly, we compute the square by first rectifying the input signal and then taking a square of the rectified signal. This is more efficient in transistor count. Further stages in the model, are additions and subtractions. By wiring the signals together, sums and differences are achieved through Kirchoff's current laws. The chip was fabricated in a standard $1.2\mu m$ CMOS process through MOSIS and the MOSFETS involved in the computational stages of the model operate in the subthreshold region keeping the power to a minimum. As subthreshold operation has an exponential $I-V$ characteristic for the MOSFET, the computations shown in the architecture are much easier to implement. We now describe the circuits used in the architecture in more detail.

4.1 Photodetection and Spatial Filtering

The photodetectors we use are the Delbrück adaptive photoreceptors (Delbrück and Mead, 1996). The adaptive photoreceptor has a high gain for transient light signals that are centered around a background adaptation point but has a low gain for steady background luminations. It encodes input light logarithmically and has a wide dynamic range of operation for input irradiance. The circuit of the adaptive photoreceptor is shown in Figure 4.2(a). M_n and M_p form an inverting amplifier. M_{fb} is a feedback transistor and M_{adap} is an adaptive element. C1 and C2 form a capacitive divider. Let us consider the case when there is a small change in light falling on the photodiode. This leads to a small increase in the photocurrent, i from the background current, I_{bg}. This increase tries to pull the voltage V_p down. This causes the voltage V_{prout} to go up A_{amp} times, where A_{amp} is the amplification factor of the inverting amplifier $M_n - M_p$. This increase in V_{prout} is coupled back onto the gate of the feedback transistor M_{fb} through the capacitive divider with a gain of $\frac{C2}{C1+C2}$, which is about 0.0916 in our case. This pulling up of the gate of M_{fb} pulls up on the source of M_{fb}, keeping the photoreceptor voltage, V_p nearly clamped. Thus a small change in the light intensity is amplified by the inverting amplifier after which it adapts back to the background. The adaptive element, M_{adap} acts as a very high resistance path for small variations in the input image intensity. But, it acts as a low resistance path for large variations in intensity. Thus small transients are coupled through the capacitive divider. The circuit adapts to large variations in the input image through the low resistance path provided by the adaptive element. Detailed analysis of the adaptive photoreceptor circuit and its noise properties are explained in (Delbrück and Mead, 1996).

Spatial filtering of the input image should ideally be performed by Gabor filters in quadrature as described in Chapter 2. But in actual hardware implementation we did not implement Gabor filters. Instead, the Gabor pair is approximated using adjacent photoreceptors in the array along with diffuser networks, which lead to an antagonistic center-surround spatial impulse response (Liu and Boahen, 1996), similar to that of a Gabor function. The width of these spatial impulse responses can be adjusted using diffuser networks, which are shown in Figure 4.2(b). These diffuser networks can be turned on by using the bias voltages V_g and V_h. In this figure $V_{fb}Left$ and $V_{fb}Right$ represent

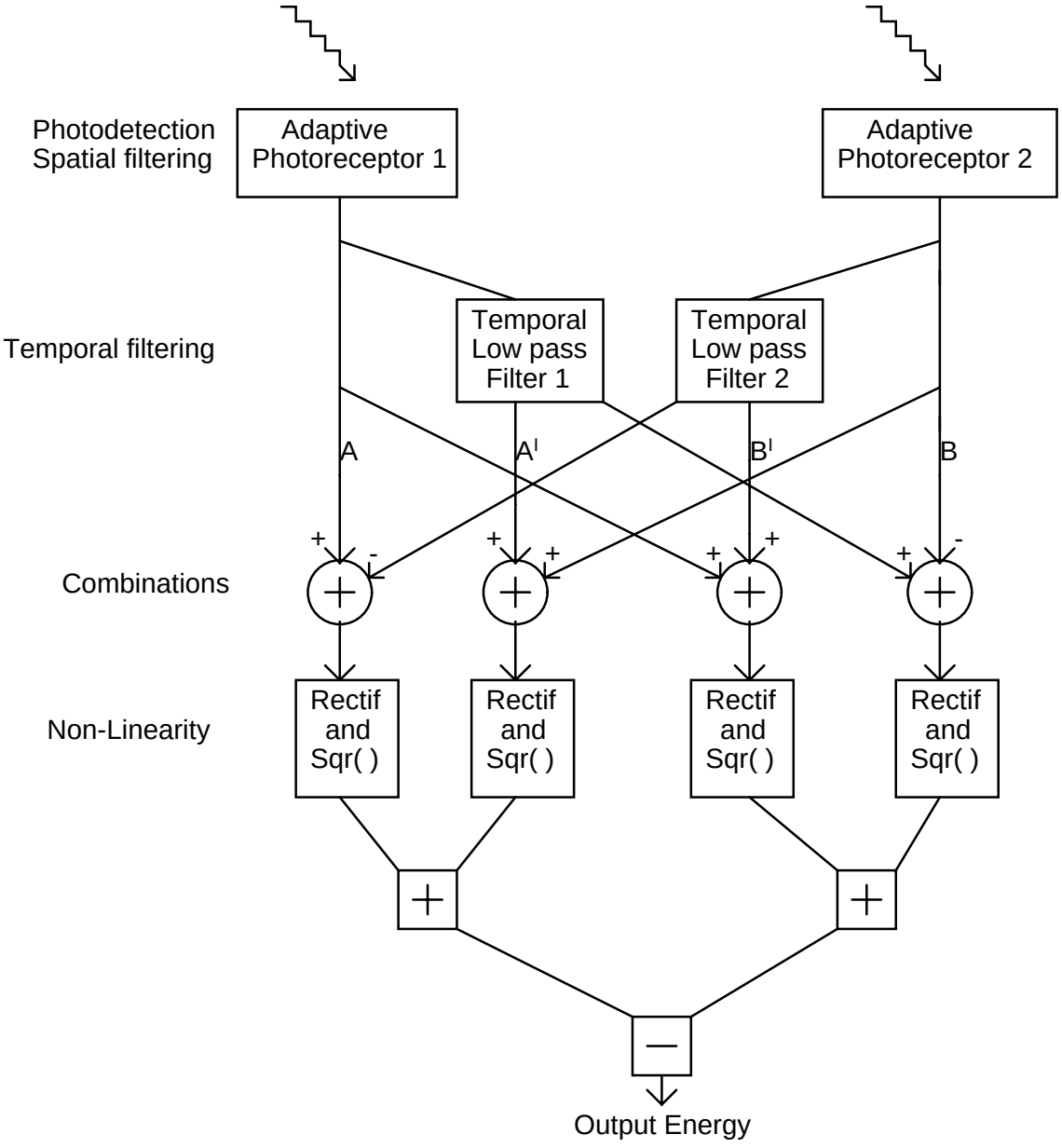
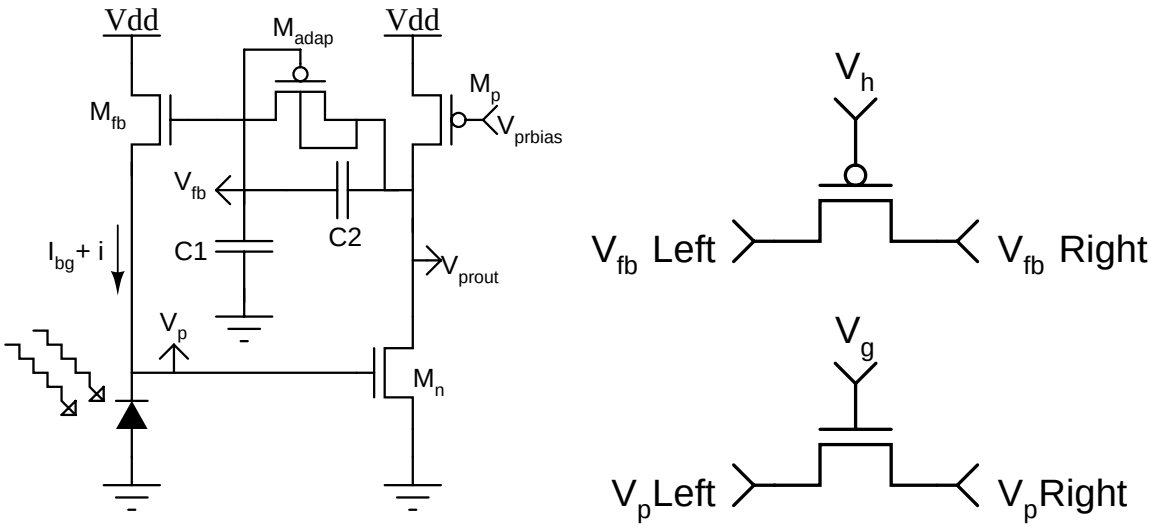


FIGURE 4.1. Architecture used in VLSI implementation of the Adelson-Bergen algorithm. The photo detection stage is performed by adaptive photoreceptors, spatial filtering is achieved by using diffuser networks. Temporal filtering is realized through voltage mode low pass filter. Non-linearity is achieved through rectification and squaring circuits. The summing and subtraction stages are performed using current mirrors and wires, as signals are kept in current mode.

the feedback voltage (V_{fb}), in the adaptive photoreceptor of the left and right pixels. Similarly V_pLeft and V_pRight represent the photoreceptor voltages of the left and right pixels respectively. By adjusting the bias voltages V_g and V_h we can control the width of the impulse responses of the spatial filters to approximate the quadrature Gabor filters. Although these diffuser networks are in place, we did not find the need to turn them on to get the exact Gabor function shape to obtain direction selectivity. As long as the mean DC value from the signals is removed the sensor has a good performance. We explain later in Section 4.4 how we achieve this.



(a) Adaptive photoreceptor

(b) Diffusers

FIGURE 4.2. (a) The adaptive photoreceptor circuit (b) Diffusers that are coupled with the photoreceptor for implementing spatial filtering.

4.2 Temporal Filtering

The delayed photoreceptor signal needed in the model is obtained by using a voltage mode low pass filter as shown in Figure 4.3. The delay is obtained by the phase lag inherent in a first order low pass filter. The circuit is a transconductance amplifier with a capacitive feed back element as shown. The output of the adaptive photoreceptor, V_{prout} is given as the input to the transistor M1 and the output, V_{prfilt} is the low passed photoreceptor voltage. We now show how this circuit acts as a low pass filter. The output current of the differential transconductance amplifier is given by (Mead, 1989):

$$C_{lpf} \cdot \frac{dV_{prfilt}}{dt} = I_b \cdot \tanh\left(\frac{\kappa(V_{prfilt} - V_{prout})}{2}\right)$$

Where, C_{lpf} is the capacitance of the feedback capacitor. I_b is the bias current in the differential pair, which can be adjusted by the bias voltage V_τ. κ is the back gate coefficient, whose value is process dependent and was found to be equal to 0.8. For small signals, tanh can be approximated by its argument as follows:

$$C_{lpf} \cdot s \cdot V_{prfilt} = \frac{I_b \cdot \kappa}{2} \cdot (V_{prfilt} - V_{prout})$$

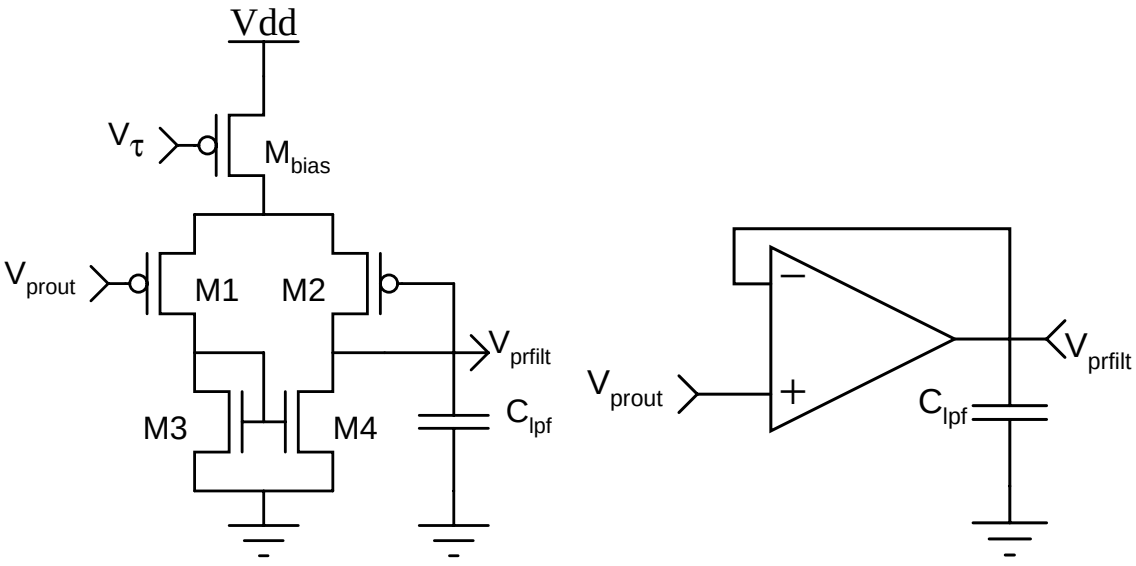
Rearranging the above equation, we can write the transfer function for the circuit as follows:

$$\frac{V_{prfilt}}{V_{prout}} = \frac{1}{\tau \cdot s + 1}$$

We can see that the circuit acts a first order low pass filter with a time constant τ, given by,

$$\tau = \frac{2 \cdot C_{lpf}}{I_b \cdot \kappa}$$

The time constant of the filter can be adjusted by changing the bias current in the circuit. The capacitor in the circuit, C_{lpf} was implemented with a MOSCAP in parallel with a poly1-poly2 parallel plate capacitor, with a combined capacitance of about 0.89 pF.



(a) Circuit (b) Symbolic representation

FIGURE 4.3. (a) Circuit to perform temporal low pass filtering on the photoreceptor signals. (b) Symbolic representation of the same circuit.

4.3 Non-Linearity

As explained previously, the original Adelson-Bergen model proposes a squaring of the incoming signals at this stage. But these signals can go both positive and negative. Although we can design circuits that can compute the square of such signals, they have a large transistor count. In order to over come this, we first fully rectify these signals by using an absolute value circuit and then perform the squaring.

The absolute value circuit is shown in Figure 4.4(a). This circuit is inspired by (Bult and Wallinga, 1987), where they propose it for above threshold operation. To understand this circuit let us first consider an input bi-directional current I_{in} at node $N1$. When I_{in} flows into the node $N1$, it flows into node $N2$ through the NFET M1. When I_{in} flows out of the node $N1$, it is taken

through the current mirror M2-M3 and flows into the node $N2$ again. Thus we can see that an input bi-directional current at node $N1$ is converted into a unidirectional current at node $N2$. Current I_{rect} always flows out of $N2$.

The squaring circuit is shown in Figure 4.4(b). The rectified current from the absolute value circuit flows into the node $N3$ and the squared current, I_{sq} flows into the node $N5$. Let the voltage at node $N3$ be V_a and the voltage at node $N4$ be V_b. This circuit utilizes the exponential $I-V$ relation of MOSFET operating in the subthreshold region to realize the squaring as shown below.

The currents flowing into the transistors M4, M5 and M6 neglecting the early effect can be written as:

$$\begin{aligned} I_{M4} &= I_0 e^{\frac{\kappa \cdot (V_a - V_b)}{V_T}} \\ I_{M5} &= I_0 e^{\frac{\kappa \cdot V_b}{V_T}} \\ I_{M6} &= I_0 e^{\frac{\kappa \cdot V_a}{V_T}} \end{aligned}$$

Where V_T is the thermal voltage ($V_T = kT/q = 25mV$, at room temperature). κ is the back gate coefficient. But, $I_{M4} = I_{M5} = I_{rect}$ and $I_{M6} = I_{sq}$. We can rearrange the above three equations as follows:

$$I_{rect} = I_0 e^{\frac{\kappa \cdot (V_a - V_b)}{V_T}} = I_0 e^{\frac{\kappa \cdot V_a}{V_T}} \cdot e^{\frac{-(\kappa \cdot V_b)}{V_T}} \tag{4.1}$$

$$I_{rect} = I_0 e^{\frac{\kappa \cdot V_b}{V_T}} \tag{4.2}$$

$$I_{sq} = I_0 e^{\frac{\kappa \cdot V_a}{V_T}} \tag{4.3}$$

Using Equations 4.2 and 4.3 in 4.1, we can draw the relation between the two currents, I_{sq} and I_{rect} as follows:

$$I_{sq} = \frac{I_{rect}^2}{I_0} \tag{4.4}$$

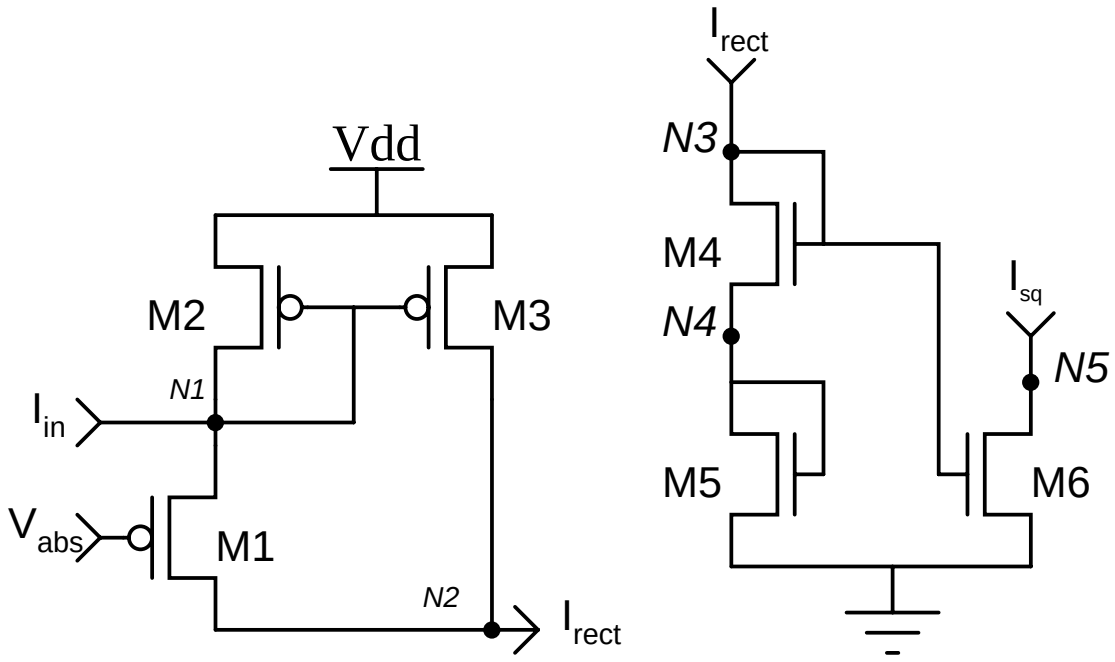
Thus we see that the circuit performs a squaring operation, scaled by a factor of I_0. However, we should note that the squaring circuit is not normalized and can operate above threshold if the current level is high after squaring. We describe a normalized squaring circuit in Chapter 7 that overcomes this problem.

4.4 Differential Current Representation

Before the current signals go into the non linear stage, we need to make sure that they do not have any offset current in them. A previous version of the implementation of Adelson-Bergen model (Higgins and Korrapati, 2000) had biases that had to be manually adjusted to subtract the offset currents. In this version we use a differential current representation scheme to get rid of the offset currents eliminating the need of extra biases in the circuit. Thus this scheme is self-compensative relative to the older version.

We can understand the differential current representation scheme from Figure 4.5. In Figure (a), we show how we obtain the four current signals, A, A', B, B' from voltage inputs of two adjacent pixels, 1 and 2 in the regular scheme. From the undelayed photoreceptor voltage V_{prout} of pixel 1 and 2, we obtain A and B. Similarly, the delayed current signals, A' and B' are obtained from the low pass filtered photoreceptor input, V_{prfilt} of pixel 1 and 2 respectively.

Figure 4.5(b) shows the generation of signals using the differential current representation. In this scheme the two voltage signals, V_{prout} and V_{fb} are used to obtain the undelayed current signals, A and B. Similarly V_{prfilt} and V_{fb} are used to obtain the delayed current signals, A' and B'. Notice



(a) Absolute value circuit

(b) Squaring circuit

FIGURE 4.4. (a) Absolute value circuit used to rectify the incoming bi-directional current, I_{in}. The output current, I_{rect} is rectified and flows out of the node $N2$. (b) The squaring circuit. It receives the rectified current, I_{rect} as the input and the output is a squared current, I_{sq} at the node $N5$.

that V_{fb} is actually a long term average of the scaled down version of V_{prout}. To obtain the undelayed current signals A and B, we take the difference of current signals generated by V_{prout} and V_{fb} as shown in Figure 4.5(b). By doing this we do not lose the transient nature of the signal, but the DC offset current is cancelled. Thus we obtain an offset free current signal which can be fed into the non-linearity stage. Similarly the figure also shows the generation of the delayed signals, A' and B'. Notice that ideally we need to take the difference between V_{prfilt} and the delayed feed back voltage (V_{fb}) to obtain the delayed signals, A' and B'. But that would involve one more temporal filter to obtain $V_{fbdelayed}$. In the actual implementation we do not do that as it would cost us more transistors. So, we approximate $V_{fbdelayed}$ to V_{fb} and take the difference as shown in the figure. Another thing to be observed in this scheme is the signal, V_{fb}. Ideally, we need to take the difference between V_{prout} and a scaled down version of V_{prout} to obtain the signals A and B. V_{fb} is not just a scaled down version of V_{prout}, but a long-term average of the scaled down version of V_{prout}. So, it does not have an identical frequency response as that of V_{prout}. We need additional circuitry to obtain an exact scaled down version of V_{prout}, which we cannot afford. So, we approximate the scaled down version of V_{prout} to V_{fb} and take the differences as shown.

4.5 Readout Circuitry

Each pixel in the two dimensional array gives out an opponent motion current and other intermediate signals. In order to read these signals from the pixel we need to scan these signals from each pixel. We use horizontal and vertical scanning circuits to do this. The scanner circuits can be seen in the layout of the chip shown in Figure 4.7. The scanner circuits in the chip are based on the scanners proposed in (Mead and Delbrück, 1991). The scanners operate on a single-phase clock going from V_{dd} to *ground*. The design of both vertical and horizontal scanner is similar. Each scanner has a shift register in it. The shift register has flip flops in it to store a binary state. Each flip flop selects a row or a column. A logic high in a flip flop of a horizontal scanner selects the particular row. Similarly a logic high in a flip flop of a vertical scanner selects the particular column. By continuously shifting bits from one flip flop to the other, we can select adjacent pixels continuously and read signals off them. We can also select a particular row and a column by sending the appropriate number of clock pulses. By selecting a particular row and a column we can read data from the same pixel continuously. More details about the circuitry involved can be found in (Mead and Delbrück, 1991).

4.6 Characterization

Using the circuits discussed in the previous sections and connecting the signals based on the AB model, we fabricated a motion sensor. The complete schematic of a pixel is shown in Figure 4.6. The layout of the chip is shown in Figure 4.7. Figure 4.8 shows the layout of a pixel detailing all the circuitry explained previously.

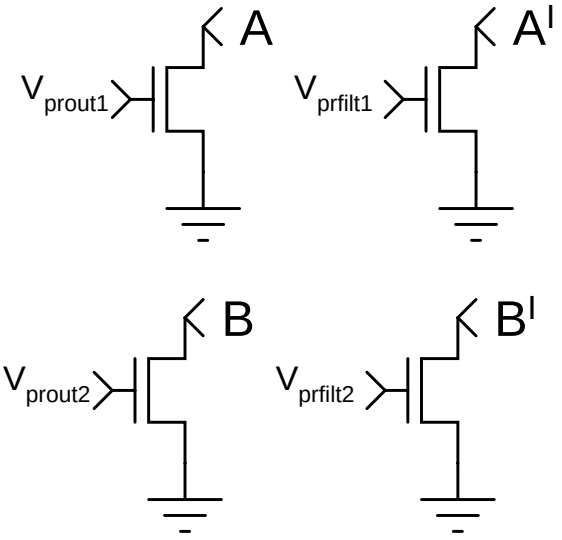
Before we describe the characterization data from the chip in detail, let us first look at the expected response from the sensor. If the input to the sensor is a sinusoidal grating with an amplitude A, contrast C, spatial frequency f_s, temporal frequency f_t and if the grating has an orientation of θ with respect to the preferred orientation of the sensor, then the input stimulus can be written as:

$$I(x,y,t) \quad = \quad A \cdot (1 + C \cdot \sin(2\pi f_t \cdot t + 2\pi f_s(cos\theta \cdot x + sin\theta \cdot y)))$$

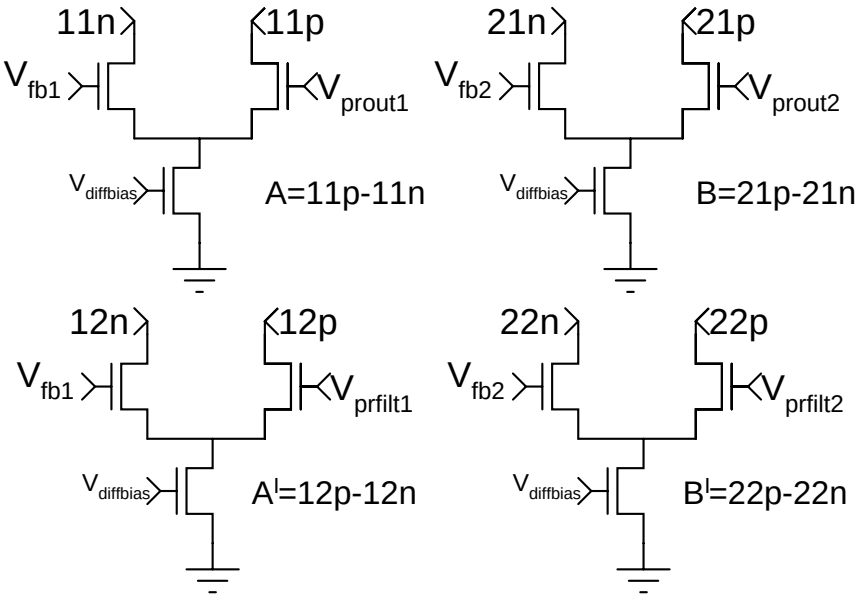
The adaptive photoreceptor circuit removes the background intensity in the stimulus and so we have the four separable signals going into the computation stages in the model:

$$\begin{aligned}
A &= A \cdot C \cdot \sin(2\pi f_t \cdot t + 2\pi f_s(cos\theta \cdot x + sin\theta \cdot y)) \\
A' &= A \cdot C \cdot H(f_t) \cdot \sin(2\pi f_t \cdot t + \phi_t(f_t) + 2\pi f_s(cos\theta \cdot x + sin\theta \cdot y)) \\
B &= A \cdot C \cdot \sin(2\pi f_t \cdot t + 2\pi f_s(cos\theta \cdot (x+\lambda) + sin\theta \cdot y)) \\
B' &= A \cdot C \cdot H(f_t) \cdot \sin(2\pi f_t \cdot t + \phi_t(f_t) + 2\pi f_s(cos\theta \cdot (x+\lambda) + sin\theta \cdot y))
\end{aligned}$$

Where $\phi_t(f_t)$ is the phase introduced by the temporal low pass filter and $H(f_t)$ is its magnitude. λ is the separation between the two pixels. The final motion opponent energy according to the model is $4(AB' - A'B)$. Substituting the expressions and denoting $\phi_t(f_t)$ as ϕ_t, we obtain:



(a) Without using the differential current representation scheme



(b) Using differential current representation

FIGURE 4.5. (a) Circuits to obtaining signals A, B, A', B' without using differential current representation. (b) Circuits to obtain signals A, B, A', B' using differential current representation.

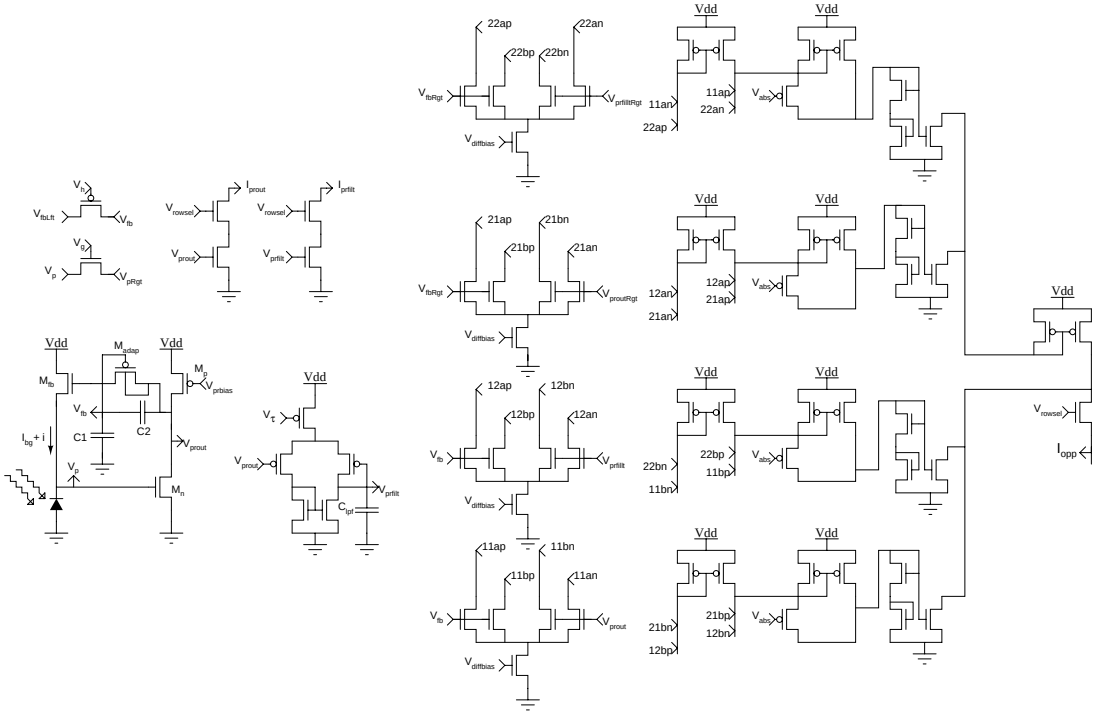


FIGURE 4.6. Schematic of a pixel showing all the circuits explained in the previous sections.

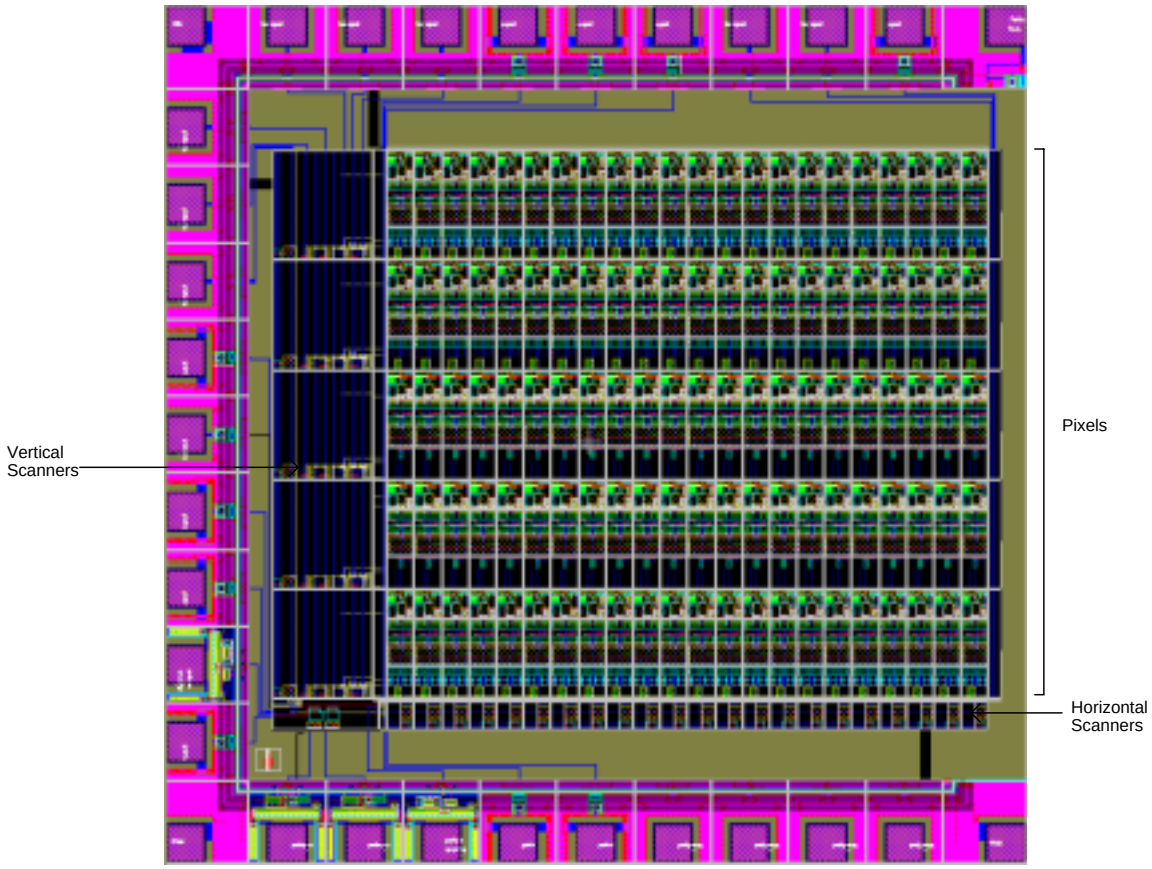


FIGURE 4.7. Layout of the chip, with a 5×22 array of pixels. The vertical and horizontal scanners are also shown in the figure. The chip was fabricated in a standard $1.2\mu m$ process and the die size was 2.2mm×2.2mm.

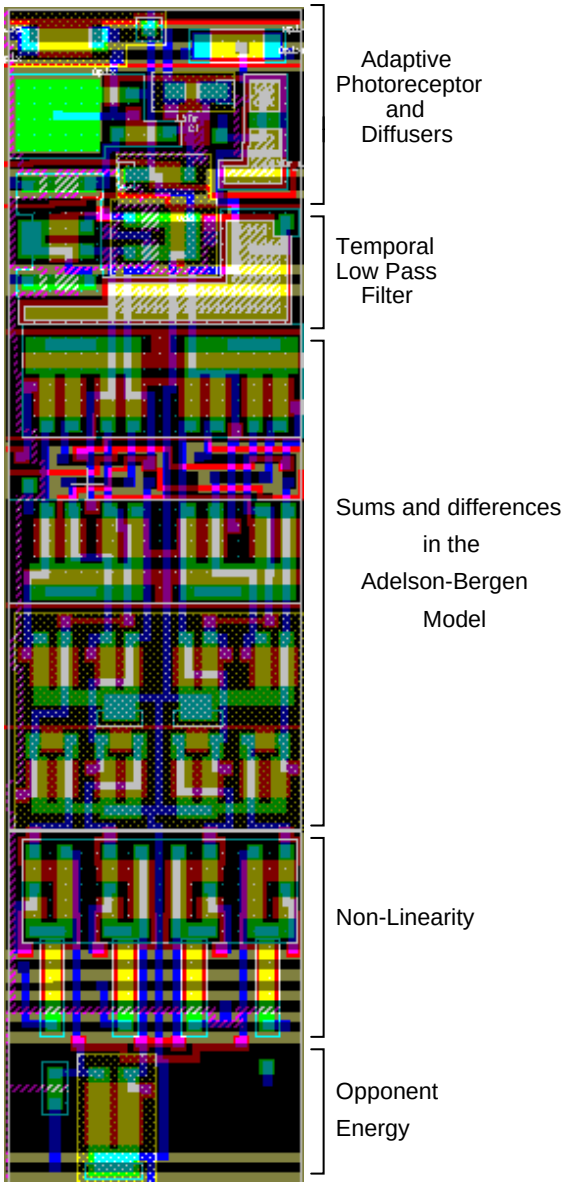


FIGURE 4.8. Layout of a pixel showing all the stages explained in the previous sections.

$$\mathbf{O} = 4 \cdot A^2 C^2 H(f_t) \begin{bmatrix} \sin(2\pi f_t \cdot t + 2\pi f_s(cos\theta \cdot x + sin\theta \cdot y)) \cdot \\ \sin(2\pi f_t \cdot t + \phi_t + 2\pi f_s(cos\theta \cdot (x+\lambda) + sin\theta \cdot y)) - \\ \sin(2\pi f_t \cdot t + \phi_t + 2\pi f_s(cos\theta \cdot x + sin\theta \cdot y)) \cdot \\ \sin(2\pi f_t \cdot t + 2\pi f_s(cos\theta \cdot (x+\lambda) + sin\theta \cdot y)) \end{bmatrix}$$

Using the trigonometric identity $2 \sin A \sin B = \cos(A-B) - \cos(A+B)$, we can simplify the above expression and rewrite the opponent motion energy as follows:

$$\mathbf{O} = 4 \cdot A^2 C^2 H(f_t) \cdot \sin(2\pi f_s \lambda cos\theta) \cdot \sin(\phi_t) \tag{4.5}$$

From this expression we can observe that the opponent energy varies quadratically to variation in contrast. Also, it varies sinusoidally with the stimulus orientation, which gives a positive response for orientations between 0 to 180^0 and a negative response for orientations between 180^0 and 360^0. The opponent energy is a large positive quantity for a stimulus in the preferred direction and a large negative quantity for a stimulus in the null direction. It would be zero for stimuli in orthogonal orientations.

When we look at the spatial frequency response, the opponent energy is maximum at a spatial frequency where $2\pi f_s \lambda \cos\theta = \pi/2$. We express the spatial frequency in cycles/pixel. So, the peak of the spatial frequency variation plot can be expected to occur at $f_s \cos\theta = 0.25$ cycles/pixel. The temporal frequency tuning plot is governed by the term $H(f_t) \cdot \sin(\phi_t(f_t))$. The temporal filter in our sensor was shown to be a first order low pass filter in Section 4.2. The product of the magnitude and sine of the phase of a first order low pass filter can be shown to be symmetric when we plot it against log frequency and peaks at the 3 dB frequency. So, we should see the peak of the frequency tuning plot at about the 3 dB frequency of the filter and it should have a symmetrical response.

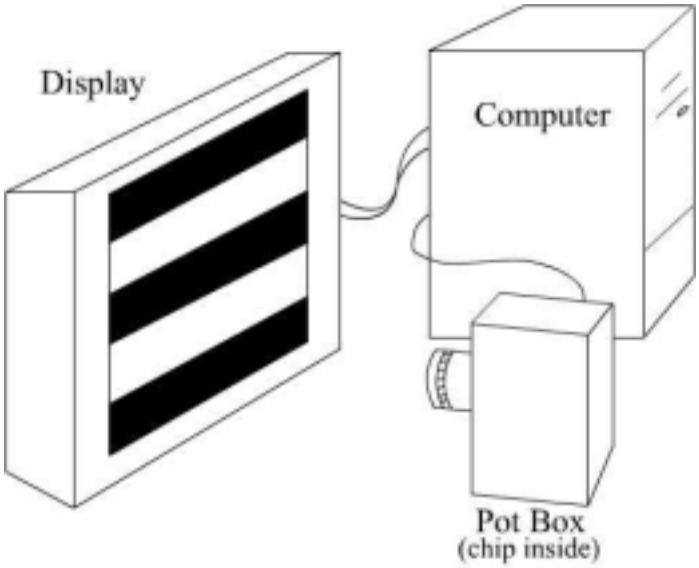
We now give results from the detailed characterization of the chip. For all the experiments the setup is shown in Figure 4.9(a). Figure 4.9(b) shows a photograph of the setup on the work bench. The chip is placed in a "pot box", which has a bread board and potentiometers on it to generate the biases for the chip. The top of the chip is covered with an 8mm CS mount lens which projects the visual scene onto the die of the chip. The die is 2.2mm×2.2mm in size and has an array of 5×22 pixels on it. The stimuli are generated with a computer and are displayed on an LCD monitor as shown.

The opponent energy from the sensor is a current output. So, we use a sense amplifier with a $20K\Omega$ resistor in the feedback path as shown in Figure 4.10 to convert the current into a voltage output. The power consumption of the chip was measured to be 41μW. A single program generates the stimulus and reads data from the chip. The data from the chip is read into the computer through a data acquisition card. During all the experiments we explain below the bias voltages are held constant. Also, when an experiment is being conducted by sweeping a particular parameter, all other parameters are held constant. The output voltages from the sensor are averaged over ten temporal cycles of the stimulus to remove the phase dependence.

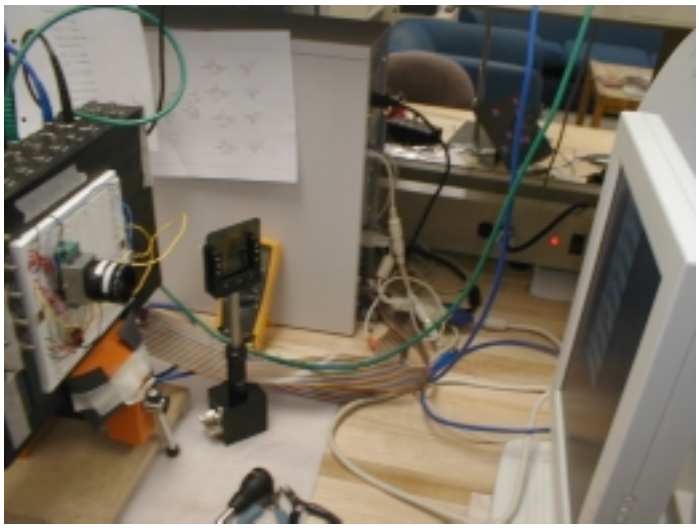
Figure 4.11 shows the raw data generated by the sensor. We can see two traces in this plot. The lighter trace in the background is the raw opponent output from the chip. The darker trace in the foreground is the temporally averaged version of the raw opponent output. During the first interval, no stimulus is displayed on the monitor and we can see the chip reacting to the background fluorescent light. The next interval shows the response of the chip to a stimulus in the preferred direction. Similarly, the response of the chip when an orthogonal stimulus is presented is shown in the third interval. Finally the last interval shows the response of the chip to a stimulus in the null direction.

The response time of the chip, that is, the time it takes for the chip to detect the direction of motion when the input image is a step, is about 50ms with a tolearnce of about 30ms because of error from sampling. The response time was calculated when the opponent motion energy increases from the base value to about 90 percent of the peak value when it detects the motion.

Figure 4.12 shows an orientation sweep of the stimulus. That is, the orientation of the stimulus with respect to the preferred direction of the sensor is swept from 0 to 360^0. As expected the response to an orientation sweep is a sinusoid. Ideally, the response should be symmetrical in both directions, but there is a slight asymmetry because of mismatch in the circuitry.



(a) Sketch of the setup



(b) Photograph of the setup

FIGURE 4.9. (a) A sketch of the setup used in conducting the experiments. The sensor is placed in the pot box and stimulus is displayed on a LCD monitor. A lens focuses the stimulus onto the sensor. Data is read into the computer through a data acquisition card. (b) Photograph of the setup on the work bench.

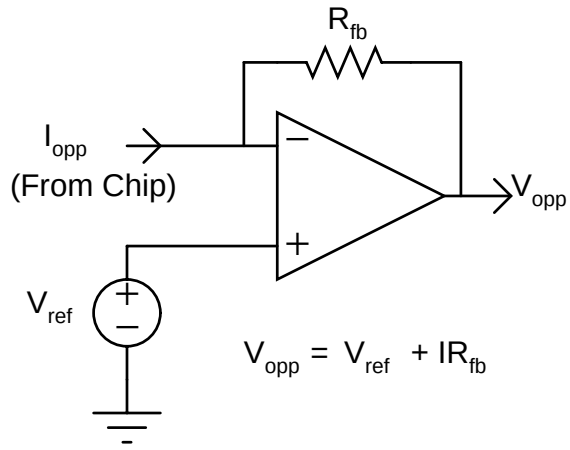


FIGURE 4.10. Sense amplifier circuit. The opponent motion current from the chip is fed into an external operational amplifier with a feedback resistor as shown to get a voltage output, V_{opp} which is fed into the data acquisition card.

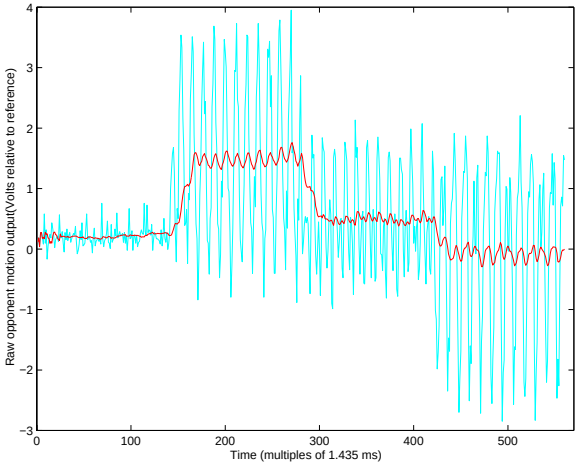


FIGURE 4.11. Raw output from the motion sensor. There are two traces, the lighter trace shows the actual raw output from the chip and the darker trace shows the temporally averaged version of the data. For the first interval there is no stimulus, during the next interval the stimulus is presented in the preferred direction, next it is presented in the orthogonal direction. In the last interval, the stimulus traverses in the null direction.

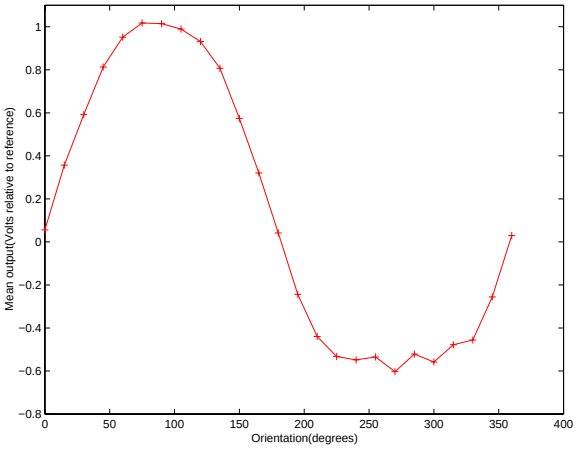


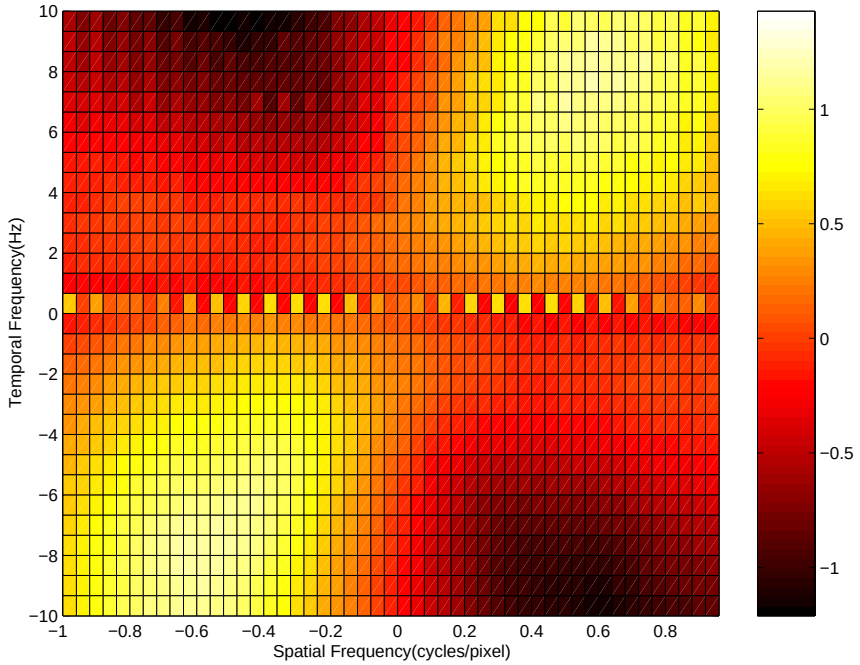
FIGURE 4.12. The opponent motion energy of the sensor is plotted when the orientation of the stimulus is varied from 0 to 360^0. The sensor is optimally tuned for a stimulus at 90^0.

Figure 4.13(a) shows the spatio-temporal frequency tuning of the sensor. That is, each point in this plot is the opponent motion output at a particular spatio-temporal frequency of the stimulus. The spatial frequencies are plotted on the X-axis and the temporal frequencies on the Y-axis. The plot shows that the performance of the chip closely resembles the theoretical prediction as discussed in Chapter 2. From the plot we can see that the opponent motion energy is positive for stimuli in the preferred directions (first and third quadrants) and negative for stimuli in the null directions (second and fourth quadrants). This clearly indicates that the chip is direction selective for a wide range of frequencies. Also, the response of the chip is maximum for a particular spatio-temporal frequency and gradually wanes as we move away from it as explained previously when discussing the theoretical model.

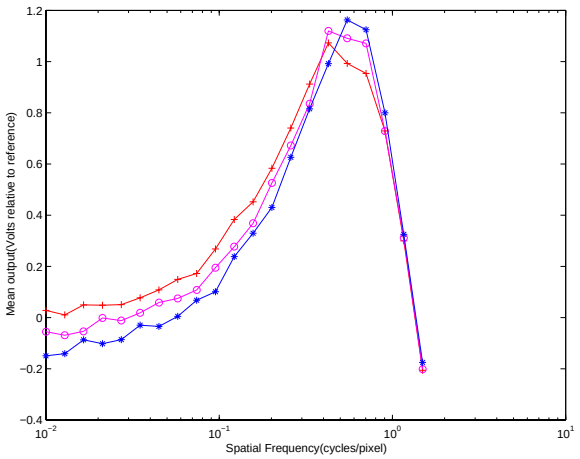
Figures 4.13(b) and 4.13(c) show the opponent motion output by varying the spatial and temporal frequencies respectively. In plot 4.13(b) we show response of the chip by varying the spatial frequency of the stimulus. There are three traces in this plot, each trace corresponds to a different temporal frequency of the stimulus. From this plot we can see that the opponent motion energy is maximum at about a spatial frequency of 0.25 cycles/pixel as expected. Similarly in plot 4.13(c) we show the response of the chip by sweeping the temporal frequency. Again each trace in the plot corresponds to a particular spatial frequency. We can see that the opponent motion is almost symmetrical as expected and has a peak at about 6Hz.

Although the sensor is tuned for a particular spatio-temporal frequency, we can adjust the time constant of the low pass filter on the chip and change the temporal frequency tuning of the sensor. In order to vary the time constant, we need to vary the bias current in the low pass filter circuit as described in Section 4.2. Figure 4.14(a) shows this variation in the temporal frequency tuning of the sensor. Each trace in the plot is obtained with a different bias setting for the low pass filter circuit.

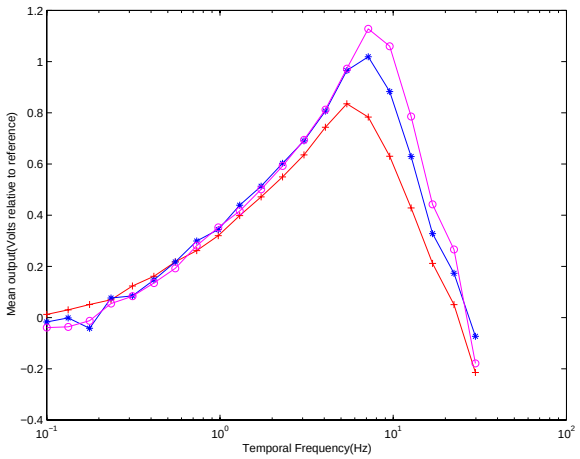
Figure 4.14(b) shows the response of the sensor when the contrast of the stimulus is varied. In this plot we show the results from varying the stimulus in both the preferred and null directions by changing the contrast of the stimulus. As expected the variation of the opponent energy with contrast is almost quadratic in the preferred direction. But, it is not strictly quadratic in the null direction because of mismatches in the circuits. We can see that the sensor can distinguish the direction of motion to approximately 10% contrast.



(a) Spatio-temporal frequency plot

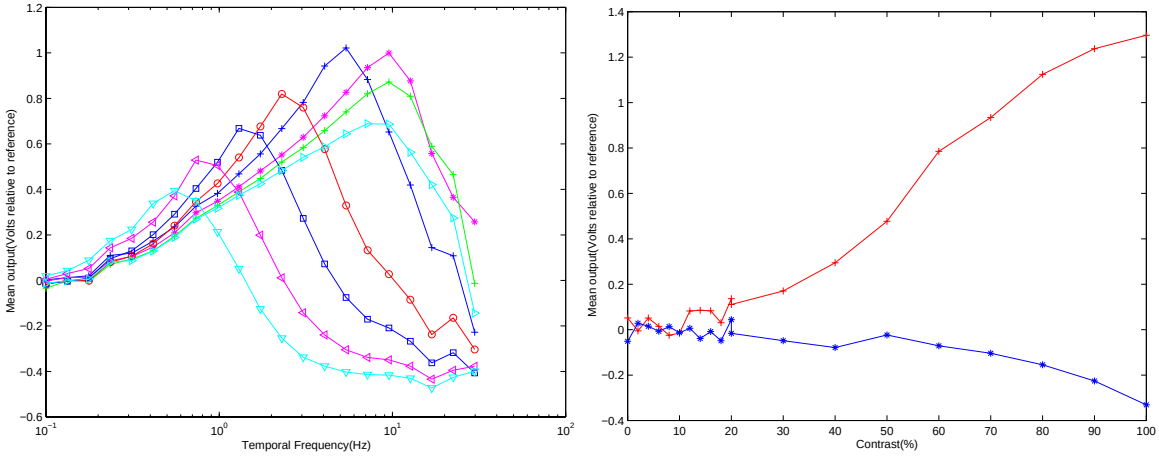


(b) Spatial frequency sweep



(c) Temporal frequency sweep

FIGURE 4.13. (a) Spatio-temporal frequency tuning of the chip: Light colors indicate positive average responses and darker colors indicate negative average responses. (b) Spatial frequency sweep showing the opponent motion output, the three traces show the motion output at three different temporal frequencies. (c) Temporal frequency sweep showing the opponent motion output, the three traces show the motion output at three different spatial frequencies.



(a) Temporal frequency tuning

(b) Contrast sweep

FIGURE 4.14. (a) Varying the temporal frequency tuning of the plot. Each trace in the plot shown here corresponds to a different bias voltage V_τ, of the low pass filter circuit, which changes its time constant. (b) The contrast of the stimulus is varied in this plot. We can see that the sensor can distinguish motion down till about 10%. The difference of motion energy between the preferred and null directions diminishes when the contrast goes low.

Chapter 5

An Active Tracking System Based on the Motion Sensor

In this chapter we describe a closed loop control mechanism for active tracking based on the motion sensor described earlier. The goal of active tracking is: the motion sensor is mounted on a base that can rotate. The base is mounted on a rotating platform. The sensor should be able to stabilize the base, i.e., cancel the effect of the rotation of the platform by controlling the rotation of the base using visual motion input from the scene. This setup is shown in Figure 5.1. However, we do not use this experimental setup. Instead, to realize this we use the same arrangement as described in Chapter 4. We do not have a rotating platform on which we mount the chip. The sensor is stationary and pointed at an LCD monitor as shown in Figure 4.9. The stimulus we present on the monitor is a sinusoidal grating moving at the relative velocity, between the platform velocity and the velocity of the base, thus producing the same effect as mounting the chip on a rotating platform.

Initially the grating on the monitor starts off with some velocity moving in either the preferred or null direction. The velocity of the platform is given as input to the control loop. The response from the chip which is the error signal, is continuously read and the velocity of the grating is corrected. That is, it is slowed down till the grating stabilizes itself on the screen and the velocity of the grating reaches zero. We used two methods for this closed loop control that can be easily translated into hardware for performing active tracking. We now describe these two methods in detail.

5.1 Method 1

The closed loop control used in this case is shown in Figure 5.2. V_{screen} is the velocity of the grating that is displayed on the monitor. $V_{stimulus}$, which is given as the input to the control loop is the velocity of the platform. The stimulus is started off with an initial velocity and is presented on the chip. The chip feeds back the opponent energy, which acts as the error signal. This signal V_{chip}, is multiplied by the feedback parameter, K_p, and is subtracted from the velocity of the platform, $V_{stimulus}$. Thus the screen velocity is corrected till it is finally stabilized and reaches zero. We can express the control loop shown in the block diagram of Figure 5.2:

$$V_{scene} = V_{stimulus} - K_p \cdot V_{chip} \tag{5.1}$$

Figure 5.3(a) shows the results from using this control scheme. This figure plots the screen velocity of the grating against time. It has three traces in it and each trace corresponds to a different value of the parameter K_p. In this experiment we let the grating start off with an initial high velocity. The control system then tries to counter the rotation of the platform using the error signal. We can see from the figure that the system stabilizes and the velocity reaches zero. When we let the system run for a while after it reaches stability, the screen velocity oscillates around zero as expected. The darker traces in the plot correspond to smaller values of K_p and the lighter traces correspond to larger values of K_p. From the figure we can see that when the value of K_p is large, the oscillation in the screen velocity after it reaches zero is large since we are now correcting the screen velocity by a larger amount (see Equation 5.1). We performed a second experiment in which we vary the feedback parameter K_p and measure the time to reach zero velocity. Figure 5.3(b) shows the results from this experiment. In this figure we plot time on Y-axis and the feedback parameter on X-axis. We can see that it takes a longer time to reach zero velocity with a smaller feedback parameter, but there is a limit beyond which increasing the value of K_p does not help anymore. It only increases the amplitude of oscillation after V_{scene} reaches zero.

5.2 Method 2

In this method we also include the rate of change of error signal in the control loop as shown in Figure 5.4(a). As in method 1, if V_{screen} is the velocity of the grating displayed on the monitor,

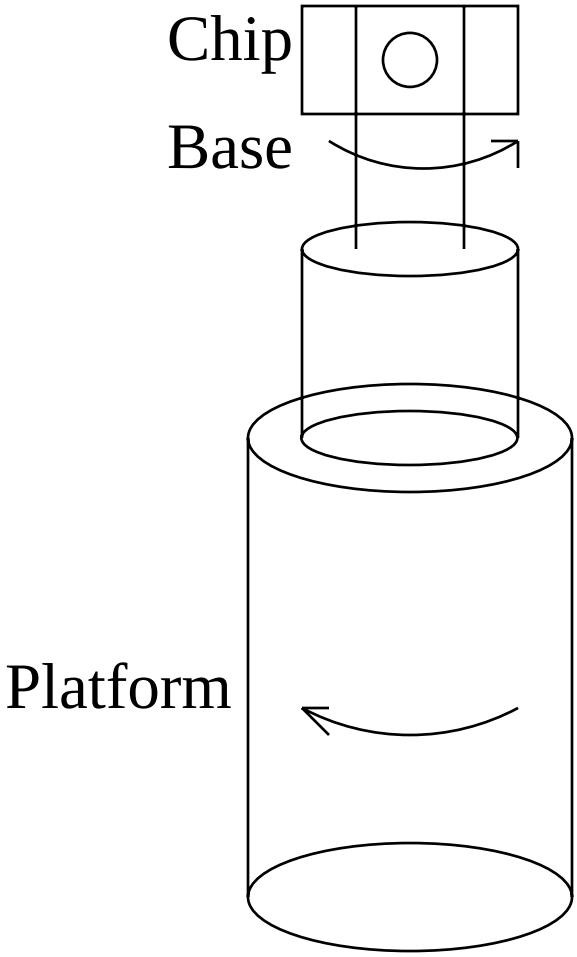


FIGURE 5.1. Experimental setup for active tracking. The chip is mounted on the base. The base is on a rotating platform. The goal of the chip is to control the velocity of the base and compensate for the rotation of the platform based on the visual motion cues in the scene.

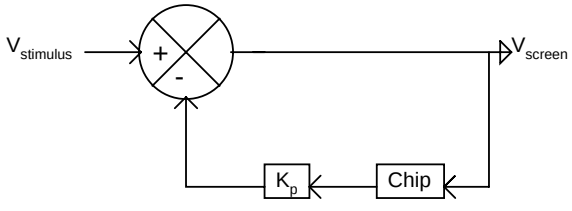
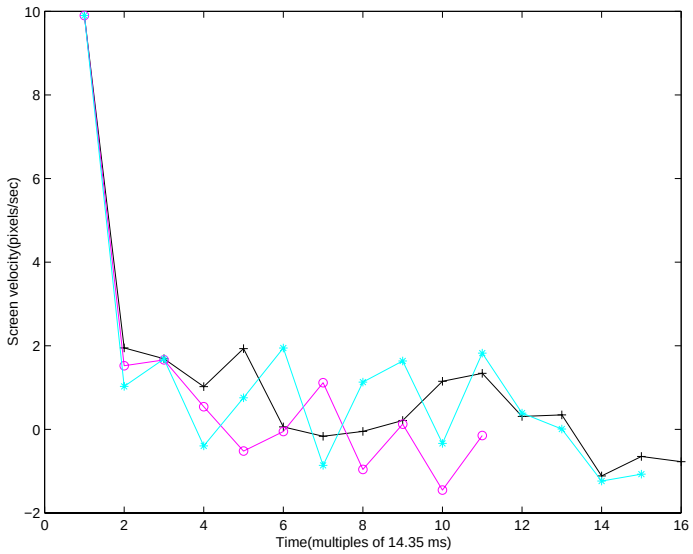
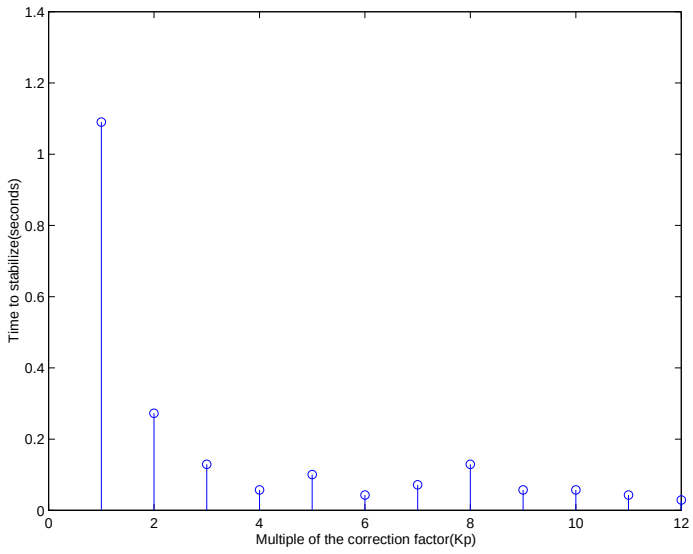


FIGURE 5.2. First form of closed loop control. The velocity of the grating to be displayed on the monitor is corrected based on the feedback from the chip.



(a) Experiment 1.



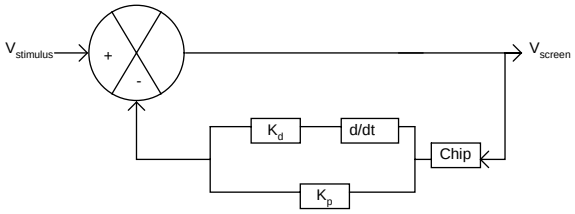
(b) Experiment 2.

FIGURE 5.3. (a) This plot shows the performance of the first closed loop scheme by varying the feedback parameter K_p. Darker traces have larger value of K_p than the lighter traces. We can see that when the feedback parameter, K_p is small, it takes a longer time for the system to stabilize and reach zero. (b) This plot shows the results from the experiment in which we measure the time to reach zero velocity by varying the feedback parameter K_p. The value of K_p is started at 0.068 and incremented in steps of 0.002.

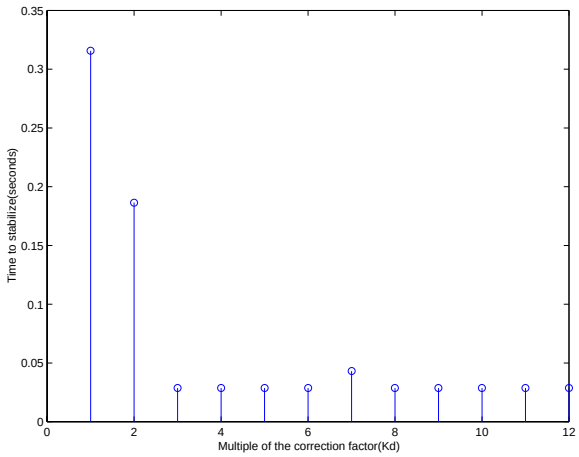
$V_{stimulus}$, the velocity of the platform and V_{chip}, the correction factor from the chip, the control loop can be expressed as:

$$V_{screen} \quad = \quad V_{stimulus} - K_p \cdot V_{chip} - K_d \cdot V'_{chip} \tag{5.2}$$

The results from using this scheme are show in Figure 5.4(b). In this experiment we vary the value of the parameter K_d (for a fixed value of the parameter K_p), and observe the time to reach zero velocity. We can also observe from this plot that the time to reach zero velocity decreases when compared with method 1. That is, when we also include the rate of change of error in the feedback of the control loop the performance of the system improves.



(a) Control system.



(b) Experimental results.

FIGURE 5.4. (a) Second form of closed loop control. The velocity of the grating to be displayed on the monitor is corrected based on both the error signal from the chip and the rate of change of error. (b) This plot shows the performance of the second closed loop scheme. We measure the time to reach zero velocity by varying the feedback parameter K_d. The value of K_p was fixed at 0.068 and K_d was incremented in steps of 0.04.

From the above two simulations we can see that this chip can be used in real-time closed-loop control to correct the velocity of a moving grating. This has obvious applications in camera image stabilization and others.

Chapter 6

Robot on a Chip

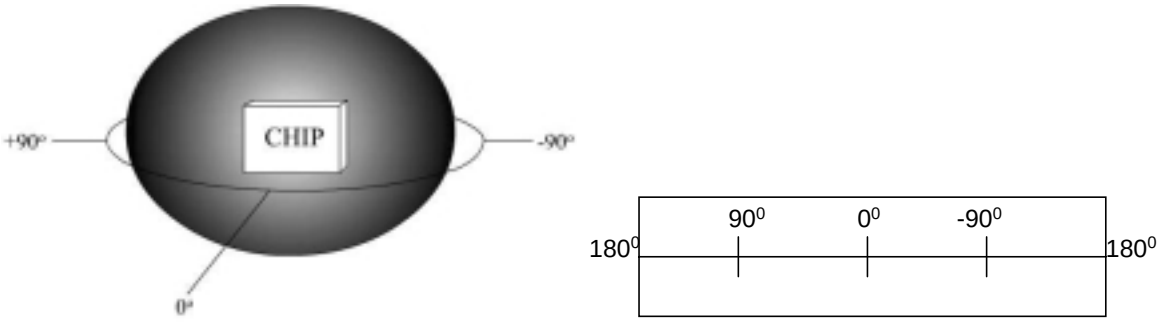
In this chapter we describe a second application based on our motion sensor. It is the design of a chip, RoaCh(Robot on a Chip). A robotic platform typically has sensors on board and a processor which fuses all the sensory information and generates the necessary commands for its navigation or any other task it has to perform. Most often this implies taking real world continuous time sensory signals, transforming them into the digital domain and feeding them into the processor. All this would mean sampling the continuous signals and then using the processor for control. One can easily imagine a situation where the sampling process might lead to a loss of information because of the very nature of sampling. Also, the processor running on a clock would be consume a lot of power. A work-around for the loss of information would be to increase the resolution of the digital signal sent by the sensors, meaning, increasing the number of bits. But this would counteract the idea of decreasing the power of the entire system as one would have more bits to deal with. Thus, the idea of moving the control system onto the single monolithic block along with the sensors is a natural extension to the project when considering the application of motion sensors to robotics.

A simple robot with the entire sensory system and control circuitry on a single chip: Robot on a Chip(RoaCh) is now described. The objective of the robot is the following: The robot is first stationary at a position, it keeps checking for motion in a 360-degree field of view. Once it detects motion in either its left or right eye, it turns the opposite direction and starts running away from it. After a while it stops and goes back to its original stationary state and keeps checking for motion. The idea of having a left and right eye on RoaCh is different from the conventional idea of having two different sensors, each acting as an eye. There are not two vision chips acting as the left and right eye. Instead, there is only one vision chip, but the entire linear array of pixels is divided into two halves. The one to the left is called the left eye and the one to the right, the right eye. This arrangement can be understood from Figure 6.1. Figure (a) shows the placement of the chip and the 360^0 field of view around it and (b) shows how the field of view is projected on the chip's one dimensional linear array. A lens-mirror system can be used to project the left hemisphere of the entire 360 degree field of view onto the left part of the array. Similarly, it can project the right hemisphere of the field of view of the robot onto the right part of the linear array (Chahl and Srinivasan, 1997). An other alternative that could be used for such an arrangement instead of lenses and mirrors is using fiber optic image conduits available commercially (Edmund Optics Online, 2001).

In this section we describe the control system needed for such a task and the circuitry needed to implement it. The results from circuit simulations using SPICE are also presented.

6.1 Control Scheme

The control scheme used for this system can be understood from the block diagram shown in Figure 6.2. As shown in the top part, the entire linear array of $2n$ motion pixels is divided into two halves, called the left eye and the right eye. The left eye has the Adelson-Bergen(AB) motion pixels $-n$ to -1 in it and the right eye has the AB motion pixels 1 to n in it. Each AB motion pixel as discussed in the previous chapters has a photodetector stage and a subsequent signal processing stage which implements the Adelson-Bergen algorithm. The output of such a motion pixel is a motion energy current. The motion energy current is denoted as $I(j)$ in the block diagram where j is the number of the pixel in the array. This motion energy current is positive for motion detected in the preferred direction and negative for motion detected in the null direction. The RoaCh does not use this information to compute the direction of motion, instead, it checks if there is motion detected at all in either eye and uses this to generate a turn in the opposite direction, i.e., turn left if motion is detected in the right eye and vice versa. It does not matter if the motion is in the preferred or null direction. So, each of these currents is taken through an absolute value circuit, shown as ABS block in the block diagram. These currents after taking the absolute value are represented as $|I_j|$ in the block diagram. The currents, $|I_j|$, from the same eye are summed together, shown as I_L for the left



(a) Positioning of chip

(b) Projection onto the chip

FIGURE 6.1. (a) 360 degree field of view around the robot. (b) Projection of the world around the chip onto its one dimensional linear array.

eye and I_R for the right. After summing them they are compared with a threshold to check if there is motion detected in that eye. If the summed current is greater than the threshold, it indicates a motion detected in that eye and a motion pulse for the opposite direction goes high as shown in the block diagram. A motion pulse, ML is generated to indicate that motion has been detected in the right eye and the robot has to turn left. Similarly MR goes high when the robot has to turn right.

After detecting motion, the RoaCh should determine how long the robot should turn. For this, the RoaCh uses a spatial position encoding circuit as shown in the block diagram. That is, the length of time the robot turns depends on the position of the pixel in the eye which detects motion. The farther the pixel is from the center of the eye (which is defined as, towards pixel 1 in the right eye and towards pixel -1 in the left eye), the smaller is the turn that the robot needs to make. The spatial position encoding circuit generates the encoded position as a voltage.

Once a motion is detected, the turn pulses have to be generated. Before going into this process, there is one more complication that needs to be addressed. When the robot starts turning, both eyes start generating motion pulses continuously, as the whole world is now turning relative to the eyes of the robot. These motion pulses are not real, i.e., not generated by an external motion, but are generated because of the robot's turning. To solve this problem of distinguishing these from real external motion, the RoaCh uses a biologically inspired technique called "saccadic suppression" (Volkman *et al.*, 1968). That is, once motion is detected in an eye, the robot ignores all other motion pulses for a while during which it makes the turn. This saccade pulse is generated using the motion pulses and external off-chip RC elements as shown in the block diagram.

Using the saccade pulse from the saccade generator block and the motion pulses, RoaCh generates the initiator pulses, turn left initiate and turn right initiate as shown in the block diagram. These are generated when there is motion detected in an eye and when the saccade pulse is low, indicating a true external motion.

The turn left initiate and the turn right initiate pulses generate the actual turn pulses, turn left and turn right for a length of time determined by the encoded position from the spatial position encoding block and by using off-chip RC elements as shown in the block diagram. The turn pulses generate a run initiation pulse. This run initiate pulse sets off the actual run pulse and the robot starts running when the turn pulses go low.

These three pulses TL, TR and Run determine the state of the robot, turning left, turning right, running or staying stationary. These three pulses are used by motor circuits which generate the signals needed for the off-chip H-Bridges, which drive the motors of the robot as shown in the block diagram.

Some of the circuits used in the control system for RoaCh are the spatial position encoding circuit, which computes how far the robot has to turn from its current position, the current comparator

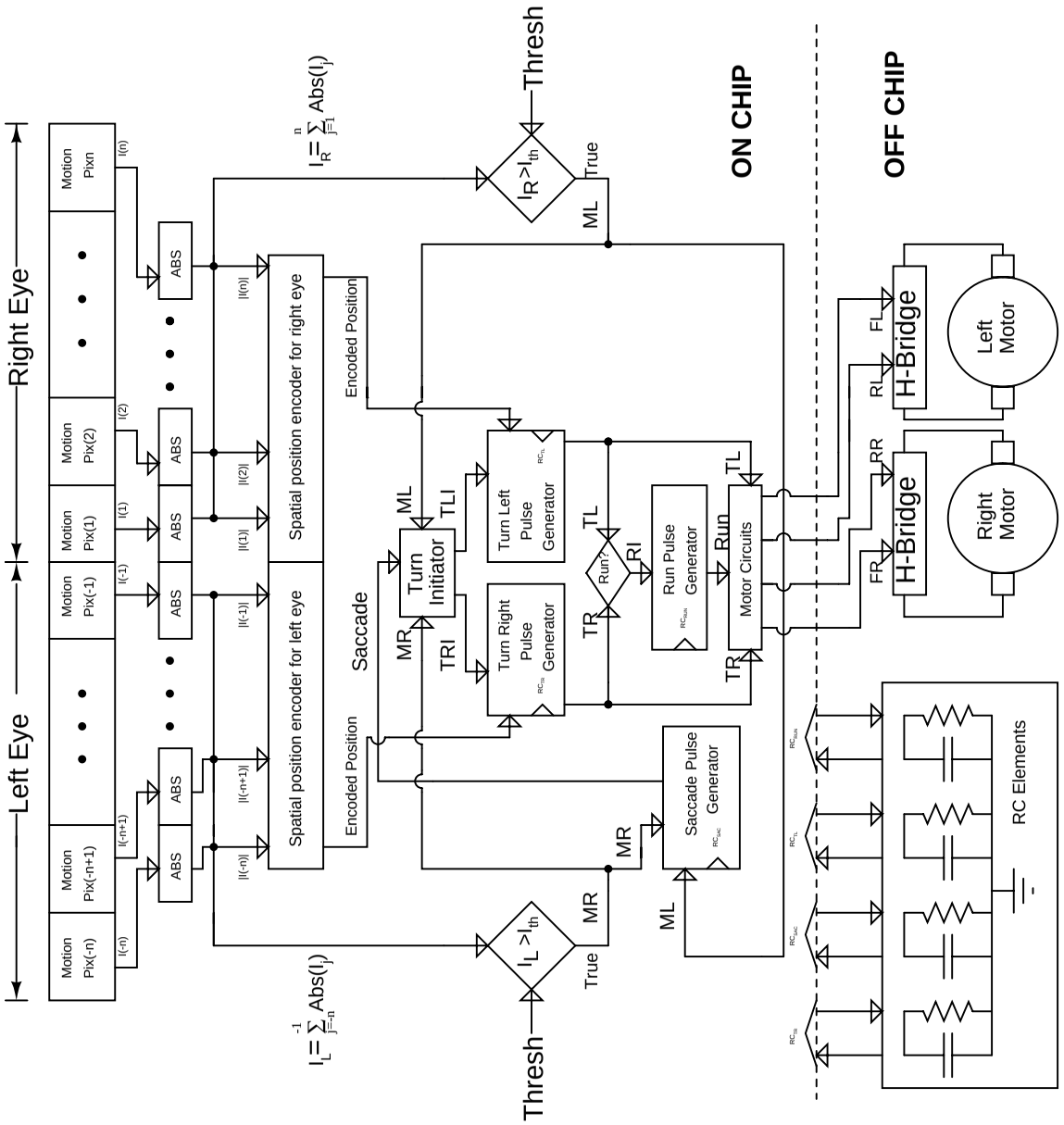


FIGURE 6.2. Block diagram of RoaCh. The entire system with both on-chip and off-chip parts is shown. The top part shows the two eyes with AB motion pixels in them. Below it lies the control circuitry, which sends signals off-chip to the motors controlling the wheels of the robot.

circuit, which determines if the robot needs to turn, the saccade pulse generation circuit to generate the saccade pulse. These and other circuits used in the control scheme are discussed next.

6.2 Circuitry

6.2.1 Absolute Value Block

In the block diagram we propose the use of an absolute value of the motion energy current. The reason for this is as follows. From Chapter 4, we see that the motion energy current is positive for the preferred direction and negative for the null direction. But RoaCh does not use this information for computing the direction of turn. The RoaCh sees if there is any motion at all in either the left or the right eye and signals the direction of turn in the opposite direction, that is turn left if motion is detected in the right eye and vice versa. This is the reason the motion energy current is taken through an absolute value circuit before using it to compute the direction of turn. The absolute value circuit is discussed in section 4.3.

6.2.2 Motion Pulse Generation Circuit

From the absolute value block, all the currents from motion pixels in each half of the linear array are summed by wiring them all together and the sum is realized by plain Kirchoff's current law. To compute the motion pulses ML or MR, we need to check if there is motion in the eye. We set an external threshold I_{th} as shown in the block diagram for this. If the summed current is greater than the threshold, ML/MR pulses go high indicating motion in that eye. If there is no motion in the eye, the summed currents are less than the threshold and the motion pulses ML/MR are low indicating the absence of motion. The current comparator circuit shown in Figure 6.3 (Traff, 1992) is used to generate the motion pulse. A description of the operation of the circuit is given below.

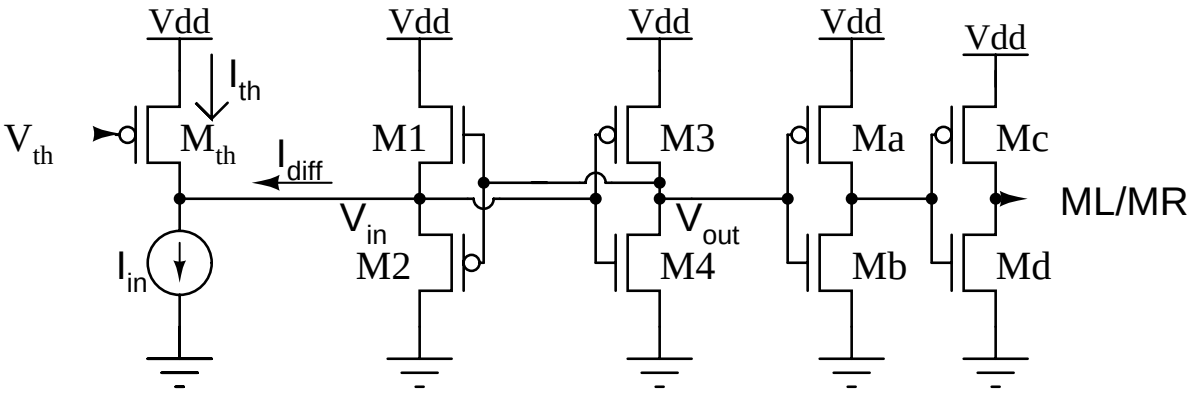


FIGURE 6.3. Current comparator circuit.

The transistor M_{th} is used to set the threshold current, I_{th} in the circuit. The summed current from the absolute value circuits is represented by I_{in}. By setting the bias V_{th} for M_{th}, we can determine the threshold current. The difference current between I_{th} and I_{in}, I_{diff} is fed into a source follower circuit (transistors M1-M2). To understand this circuit, let us start by considering the case when the difference current, I_{diff} flows into the node V_{in}. The transistor M2 sinks this current, pulling up the node voltage of V_{in}. This causes the voltage on the node V_{out} to go low as M3-M4 is an inverter. The transistor pair M1-M2 has linear transfer characteristics between its input voltage and output voltage. Thus a decrease in the voltage at node V_{out} would decrease the voltage at node V_{in}. However, in this process, the node voltage at V_{in} is pulled up high enough to flip the state of the inverter M3-M4 and make it to go low. Node V_{out} is fed to an inverter pair Ma-Mb and Mc-Md which generate a nice peak to peak output pulse ML/MR. similarly, one can analyze the case when the difference current, I_{diff} flows out of the node V_{in}. The node voltage

V_{in} is pulled low, causing the voltage at node V_{out} to go high which leads to pulling up the voltage at node V_{in}. From these cases we can see that feedback prevents the voltage on the node V_{in} to move all the way to the rails. Instead, it moves just enough to flip the state of the inverter M3-M4. By keeping the changes in this voltage small, the circuit can detect changes even in small current signals. Two such current comparator circuits are needed, one to generate the MR signal from the left eye and the other to generate the ML signal from the right eye.

Simulation Results: The results from the simulation of the circuit are shown in Figure 6.4. All the important voltage traces are plotted against the difference current I_{diff} shown in Figure 6.3. As explained earlier, one can see that the voltage V_{in} stays relatively constant at about 2.5V. The voltage V_{out} changes enough above and below the threshold voltage of the next inverter such that it can flip its state. This leads to a rail to rail voltage swing of the output motion pulse ML/MR. The current at which ML/MR flips its state, which is about 20nA in this case can be adjusted by changing the bias voltage of the transistor, M_{th}.

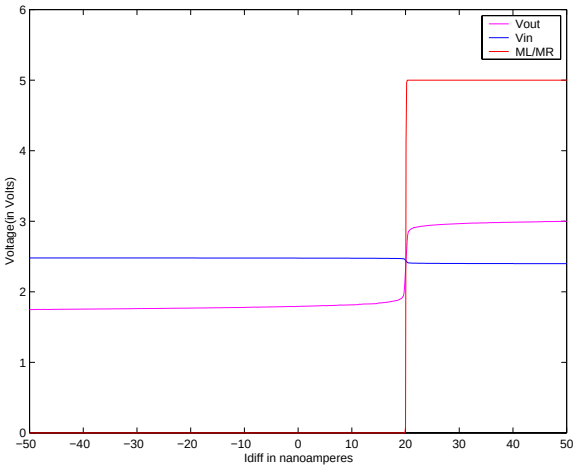


FIGURE 6.4. SPICE simulation results of Current comparator circuit: The difference current I_{diff} between I_{th} and I_{in}, at node V_{in} is plotted on the X-axis. The three voltage traces, V_{in}, V_{out} and ML/MR are plotted on the Y-axis. We can see V_{in} not changing much, and V_{out} flipping enough to change the state of the next inverter and the final ML/MR pulse going from rail to rail.

6.2.3 Spatial Position Encoding Block

As discussed earlier, the length of time the robot turns depends on the position of the motion pixel in the eye. If the pixel is farther from the center of the eye, the robot has to make a shorter turn away from it. The spatial position encoding circuit is used to detect the pixel that detects motion in the field of view and it outputs a voltage corresponding to the position of the pixel in the eye. The circuit is shown in Figure 6.5 (Deweerth, 1992). The circuit consists of a series of differential pairs, using a common global current mirror. Each differential pair receives a reference voltage input from a resistive network formed by the voltage sources V_0, V_n and the resistors $R_1, R_2, \cdots, R_n$. The currents $|I(1)|, |I(2)|, \cdots, |I(n)|$ are the currents generated by the absolute value block in the left or right eye. These currents provide the bias current for the differential pairs. The pixel that detects motion has the largest current. All other differential pairs are inhibited, except the one which detects motion. The output of the spatial position encoding circuit, $V_{centroid}$, thus represents the position of the pixel which detects the motion in the eye. There are two such spatial position encoding circuits, one for each eye.

There is one more design issue that has to be considered for the spatial position encoding circuit, which is the design of the resistances in the resistive ladder. If all the resistors are equal, the centroid

voltage, $V_{centroid}$ would be linear with the position. That is, $V(j)_{centroid}$ for the j^{th} position in the linear array of pixels would be,

$$V(j)_{centroid} = \frac{\sum_{i=1}^{j}(R_i)}{\sum_{i=1}^{n}(R_i)} \cdot (V_n - V_0) + V_0 \tag{6.1}$$

This voltage is used in the turn generation circuits to charge up an external capacitor up to $V(j)_{centroid}$. The turn pulse then goes high and then the capacitor is discharged through a resistor (R_{cent} in Figure 6.11) to ground. The turn pulse goes low when the capacitor discharges down to a threshold voltage of V_{th}.

This spatial encoding circuit where the centroid voltage is linear has been used for visual tracking using edge detection in (Indiveri, 1999). But the RoaCh needs a linearity in time and not in centroid voltage, which means, the length of time the turn pulse stays high should be linear with the position of the pixel in the array, but not the centroid voltage. So the centroid voltages should be designed appropriately.

The discharge time of a capacitor to ground through a resistor is not linear, but logarithmic instead. Rearranging the expression for an exponential decay of a voltage with time in a RC circuit, we can express the time of decay as shown:

$$t = R \cdot C \cdot \log \frac{V_{init}}{V_{final}} \tag{6.2}$$

Where, $R \cdot C$ is the time constant, V_{init} is the initial centroid voltage the capacitor is charged to. V_{final} is the threshold voltage to which the capacitor discharges to. To compensate for the exponential decay of voltage through a capacitor, we need to design the resistances in such a way that the centroid voltages are encoded exponentially, which would in effect linearize the length of the time the turn pulse stays high. We can design the resistors to make this encoding exponential. The derivation to determine the resistance of these resistors is as follows. The time it takes for a centroid voltage V_j to discharge to a threshold voltage V_{th} is given from Equation 6.2 as,

$$t_j = R \cdot C \cdot \log \frac{V_j}{V_{th}}$$

Let $\Gamma = \sum_{i=1}^{n} R_i$. Using Equation 6.1 in Equation 6.2 we can write,

$$t_j = R \cdot C \cdot \log \left[\frac{(V_n - V_0) \cdot \sum_{i=1}^{j} R_i}{V_{th} \cdot \Gamma} + \frac{V_0}{V_{th}} \right] \tag{6.3}$$

The above equation can be rearranged as follows:

$$t_j = R \cdot C \log \left[(V_n - V_0) \cdot \sum_{i=1}^{j} R_i + \Gamma \cdot V_0 \right] - R \cdot C \log (V_{th} \cdot \Gamma)$$

Which can be rewritten as:

$$(V_n - V_0) \cdot \sum_{i=1}^{j} R_i = \exp \left(\frac{t_j}{R \cdot C} + \log (V_{th} \cdot \Gamma) \right) - \Gamma \cdot V_0 \tag{6.4}$$

When j=n, which is the case when motion is detected in the farthest pixel from the center, the following hold:

$$V_j = V_0$$
$$t_n = n \cdot \Delta t = R \cdot C \cdot \log \frac{V_n}{V_{th}}$$

Where Δt is the incremental time slice. That is, the turn pulse for a pixel position stays high more than the previous pixel by a time Δt. Let $\psi = \frac{1}{V_n - V_0}$. Rearranging Equation 6.4, we can express the resistance of the resistors in terms of time as:

$$\sum_{i=1}^{j} R_i = \psi \cdot \exp\left(\frac{j\Delta t}{R \cdot C} + \log\left(V_{th} \cdot \Gamma\right)\right) - \psi \cdot \Gamma \cdot V_0 \tag{6.5}$$

An initial Γ can be chosen based on the layout area available and we can solve the following n equations to determine the n resistor values:

$$\begin{aligned}
\text{When j=1, } R_1 &= \psi \cdot \exp\left(\frac{\Delta t}{R \cdot C} + \log(V_{th} \cdot \Gamma)\right) - \psi \cdot \Gamma \cdot V_0 \\
\text{When j=2, } R_1 + R_2 &= \psi \cdot \exp\left(\frac{2\Delta t}{R \cdot C} + \log(V_{th} \cdot \Gamma)\right) - \psi \cdot \Gamma \cdot V_0 \\
\text{When j=3, } R_1 + R_2 + R_3 &= \psi \cdot \exp\left(\frac{3\Delta t}{R \cdot C} + \log(V_{th} \cdot \Gamma)\right) - \psi \cdot \Gamma \cdot V_0 \\
&\vdots \\
\text{When j=(n-1), } R_1 + R_2 + R_3 + R_{n-1} &= \psi \cdot \exp\left(\frac{(n-1)\Delta t}{R \cdot C} + \log(V_{th} \cdot \Gamma)\right) - \psi \cdot \Gamma \cdot V_0 \\
R_n &= \Gamma - \sum_{i=1}^{n-1} R_i
\end{aligned}$$

Using the above equations, Figure 6.6 shows the values for resistances in a design with 60 pixels in the linear array. We can see that for designing such a resistive ladder, we need to be very precise in the values of the resistances. However, in actual practice, it is not always possible to obtain precise resistance values. For example, in the current process we use for our chips, the minimum denomination of resistance per square we can obtain from poly1 is 26.6 $\Omega/\square$. From Figure 6.6 we can see that the minimum resolution of resistances needed is in the order of about 4 ohms. So, if we try designing a resistive ladder which has a minimum resolution above 26.6 ohms between its resistances, the area needed for the resistive ladder becomes very large making it impractical to realize on a chip.

A second approach to solve the problem of obtaining linearity in time with pixel position comes not by designing the resistors specially, but by using a transistor in parallel with the turn capacitor for the discharge as shown in Figure 6.7. The resistors can all be designed equally in this case to obtain a linear voltage encoding of position. Instead of using an off chip resistor and capacitor to generate the delay for the turn pulse as shown in Figure 6.11, we now use an off chip capacitor and an on chip transistor for the decay. This is a more elegant solution for the problem, as one can even adjust the length of the time slice by changing the bias voltage V_{leak}. But, this comes with a price of an added bias voltage costing an extra pin. The simulation results for this circuit with SPICE are shown in Figure 6.7(b). The top plot shows the switch which is closed initially at about 10 microseconds and opened at 200 microseconds. The capacitor $C_{centroid}$ gets charged to the input voltage from the spatial position encoding circuit. The bottom plot shows this voltage being discharged as time progresses. In this plot we sweep the input voltage from the spatial position encoding circuit from 1V to 4V. We can see that with varying input voltages, the capacitor discharges linearly in time, which is what is needed.

6.2.4 Saccade Pulse Generator Circuit

The way "Saccadic Suppression" is implemented is as follows. Whenever motion is detected, the robot starts turning in the opposite direction ignoring all other motion pulses for a certain predetermined time. Thus the RoaCh can suppress the effect of additional motion pulses once the robot starts

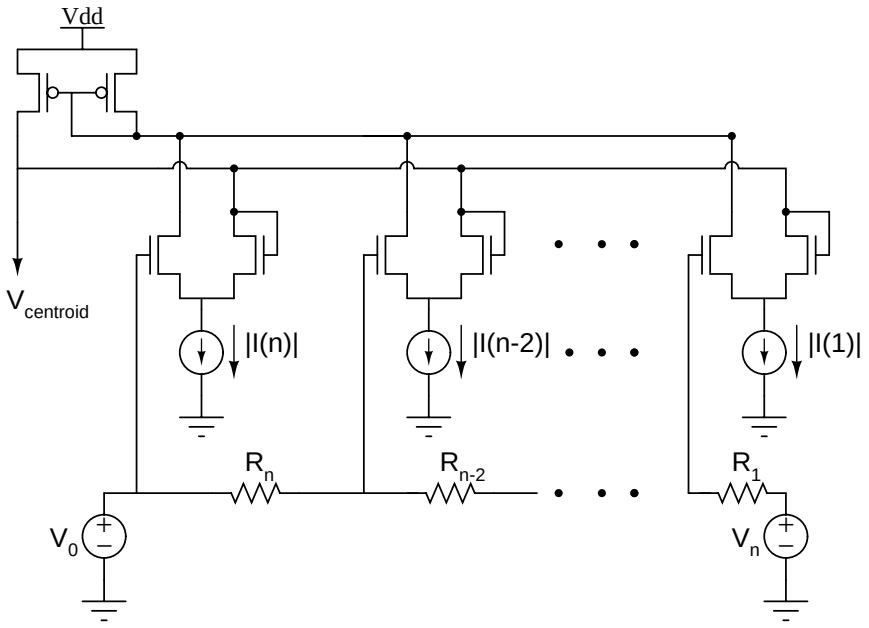
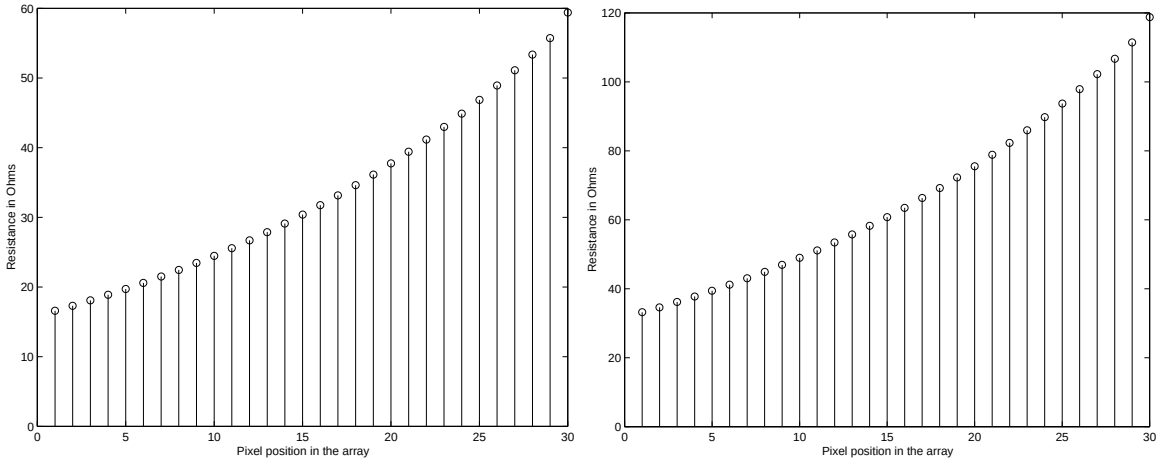


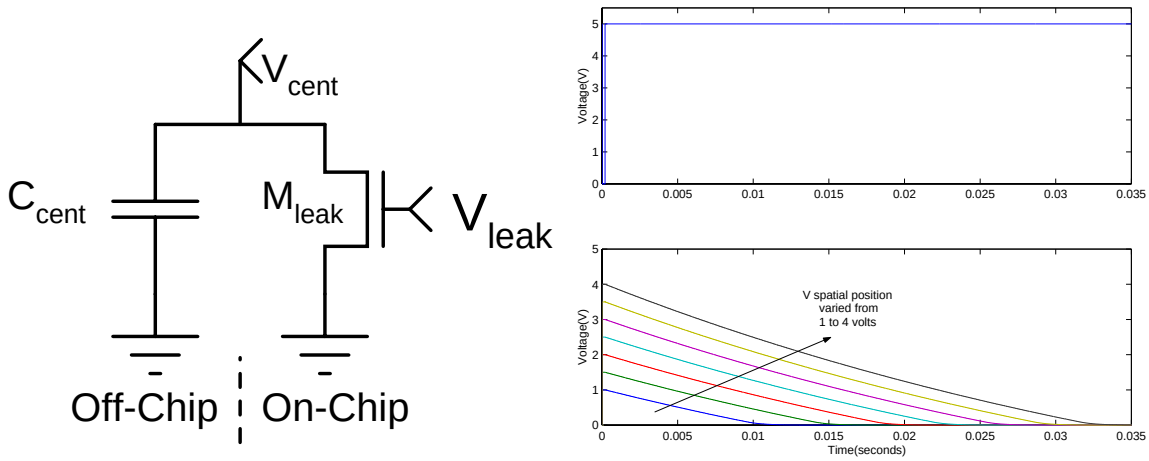
FIGURE 6.5. Centroid circuit to encode spatial position. It consists of a series of differential pairs with a common global current mirror. The input bias currents are obtained from the previous stage. The resistive ladder sets the reference voltage for each differential pair.



(a) For $\Gamma = 1000\Omega$

(b) For $\Gamma = 2000\Omega$

FIGURE 6.6. Resistor values for the spatial position encoding circuit with 60 pixels in the linear array. Which means, there are 30 resistors in each spatial position encoding circuit. For the above plots, we chose $V_n = 4.4V, V_0 = 1.2V, R \cdot C = 1.25sec$. (a) Shows the case when total resistance, $\Gamma = 1K\Omega$. (b) Shows the case when total resistance, $\Gamma = 2K\Omega$.



(a) Circuit for obtaining a linear decay

(b) SPICE simulation results of the circuit

FIGURE 6.7. (a) Circuit to implement linearity of time in the spatial position encoding circuitry. (b) Results from SPICE simulation of the circuit. The top plot shows the voltage signal (TLI/TRI) that controls the charging of the capacitor. The switch to charge the capacitor C_{cent} closes at 10 μs and opens at 200 μs as shown. In the bottom trace we show the time of discharge of the capacitor through M_{leak}, for various voltages of V_{cent}, to which the capacitor gets charged to.

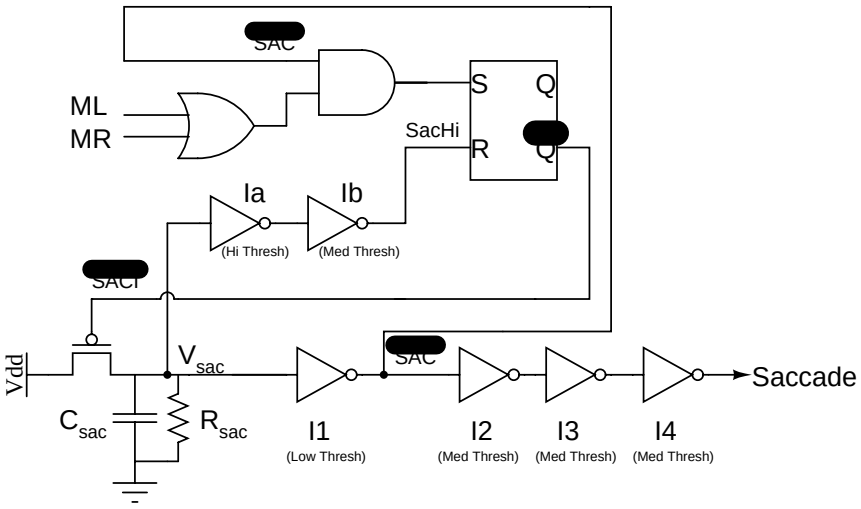


FIGURE 6.8. Circuit to generate saccade pulse.

turning and avoid triggering of false turn initiation pulses. The schematic of the circuit is shown in Figure 6.8.

A saccade pulse has to be generated when there is an ML or an MR pulse, which tells the robot that there is motion detected and it needs to turn left or right. But, the robot should not turn for every ML or MR pulse. It needs to turn only when there is no motion detected already. So, we need a "memory" of the previous state of the robot. This can be achieved by using an SR flip-flop. In order to determine how long the saccade pulse stays high, we need to measure time, which is achieved through a capacitive decay with the help of a capacitor C_{sac} and a resistor R_{sac} as shown in Figure 6.8.

The operation of the circuit can be understood from the timing diagram shown in Figure 6.9 where we plot all the signals occurring in the circuit against time. The saccade pulse shown in the last trace goes high when the voltage on the capacitor, V_{sac} (shown in the seventh trace) is above a certain fixed threshold voltage of 1.2V. This threshold is set by the inverter $I1$ which flips its state at this low threshold value. A series of three other inverters which flip their state at a medium threshold value of about 2.5V are needed to obtain the correct logic sense. Ideally just one transistor after $I1$ is sufficient to obtain the correct logic sense for the saccade pulse. But, the inverters $I2$ and $I3$ are needed to obtain steep rising and falling edges for the saccade pulse. The top two traces show the motion pulses, ML and MR. The third trace shows the $\overline{SAC}$ pulse, which is the output from the inverter $I1$. The SR flip-flop generates the signal $\overline{SACI}$, which is used to control the switch that charges up the capacitor C_{sac}. The switch needs to be closed when there is a motion pulse and when the saccade pulse is already not high. In other words, the input to the flip-flop which sets its next state should be:

$$S \quad = \quad (ML + MR) \cdot (\overline{SAC})$$

This signal S which sets the next state of the flip-flop is shown as the fourth trace. There is one more thing that needs to be taken care of for this circuit, which is the reset signal for the flip-flop, $SACHI$. The flip-flop needs to be reset i.e., the switch charging the capacitor needs to opened when the saccade pulse reaches a high threshold voltage. This signal is obtained from an inverter, Ia which has a high threshold value at which it flips its state. The output of this inverter goes low when the voltage on the capacitor, V_{sac} is above a high threshold, designed to be about 4V. The output of Ia is again complemented by Ib so that, $SACHI$ goes high when V_{sac} goes above 4V, and it goes low when it falls below 4V. $SACHI$ is plotted as trace five. Once $SACHI$ goes low,

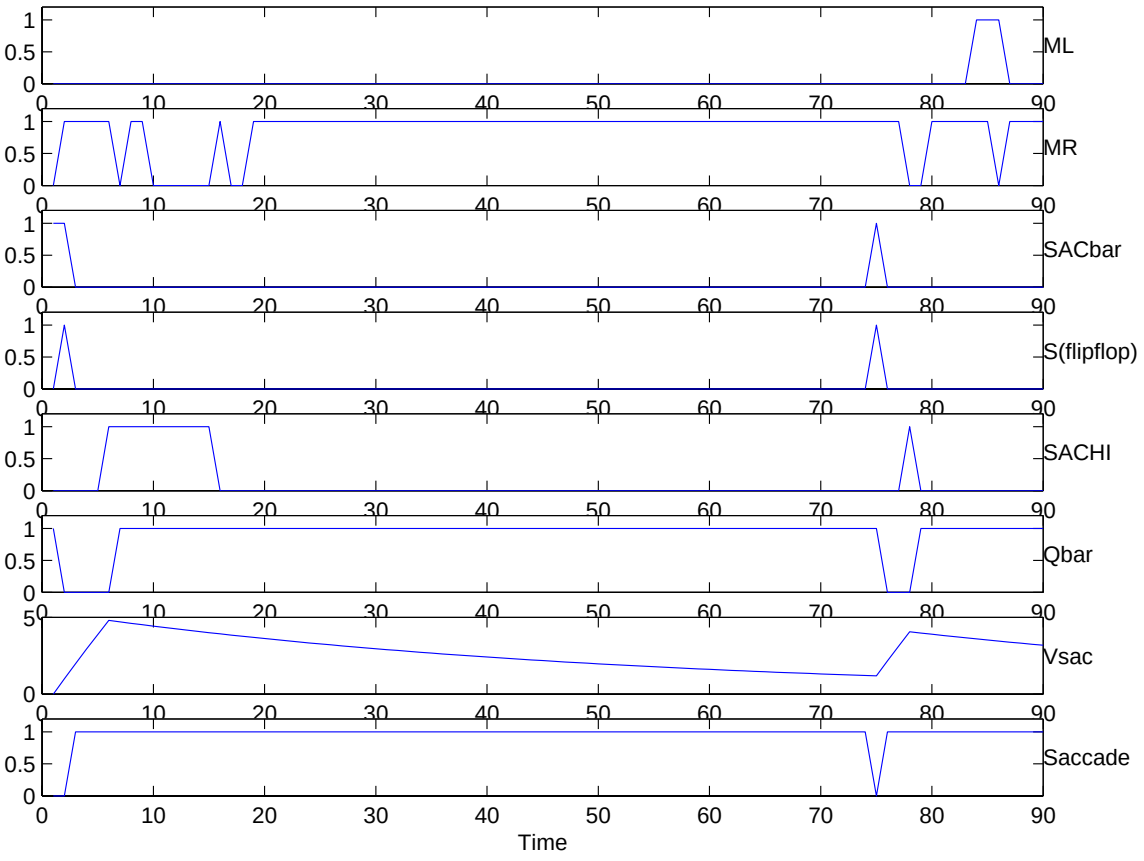


FIGURE 6.9. Timing diagram showing all the pulses in the saccade pulse generation circuit.

the saccade pulse is still high. The inputs to the flip-flop are now both low and the switch charging C_{sac} remains open till V_{sac} drops to the low threshold value of 1.2V. At this point, $\overline{SAC}$ goes high and the flip-flop is now ready to detect new motion pulses that are generated.

6.2.5 Initiation Pulse Generation Circuits

These are the circuits that generate the turn initiation and run initiation pulses. The schematics are shown in Figure 6.10. A turn initiate pulse, either turn left initiate (TLI) or turn right initiate (TRI) should be generated when the saccade pulse is low and when the corresponding motion pulse, ML or MR goes high. Or, in other words

$$\begin{aligned} TLI &= ML \cdot \overline{Saccade} \\ TRI &= MR \cdot \overline{Saccade} \end{aligned}$$

Initially when the robot is at rest, the saccade pulse is low and when the robot sees any motion, the turn initiate pulse goes high. When the robot starts to turn, as the motion pulses are continuously high, the saccade pulse stays high but the turn initiate pulse goes low. Similarly, the run initiation pulse would go high when either TL or TR goes high.

$$RI = TL + TR$$

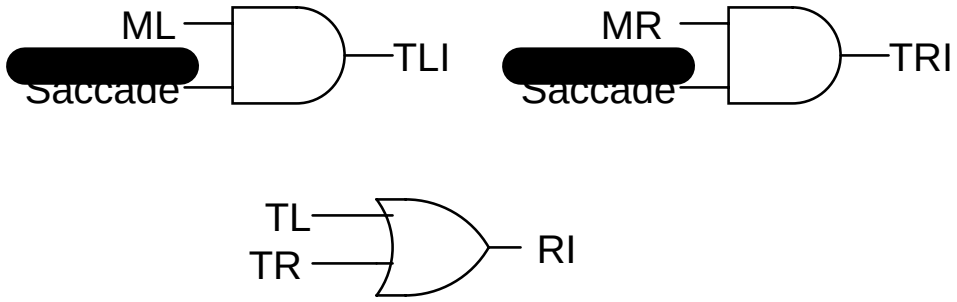


FIGURE 6.10. Circuits to generate initiation pulses. The top two circuits generate the turn left initiate and the turn right initiate pulse, which trigger the turn left and the turn right pulses, The bottom circuit generates the run initiation pulse which triggers the Run pulse.

6.2.6 Turn Pulse Generator Circuit

Once a turn initiate pulse is generated, the actual turn pulse has to be generated for a length of time determined by the encoded position as discussed earlier in the control scheme. The circuit shown in Figure 6.11 does this job. This circuit shown here generates a turn left, TL pulse. A similar circuit is used to generate the turn right pulse, TR. The TLI pulse is complemented, called $\overline{TLI}$ and these are used as the control signals of a transmission gate as shown in the figure. The input to the transmission gate comes from the spatial position encoding circuit. This input voltage encodes the position of the motion pixel in the right eye. When TLI is high, the transmission gate allows an external capacitor C_{cent} to charge up to the input voltage. Once the turn initiation pulse goes low, the transmission gate shuts off. The capacitor C_{cent} then starts discharging through a resistor R_{cent} to ground. This discharge time would determine how long TL stays high. The voltage on this capacitor is applied to the input of an inverter I1, which is sized such that it flips its state at an input voltage of 1.2V. The inverter I4 generates the turn left, TL pulse. The TL pulse is a strong high when the input to I1 is above 1.2V and strong low when the input is below 1.2V. The length of time the robot turns is also determined by the input voltage to which the capacitor C_{cent} is charged to, thus encoding the position of the motion pixel. Another important thing to be observed is, the voltage at which I1 flips, 1.2V sets the lower limit V_0 of the reference voltage for the spatial position encoding circuit shown in Figure 6.5.

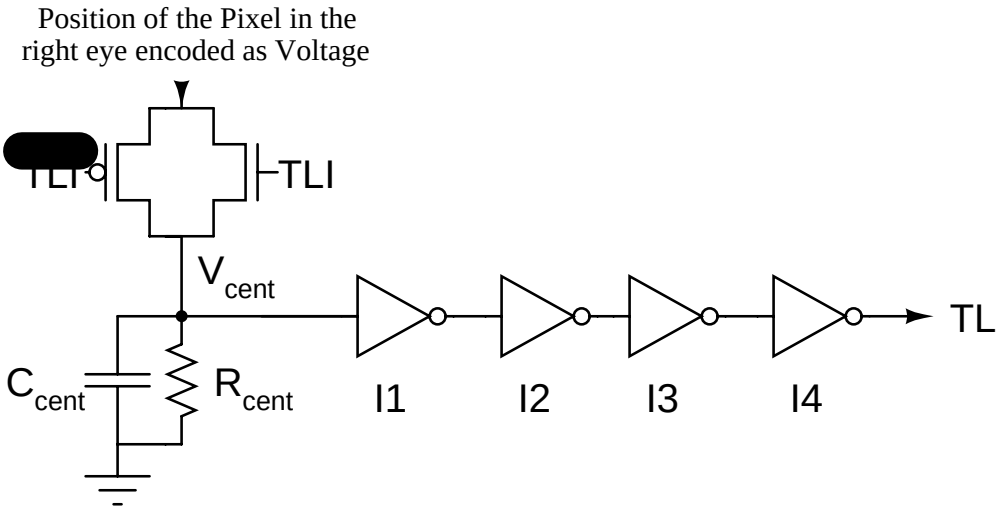


FIGURE 6.11. Circuit to generate turn pulse.

6.2.7 Run Pulse Generator Circuit

The circuit shown in Figure 6.12 generates the Run pulse. This circuit is quite similar to the other pulse generator circuits. The run initiation pulse acts as the control signal for generating the Run pulse. A PFET pass transistor acts as the switch to pull the voltage of the capacitor C_{run} all the way up to V_{dd}. Once the RI pulse goes low, the switch is turned off and the capacitor C_{run} starts discharging to ground through R_{run}. The inverter chain is designed such that any voltage below 1.2V puts out a strong low run pulse and any voltage above 1.2V on C_{run} would put out a strong high run pulse.

6.2.8 Motor Command Generator Circuits

All the circuits discussed previously generate the necessary signals for turning and running. These signals have to be translated into real world motor control signals. The RoaCh generates H-Bridge control signals for controlling the motors for the wheels of the robot. A typical H-Bridge is shown in Figure 6.13. A H-Bridge takes a DC supply voltage and provides 4-quadrant control to a load connected between two pairs of power switching transistors (Regan, 1999). In this research the load is a DC motor shown between + and − in Figure 6.13. A H-Bridge can be controlled using different configurations, but the one used here is as follows, when the wheel has to move forward, the switches 1a and 2b have to be closed, similarly 2a and 1b have to be closed for reverse motion. For sake of notation, the pair 1a and 2b is called F (for forward) and the pair 2a and 1b is called R (for rear) from here on. There are two motors to be controlled, left motor and right motor. Let FL and RL represent the switches of left motor and FR and RR represent the switches of right motor. The various possible states for the motors are Stay, turn left, turn right, Run. Table 6.1 describes the generation of the H-Bridge signals.

Though some of the cases described in the Table 6.1 cannot occur if the external capacitors and resistors are chosen appropriately, those are included to avoid any indeterminate state, if it ever occurs. From Table 6.1, we can generate the necessary boolean functions for generating the H-Bridge controls. These are as follows:

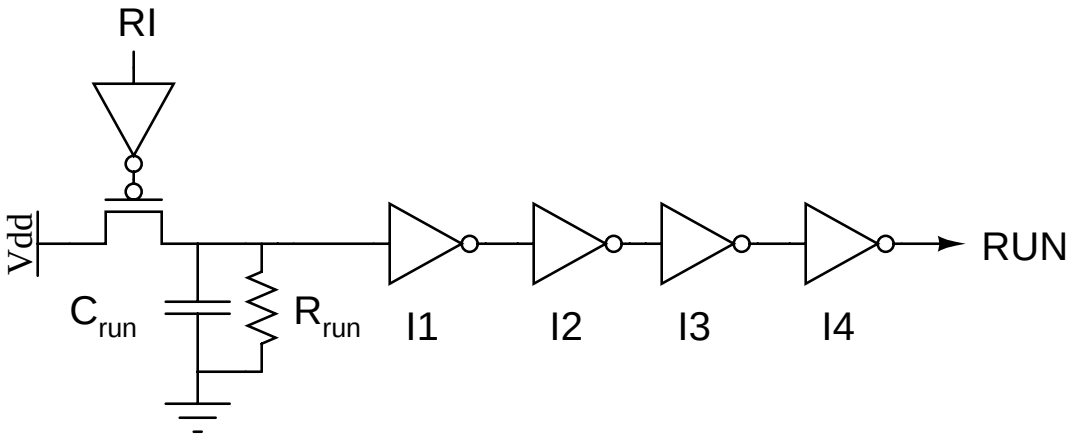


FIGURE 6.12. Circuit to generate Run pulse.

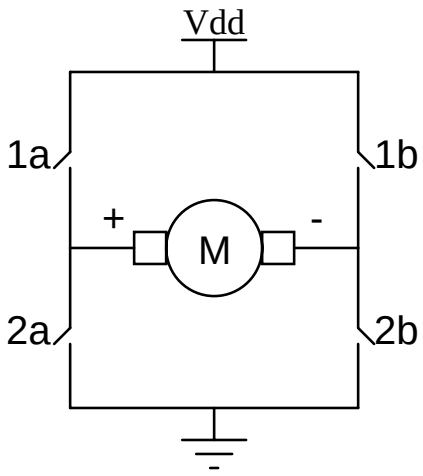


FIGURE 6.13. A typical H-Bridge for a DC motor. 1a and 2b need to be closed for forward direction; 2a and 1b need to be closed to for reverse direction.

TL	TR	RUN	STATE	FL	RL	FR	RR
0	0	0	Stay	0	0	0	0
0	0	1	Run	1	0	1	0
0	1	X	Right	1	0	0	1
1	0	X	Left	0	1	1	0
1	1	X	Stay	0	0	0	0

TABLE 6.1. State Table of RoaCh

$$
\begin{aligned}
FL &= RUN \cdot \overline{TL} + TR \cdot \overline{TL} \\
RL &= TL \cdot \overline{TR} \\
FR &= RUN \cdot \overline{TR} + TL \cdot \overline{TR} \\
RR &= TR \cdot \overline{TL}
\end{aligned}
$$

From the above equations, we can synthesize CMOS logic circuits to implement them as shown in Figure 6.14.

Simulation Results: Figures 6.15 and 6.16 show the results from simulating the entire control circuit of RoaCh in SPICE. All the plots are from a transient analysis and so show signals against time in seconds. The top trace in Figure 6.15(a) shows the voltage on the saccade capacitor V_{sac}, the second trace shows the saccade pulse being generated from it. We can see that the saccade pulse is always high for a fixed period of time after it detects motion in either eye. The next three traces are of TL, TR and RUN which control the state of the RoaCh at any time.

Figure 6.15(b) shows the generation of TL pulse. The top trace shows the voltage on the capacitor $C_{centroid}$, $V_{centroid}$ described earlier. This voltage is named as V_{tl} here to indicate that this generates the TL pulse. We can see that at about 0.1 seconds, the TLI pulse goes high and the capacitor is charged up to about 1.8V. This causes the TL pulse to go high and it stays high till the voltage V_{tl} discharges to about 1.2V, and at about 0.25 seconds, the TL pulse goes low. At about 2.2 seconds TLI goes high again. But this time motion is detected in a pixel closer to the center of the eyes and so V_{tl} gets charged to about 2V and so TL stays high for a longer time till about 2.6 seconds as shown.

Figure 6.15(c) shows the generation of TR pulse. Again the top trace shows $V_{centroid}$, which is termed as V_{tr} here as it generates the TR pulse. In this figure, we can see that at about 1.3 seconds TRI goes high. V_{tr} gets charged up to about 2.1V as motion is now detected at a pixel much closer to the center of the eye. So, TR stays high for a long time. After V_{tr} discharges to about 1.2V, TR goes low. At about 3.1 seconds TRI goes high again, but TR stays high for a shorter time now as motion is detected in a pixel farther from the center of the eye.

Figure 6.15(d) shows the generation of the RUN pulse. The top trace shows the voltage V_{run} on the run capacitor C_{run}. The second trace shows $\overline{RI}$. So, C_{run} gets charged when $\overline{RI}$ is low. The RUN pulse shown in the bottom trace goes high when the voltage V_{run} is above 1.2V.

Figure 6.16 shows the actual motor command signals against time. The top three traces are the turn left, right and run pulses, explained previously. The bottom four pulses are the signals generated from the motor circuits which would feed the H-Bridge and control the motors. From the state table of the RoaCh shown in Table 6.1, one can see that the four H-Bridge pulses, FL, RL, FR and RR are in accordance with it.

6.3 Simulation of the Entire RoaCh System

The simulation results for the entire RoaCh system in MATLAB are given in Figures 6.17 and 6.18.

In Figure 6.17 we show the results from the simulations of the two eyes of RoaCh, that is, from the linear array of A-B motion pixels. Figures 6.17(a) and 6.17(b) show the inputs and outputs of the eyes at progressive times. The top plot in both figures shows the 360^0 field of view around the robot. The stimulus versus degrees surrounding the robot is plotted on the x-axis. One can see a source of stimulus at about 300^0 in the figure. The second plot is similar to the first plot, but this time, the stimulus is plotted in the robot's frame of reference (so that one can see the external source in both the eyes). As time progresses, the robot turns and the stimulus in the robot's eyes changes as seen in the plot. The third plot shows the same image as in the second plot, but, with the mean value of the background filtered out. The fourth plot shows the response of the photoreceptors of the pixels. Notice that in this figure the photoreceptor response versus the pixels in the array is plotted. The linear array has 40 pixels as shown in the plot, pixels 1 to 20 constitute the left eye and pixels 21 to 40 constitute the right eye. The fifth plot shows the low pass filtered photoreceptor output. The sixth plot has two traces in it, the final opponent energy, and the absolute value of the opponent energy.

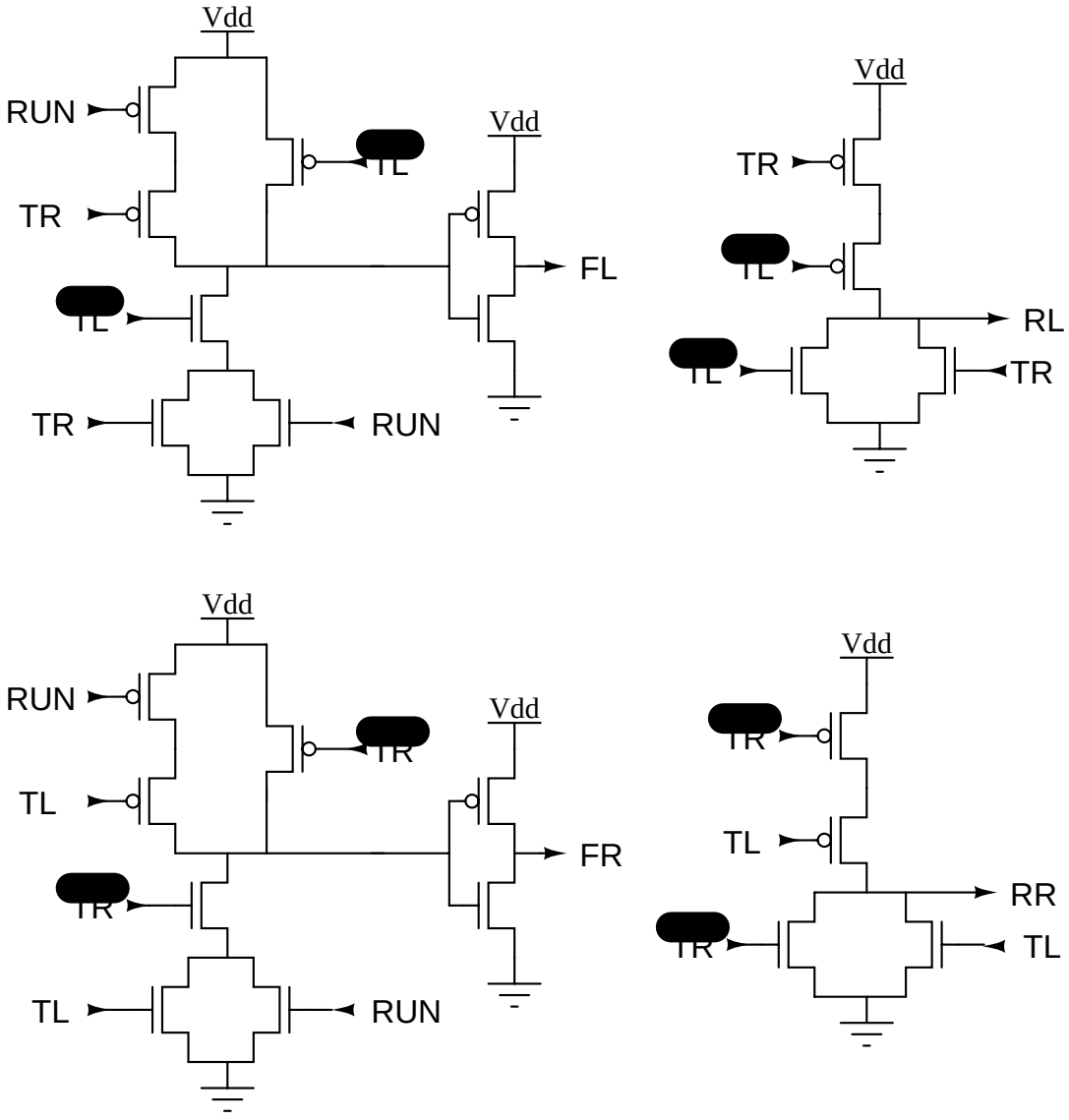
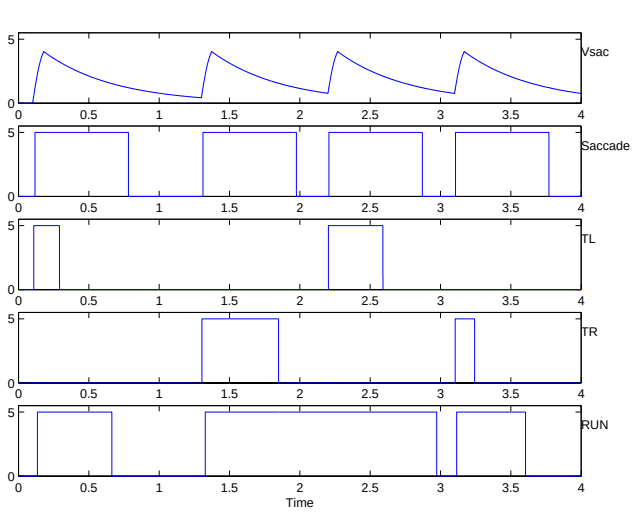
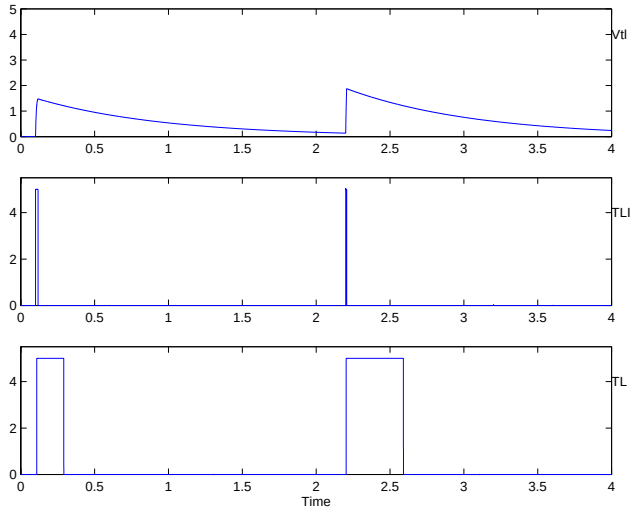


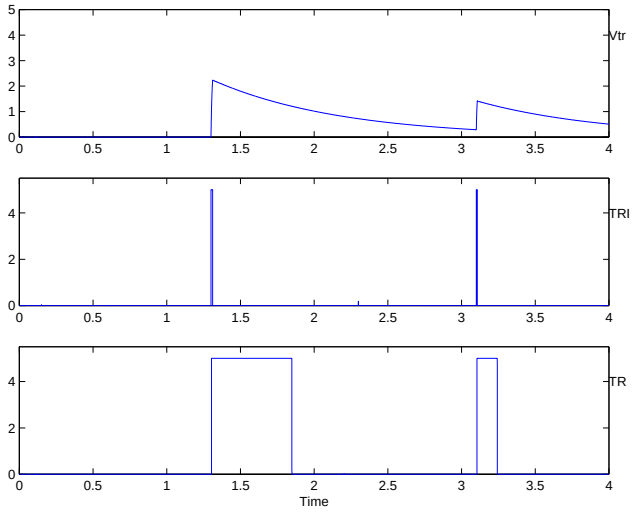
FIGURE 6.14. Circuits to generate motor control pulses.



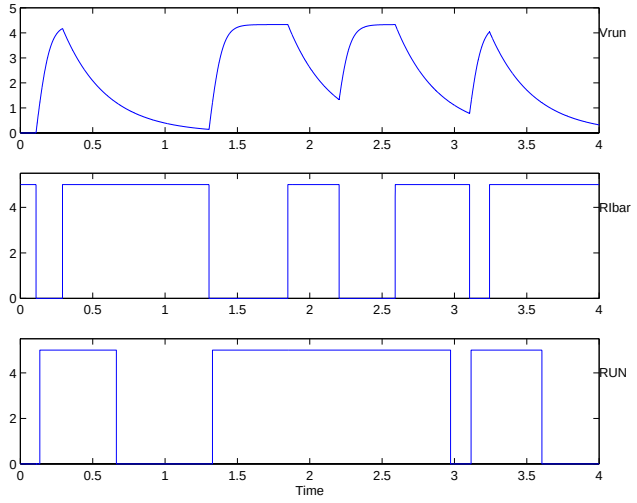
(a) Pulses governing the state of RoaCh



(b) Generation of TL pulse



(c) Generation of TR pulse



(d) Generation of RUN pulse

FIGURE 6.15. Results from simulating the entire control scheme of RoaCh in SPICE: (a) Shows the pulses TL, TR, RUN which determine the state of RoaCh and also the saccade pulse. (b) Shows the generation of the TL pulse based on V_{tl} and TLI pulse. (c) Shows the generation of the TR pulse based on V_{tr} and TRI pulse (d) Shows the generation of the RUN pulse based on V_{run} and $\overline{RI}$ pulses.

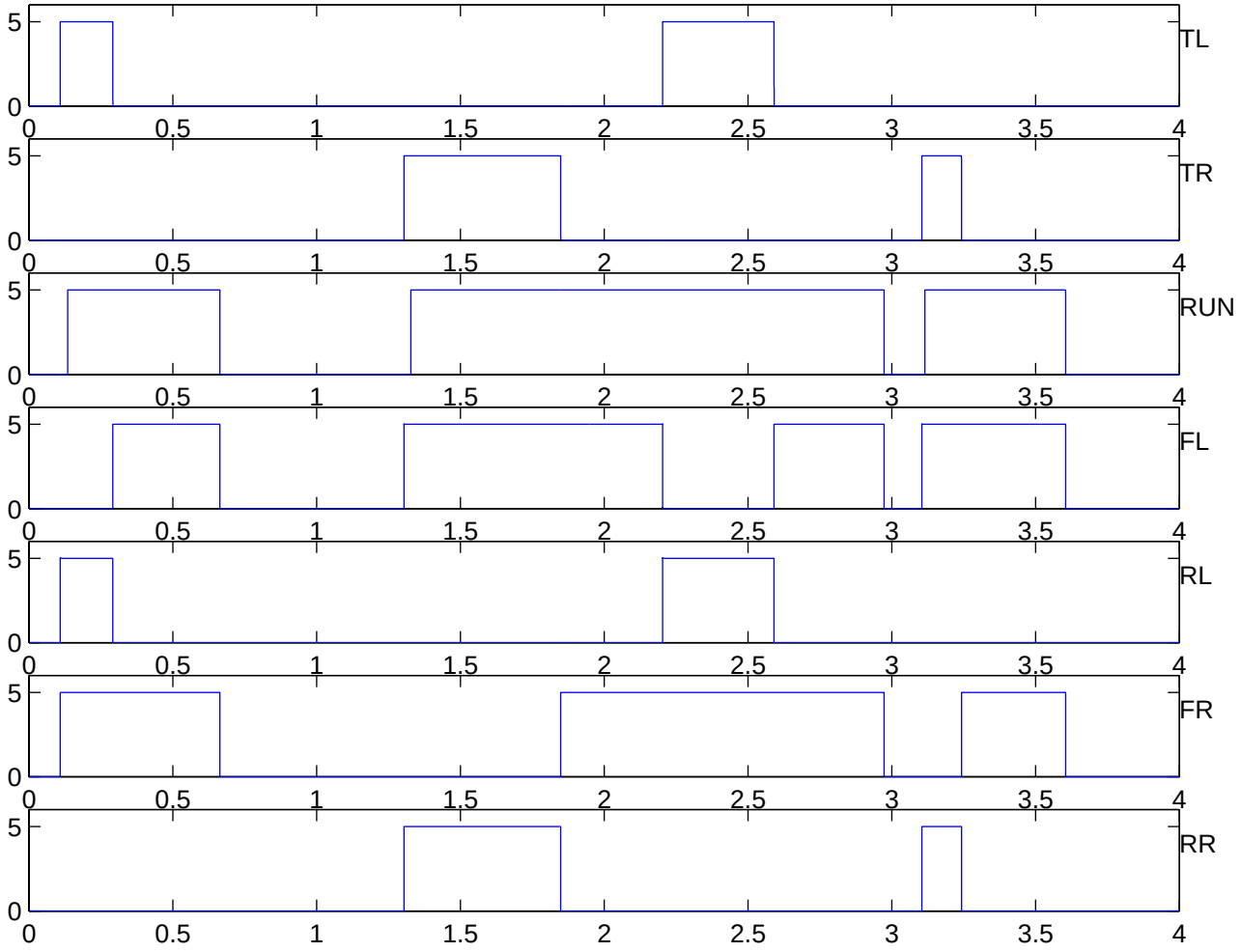
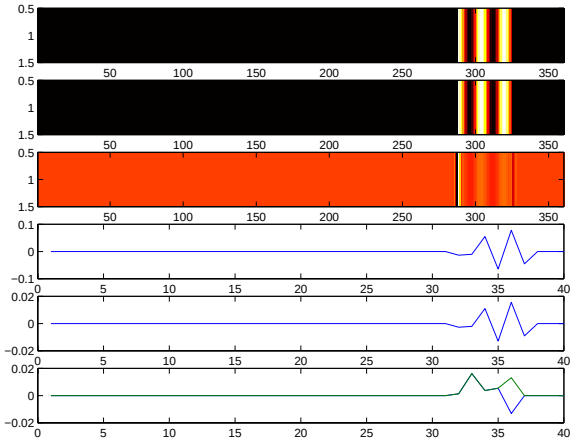


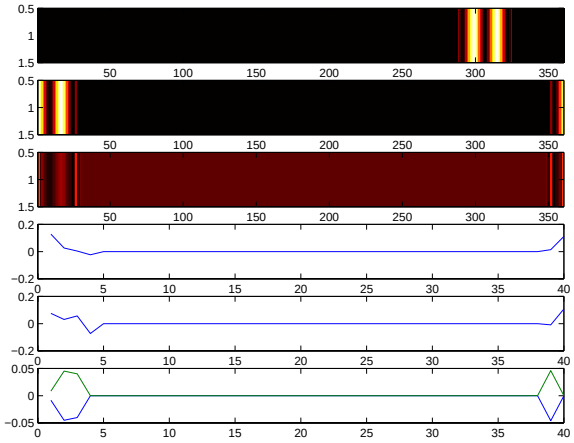
FIGURE 6.16. Results from SPICE simulation of the entire RoaCh control circuit : The top three traces show the turn pulses TL, TR and the Run pulse. The bottom four traces show FL, RL, FR, RR which are the inputs to the two H-Bridges that control the motors of the robot. These four traces follow the state table of the RoaCh given in Table 6.1.

In Figure 6.18(a) the saccade pulse, and the three pulses, TL, TR and RUN, which govern the state of the robot are plotted against time. The saccade pulse goes high whenever there is motion detected in either eye. It stays high for a certain fixed period of time as shown in the figure. When either the turn pulse is high or the robot turns in that direction. The robot starts running when the RUN pulse is high (and when the turn pulse goes low).

The Figures 6.18(c) and 6.18(d) show the generation of the TR, TL pulses respectively. As explained earlier, the length of time TR and TL stay high is determined by the spatial position encoding circuit. The voltages V_{tr} and V_{tl} are obtained from the spatial position encoding circuit of each eye and these are used to charge up a capacitor as explained before. The capacitor, C_{cent} is then discharged to ground through a resistor, R_{cent}. V_{tr} and V_{tl} are shown in the first plot of both figures (c) and (d). One can observe that, these voltages decay with time, as these are the voltages on the capacitor C_{cent}. The turn pulses, TR and TL are shown in the third plot in both figures (c) and (d). These stay high as long as the voltages V_{tr} and V_{tl} are above a certain threshold and go low when V_{tr} and V_{tl} reach the threshold. The initiation pulses, TRI and TLI are shown in the third plot in both the figures. Similarly, Figure 6.18(b) shows the RUN pulse and the associated pulses, V_{run} and RI.

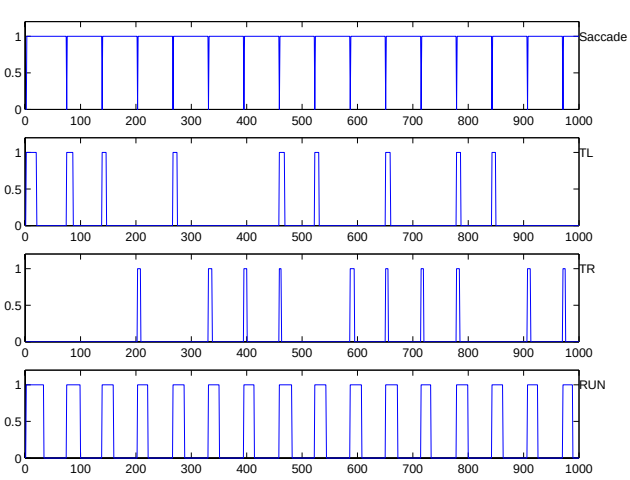


(a) At time = 1

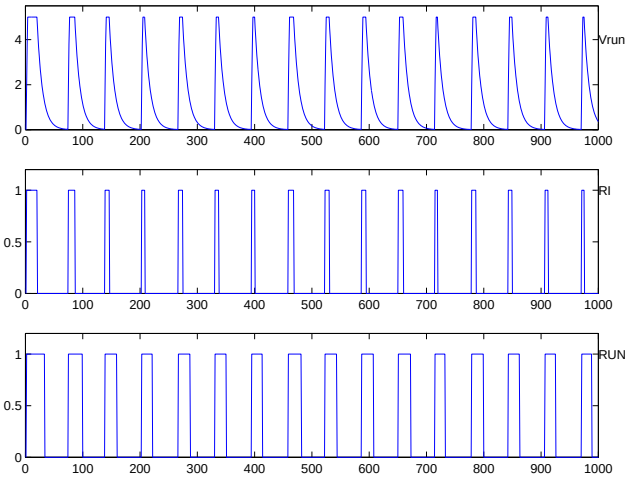


(b) At time = 200

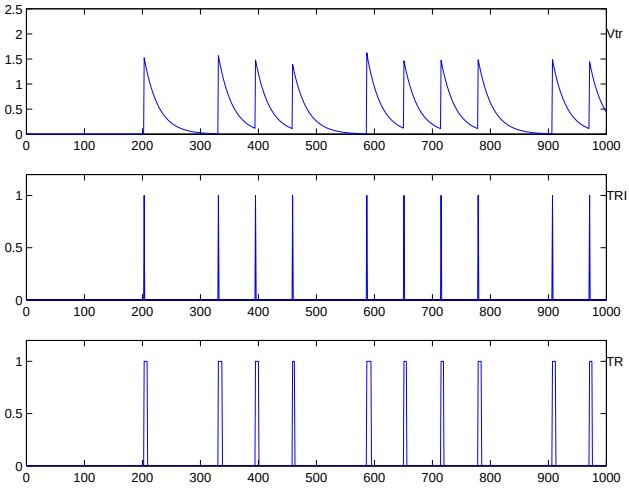
FIGURE 6.17. MATLAB simulations of RoaCh showing the external field of view and outputs from computation stages in the A-B motion pixels at progressive times.



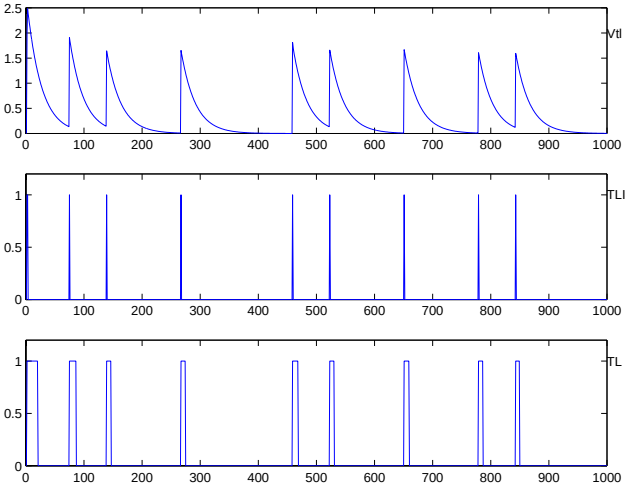
(a) Pulses governing the state of RoaCh



(b) Generation of RUN pulse



(c) Generation of TR pulse



(d) Generation of TL pulse

FIGURE 6.18. MATLAB simulations of RoaCh (a) Shows the three pulses TL, TR, RUN which determine the state of RoaCh and also the saccade pulse. (b) Shows the generation of the RUN pulse based on V_{run} and RI pulses. (c) Shows the generation of the TR pulse based on V_{tr} and TRI pulse (d) Shows the generation of the TL pulse based on V_{tl} and TLI pulse.

Chapter 7

Discussion

In the previous chapters the implementation of the Adelson-Bergen algorithm was explained in detail. Now some improvements that could be made so that a better VLSI implementation of the Adelson-Bergen algorithm results will be discussed. Some of the issues when doing system level design using the sensor will also be discussed.

7.1 Circuit Level Improvements

The squaring circuit was discussed in Section 4.3. From Equation 4.4 one can see that the squared current is scaled by a factor of I_0. But, this current level can still be high enough that it will increase the power consumption of the chip. With two extra transistors, one can get this current level down, decreasing the power consumption of the chip. A normalized squaring circuit that can achieve this is shown. Figure 7.1 shows the circuit.

The relation between the rectified current input to this circuit, I_{rect} and the squared output current, I_{sq} is now derived. As in Section 4.3, the early effect for these transistors operating in subthreshold region is neglected. Let the voltages at node $N3$, N_4 and N_5 be V_a, V_b and V_c respectively. The expressions for currents through the transistors M4, M5, M6, M7 and M8 are:

$$I_{M4} = I_0 e^{\frac{\kappa \cdot (V_a - V_b)}{V_T}} \tag{7.1}$$

$$I_{M5} = I_0 e^{\frac{\kappa \cdot V_b}{V_T}} \tag{7.2}$$

$$I_{M6} = I_0 e^{\frac{\kappa \cdot (V_a - V_c)}{V_T}} \tag{7.3}$$

$$I_{M7} = I_0 e^{\frac{\kappa \cdot V_{norm}}{V_T}} \tag{7.4}$$

$$I_{M8} = I_0 e^{\frac{\kappa \cdot V_c}{V_T}} \tag{7.5}$$

We can see that $I_{M4} = I_{M5} = I_{rect}$; $I_{M8} = I_{sq}$. The current through the transistors M6 and M7 is equal and let it be I_{norm}. From Equation 7.5, one can write:

$$V_c = \frac{V_T}{\kappa} \cdot \log \frac{I_{sq}}{I_0} \tag{7.6}$$

Similarly, as the currents I_{M4} and I_{M5} are equal, equating the right hand sides of Equations 7.1 and 7.2, one can write

$$V_a = 2 \cdot V_b \tag{7.7}$$

Substituting the results from Equations 7.6 and 7.7 into Equation 7.3 and after simplification, one obtains the final result:

$$I_{sq} = \frac{I_{rect}^2}{I_{norm}} \tag{7.8}$$

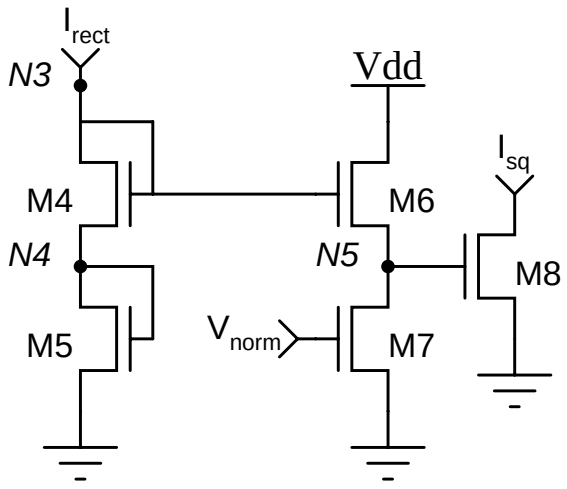


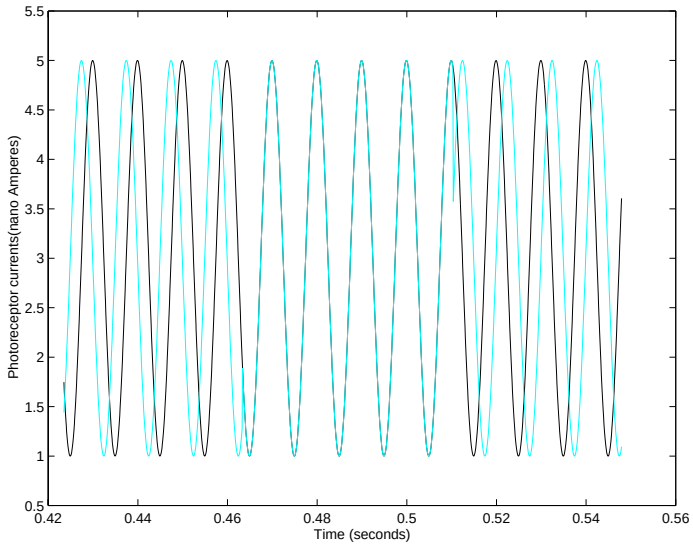
FIGURE 7.1. Normalized squaring circuit. The current level in the circuit can be adjusted using the bias voltage V_{norm}.

from which one can see the advantage of using this circuit. The current I_{norm} can change and have control over the order of magnitude of the squared current I_{sq}. This current, I_{norm} can be adjusted with the bias voltage V_{norm}.

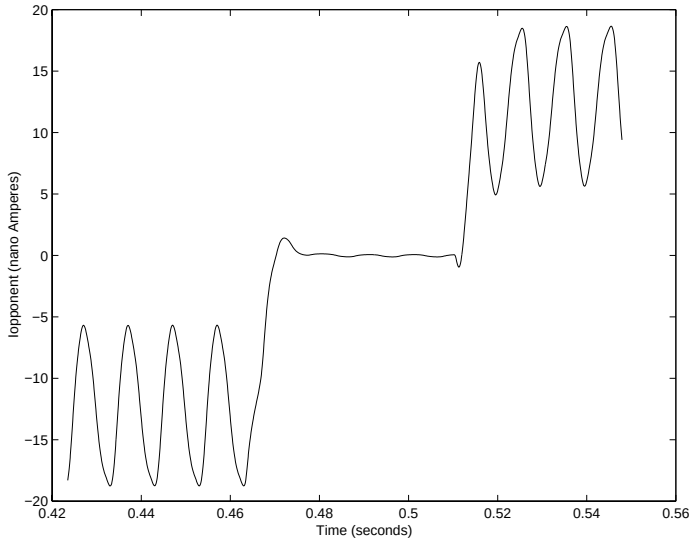
Adelson and Bergen propose the use of squaring the oriented linear responses, $(A - B')$, $(A' + B)$, $(A + B')$, $(A' - B)$ before taking the sums and difference to obtain the opponent motion energy. As explained earlier, in the actual hardware implementation we first rectify these signals to obtain, $|A - B'|$, $|A' + B|$, $|A + B'|$, $|A' - B|$ and then computed the squares. It was seen in Section 4.3 that squaring these signals involved the use of twelve additional transistors per pixel. Another modification that could be done to the hardware implementation is to eliminate the squaring stage altogether, thus getting rid of all the transistors needed for squaring. Thus, the non-linearity needed in the algorithm would now be computed by the rectification stage alone. Though this is a deviation from the original form of the algorithm, it is more amenable for VLSI implementation, since a smaller transistor count would mean smaller pixel size leading to a denser pixel array and better resolution.

The results are now shown for the simulation of the entire pixel circuitry in ANALOG, a circuit simulation tool (Chipmunk Tools, 1988). Figure 7.2 shows the results from using the non-linearity that was used to implement chip, that is, squaring the rectified oriented linear responses, $|A - B'|$, $|A' + B|$, $|A + B'|$, $|A' - B|$. In Figure 7.2(a), the photocurrents from two adjacent pixels are plotted. From 0.28 to 0.36 seconds, the stimulus traverses in the null direction (observe that the lighter trace leads the darker trace), from 0.36 to 0.42 seconds the stimulus is orthogonal (observe that the phase difference between the photocurrents of adjacent pixels is zero), and finally from 0.42 to 0.52 seconds the stimulus traverses in the preferred direction (observe the darker trace leading the lighter trace). In Figure 7.2(b) the final opponent motion output current from the motion pixel is plotted. As expected, the opponent motion current is negative when the stimulus is in the null direction, zero when the stimulus is orthogonal and positive when the stimulus moves in the preferred direction.

Similarly, Figure 7.3 shows the results from using the non-linearity proposed earlier, that is, using just the rectification as non-linearity and not squaring the rectified signals. In Figure 7.3(a), the photocurrents from two adjacent pixels are plotted as previously done. From 1.37 to 1.49 seconds, the stimulus traverses in the null direction (observe that the lighter trace leads the darker trace), from 1.49 to 1.58 seconds the stimulus is orthogonal (observe that the phase difference between the photocurrents of adjacent pixels is zero), and finally from 1.58 to 1.74 seconds the stimulus traverses



(a) Photo currents of adjacent pixels



(b) Opponent motion energy current

FIGURE 7.2. Simulation results using both rectification and squaring in the non-linearity stage. (a) In this plot each trace corresponds to the photocurrent detected in adjacent pixels. During the first time interval, the stimulus is in the null direction (light trace leads the darker trace), then it is orthogonal (the traces have no phase difference), after which the stimulus is in the preferred direction (darker trace leads the lighter trace). (b) The opponent motion energy current is plotted in this trace.

in the preferred direction (observe the darker trace leading the lighter trace). In Figure 7.3(b) the final opponent motion output current from the motion pixel is plotted. As expected, the opponent motion current is negative when the stimulus is in the null direction, zero when the stimulus is orthogonal and positive when the stimulus moves in the preferred direction.

If one looks at the opponent motion current plots in Figure 7.2(b) and in Figure 7.3(b), one can notice the difference between the two methods. In the first method, when the squaring after rectification is included, a nice distinction between the opponent currents in the preferred and null directions is achieved. That is, the opponent current in the preferred direction oscillates above 7 nano amperes and the opponent current in the null direction oscillates below -7 nano amperes. There is a dead zone between 7 and -7 nano amperes. When the squaring after rectification is not computed, one can see that there is no dead zone. The opponent current in the preferred direction oscillates above -25 nano amperes and it oscillates below 25 nano amperes in the null direction. But one can still clearly distinguish the direction of motion by doing a temporal averaging. This is the price one pays for reducing the transistor count.

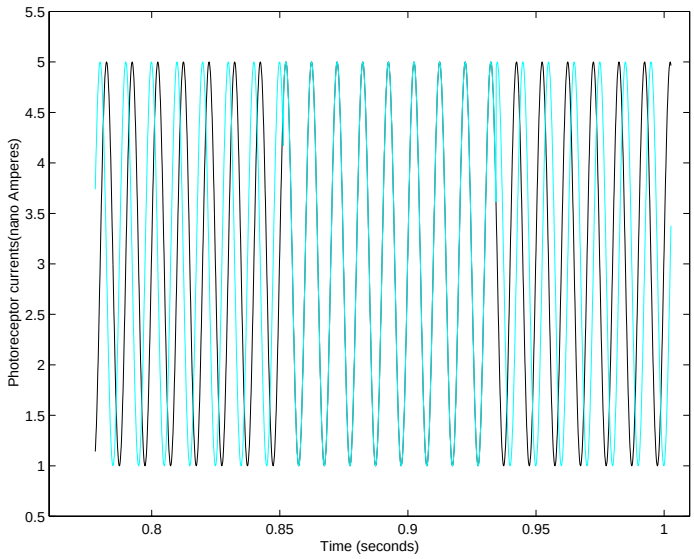
In this section two circuit level changes are described, one is the use of a normalized squaring circuit and the second change is getting rid of the squaring completely, thus using only rectification in the non-linearity stage. A revised version of the motion sensor was fabricated which incorporated the above changes.

7.2 Issues in System Level Design

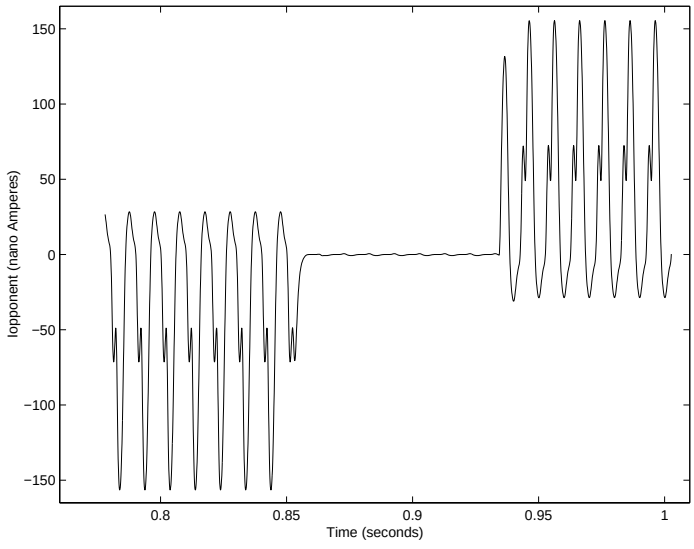
In Chapters 5 and 6, two applications of system level design based on the AB sensor were described. Though only two applications in this work were mentioned, as described in Chapter 1, one can also couple motion with other visual tasks like stereo disparity measurement to realize more complex behaviors. In (Schwager, 2000), the author shows how Adelson-Bergen algorithm can be used in stereoscopic vergence control and tracking. As the initial computation stages of Adelson-Bergen algorithm in this sensor already exist, one can easily integrate both motion and stereo-disparity for obtaining complex behaviors. Though this sensor is on a single monolithic block, one can implement the Adelson-Bergen algorithm as a multi-chip system. By using different chips to perform different stages of computation, one can couple motion and other visual tasks more efficiently. Work on such a multi chip system is already in progress.

7.3 Summary

In this work a motion sensor based on a biologically inspired algorithm for motion detection, the Adelson-Bergen algorithm was described. A detailed description of its VLSI implementation was also given. Results from extensive characterization of the chip were reported. Two application examples of this sensor were described. One was an active tracking mechanism using the chip that was fabricated. The second was the design of a single monolithic robot chip (RoaCh), which combines the motion sensor and a control scheme on it. RoaCh can be used on any commercially available robot like a Khepera (K-Team Inc Online, 2001) or a custom made robot. The details of the modeling of an early visual pathway of the fly which is thought to be involved in motion computation was also described.



(a) Photo currents of adjacent pixels



(b) Opponent motion energy current

FIGURE 7.3. Simulation results using rectification alone in the non-linearity stage. (a) In this plot each trace corresponds to the photocurrent detected in adjacent pixels. During the first time interval, the stimulus is in the null direction (light trace leads the darker trace), then it is orthogonal (the traces have no phase difference), after which the stimulus is in the preferred direction (darker trace leads the lighter trace). (b) The opponent motion energy current is plotted in this trace.

References

Adelson, E.H. and J.R. Bergen (1985). Spatiotemporal energy models for the perception of motion. *J. Optical Society of America A* 2(2): 284–299.

Anandan, P. (1989). A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision* 2: 283–310.

Andreou, A.G. and K. Strohbehn (1990). Analog VLSI implementation of the Hassenstein-Reichardt-Poggio model for vision computation. In *Proc. IEEE Intnl. Conf. on Systems, Man, and Cybernetics*, pp. 708–710, Los Angeles, CA.

Barlow, H.B. and W.R. Levick (1965). The mechanism of directionally selective units in the rabbit's retina. *J. Physiology* 178: 447–504.

Barnard, S.T. and W.B. Thomson (1980). Disparity analysis of images. *IEEE Trans. Pattern Analysis and Machine Intelligence* 2(4): 333–340.

Barrows, G.L. (1998). Feature tracking linear optic flow sensor for 2-d optic flow measurement. In *Proceedings of the International Conference on Automation, Robotics and Computer Vision*, pp. 732–736.

Barrows, G.L., K.T. Miller, and B. Krantz (1999). Fusing neuromorphic motion detector outputs for robust optic flow measurement. In *International Joint Conference on Neural Networks*, Vol. 4, pp. 2296–2301.

Benson, R.G. and T. Delbrück (1991). Direction selective silicon retina that uses null inhibition. In Lippman, R.P., J.E. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pp. 756–653. Morgan Kaufmann, San Mateo, CA.

Borst, Alexander. (1990). How do flies land? *Bioscience* 40(2): 292–299.

Borst, Alexander. and Martin. Egelhaaf (1989). Principles of visual motion detection. *Trends in Neuroscience* 12: 297–306.

Bult, K. and H. Wallinga (1987). A class of analog CMOS circuits based on the square-law characteristic of a MOS transistor in saturation. *IEEE Journal of Solid-State Circuits* SC-22(3): 357–365.

Chahl, J.S. and Mandyam Srinivasan (1997). Reflective surfaces for panoramic imaging. *Applied Optics* 36: 8275–8285.

Chipmunk Tools (1988). The chipmunk system for electronic circuit simulation `http://pcmp.caltech.edu/chipmunk/index.html`. No author listed.

Cohen, A., Shihab Shamma, G. Indiveri, and T. Horiuchi (2000). *NSF Neuromorphic Engineering Workshop Report*. Telluride, CO.

Cummings, R.E., Viktor Gruev, and Mohammed Abdel Ghani (1998). VLSI implementation of motion centroid localization for autonomous vehicles. In *Advances in neural information processing systems*, Vol. 11, pp. 685–691. MIT Press.

Cummings, R.E., Jan Van der Spiegel, and Paul Mueller (1996). A visual smooth pursuit tracking chip. In Touretzky, D, M. Mozer, and M. Jordan, editors, *Advances in Neural Information Processing Systems 8*, pp. 706–712. MIT Press.

Delbrück, T. (1993). Silicon retina with correlation-based, velocity-tuned pixels. *IEEE Trans. Neural Networks* 4: 529–541.

Delbrück, T. and C.A. Mead (1996). Analog VLSI phototransduction by continuous-time, adaptive, logarithmic photoreceptor circuits. Technical report 30, Department of Computation and Neural Systems, California Institute of Technology.

Deutschmann, R. A. and C. Koch (1998). Compact real-time 2-D gradient based analog VLSI motion sensor. In *Proceedings of the Int. Conf. on Advanced Focal Plane Arrays and Electronic Cameras*, Zurich/Switzerland.

Deweerth, Stephen P. (1992). Analog VLSI circuits for localization and centroid. *International Journal of Computer Vision* 8(2): 191–202.

Douglass, J.K. and N.J. Strausfeld (1995). Visual motion detection circuits in flies: Peripheral motion computation by identified small field retinotopic neurons. *J. Neurosci.* 15: 5596–5611.

Douglass, J.K. and N.J. Strausfeld (1996). Visual motion detection circuits in flies: Parallel direction- and non-direction sensitive pathways between the medulla and lobula plate. *J. Neurosci.* 16: 4551–4562.

Douglass, J.K. and N.J. Strausfeld (1998). Functionally and anatomically segregated visual pathways in the lobula complex of a calliphorid fly. *J. Comp. Neurol.* 396: 84–104.

Edmund Optics Online (2001). Fiber optic image conduits to transmit images between polished surfaces `http://www.edmundoptics.com/iod/displayproduct.cfm?productid=1355`. No author listed.

Egelhaaf, M., K. Hausen, W. Reichardt, and C. Wehrhahn (1988). Visual course control in flies relies on neuronal computation of object and background motion. *Trends in Neuroscience* 11: 351–358.

Egelhaaf, Martin, Alexander Borst, and Werner Reichardt (1989). Computational structure of a biological motion-detection system as revealed by local detector analysis in the fly's nervous system. *J. Opt. Soc. Amer. A* 6: 1070–1087.

Etienne-Cummings, R., J. Spiegel, and P. Mueller (1997). A focal plane visual motion measurement sensor. *IEEE Trans. on Circuit and Systems I* 44(1): 55–66.

Etienne-Cummings, R, J. Spiegel, and P. Mueller (1999). Hardware implementation of a visual-motion pixel using oriented spatiotemporal neural filters. *IEEE Transactions on Circuits and Systems–II* 46(9): 1121–1136.

Fang, Y., M. Mizuki, M. Masaki, and B. Horn (2000). TV camera-based vehicle motion detection and its chip implementation. In *Proceedings of the IEEE Intelligent Vehicles Symposium.*

Franceschini, N, N Riehle, and A. Le Nestour (1989). Directionally selective motion detection by insect neurons. In Stavenga, DG and RC Hardie, editors, *Facets of Vision*, pp. 360–390. Springer, Berlin, Heidelberg.

Harrison, R.R. (2000). An Analog VLSI Motion Sensor Based on the Fly Visual System. Ph.D. diss., Department of Computation and Neural Systems, California Institute of Technology, Pasadena, CA.

Harrison, R.R. and C. Koch (1998). An analog VLSI model of the fly elementary motion detector. In Jordan, M. I., M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pp. 880–886. MIT Press.

Hassenstein, B. and W. Reichardt (1956). Systemtheoretische analyse der zeit-, reihenfolgen-, und vorzeichenauswertung bei der bewegungsperzeption des rüsselkäfers. *Chlorophanus. Z. Naturforch* 11b: 513–524.

Haykin, Simon. (1996). *Communication Systems.* John Wiley and Sons, New York, NY, third edition.

Heeger, D.J., E.P. Simoncelli, and J.A. Movshon (1996). Computational models of cortical visual processing. *Proc. Natl. Acad. Sci.* 93: 623–627.

Higgins, C., T Vanneck, P. Joshi, and Strausfeld N.J. (2001). A model for direction selectivity in an insect based on non-directional motion cells and appropriate to cross phyla comparisions. In *Society for Neuroscience Abstracts*, San Diego,CA.

Higgins, C. M., R. A. Deutschmann, and C. Koch (1999). Pulse-based 2D motion sensors. *IEEE Transactions on Circuits and Systems II* 46(6): 677–687.

Higgins, C. M. and C. Koch (1999). An integrated vision sensor for the computation of optical flow singular points. In Kearns, M. S., S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, Vol. 11, Cambridge, MA. MIT Press.

Higgins, C.M. and C. Koch (1997). Analog CMOS velocity sensors. In *Proceedings of Electronic Imaging '97 (SPIE volume 3019).*

Higgins, C.M. and S. Korrapati (2000). An analog VLSI motion energy sensor based on the Adelson-Bergen algorithm. In *Proceedings of the International Symposium on Biologically-Inspired Systems.*

Horiuchi, T., J. Lazzaro, A. Moore, and C. Koch (1991). A delay-line based motion detection chip. In Lippman, R.P., J.E. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems*, pp. 406–412. Morgan Kaufmann, San Mateo, CA.

Horn, B.K.P. and B.G. Schunck (1981). Determining optic flow. *Artificial Intelligence* 17: 185–203.

Indiveri, G., J. Kramer, and C. Koch (1996). Parallel analog VLSI architectures for computation of heading direction and time-to-contact. In Touretzky, D.S., M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, Vol. 8, pp. 720–726, Cambridge, MA. MIT.

Indiveri, Giacomo (1999). Neuromorphic analog VLSI sensor for visual tracking: Circuits and application examples. *IEEE Trans. on Circuit and Systems II* 46(11): 1337–1347.

K-Team Inc Online (2001). Khepera family of robots `http://www.k-team.com`. No author listed.

Koch, Christof (1999). *Biophysics of Computation: Information Processing in Single Neurons.* Oxford University Press: New York.

Kramer, J. (1996). Compact integrated motion sensor with three-pixel interaction. *IEEE Trans. Pattern Analysis and Machine Intelligence* 18: 455–460.

Kramer, J., R. Sarpeshkar, and C. Koch (1995). An analog VLSI velocity sensor. In *Proc. Int. Symp. Circuit and Systems (ISCAS)*, pp. 413–416, Seattle, WA.

Kramer, J., R. Sarpeshkar, and C. Koch (1997). Pulse-based analog VLSI velocity sensors. *IEEE Trans. Circuits and Systems II* 44: 86–101.

Laughlin, Simon B. (1989). Coding efficiency and design in visual processing. In Stavenga, DG and RC Hardie, editors, *Facets of Vision*, pp. 213–234. Springer, Berlin, Heidelberg.

Little, J.J., H.H. Bulthoff, and T.A. Poggio (1988). Parallel optical flow using local voting. In *Proceedings of 2nd international conference on Computer Vision*, pp. 454–459.

Liu, S. and K. Boahen (1996). Adaptive retina with center-surround receptive field. In Touretzky, D.S., M.C. Mozer, and M.E. Hasselmo, editors, *Advances in neural information processing systems*, Vol. 8. MIT Press.

Liu, S.C. (1998). Silicon retina with adaptive filtering properties. In Jordan, M.I., M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems*, Vol. 10, pp. 712–718, Cambridge, MA. MIT Press.

Mead, C.A. (1989). *Analog VLSI and Neural Systems.* Addison-Wesley, Reading.

Mead, Carver A. and Tobias. Delbrück (1991). Scanners for visualizing activity of analog VLSI circuitry. Technical report 11, Department of Computation and Neural Systems, California Institute of Technology.

Moshnyaga, V.G. and K. Tamaru (1997). A memory efficient array architecture for full-search block matching algorithm. In *Proc. of the 1997 IEEE International conference on Acoustics, Speech, and Signal processing, ICASSP-97.*, Vol. 5, pp. 4109–4112.

Nowlan, S.J. and T.J. Sejnowski (1994). Filter selection model for motion segmentation and velocity integration. *J. Optical Soc. America* 11(12): 3177–3200.

Qian, N., R.A. Andersen, and E.H. Adelson (1994). Transparent motion perception as detection of unbalanced motion signals. III. modeling. *J. Neuroscience* 14(12): 7381–7392.

Regan, Tim (1999). A DMOS 3A, 55V, H-Bridge: The LMD18200 Application Note. Technical report 694, National Semiconductor.

Sarpeshkar, R., J. Kramer, G. Indiveri, and C. Koch (1996). Analog VLSI architectures for motion processing: From fundamental limits to system applications. *Proceedings of the IEEE* 84: 969–987.

Schwager, M.A. (2000). Stereoscopic Vergence Control and Horizontal Tracking using Biologically Inspired Filters. Ph.D. diss., Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ.

Strausfeld, Nicholas James (1976). *Atlas of an Insect Brain.* Springer-Verlag, New York, NY.

Tanner, J. and C. Mead (1986). An integrated analog optical motion sensor. In *VLSI Signal Processing, II*, pp. 59–76. IEEE Press.

Traff, H. (1992). Novel approach to high speed CMOS current comparators. *Electronics Letters* 28(3): 310–312.

Van Santen, Jan P. H. and George Sperling (1985). Elaborated Reichardt detectors. *Journal of the Optical Society of America A* 2: 300–320.

Volkman, F., A. Schick, and L. Riggs (1968). Time course of visual inhibition during voluntary saccades. *Journal of the 0ptical Society of America* 58: 1410–1414.

Wang, Bor-Min, Jui-Cheng Yen, and Chang Shyang (1994). Zero waiting-cycle hierarchical block matching algorithm and its array architectures. *IEEE Transactions on circuits and systems for video technology* 4(1): 18–28.

Zhang, Xiao-Dong and Tsui Chi-Ying (1997). An efficient and reconfigurable vlsi architecture for different block matching motion estimation algorithms. In *Proc. of the 1997 IEEE International conference on Acoustics, Speech, and Signal processing, ICASSP-97.*, Vol. 1, pp. 603–606.

Zigmond, Michael J., Floyd E. Bloom, Story C. Landis, James L. Roberts, and Larry R. Squire (1999). *Fundamental Neuroscience*. Academic Press, San Diego, CA.