A Space-Variant Architecture for Active Visual Target Tracking

David Claveau

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

August 2007

Your file Votre référence
ISBN: 978-0-494-31132-5
Our file Notre référence
ISBN: 978-0-494-31132-5

ABSTRACT

A Space-Variant Architecture for
Active Visual Target Tracking

David Claveau,
Concordia University, 2007

An active visual target tracking system is an automatic feedback control system that can track a moving target by controlling the movement of a camera or sensor array. This kind of system is often used in applications such as automatic surveillance and human-computer interaction. The design of an effective target tracking system is challenging because the system should be able to precisely detect the fine movements of a target while still being able to detect a large range of target velocities. Achieving this in a computationally efficient manner is difficult with a conventional system architecture.

This thesis presents an architecture for an active visual target tracking system based on the idea of space-variant motion detection. In general, space-variant imaging involves the use of a non-uniform distribution of sensing elements across a sensor array, similar to how the photoreceptors in the human eye are not evenly distributed. In the proposed architecture, space-variant imaging is used to design an array of elementary motion detectors (EMDs). The EMDs are tuned in such a way as to make it possible to detect motion both precisely and over a wide range of velocities in a computationally efficient manner. The increased ranges are achieved without additional computational costs beyond the basic mechanism of motion detection. The technique is general in that it can be used with different motion detection mechanisms and the overall space-variant structure can be varied to suit a particular application.

The design of a tracking system based on a space-variant motion detection array is a difficult task. This thesis presents a method of analysis and design for such a tracking system. The method of analysis consists of superimposing a phase-plane plot of the continuous-time dynamics of the tracking system onto a map of the detection capabilities of the array of EMDs. With the help of this 'sensory-motor' plot, a simple optimization algorithm is used to design a tracking system to meet particular objectives for settling time, steady-state error and overshoot. Several simulations demonstrate the effectiveness of the method. A complete active vision system is implemented and a set of target tracking experiments are performed. Experimental results support the effectiveness of the approach.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Professor Chunyan Wang for supporting this research and for all of the support she has provided for over seven years. I would also like to thank the members of my committee: Professor M.Omair Ahmad, Professor Wei-Ping Zhu, Professor Chun Yi Su and Professor Magdy Bayoumi. In particular I would like to thank Professor Bayoumi for making time in his schedule to come to Montreal to serve as the external examiner. I would also like to thank Engineering Specialist Ted Obuchowicz for all of his help. Most of all I would like to thank the people of Canada for supporting a system of education that is accessible to an ordinary person like me.

Contents

List of Figures

List of Abbreviations

CCD	charge-coupled device
CMOS	complementary metal oxide semiconductor
CPU	central processing unit
DSP	digital signal processor
VLSI	very large scale integration
EMD	elementary motion detector
FPGA	field programmable gate array
DC	direct current

List of Symbols

$\mathbf{A}$	state matrix for tracking system model
$\mathbf{B}$	input matrix for tracking system model
d	distance from target plane to image sensor plane
δx	elementary spatial unit
δt	elementary temporal unit
Δt	time delay for motion detection
Δx	center-to-center distance between adjacent pixels or photoreceptors
f_t	temporal frequency
f_x	spatial frequency
$I(x,t)$	spatio-temporal input signal to EMD
$\mathbf{K}(q,n)$	gain vector for EMD map
l	length of target
L	total length of the sensor array
m_1	result of signal multiplication in the Reichardt EMD
m_2	result of signal multiplication in the Reichardt EMD
n	rectangle number in EMD map
p_1	signal at a pixel or photoreceptor
p_2	signal at a pixel or photoreceptor
$p_{1'}$	delayed signal at a pixel or photoreceptor
$p_{2'}$	delayed signal at a pixel or photoreceptor
q	quadrant number in EMD map
R	response of the Reichardt EMD
$R_{12}(\tau)$	cross-correlation
$R_{11}(\tau)$	auto-correlation
s	complex frequency variable for Laplace transforms
t	time variable
τ	correlation lag
$u(q,n)$	control signal for tracking
v	target velocity in the image plane
$v_a(t)$	array velocity
v_{peak}	target velocity that elicits peak response in the EMD
v_r	relative velocity between target and array
$v_t(t)$	target velocity in image plane
$V_t(t)$	target velocity in the world
x	space variable
$x_a(t)$	array position
x_r	relative position between target and array
$\mathbf{x}_r(t)$	relative state vector between target and array
$\hat{\mathbf{x}}_r(t)$	estimate of the relative state vector
$x_t(t)$	target position in image plane
$X_t(t)$	target position in world

1

Introduction

1.1 Active Visual Target Tracking

The ability to detect and track moving objects is an important part of our ability to interact with the world around us. Our vision system allows us to detect motion at a distance. Light from a moving object reaches our eyes and we are able to shift our gaze to keep the object within our field-of-view. Actively shifting our gaze allows us to track the motion of the object across a much larger field-of-view than our eyes provide. These abilities have inspired researchers and engineers to build active vision systems that can track a moving target by controlling the movement of a camera or sensor array to keep its optical axis aligned with the target. This kind of work has been motivated by useful applications in automatic surveillance [1][2], human-computer interaction [3][4], and mobile systems such as exploratory robots that can use active visual tracking to perform a

variety of visually-guided tasks [5][6]. Even entertainment robots are easier to interact with when given the ability to visually track objects in their environment [7].

An active visual target tracking system is a special type of automatic feedback control system consisting of three functional components: i) vision, ii) control and iii) actuation, as shown in Fig. 1.1. Particular to this type of control system is the vision component which consists of a sensing stage and a processing stage. Sensing of the light reflected or emitted by a target is usually performed by a 2-D array of light-sensitive elements. Over the last 50 years there have been three main types of sensor arrays: the vidicon tube, the CCD sensor and the CMOS sensor.

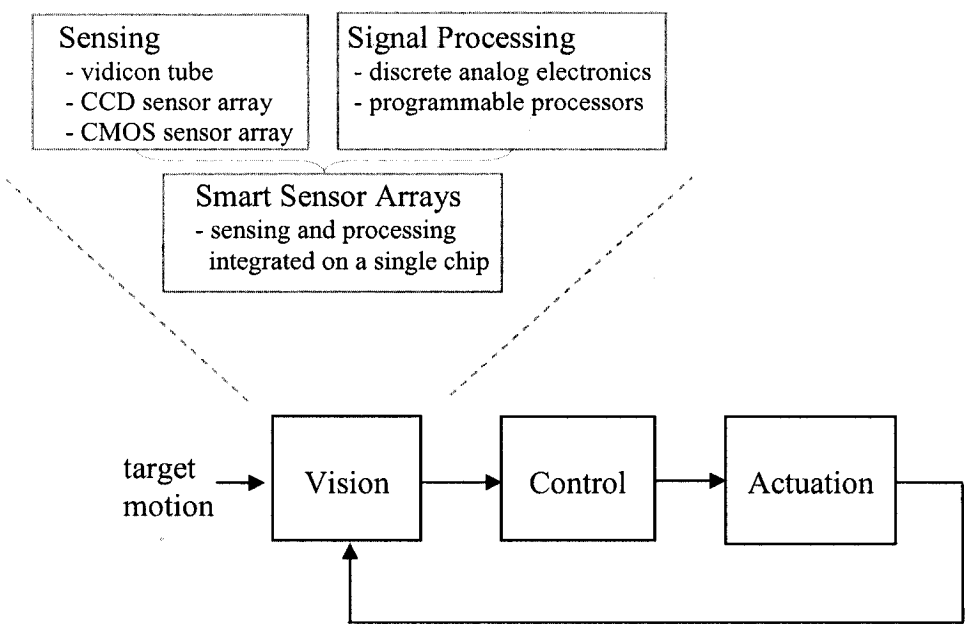


Fig. 1.1. Functional block diagram of an active visual tracking system listing the sensing and signal processing technologies typically used.

The signal processing stage operates on the image signal produced by the sensor array. Information about the target is extracted from the image signal. Usually a tracking error, which may consist of a position error and a velocity error, is computed. The control component uses the tracking error to generate a control signal for an actuator, such as a motor, which physically moves the vision system accordingly. Subsequent observations by the vision component are affected by this movement, closing the loop between sensing and acting.

The signal processing stage is critical to the operation of the system because it must compute the tracking error in real-time. A variety of strategies have been used for this processing, ranging from discrete analog circuits to programmable digital computers. With the advent of very large scale integration (VLSI) in the last 20 years there have been attempts to integrate the signal processing with the sensing on a single chip. These smart sensors have become a viable implementation option in applications that require small size and low power consumption [8].

By far the most common strategy has been to use a video camera, based on a CCD or CMOS image sensor, connected to a digital computer. A block diagram of such a system is shown in Fig. 1.2. Digitized images from a camera are stored in a frame buffer where they are processed by a computer. The computer may be programmable or it may be an application specific integrated circuit (ASIC). The computer can make use of various algorithms to extract useful information from the images. However, despite advances in algorithm development and improvements in technology, active visual tracking remains a difficult problem. The specific challenges are described in the next section.

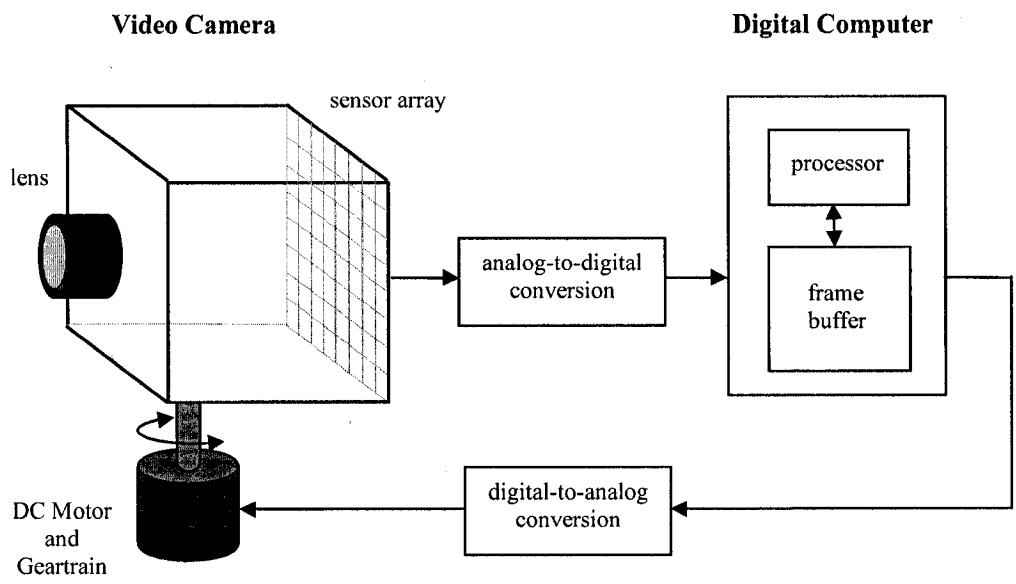


Fig. 1.2. Component block diagram of an active vision system that consists of a video camera connected to a digital computer. Digitized images from the camera are stored in a frame buffer and the computer makes use of various algorithms to extract useful information from the images.

1.2 Challenges of Active Tracking

The design and implementation of an effective tracking system involves several challenges. First, an effective target tracking system should be able to detect a large range of target velocities while still being able to precisely detect the fine movements of a target. This combination of large range of operation and high precision is difficult to achieve. Another challenge is the need to extract data from an image signal that can be used to produce a useful action in a real-time control loop. This type of sensory-motor coordination is challenging because it is often difficult for the designer to visualize how the sensing capabilities of a system relate to the motor capabilities. Yet another challenge

is the need for all computation to be sufficiently fast so as not to incur delays in the overall feedback loop that will lead to instability and loss of tracking. This is particularly true for computations that involve successive images acquired by the vision system. The volume of data is usually very large and strategies are needed to keep the necessary computations simple and efficient. Often the image capture rate is 30 frames per second as shown in Fig. 1.3. A typical image resolution will lead to a very high bit rate. Processing such a signal requires high speed circuits and high clock frequencies, depending on the algorithms being used for the tracking. Power-hungry computational resources cannot be used on mobile systems that have obvious constraints on power consumption.

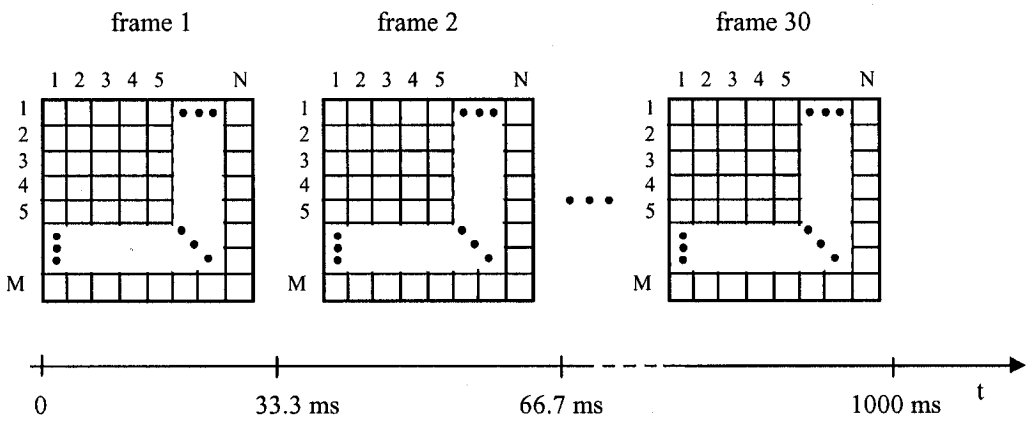


Fig. 1.3. Typical video camera: frames processed every 33.3 ms. The throughput of such a system can be very high even at modest resolutions.

Designing and implementing an active visual target tracking system is a multidisciplinary task. Techniques from computer vision, control systems and electrical and mechanical engineering must be applied to the design of a complete system. Integrating these techniques into a cohesive approach to system analysis and design is a

major challenge. A complete system model is essential for the design and optimization of such a system for a particular application.

One way of addressing all of these challenges is by taking inspiration from nature. Clearly our own vision system is capable of active visual target tracking over an impressive range of target speeds. Sensory-motor coordination is performed effortlessly. This is also true for many other animals with vision systems similar to our own. Researchers have begun to consider how strategies found in nature can be adapted to the design of active vision systems [9][10]. This thesis addresses all of the challenges described in this section by applying strategies found in biological vision systems.

1.3 Scope and Organization of the Thesis

The objective of this thesis is to achieve active visual target tracking with both precision and over a wide range of speeds in a computationally efficient manner. Another objective is to develop an approach to the analysis and design of such a system and to address the difficulties of sensory-motor coordination. The approach is bio-inspired. It makes use of strategies for: i) sensing, ii) motion detection and iii) sensory-motor coordination that are inspired by what is known about biological vision systems. Using these strategies this thesis describes an architecture for a space-variant motion detection array that is designed to efficiently track a target over a wide range of speeds while still being able to precisely track its fine movements. Although it is possible to solve this problem by using faster and more powerful computers, this leads to systems that are inefficient in terms of resource usage. The approach in this thesis solves the problems of active visual tracking without using excessive computational resources. Here we use the

word 'architecture' to describe the 'sensing scheme' or the structure of the sensing that will lead to more efficient computation.

The thesis is organized as follows. Chapter 2 consists of a review of background topics including a survey of the basic motion detection techniques and a discussion of related work done by other researchers. Chapter 3 describes the spatio-temporal frequency response of a commonly used motion detector and how it can be incorporated into a space-variant array to provide both range and precision for target tracking. Chapter 4 describes the proposed target tracking system along with a method of analysis and design for the system and how it can be used to meet performance goals for settling time, steady-state error and overshoot. In chapter 5 an experimental system is described along with the results of target tracking experiments. Chapter 6 contains the conclusion and a discussion of future work.

2

Background and Related Work

The design and implementation of an effective visual tracking system involves techniques from fields such as computer vision and control systems. At the core of visual tracking is the process of extracting information about the target's motion from the image signal. This chapter provides background information on a variety of techniques that have been used for visual motion detection for target tracking. In section 2.1 several motion detection techniques are considered from a target tracking perspective highlighting some of their advantages and disadvantages. Particular attention is paid to the spatio-temporal frequency-based techniques of section 2.1.5, as the work of this thesis is based on these techniques. Other researchers have attempted to use some of these techniques and a review of the approaches they have taken is provided in section 2.2. Section 2.3 concludes the chapter with a brief description of some bio-inspired strategies that are central to the approach taken in this thesis.

2.1 Measuring Motion for Visual Tracking

This section is a survey of several commonly used techniques for visual motion detection and estimation. The goal of these techniques is to extract useful motion information from time-varying imagery. The patterns of visual motion include those produced by the target's motion relative to the sensor and also those produced by the sensor's motion relative to the background. It is assumed that motion due to the background is insignificant compared to motion due to the target. For target tracking, relevant motion information includes the *position* of the motion in the image, the *direction* of motion and the *speed*. The techniques are mathematically very similar [11][12] though they differ in terms of their computational complexity, their generality, and their precision and range.

2.1.1 Techniques Based on Matching

This section briefly describes three commonly used motion estimation techniques based on matching: high-level template matching, feature matching and block matching. High-level template matching techniques make use of a 2-D template or a small image of the target to search for the position of the target in each image as shown in Fig. 2.1. This is done by 'sliding' a template, T, over an image, I, and finding the best match by minimizing a distance measure such as the sum of square differences (SSD):

$$SSD(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} [T(i,j) - I(x+i, y+j)]^2 \tag{2.1}$$

in which (x,y) is the position of the candidate window in I and $2k+1$ is the size of both the template and the candidate window. The SSD has convenient mathematical properties but may be less efficient than the sum of absolute differences (SAD):

$$SAD(x,y) = \sum_{i=-k}^{k}\sum_{j=-k}^{k} |T(i,j) - I(x+i,y+j)| . \tag{2.2}$$

Alternatively the best match may be found by maximizing a similarity measure such as the cross-correlation:

$$R(x,y) = \sum_{i=-k}^{k}\sum_{j=-k}^{k} T(i,j)\cdot I(x+i,y+j) . \tag{2.3}$$

The maximum of $R(x,y)$ indicates that the best match is at location (x,y).

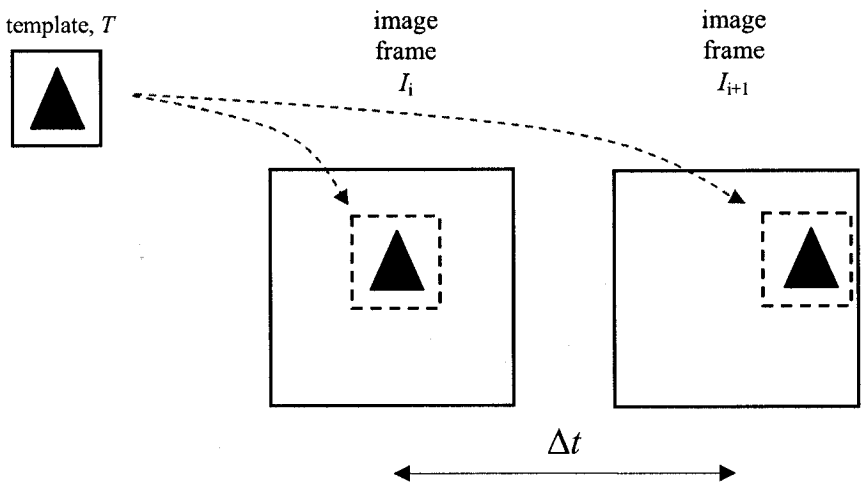


Fig. 2.1. Template matching techniques begin with a template image of the target and search each image frame for the best match.

As described above, by searching for the best match it is possible to find the position of the target in each image. The target speed can then be estimated by dividing the change in position by the amount of time between images. If the entire image is searched

then the technique can detect a wide range of velocities. One drawback is that a template or model of the target must be available *a priori*. Another drawback is that the target may appear differently in the image and multi-resolution techniques may be needed. Also, the accuracy of matching is usually better when more pixels are compared, leading to an accuracy/computational efficiency tradeoff.

Examples of template matching being used for target tracking include some of the first visual tracking systems developed for classified military applications such as missile guidance [13]. The availability of digital computers and CCD-based video cameras in the 1970's and 1980's lead to several other systems based on template matching [14][15], though usually limited to contexts in which the target's appearance is known.

Feature matching techniques begin by extracting features such as edges [16] or corners [17] in each image. For each feature in the current image, the corresponding feature in the previous image must be searched for. If features can be properly matched between the current image and the previous image then the motion vector for that feature can be estimated. This is called the correspondence problem [18] and it is an example of an ill-posed problem for which there is no unique solution; multiple features could be legitimate matches. This technique also depends on the presence of a sufficient number of features in the images to compute motion. Feature extraction itself can be computationally demanding and susceptible to noise.

Block matching techniques [19] are very similar to template matching techniques. Each image in a sequence is partitioned into an array of blocks. Each block is then searched for in the previous image frame, within a search window, as shown in Fig. 2.2.

The best match can be determined using the SSD, SAD or correlation. For example, the sum of absolute differences (SAD) may be computed as:

$$SAD(p_1, p_2) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} | I_2(x_2 + i, y_2 + j) - I_1(x_1 + i, y_1 + j) | \tag{2.4}$$

in which $p_1 = (x_1, y_1)$ is the center of the candidate block in I_1, $p_2 = (x_2, y_2)$ is the center of the block in I_2, and the block size is $2k+1$. This must be computed for each possible position in the search window to find the best match.

A motion vector can be assigned to each block producing a motion vector field sometimes referred to as an estimate of the optical flow. The size of the blocks and the search window are adjustable. A larger block size means more accurate motion vectors but results in a sparser vector field and requires more computations. A sparse vector field means that only coarse image features can be assigned a motion vector.

In summary, all of the matching techniques can be used to detect motion over a wide range of velocities however they exhibit a tradeoff between accuracy and computational efficiency.

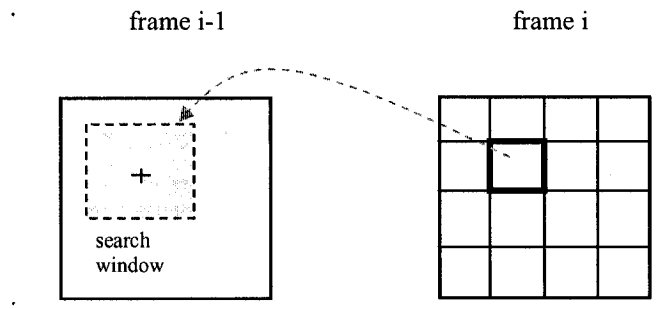


Fig. 2.2. Block matching techniques partition each image into an array of blocks each of which is then searched for within a search window in the previous image frame.

2.1.2 Gradient-Based Techniques

A simple way to detect motion is to detect a temporal change at each pixel. This does not require any searching or sliding window computations. All that is needed is an approximation to a temporal derivative of the input at each pixel:

$$\frac{d}{dt}I(x,y,t) = \lim_{\Delta t \to 0} \frac{I(x,y,t) - I(x,y,t-\Delta t)}{\Delta t}. \tag{2.5}$$

A pixel-level block diagram in Fig. 2.3 is used to show the computations needed to approximate this derivative. Only one spatial dimension is shown and the inputs, p_1 and p_2, are the values of image intensity at two adjacent pixels. The blocks can be implemented with either digital or continuous-time components and all that is needed is a delay and a subtraction. Similar to a simple feedforward comb filter, the output for each pixel is formed as a linear combination of the input and output of a delay line.

If the input is considered to be a sequence of images, as in a digital video signal, then the previous image must be subtracted from the current image, pixel by pixel. The position of motion can be estimated in this way. The estimate becomes less precise as the speed of image motion increases since many pixels will have changed after a delay of Δt. Variations on this simple technique have been used for active visual tracking [20].

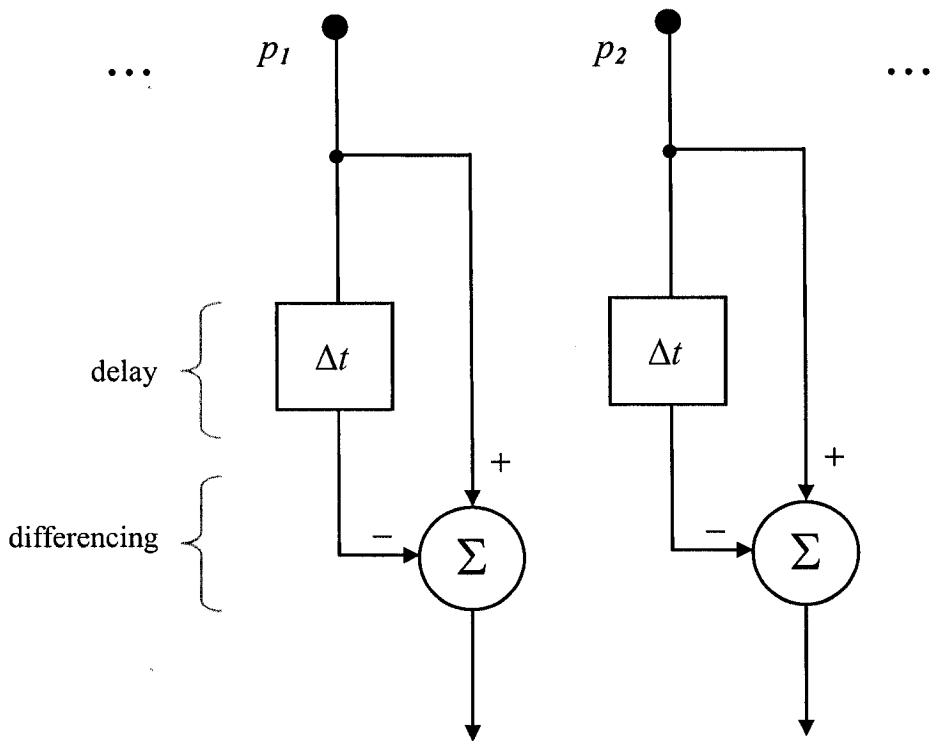


Fig. 2.3. Block diagram of the computations needed to approximate a temporal derivative at each pixel. Only one spatial dimension is shown.

In order to estimate more than just the position of motion, an approximation to a spatial derivative at each pixel is needed. For example, in the x-direction:

$$\frac{d}{dx}I(x,y,t) = \lim_{\Delta x \to 0} \frac{I(x,y,t) - I(x-\Delta x,y,t)}{\Delta x}. \tag{2.6}$$

An estimate of motion speed in one dimension can be computed as a ratio of the temporal and spatial derivatives [21][22]:

$$v = \frac{dx}{dt} = \frac{\frac{dI}{dt}}{\frac{dI}{dx}} = \frac{I_t}{I_x} \tag{2.7}$$

in which I_t and I_x denote the spatial and temporal derivatives of I respectively. A pixel-level block diagram in Fig. 2.4 shows the computations for two adjacent pixels in one spatial dimension. This technique has been extended to 2-D [23]. The usual derivation begins by assuming that image intensity remains constant:

$$\frac{d}{dt}I(x,y,t)=0 \tag{2.8}$$

then, using the Taylor expansion:

$$\frac{\partial I}{\partial x}v_x+\frac{\partial I}{\partial y}v_y+\frac{\partial I}{\partial t}=0. \tag{2.9}$$

in which v_x and v_y are the local velocities in the x and y directions respectively. This can be written as:

$$I_x v_x+I_y v_y+I_t=\nabla I\cdot\mathbf{v}+I_t=0. \tag{2.10}$$

The velocity perpendicular to the spatial gradient can be estimated as:

$$\mathbf{v}=\frac{-I_t}{|\nabla I|}=\frac{-I_t}{\sqrt{I_x^2+I_y^2}}. \tag{2.11}$$

Other components of the velocity cannot be estimated. This is referred to as the aperture problem. To solve for both v_x and v_y, assumptions are usually made about the 'smoothness' of the image motion, but these are only valid for low image velocities. Another problem is that pixel-level estimates of the spatial derivative are often noisy and unreliable, resulting in poor accuracy. These factors have made it difficult to apply this technique to visual tracking. On the other hand, the technique is conceptually simple and requires only basic arithmetic operations. There have been successful implementations of this technique including an integrated smart sensor developed by Tanner and Mead [24] and one developed by Stocker [25].

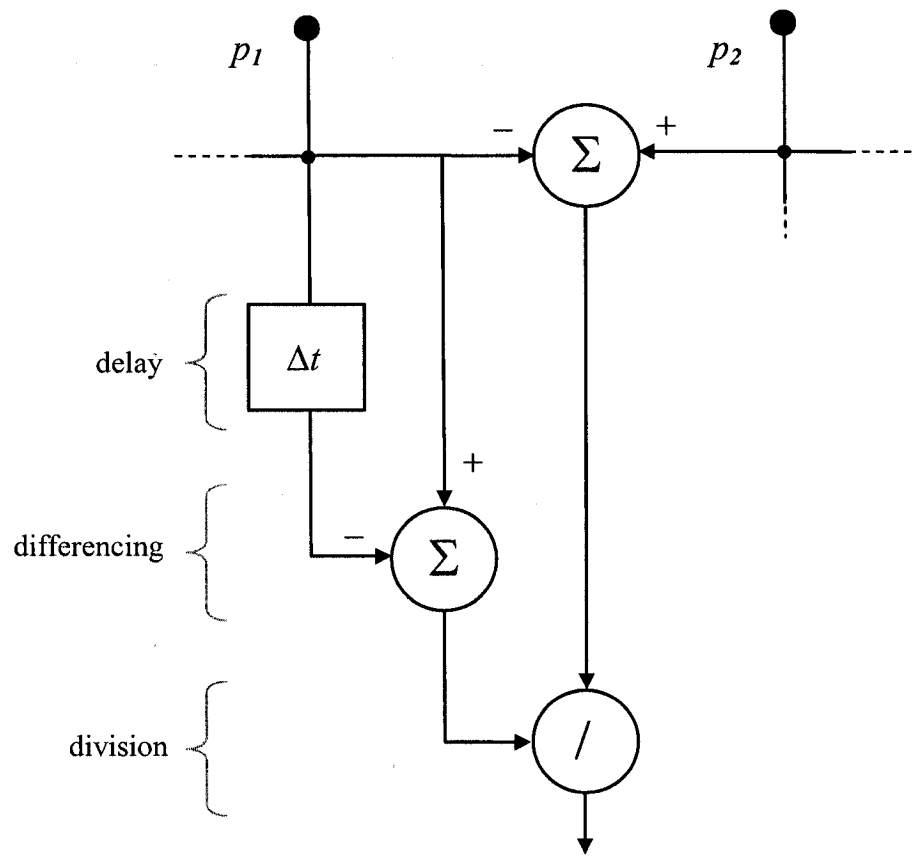


Fig. 2.4. Block diagram of the computations needed to estimate motion speed in one dimension as a ratio of the temporal and spatial derivatives.

2.1.3 Spatio-temporal Frequency-Based Models

Biological vision systems appear to perform motion detection in a way that is different from the matching and gradient techniques. Evidence suggests that these systems compute the spatio-temporal frequency content of local image regions [26][27]. Individual spatio-temporal frequency filters are arranged in an array. Each filter is selectively activated by a range of spatio-temporal frequency patterns produced by the motion of a target. This allows motion to be computed across the array without restrictive assumptions. One commonly used filter structure is based on correlation [26] and another

is based on 'motion energy' [28]. They are mathematically equivalent, however, the work of this thesis is based on the correlation-type filter because of its simplicity.

Cross-Correlation is a measure of the degree to which two signals agree in behavior [29]. It is used as a similarity measure for the template matching techniques of section 2.1.1. If we consider two arbitrary continuous-time signals, $f_1(t)$ and $f_2(t)$, the cross-correlation, $R_{12}(\tau)$, can be expressed as:

$$R_{12}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_0^T f_1(t-\tau) \cdot f_2(t)\, dt \,. \tag{2.12}$$

In actual practice, the average is taken over a finite length of time, T, and what is usually measured is:

$$R_{12}(\tau, T) = \frac{1}{T} \int_0^T f_1(t-\tau) \cdot f_2(t)\, dt \,. \tag{2.13}$$

The block diagram of this computation is shown in Fig. 2.5 (a). When the two signals are the same, the operation is called an autocorrelation and it will peak for a lag of $\tau = 0$.

A similar configuration can be used for pixel-level motion detection, as shown in Fig. 2.5 (b). The two signals, p_1 and p_2, are point samples of the image plane, separated by a distance Δx. The image intensity at p_1 is delayed by Δt and then multiplied by the intensity at p_2. If a target, with a uniform intensity and a size that is smaller than Δx, moves rightward at velocity v then it will travel from p_1 to p_2 in time $\Delta x/v$. This operation is similar to an autocorrelation for which the 'effective lag' τ depends on Δx, Δt and v:

$$\tau = \Delta t - \frac{\Delta x}{v} \,. \tag{2.14}$$

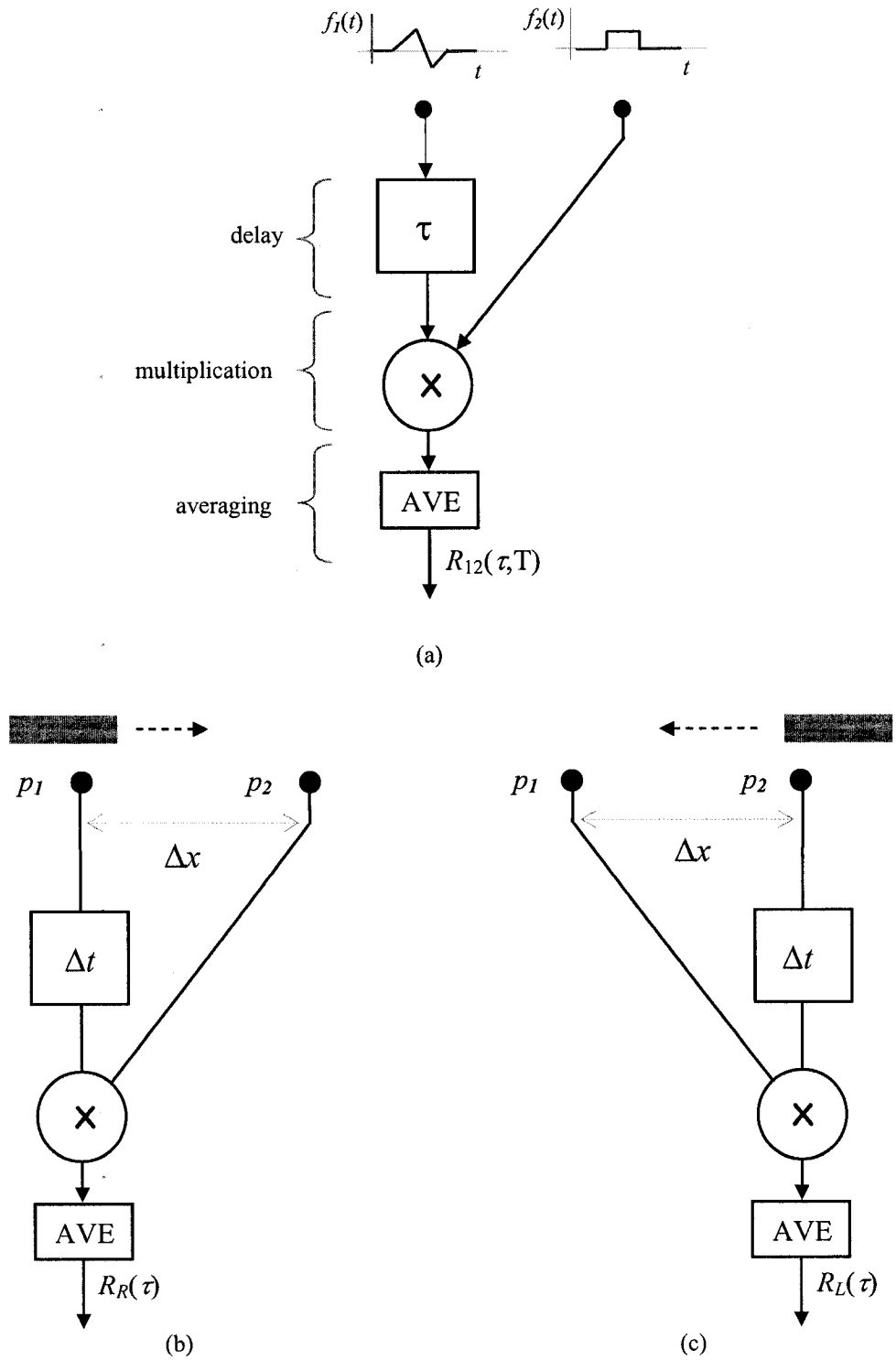


Fig. 2.5. (a) Block diagram of a cross-correlation operation. (b) If p_1 and p_2 are signals from adjacent pixels then this simple computation can be used to detect rightward motion and (c) leftward motion.

If the time it takes for the target to pass from p_1 to p_2 is approximately the same as Δt, then the amplitude of the output of the correlator will likely be large resulting from the product of the two signals. It will peak for a target velocity of $\Delta x/\Delta t$. For a range of target velocities the effective lag will be small and the output of the correlator will be close to the peak value. If the target moves leftward from p_2 to p_1 then the effective lag will increase and the configuration in Fig. 2.5 (b) will not respond to this motion. Rather, if p_2 is delayed by Δt instead of p_1, as in Fig. 2.5 (c), then the correlator will be sensitive to leftward motion rather than rightward motion. This type of 'delay-and-compare' motion detection was first reported by Reichardt [26] in the 1950's after experimenting with the visual capabilities of insects.

Reichardt combined two correlators in a symmetric fashion as shown in Fig. 2.6. This Reichardt elementary motion detector (EMD) computes the difference between the output of two correlators that are sensitive to motion in opposite directions. This 'motion-opponency' results in a positive output for rightward motion or a negative response for leftward motion and eliminates the response to full-field flicker. If a target with uniform intensity and a size that is several times larger than Δx passes over the EMD then the motion of its leading edge will be detected but when the object completely covers p_1 and p_2 then the two sides of the EMD will cancel and the response will be zero.

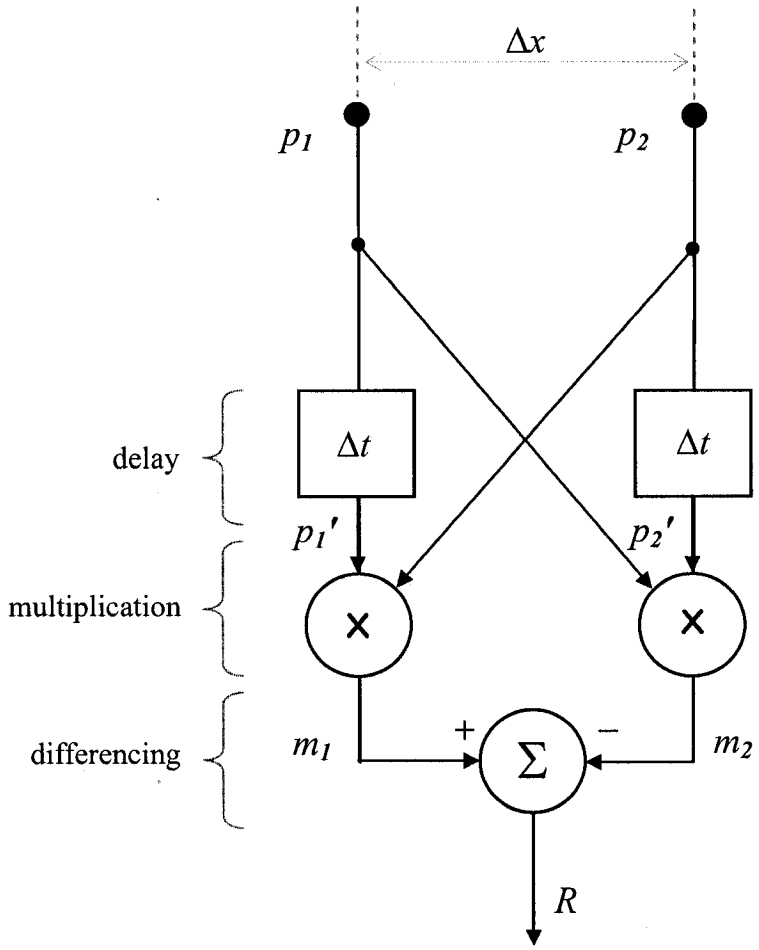


Fig. 2.6. Block diagram of a Reichardt elementary motion detector (EMD).

Real-world objects and scenes have complex patterns. In order to use the Reichardt EMD to track real-world targets it is important to understand how it responds to complex patterns. To begin, consider a target with a simple sinusoidal pattern on its surface as shown in Fig. 2.7 (a). Assume the length of the target to be just large enough to see a few wavelengths of the pattern. For simplicity consider the intensity profile, I(x), along the x-dimension. If the average or DC component is subtracted from the intensity profile then it can be represented with a zero-mean sinusoid:

$$I(x) = \sin(2\pi \cdot f_x x) \tag{2.15}$$

in which f_x, the spatial frequency, is measured as the number of times the spatial pattern repeats itself over some unit distance. The target is moving at a constant velocity, v, which means the sinusoid is being shifted to the right:

$$I(x,t) = \sin(2\pi \cdot f_x x - 2\pi \cdot f_x vt). \tag{2.16}$$

A space-time plot of the sinusoid moving to the right is shown in Fig. 2.7 (b). The initial position is shown in bold along the x-axis. The sinusoid shown in bold along the time axis is the signal that would be observed at a given point on the x-axis, in this case the 0 point, as the wave moves to the right. Its temporal frequency f_t is related to the spatial frequency f_x and velocity v by $f_t = f_x v$. If the wave moves to the right at a higher velocity, as shown in Fig. 2.7 (c), then the temporal frequency will increase. Likewise f_t will increase if f_x is increased as in Fig. 2.7 (d) and v is held constant, or if both v and f_x are increased as in (e). Equation (2.14) can now be written as:

$$I(x,t) = \sin(2\pi \cdot f_x x - 2\pi \cdot f_t t) \tag{2.17}$$

and the sinusoid along the time axis is:

$$I(t) = -\sin(2\pi \cdot f_t t)\big|_{x=0}. \tag{2.18}$$

The motion of a target will produce a particular pattern of spatial and temporal frequencies depending on its appearance, its direction of motion and its speed. The Reichardt EMD can be tuned to detect a particular spatio-temporal frequency pattern by changing its distance parameter Δx and its delay parameter Δt.

Now consider the classical autocorrelation of a sinusoidal function of time:

$$R(\tau) = \frac{1}{2T}\int_{-T}^{T} \sin(2\pi f_t \cdot (t-\tau)) \cdot \sin(2\pi f_t \cdot t)\, dt$$
$$= \frac{1}{2}\cos(2\pi f_t \cdot \tau). \tag{2.19}$$

This function is periodic in τ and has a peak at $\tau = 0$. Again, consider the simple one-sided correlator which detects rightward motion, shown previously in Fig. 2.5 (b), but this time with a sinusoid passing over it. If the crest of the sinusoid moves rightward at velocity v then it will travel from p_1 to p_2 in time $\Delta x/v$. As before we have an 'effective lag' τ:

$$\tau = \Delta t - \frac{\Delta x}{v} \tag{2.20}$$

which can be substituted into equation (2.19) to give us the output of the correlator for rightward motion:

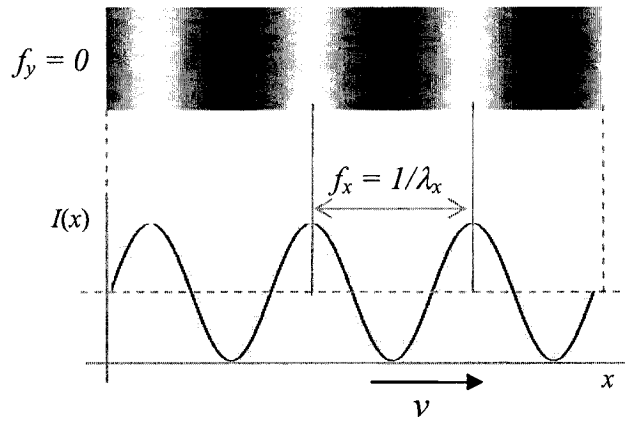
$$R_R(\tau) = \frac{1}{2}\cos\left(2\pi f_t \cdot \left(\Delta t - \frac{\Delta x}{v}\right)\right) \tag{2.21}$$

which simplifies to:

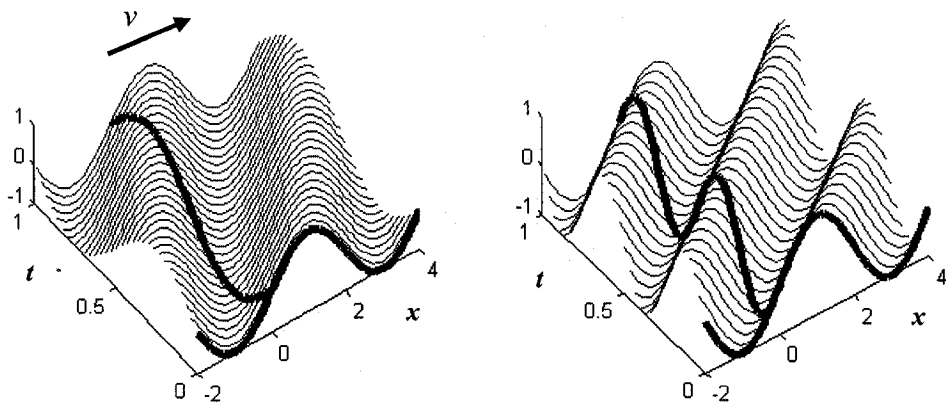
$$R_R(\tau) = \frac{1}{2}\cos(2\pi f_t \cdot \Delta t - 2\pi f_x \Delta x). \tag{2.22}$$

Using the trigonometric identity for the sum of two angles we have:

$$R_R(\tau) = \frac{1}{2}(\cos(2\pi f_t \cdot \Delta t)\cos(2\pi f_x \Delta x) + \sin(2\pi f_t \cdot \Delta t)\sin(2\pi f_x \Delta x)). \tag{2.23}$$



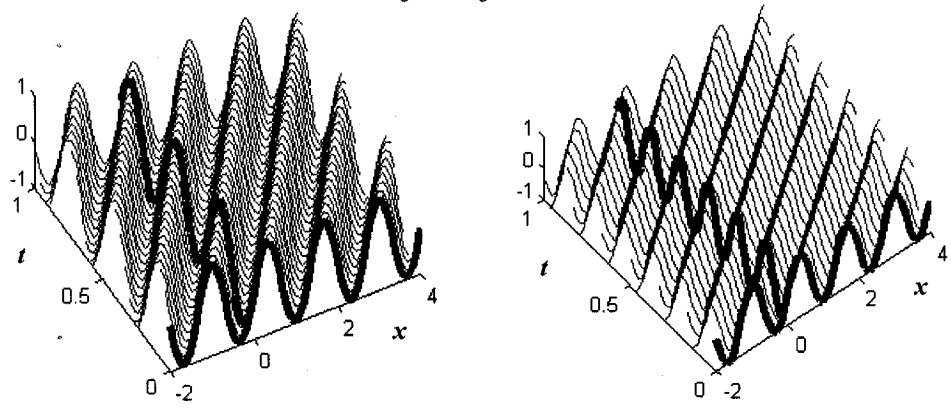
(a)



(b)

(c)

$$f_t = f_x v$$



(d)

(e)

Fig. 2.7. (a) A simple target with a sinusoidal pattern moving to the right with velocity v. (b) A space-time plot of the sinusoid moving to the right. The sinusoid along the time axis is the signal that would be observed at 0 on the x-axis. In (c) the velocity is increased and in (d) the spatial frequency is increased. In (e) both spatial frequency and velocity are increased.

For the leftward sensitive correlator in Fig. 2.5 (c), the 'effective lag' τ will be:

$$\tau = \Delta t + \frac{\Delta x}{v} \tag{2.24}$$

and the output after substitution will be:

$$R_L(\tau) = \frac{1}{2}\left(\cos(2\pi f_t \cdot \Delta t)\cos(2\pi f_x \Delta x) - \sin(2\pi f_t \cdot \Delta t)\sin(2\pi f_x \Delta x)\right). \tag{2.25}$$

A complete Reichardt EMD computes the difference between equations (2.24) and (2.25) resulting in a function of spatial and temporal frequency:

$$R(f_x, f_t) = \sin(2\pi f_t \cdot \Delta t)\sin(2\pi f_x \Delta x). \tag{2.26}$$

The plot in Fig. 2.8 (a) shows how the output of an EMD will vary with f_x. For simplicity, we set Δx = 1 'spatial unit' and express f_x in units of cycles/Δx. Likewise Fig. 2.8 (b) shows how the output will vary with f_t which is expressed in units of cycles/Δt. The response peaks for a spatial frequency of $0.25\Delta x$ and a temporal frequency $0.25\Delta t$. The plot in Fig. 2.8 (c) shows why the EMD is referred to as a spatio-temporal filter. The EMD will respond most strongly to a particular range of spatial and temporal frequencies depending on the values of Δx and Δt. As such it is a bandpass filter. The ranges in which the response is valid are $0 < f_x < 0.5$ and $0 < f_t < 0.5$. The response actually becomes negative for $0.5 < f_x < 1$, incorrectly indicating leftward motion since the actual motion is rightward. Spatial prefiltering can be used to eliminate this incorrect portion of the response. This is discussed in more detail in section 3.1 of chapter 3.

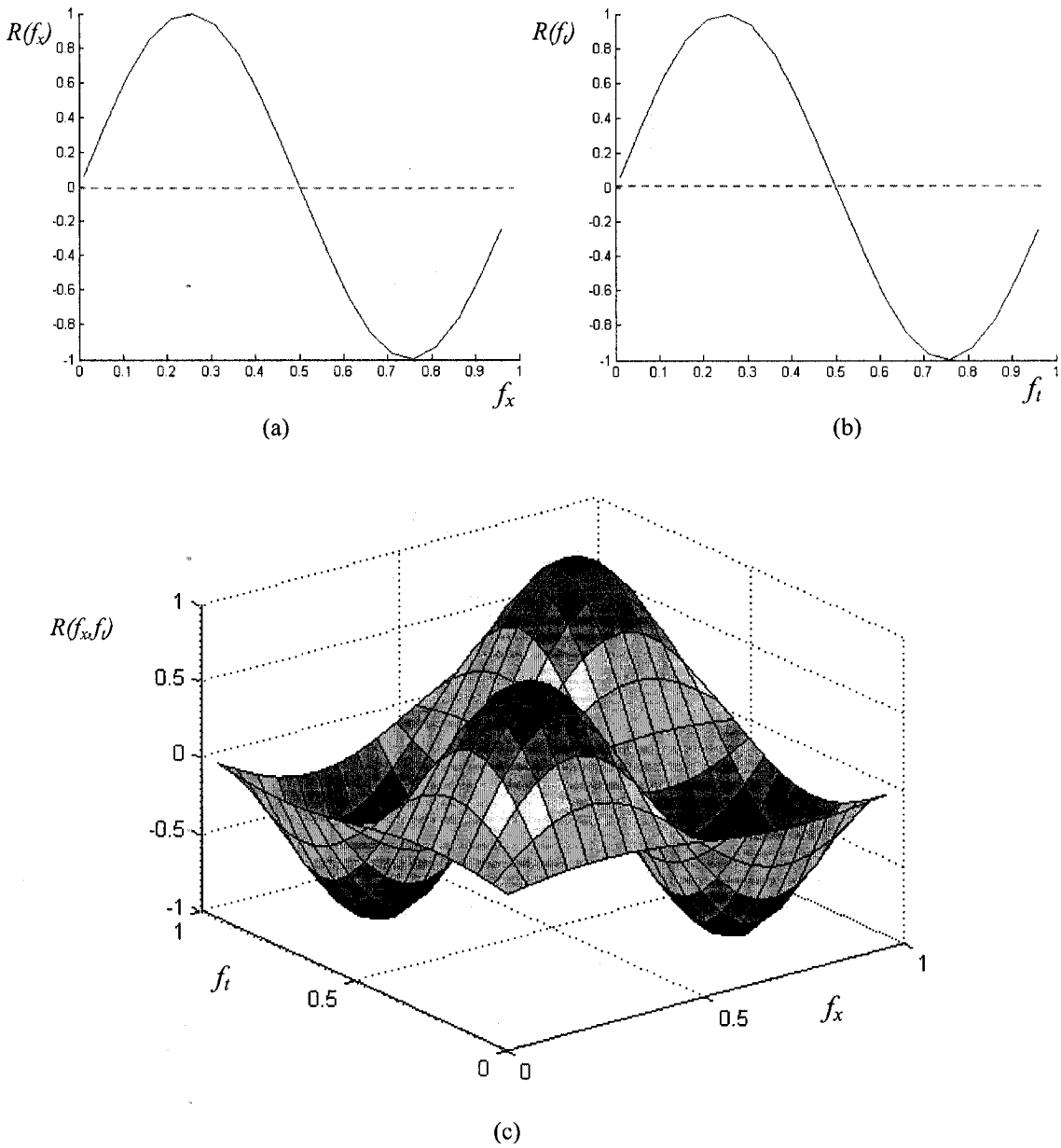


Fig. 2.8. (a) Plot of EMD response as a function of spatial frequency and (b) as a function of temporal frequency. (c) The complete response characteristic as a function of spatial and temporal frequency. For all plots, $\Delta x = 1$ spatial unit and f_x is expressed in units of cycles/Δx. Likewise, $\Delta t = 1$ time unit and f_t is expressed in units of cycles/Δt.

The above analysis was based on an ideal delay, Δt, which can be achieved in a discrete-time system. In a continuous-time system, Δt can be achieved by using the inherent phase delay of a first-order lowpass filter [30]:

$$D(f_t) = A(f_t)e^{-j\Theta(f_t)} \tag{2.27}$$

in which $A(f_t)$ and $\Theta(f_t)$ are real-valued functions representing the amplitude and phase respectively. The delayed signals, as shown in Fig. 2.6, are then:

$$p_1'(t) = A(f_t)\sin(2\pi \cdot f_t t - \Theta(f_t)) \tag{2.28}$$

$$p_2'(t) = A(f_t)\sin(2\pi \cdot f_t t - 2\pi \cdot f_x \Delta x - \Theta(f_t)) \tag{2.29}$$

and:

$$R(f_x, f_t) = p_1'(t) \cdot p_2(t) - p_2'(t) \cdot p_1(t) \tag{2.30}$$

$$= A(f_t)\sin[\Theta(f_t)]\sin(2\pi f_x \Delta x) \tag{2.31}$$

If the frequency response of the filter is:

$$D(f_t) = A(f_t)e^{-j\Theta(f_t)} = \frac{1}{\sqrt{1+(2\pi \cdot f_t \Delta t)^2}} e^{-j\arctan(2\pi \cdot f_t \Delta t)}, \tag{2.32}$$

where again $A(f_t)$ denotes the magnitude response and $\Theta(f_t)$ denotes the phase, then:

$$\sin[\Theta(f_t)] = \frac{2\pi f_t \Delta t}{\sqrt{1+(2\pi \cdot f_t \Delta t)^2}}, \tag{2.33}$$

and the response is given by:

$$R(f_x, f_t) = \frac{1}{2\pi\Delta t} \frac{f_t}{f_t^2 + 1/(2\pi\Delta t)^2} \sin(2\pi \cdot f_x \cdot \Delta x) \tag{2.34}$$

Fig. 2.9 shows the plot of this response as a function of spatial frequency in (a), as a function of temporal frequency in (b), and the complete response in (c). This is the typical response characteristic that appears in the literature. A complete derivation of the

response is provided in Appendix A. Equation (2.34) will be used in chapter 3 to show how the filter can be tuned to different ranges of velocity and spatio-temporal frequency by varying the parameters Δx and Δt.

A single EMD or an array of identical EMDs will be limited to detection over a very specific range of spatial and temporal frequency patterns and therefore to a specific range of velocities. This is common to all spatio-temporal frequency-based techniques. However, their ability to individually detect the spatio-temporal frequency content of local image regions makes them well suited to target tracking applications. What is needed is a study to find a way to design an efficient architecture for a motion detection system, based on these filters, that can perform detection over a wide range of velocities and for both coarse and fine spatial details.

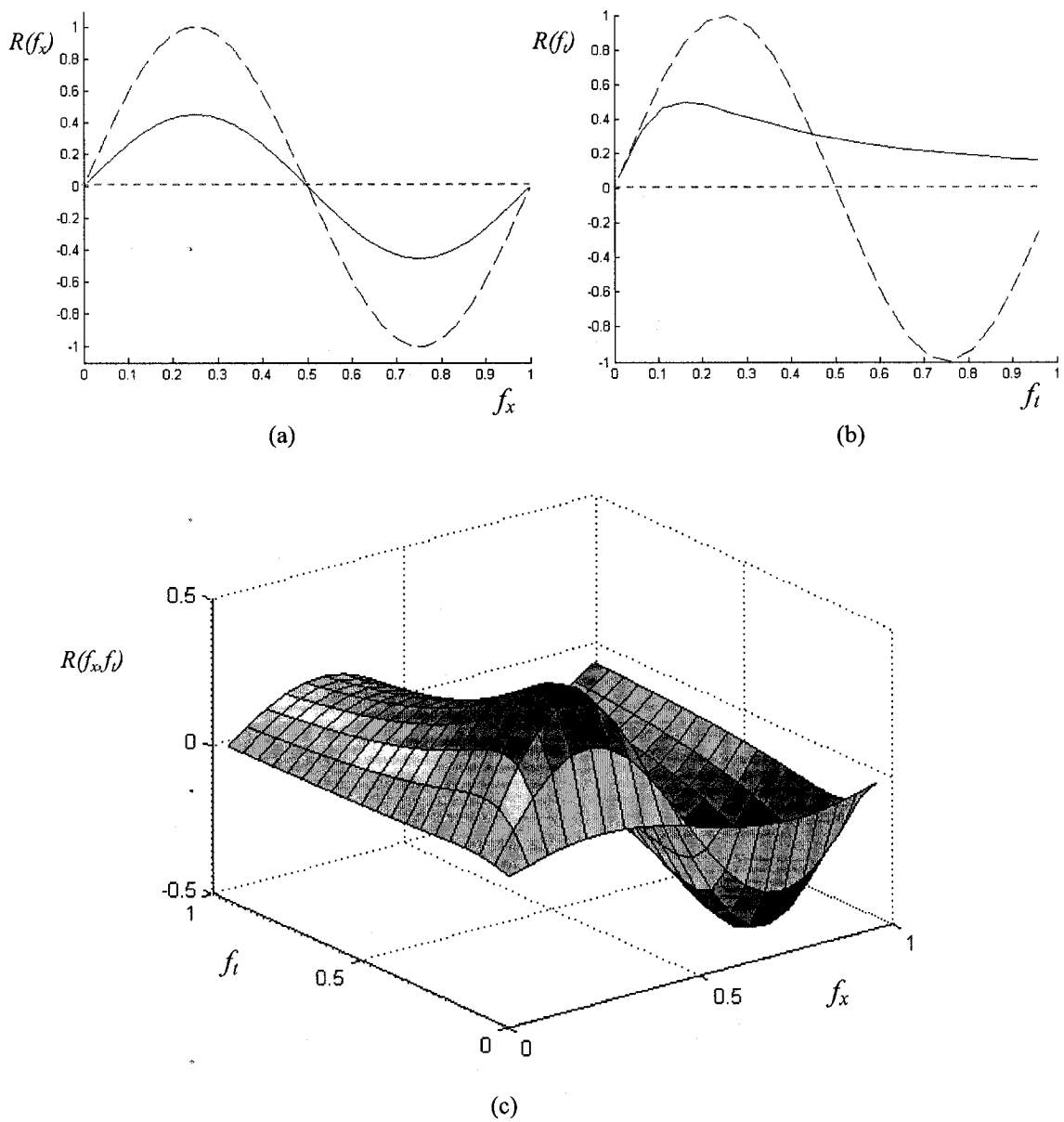


Fig. 2.9. (a) Plot of EMD response as a function of spatial frequency and (b) as a function of temporal frequency. The dashed lines indicate the response curves from Fig. 2.8. (c) The complete response characteristic as function of spatial and temporal frequency. All plots were produced using a simulated continuous-time low-pass filter to create a delay, $\Delta t = 1$ time unit. Also, f_x is expressed in units of cycles/Δx and f_t is expressed in units of cycles/Δt.

2.2 Related Work on Motion Detection for Tracking

None of the visual motion detection techniques described in this chapter are perfectly suited to target tracking. All of them have some deficiencies and researchers have tried to work around these deficiencies in two different ways. Sometimes the basic computation performed by the motion detector itself is modified. These computational level modifications are usually an attempt to devise a new motion detection mechanism that is still based on one of the basic principles such as matching or correlation. Alternatively, sometimes the basic motion detector is used unmodified, but it is incorporated into a new architecture that leads to more effective motion detection and tracking. In this section both of these approaches are considered in terms of work that has been previously reported by other researchers. The emphasis is on work related to correlation-based motion detection, as performed by the Reichardt EMD, since this form of motion detection is used for the work of this thesis.

2.2.1 Related Work at the Computational Level

A basic problem of motion detectors that are based on spatio-temporal filtering is that they do not directly measure the speed of a target, rather they compute the spatio-temporal frequency content of a local region. This makes it difficult to distinguish changes in a target's velocity from changes in its spatial frequency pattern. To solve this problem, a detection mechanism reported by Delbruck [31] extends the usual pairwise correlation model to a sequence of several photoreceptors that are coupled into a delay line. The basic scheme is shown in Fig. 2.10. The unidirectional delay line composed of

low-pass filters serves as a tuned filter for a particular target velocity. If the motion of a passing edge matches the tuning of the line then the amplitude at each output, labeled M_0, M_1,..., is reinforced and increases along the delay line; otherwise the output decays. In this way the amplitude of the output serves as a measure of the velocity of the target. The use of a delay line to aggregate a signal across space and time improves the ability to estimate velocity, however, differently tuned delay lines are still needed to estimate target velocity over a wide range.

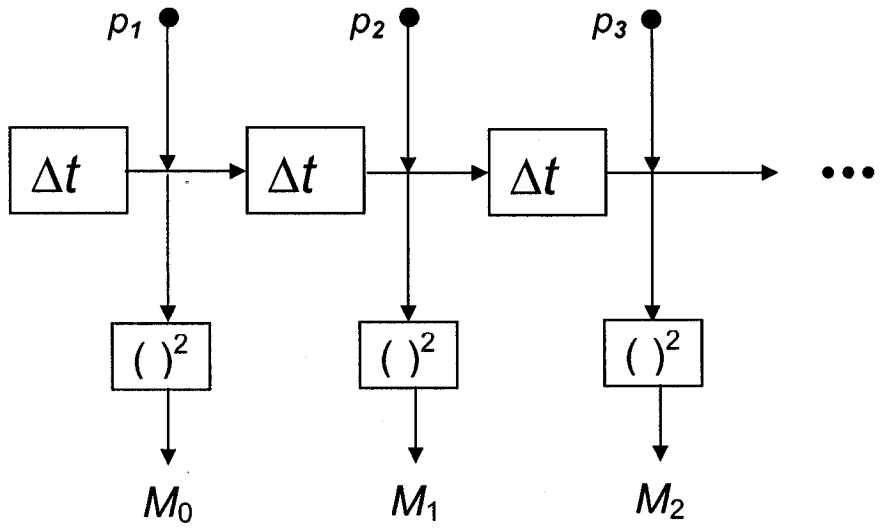


Fig. 2.10. Delbruck's correlation-based motion detector [31]. A delay line is used to measure the velocity of a target passing over the photoreceptors. The $( )^2$ symbol represents a squaring operation.

A different scheme that resembles the Reichardt EMD in structure, but operates in a different way, is the facilitate-and-sample velocity sensor [32] shown in Fig. 2.11 (a). The first stage consists of edge detectors which pass their signals on to pulse-shaping circuits which then produce both a slowly decaying pulse, *P1*, and a narrow sampling

pulse, *P2*. If an edge is moving to the right then its decaying pulse will be sampled by the narrow pulse at a voltage that encodes the velocity of the edge, as shown in Fig. 2.11 (b). Example systems based on the scheme still rely on mainly qualitative properties of the computed optical flow despite the use of more complex computations. This is due mainly to the vulnerability of the scheme to noise and mismatch.

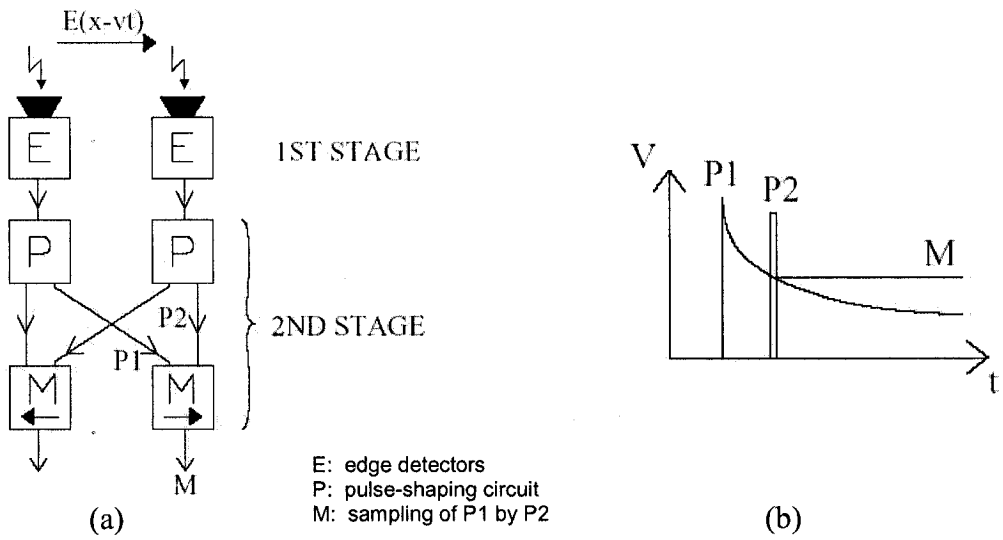
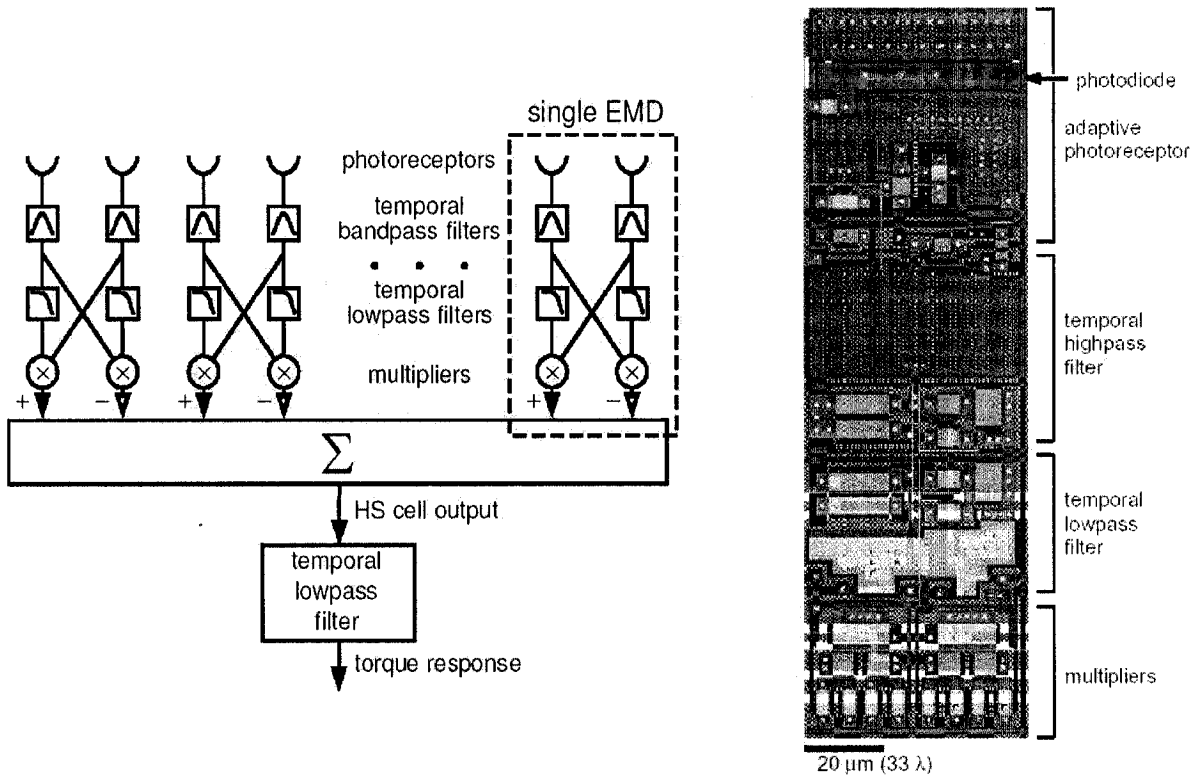


Fig. 2.11. The facilitate-and-sample velocity sensor [32]. (a) Block diagram. (b) Voltage signals. The slowly decaying facilitation pulse is P1, the sampling pulse is P2 and M is the output voltage which encodes the velocity of the edge.

2.2.2 Related Work at the Architectural Level

Researchers have attempted to combine individual motion detectors into architectures that lead to some performance improvement. Simply implementing a complete 1-D array of basic Reichardt EMDs has been shown to be feasible in a standard VLSI process by Harrison and Koch [33] and others [34]. The Harrison system is shown in Fig. 2.12 (a). The layout of a single EMD is shown in Fig. 2.12 (b). The outputs of the array of EMDs

are linearly summed and then passed through a temporal low-pass filter. The spatial summation is a simple way to produce a motor control signal from the local measurements of the individual EMDs. The arrangement was used in a complete sensory-motor control system to simulate the ability of a fly to estimate self-motion and stabilize its body during flight.



(a) (b)

Fig. 2.12. Harrison and Koch's array of EMDs [33]. (a) Block diagram of the sensory-motor system that uses the array to produce a motor signal. (b) Layout of a single EMD.

Perhaps the closest system to the one described in this thesis is that of Higgins and Shams [35]. They have proposed to use a number of parallel motion detection arrays to detect both fine and coarse motion. The architecture is shown in Fig. 2.13. An image is

focused onto a sensor array which outputs changing contrast information to four parallel arrays of EMDs. The arrays are distinguished by differences in EMD orientation, speed tuning, spatial frequency tuning or other properties. The outputs of these EMDs is multiplied by weighted matrices called innervation matrices (IMs) and then spatially integrated to form the system's output. One drawback of the system is that it requires some non-trivial coordination between the separate arrays of EMDs. It also requires significant computational resources, i.e. circuits and memory, for its tuning by using four complete and separate arrays.

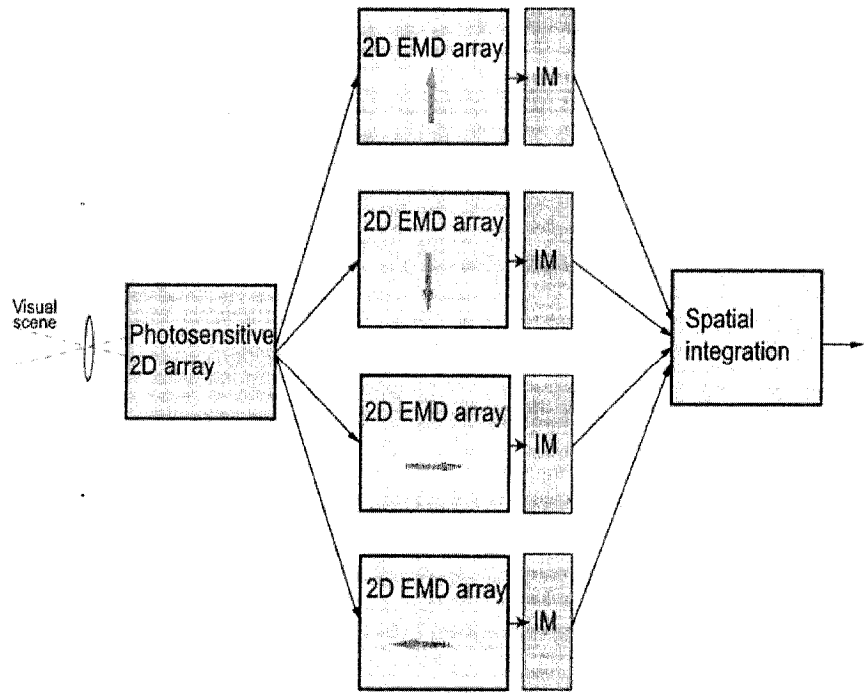


Fig. 2.13. Architecture for a set of parallel motion detection arrays [35].

2.3 Bio-Inspired Strategies for Visual Tracking

Despite advances at both the computational and architectural levels, visual tracking remains a difficult problem. The challenging aspects, as described in chapter 1, include the need for both precision and range in the measurements of target position and velocity. Another challenge is to find an efficient way to map these measurements to a control signal for tracking. It appears that biological vision systems have met these challenges, given the fact that humans and many other animals are able to easily perform target tracking. This has led researchers to pursue solutions that draw inspiration from biological vision systems.

Researchers and engineers have been taking inspiration from nature for thousands of years. In the last 50 years many attempts have been made to imitate, with electronics and computers, the structures and strategies found in biological systems. In the 1950's and 1960's the field of cybernetics developed many interesting robotic systems based on negative feedback [36]. In the late 1980's the field of neuromorphic engineering began to use analog VLSI to emulate the computational strategies found in the nervous system [37]. This section briefly describes some bio-inspired strategies that are relevant to active visual target tracking.

2.3.1 Active Vision

Active vision systems are those in which the parameters of an image sensor or camera, such as the orientation or the focal length, can be dynamically changed to meet the needs of a task [38-40]. Rather than trying to extract a maximum amount of information from any one image, the camera is actively controlled to sense different

aspects of the scene. Bajcsy [39] observed that many traditional vision problems could be solved with less computation by controlling the motion of the camera. It was also proposed that a vision system should be designed to actively serve a purpose rather than being a general image understanding system. The overall strategy is based on ideas found in the oculomotor control system of biological vision systems.

2.3.2 Space-Variant Imaging

Researchers have been inspired by the observation that the eyes of humans and many animals are space-variant sensors [41]. The density of photoreceptors is highest in the center and decreases with distance from the center of the sensor as shown in Fig. 2.14 (a) and (b). The central region can be used for tasks which require high resolution, while the lower resolution periphery can provide information about the background, reducing the number of pixels required for computation and transmission. For real-time target tracking, the most important property of a space-variant sampling structure is the reduction of visual information that must be processed. This reduction is achieved without a reduction in the field-of-view or a reduction in the resolution at the center of the sensor. A review of space-variant image processing can be found in [42].

The most commonly used sampling geometry is the log-polar structure which has been explored with both custom sensors and software simulations for over 20 years [43][44]. It is a model of the mapping of the photoreceptors from the human retina to the primary visual cortex. A constant number of photoreceptors are arranged over concentric rings producing a linear increase in the spacing with distance from the center of the

sensor. There have also been examples of using the mapping for active target tracking [45][46].

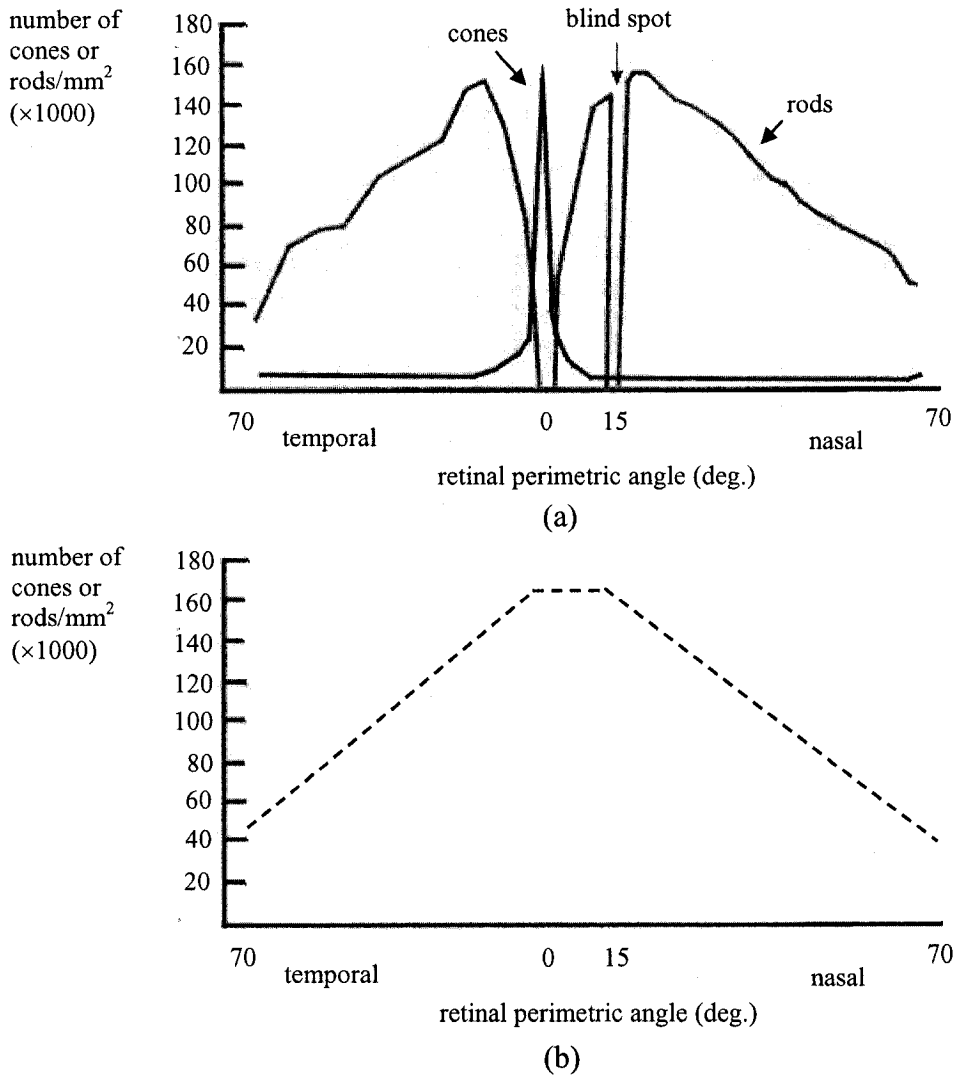


Fig. 2.14. (a) The density of cones and rods across the human retina [41] and (b) the general form of the density of both types of photoreceptors combined.

Other sampling structures may offer advantages for particular applications. For example, various animals have eyes that are adapted to their ecological niche, such as birds that have two foveal regions to allow for hunting and flying at the same time [47].

Following this idea, there has been a foveated sensor with a sampling structure that was designed to prioritize obstacles in a robot's path [48]. The sensor structure makes it easier to make decisions about steering away from obstacles because of the way it filters the visual information to emphasize obstacles which are close or those which are distant but very large.

An interesting example of a space-variant sensor consists of a central region of a single high resolution surrounded by a region of a single low resolution [49]. The low resolution region performs target detection based on image intensity while the central region performs target tracking using an EMD-type motion detection circuit. What is still needed is a systematic method to design space-variant sensing to match different applications.

Once motion information has been extracted from an image signal, the task that remains is to produce an appropriate control signal to drive the vision system in such a way as to track the target. This sensory-motor coordination can be difficult to design, especially in a system that uses non-uniform, space-variant imaging. Both sensing and motor functions are often characterized by nonlinearities that make analysis difficult. What is needed is a tool to help visualize how the sensing and motor functions interact. Research in neurophysiology has shown that sensory-motor regions of the vertebrate brain, such as the superior colliculus, receive inputs from the eye in a retinotopic map that is aligned with a 'motor map' of neurons that project to brainstem areas which then control eye and head movement [50]. This alignment of sensory and motor maps serves as inspiration for a method of analysis and design presented in this thesis. It combines a map of the motion detection capabilities of the system with a phase-plane plot of the

tracking dynamics into a single sensory-motor map that can be used to design the tracking control signal.

2.4 Summary

A variety of techniques have been used for visual motion detection for target tracking. These include techniques based on matching, gradient-based techniques and spatio-temporal frequency-based techniques. The techniques based on matching can be used to detect motion over a wide range of velocities however they exhibit a tradeoff between accuracy and computational efficiency. Gradient-based techniques are conceptually simple but problems with accuracy and reliability limit their applicability to target tracking. Spatio-temporal frequency-based techniques compute the spatio-temporal frequency content of local image regions. They use filters which are selectively activated by a range of spatio-temporal frequency patterns produced by the motion of a target. A simple and commonly used example is the Reichardt EMD. These techniques are very useful for target tracking but an individual detector of this type is limited to a particular range of spatial and temporal frequency and therefore to a particular range of target velocity. Researchers have attempted to work around this by modifying the basic detector at the computational level or by incorporating the basic detector into an architecture that supports a wider range of motion detection. While these ideas have led to improvements, further investigation is needed to find a way to design an efficient architecture for a motion detection system based on these filters. Many promising approaches draw inspiration from what is currently known about biological vision systems.

3

Spatio-Temporal Frequency Tuning of Reichardt EMDs for Space-Variant Motion Detection Arrays

The goal of this thesis is to achieve active visual target tracking with precision and also over a wide range of speeds in a computationally efficient manner. Chapter 2 described how spatio-temporal filtering can be used to detect a specific range of spatial and temporal frequency patterns. What is needed is an efficient way to incorporate individual spatio-temporal filters into an array and to tune the individual filters to different frequency and velocity ranges. This chapter provides an analysis of an example of a spatio-temporal filter known as the Reichardt elementary motion detector or EMD. Based on this analysis we propose that space-variant imaging be used to build an array of EMDs that are tuned to different velocity ranges. We also propose a graphical technique, called the EMD map, to visualize the different velocity ranges across the array.

3.1 Tuning a Reichardt EMD

A diagram of a simple Reichardt EMD, as shown previously in Fig. 2.6, is shown again in Fig. 3.1. It has two input signals, p_1 and p_2, acquired from two points in the image plane separated by a distance Δx. The inputs are assumed to come from photodetectors and represent the image irradiance at two adjacent points in the image

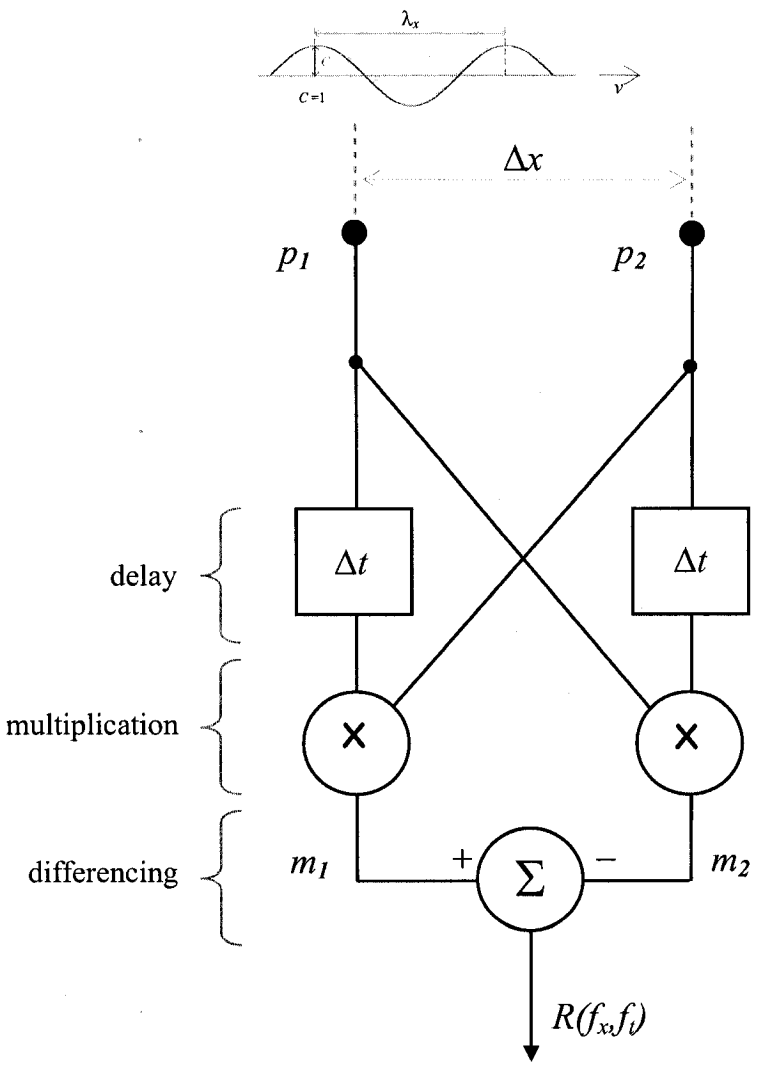


Fig. 3.1. Block diagram of a simplified Reichardt EMD showing the two adjustable parameters: Δx, the distance between inputs and Δt the delay. The input sinusoid, at top, could represent a target passing over the EMD.

plane. The input p_1 is delayed by Δt and is multiplied with the undelayed input p_2, producing a response that is sensitive to rightward motion. The same delay-and-multiply operation is performed for the opposite direction. The outputs of the two sides are subtracted to reduce the response to static patterns and full-field flicker. For the configuration in Fig. 3.1 the output will be positive for rightward motion and negative for leftward motion. A detailed discussion of the operation of the EMD is provided in section 2.1.3 in chapter 2.

The two parameters that can be adjusted in the simplified EMD are the distance, Δx, between the two points where the inputs are acquired, and the delay, Δt. To see how they affect the response of the EMD to motion it helps to assume that the input to the EMD is a zero-mean sinusoid that has unit amplitude and a spatial frequency of $f_x = 1/\lambda_x$ as shown at the top of Fig. 3.1. This assumes that a phototransduction stage is able to produce such a signal from the image of an actual target and allows us to ignore for now the effect of contrast on the response. If the sinusoid is shifted to the right with velocity v then a temporal frequency of $f_t = f_x v$ will be induced at each input. The shifting sinusoid could represent a target passing over the EMD. If the delay is modeled as the inherent phase delay of a first-order low pass filter with time constant Δt, then the response can be expressed as [30]:

$$R(f_x, f_t) = \frac{C^2}{2\pi \cdot \Delta t} \cdot \left[\frac{f_t}{f_t^2 + 1/(2\pi \cdot \Delta t)^2} \right] \cdot [\sin(2\pi \cdot f_x \cdot \Delta x)] \qquad (3.1)$$

in which C, the 'contrast' or amplitude of the input sinusoid is equal to one. Physical

realizations of the EMD are usually limited to having some minimum value for Δx and Δt. For the following analysis we define the smallest time delay as a unit δt and the smallest distance as a unit δx. A plot of the response for $\Delta x = 1\delta x$ and $\Delta t = 1\delta t$ was shown in Fig. 2.9 and is shown again in Fig. 3.2. Spatial frequency, f_x, is expressed in units of cycles/δx and temporal frequency in units of cycles/δt.

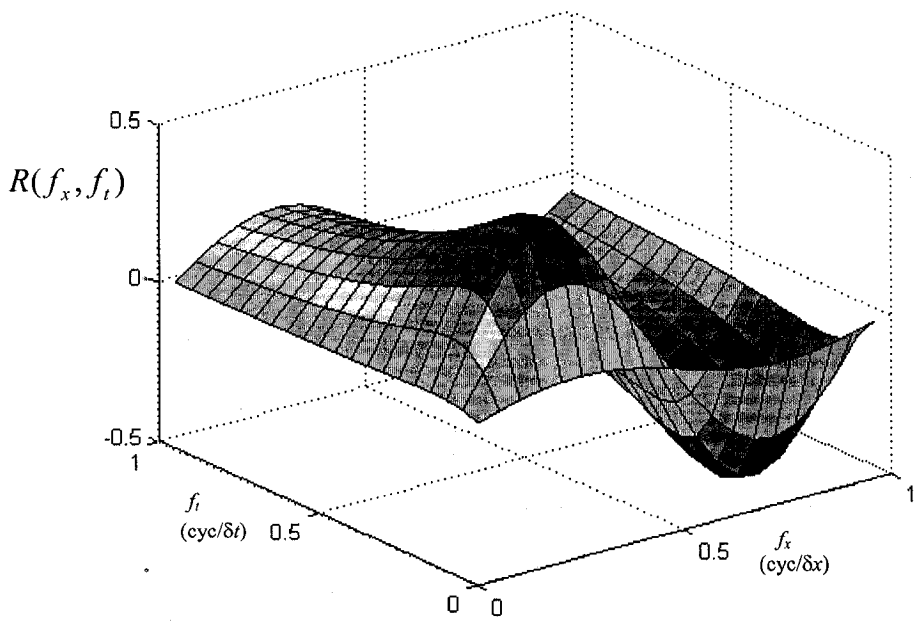


Fig. 3.2. Plot of EMD response to a unit-amplitude sinusoid as a function of spatial and temporal frequency. The response is for the EMD of Fig. 3.1 with $\Delta x = 1\delta x$ and $\Delta t = 1\delta t$.

As seen in Fig. 3.2, the response peaks for particular values of spatial and temporal frequency. For a zero-mean sinusoidal pattern the response has a maximum value of 0.5 which is reached when the spatial frequency of the pattern is:

$$f_{x,peak} = \frac{1}{4 \cdot \Delta x} \tag{3.2}$$

and when the temporal frequency is:

$$f_{t,peak} = \frac{1}{2\pi \cdot \Delta t}. \tag{3.3}$$

At higher spatial frequencies, a form of aliasing occurs and the response becomes negative for rightward motion. This can be eliminated by spatially prefiltering the inputs. If the image plane is not point sampled but each sample extends to fill the space between samples then the photoreceptor itself performs this prefiltering. Assuming that the photoreceptor averages the light intensity falling on its surface, then for one dimension in the spatial domain the filter is simply a rectangular function as shown in Fig. 3.3 (a). In the frequency domain it is described with a sinc function:

$$S(f_x) = \frac{\sin(\pi \cdot \Delta x \cdot f_x)}{\pi \cdot \Delta x \cdot f_x} \tag{3.4}$$

in which Δx is both the separation and the physical extent of the photoreceptors, and f_x is the spatial frequency of the input sinusoid. The first zero occurs at $f_x = \Delta x = 1$. The magnitude response for $0 < f_x < 1$ is shown in Fig. 3.3 (b). Spatial frequencies above the Nyquist frequency of $0.5\Delta x$, which is twice the spatial frequency for the peak response, are strongly attenuated. Fig. 3.3 (c) shows the EMD response without prefiltering while Fig. 3.3 (d) shows the response with prefiltering. The spatial frequencies above $0.5\Delta x$ are noticeably attenuated. Now a threshold can be used to eliminate the response due to aliasing. Fig. 3.3 (e) shows a contour plot of the unfiltered response of (c) after thresholding while Fig. 3.3 (f) shows the filtered response of (d) after thresholding. Only the positive portion of the response is retained.

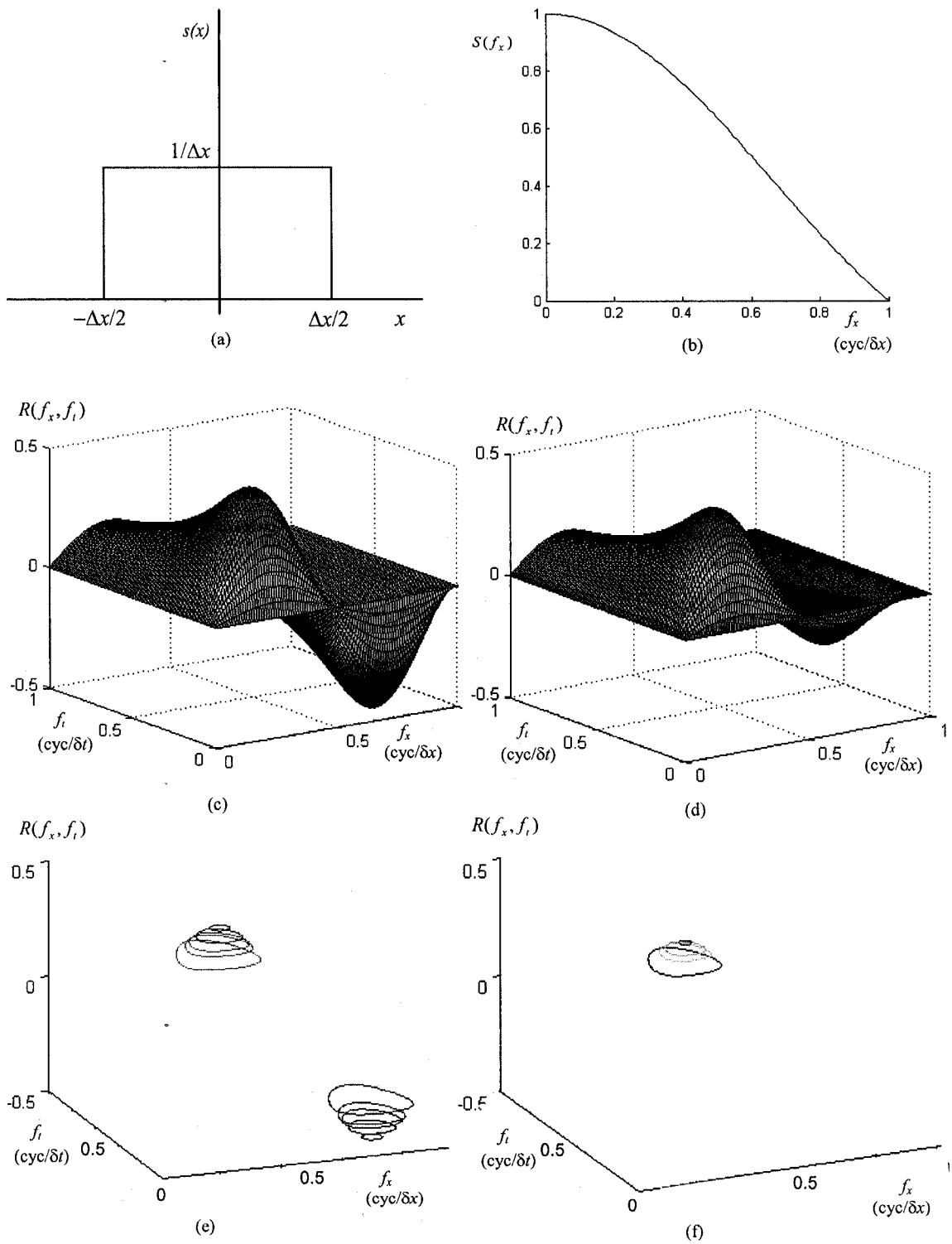


Fig. 3.3. Effect of spatial prefiltering on EMD response. (a) A one-dimensional spatial prefilter in the spatial domain and (b) in the spatial frequency domain. (c) and (e) show the response and the thresholded response without prefiltering. (d) and (f) show contour plots of the response and the thresholded response with prefiltering. The negative response due to aliasing has been removed.

Now we will consider how the two parameters Δx and Δt affect the frequency response and the range of detectable target velocities for an EMD. To see the effect of varying Δt, Fig. 3.4 shows contour plots of the responses for three EMDs with different values of Δt but the same Δx. The y-axis represents f_t in units of cycles/δt and the x-axis represents f_x in units of cycles/δx. Each set of concentric response contours represents the response of a single EMD with a particular Δt. The three EMDs have delays of $\Delta t = 1\delta t$, $2\delta t$ and $4\delta t$. For each EMD three contours and a peak are shown. Each contour represents a combination of spatial and temporal frequency of the input signal that produces the same average response. The peak average response at the center is 0.5 and the outermost contour represents a response of 0.4. This value has been chosen as a threshold to indicate that a detection is made only if the response is above that value. The choice of the threshold defines the passband of the filter. The slope of the line from the origin to any point on a contour represents the velocity, in units of $\delta x/\delta t$, of the input sinusoid corresponding to that point.

If an array consists of EMDs with the three values of Δt shown in Fig. 3.4 then the range of detectable velocities for the array would be given by the difference between the slopes of the lines that pass just above and below the outer contours of the three responses as shown on the plot. The total velocity range achievable with the three different values of Δt is greater than for a single fixed Δt but the spatial frequency range or bandwidth, shown on the x-axis, is the same as for any single Δt. An array of EMDs with different values of Δt can be arranged to cover a large range of velocities, however, the detectable spatial frequency range will not widen. Also, in practice it may be difficult to vary Δt in an efficient and reliable way. In a standard imaging system Δt would be

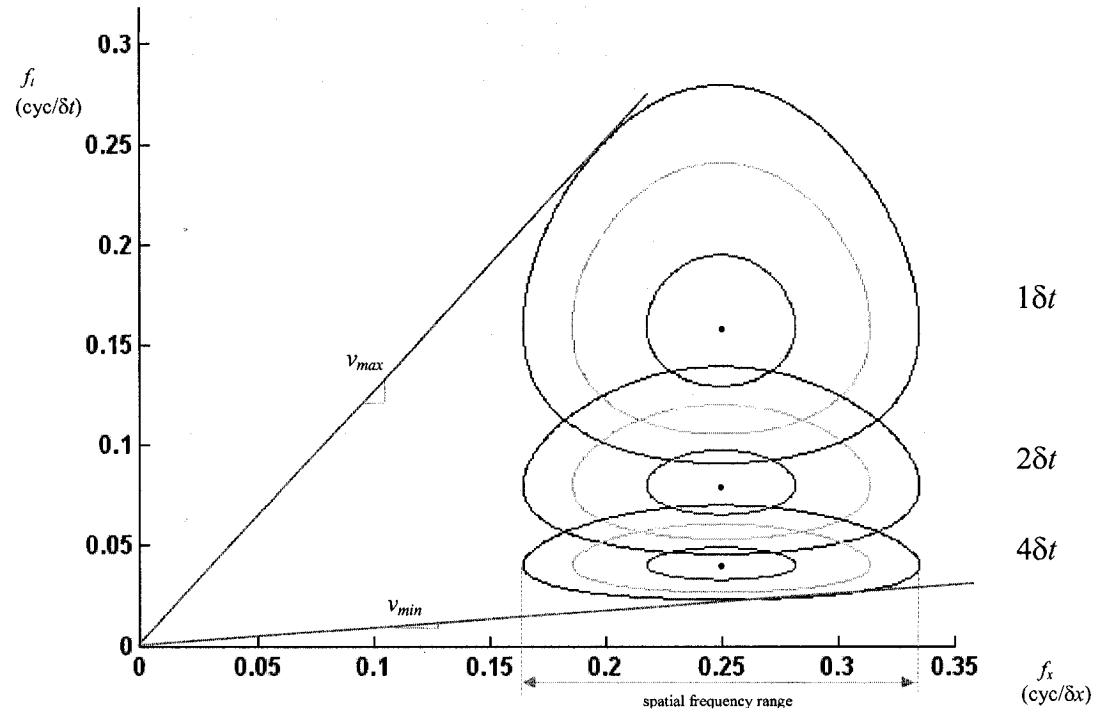


Fig. 3.4. Contour plot of the responses of three EMDs with delays of $\Delta t = 1\delta t$, $2\delta t$ and $4\delta t$. The range of detectable velocities for the three EMDs together is the difference in the slopes of the v_{max} line and the v_{min} line. The range of detectable spatial frequencies, shown on the x-axis, is the same as for a single EMD.

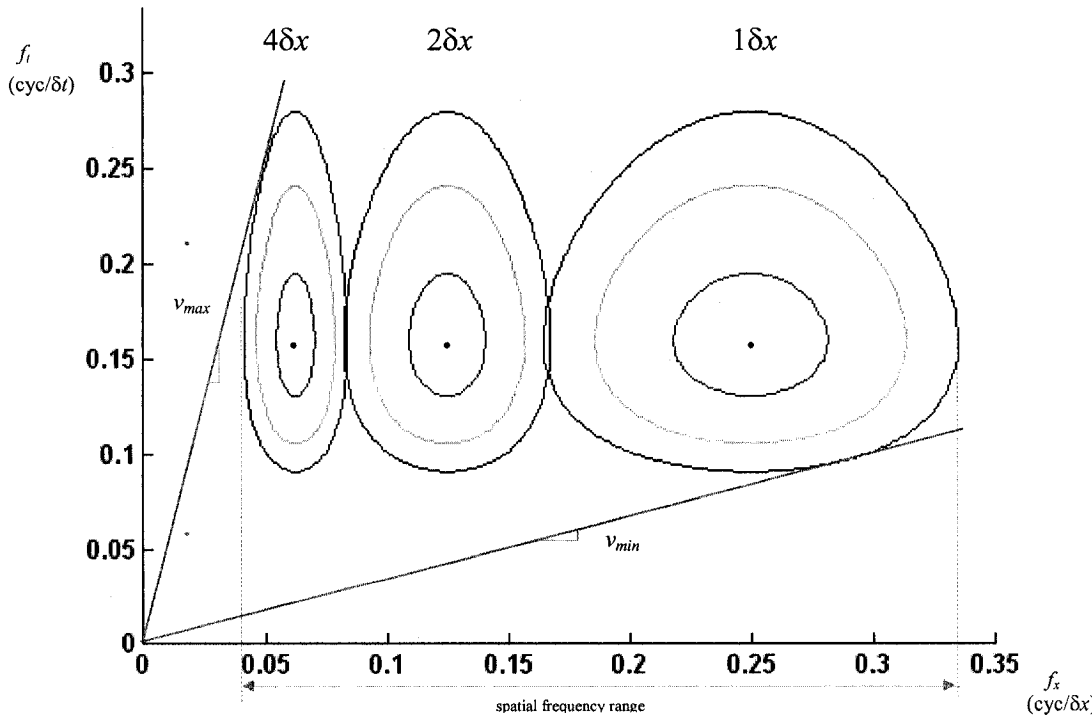


Fig. 3.5. Contour plot of the responses of three EMDs with input distances of $\Delta x = 1\delta x$, $2\delta x$ and $4\delta x$. The range of detectable velocities for the three EMDs together is the difference in the slopes of the v_{max} line and the v_{min} line while the range of detectable spatial frequencies, shown on the x-axis, is the combined ranges of the three EMDs.

constrained to be an integer multiple of the frame rate. In an integrated smart sensor implementation, the range for Δt would be constrained by the size of the on-chip capacitors that would be needed to implement a low-pass filter.

To see the effect of varying Δx, Fig. 3.5 shows contour plots of the responses for three EMDs with different values of Δx but the same Δt. The EMDs have input distances of $\Delta x = 1\delta x$, $2\delta x$ and $4\delta x$. Each EMD is sensitive to a specific range of velocities and spatial frequencies. An EMD with $\Delta x = 1\delta x$ is sensitive to lower velocities than an EMD of $2\delta x$. It is also sensitive to higher spatial frequencies. If an array contains EMDs with these three values of Δx then both the velocity range and the spatial frequency range will be expanded. An array of EMDs with different values of Δx can be designed to cover a wide range of velocities and spatial frequencies. Varying Δx can be easily achieved either by varying photodetector spacing or, in the case of a standard imaging system, by grouping and averaging pixels into pixel aggregates, resulting in a space-variant imaging system.

3.2 Space-Variant Motion Detection Arrays and the EMD Map

An array of EMDs with different values of Δx can be easily achieved with a space-variant imaging system. A diagram of a very simple 1-D space-variant motion detection array is shown in Fig. 3.6 (a). The array is arranged such that the EMDs with the smallest value of Δx are at the center and those with the largest value of Δx are at the periphery. The EMDs have values of Δx that are in the same ratio of 1:2:4 as used in Fig. 3.5. Such a progression from the center to the periphery is similar in principle to that found in the

vision systems of many animals and is often referred to as a foveal array. Many variations are possible and the diagram is meant only to show the idea of a space-variant motion detection array of tuned motion detectors. The ease with which Δx can be varied in an implementation makes many different progressions possible. The diagram also shows that the image plane is not point sampled but that each sample extends to fill the space between samples.

To visualize the overall velocity range of an array like that of Fig. 3.6 (a) we can expand the diagram to show the velocity ranges of the individual EMDs as in Fig. 3.6 (b). The x-axis represents distance along the array in units of δx and the y-axis represents the velocity range in the image plane in units of $\delta x/\delta t$. The velocity can be either positive or negative and so each EMD is represented by a pair of rectangles. The width of each rectangle represents the width in the image plane occupied by the pair of photodetectors for that EMD in units of δx. The width of each rectangle also provides an indication of the spatial frequency bandwidth for that EMD. The height of each rectangle represents the velocity range over which the EMD is able to detect motion. In an active vision system, the array itself is moving to aim at the target so that velocities and positions are always relative to the velocity and position of the array. The y-axis is therefore labeled v_r for relative velocity and the x-axis is labeled x_r for relative position. As a target passes from left to right relative to the array we assume it has a positive relative velocity and if it passes from right to left it has a negative relative velocity. With this EMD map we now see that the periphery of the array covers higher relative velocities and lower spatial frequencies than the center of the array which covers lower relative velocities and higher spatial frequencies. This selectivity can help detect the motion of real-world targets with

complex spatial frequency patterns. The lower spatial frequencies represent the gross overall pattern of the target while the higher spatial frequencies represent the fine details of the target and should only be resolved when the relative velocity between target and sensor is low. These ideas have appeared in our previously published work [51][52].

In comparison, consider the EMD map for a uniform array. A simple 1-D uniform array of EMDs with $\Delta x = 2\delta x$ is shown in Fig. 3.7 (a). Its EMD map is shown in Fig. 3.7 (b). All of the rectangles are of the same size and so the overall velocity range of the array is the same as for any individual EMD. The spatial frequency bandwidth is also the same for all EMDs. In this case there is no selectivity in the motion detection. Whenever a detection is made by any EMD in the array, nothing unique can be said about the speed or spatial frequency of the target. Information of that sort must be computed based on other data such as the location of previous detections. On the other hand, the foveal array provides information unique to each detection without additional computation, simply by virtue of the structure of the array.

The progression of Δx values of 1:2:4 is an example of an exponential progression. Other progressions are possible and their usefulness depends on the application. An EMD map can be drawn for any progression to visualize the ranges of detection. For example, Fig. 3.8 (a) shows an example of a linear progression which provides more overlap of velocity range for adjacent EMDs. Fig. 3.8 (b) shows that a custom array can be designed for any special detection ranges. Fig. 3.8 (c) shows an array with a linear progression that increases from left to right. This could be used for applications in which target motion will always occur in only one direction.

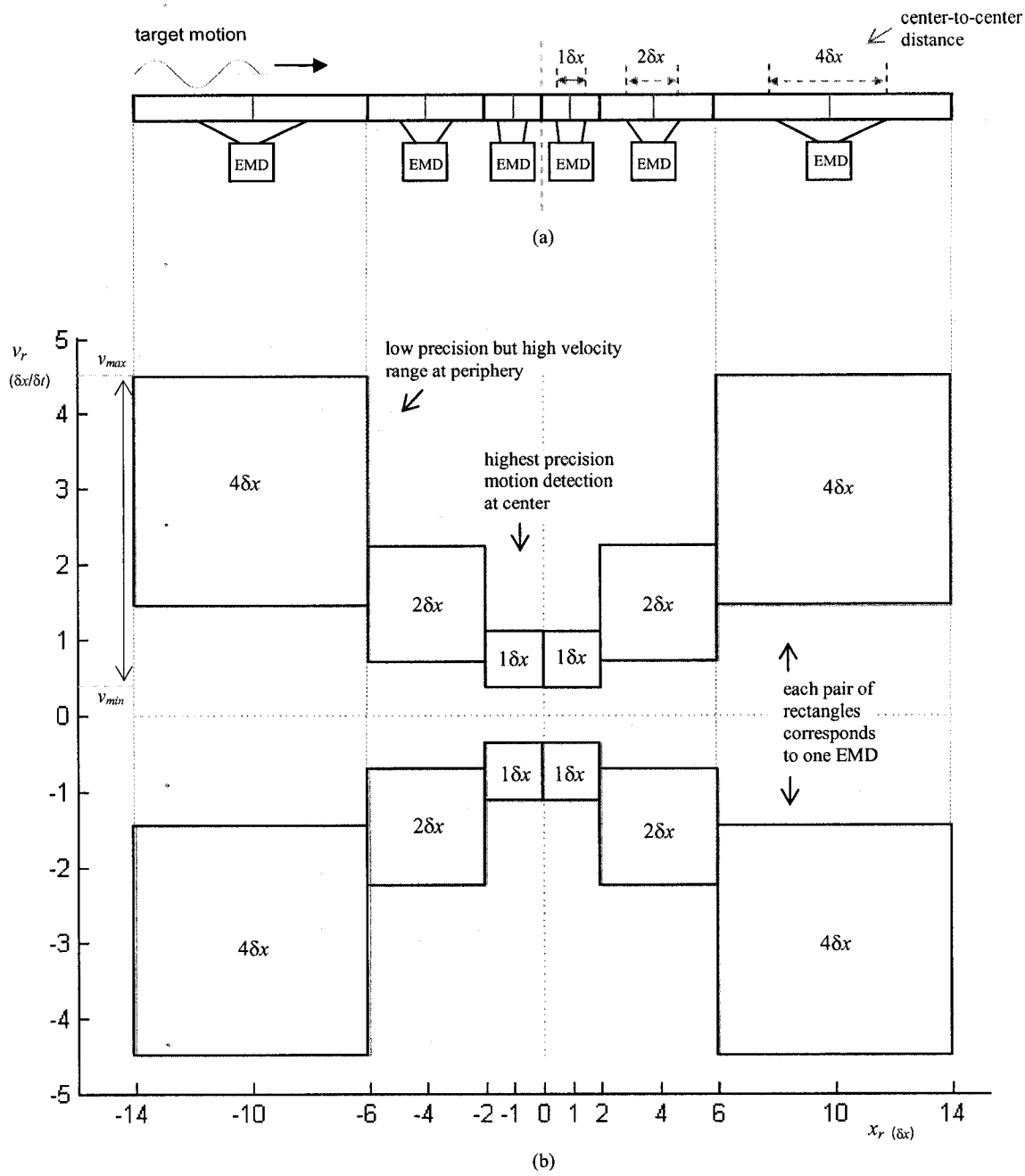


Fig. 3.6. A 1-D motion processing array (a) with an arrangement of EMDs symmetric about the center with δx ratios of 1:2:4 and a plot (b) of the velocity ranges for the same 1-D array. Each rectangle and its mirror image about the x-axis represents a single EMD. The height of each rectangle represents the velocity range over which the EMD is able to detect motion.

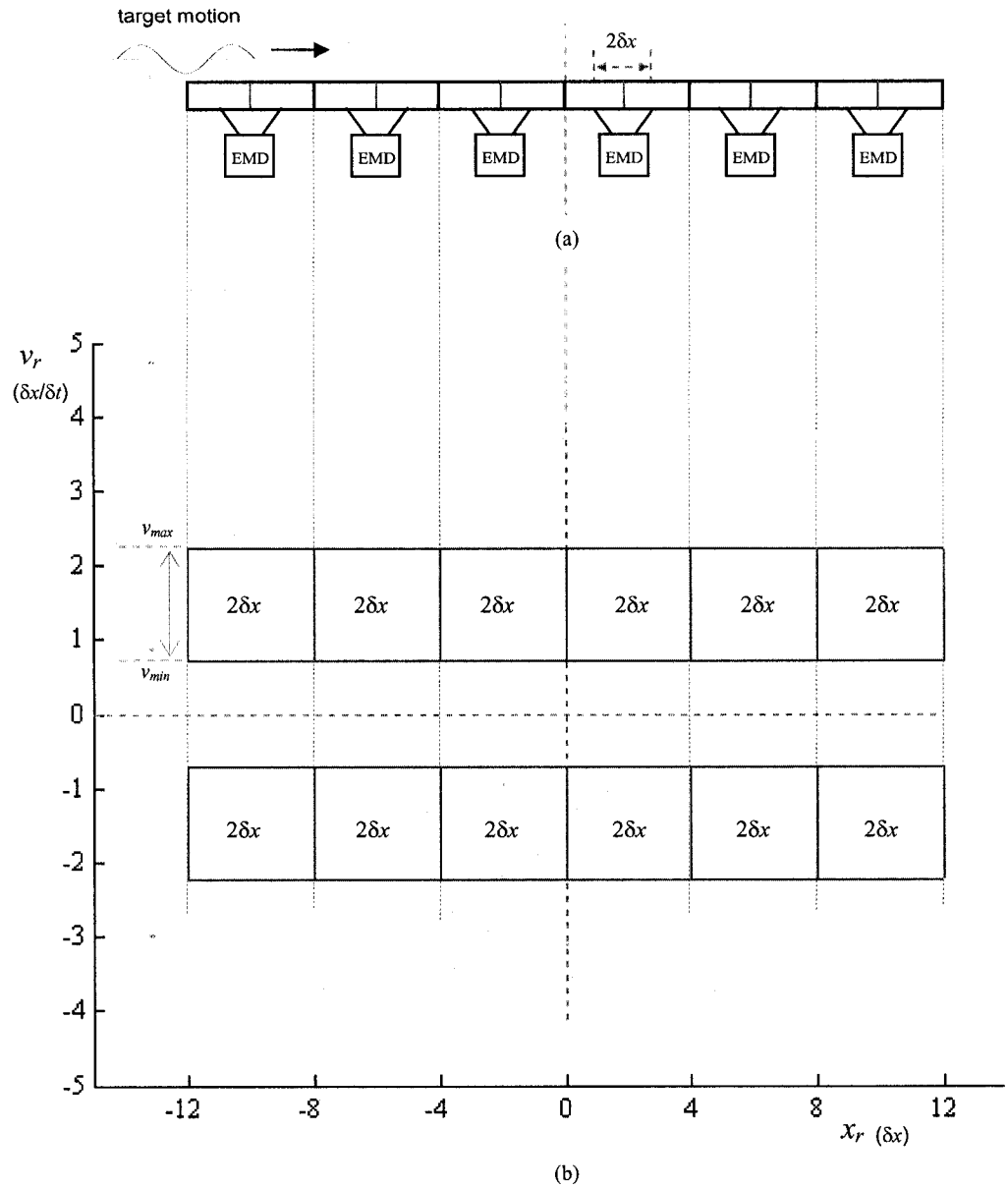


Fig. 3.7. A 1-D motion processing array (a) with a uniform arrangement of EMDs and a plot (b) of the velocity ranges for the same 1-D array. The total velocity range of the array is the same as the range for any individual EMD. The spatial frequency range is also limited to the range for any individual EMD.

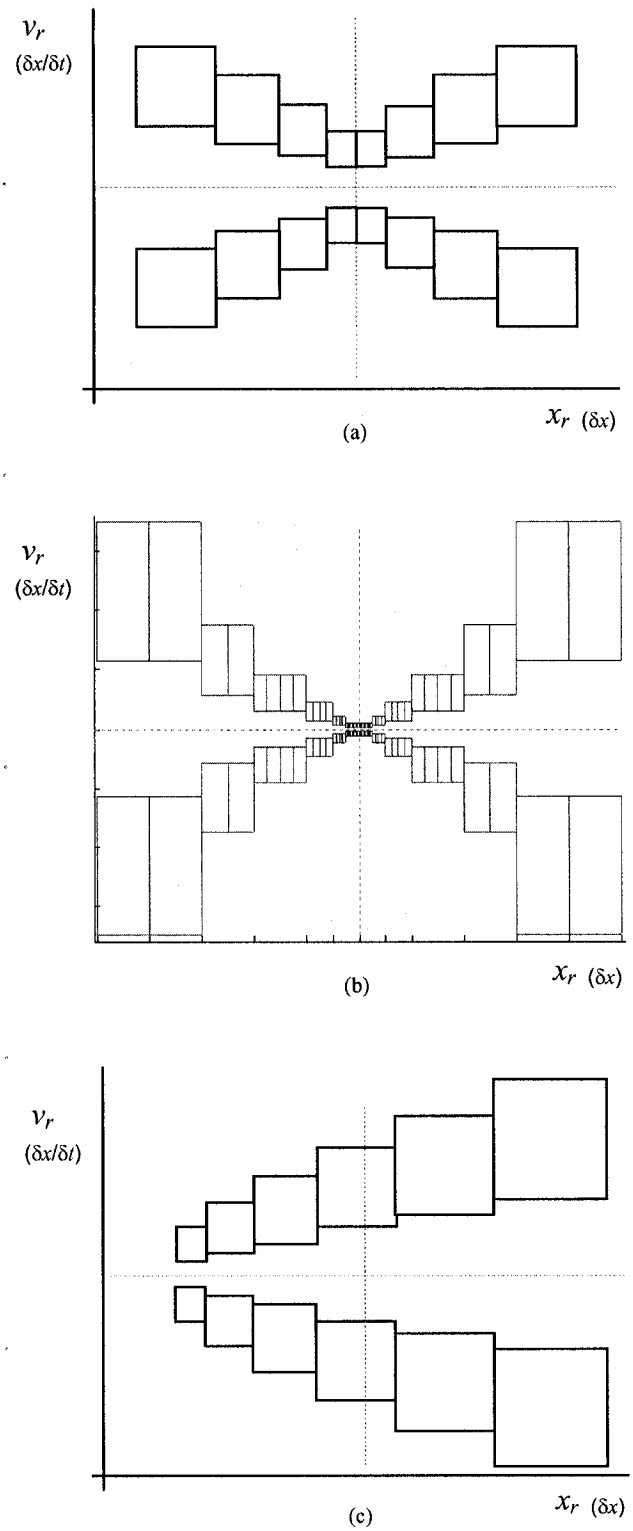


Fig. 3.8. Examples of alternative EMD arrays. (a) A linear array, (b) a custom array and (c) an array that linearly increases from left to right.

For target tracking, a foveal array is expected to be more useful than a uniform array. Clearly the foveal arrangement shown in Fig. 3.6 has an overall velocity range and spatial frequency range that would exceed the ranges for an array of uniform EMDs. The challenge with such a foveal array is to somehow make use of the increased range of detection to control the movement of the array so that as the target becomes more aligned with the center of the array, the relative velocity will be lower. The target can then be detected with more precision by the smaller EMDs. This type of sensory-motor coordination is difficult to achieve without a model of the complete tracking system. The next section describes a model of a tracking system that incorporates a foveal array of EMDs and that enables a method of analysis and design for the type of sensory-motor coordination needed for target tracking. The analysis and the results are all based on 1-D sensing arrangements. Extensions to 2-D are discussed in Chapter 6, however 1-D sensing can find use in various industrial applications, for example in tracking objects on conveyor belts and for tracking projectiles.

3.3 Summary

The goal of this thesis is to achieve a system that is able to detect targets both precisely and over a wide range of speeds in a computationally efficient manner. The analysis of this chapter has shown that a simple EMD can be tuned to detect different velocity and spatial frequency ranges by varying the inter-receptor distance, Δx. This can be achieved for an array of EMDs by varying the size and spacing of the photoreceptors across the array. Such a space-variant array can be designed to provide sensitivity to different velocity and spatial frequency ranges in order to improve a task such as active

visual target tracking. A plot of the detectable ranges across an array, the EMD map, was introduced as a visualization tool for the design of space-variant motion detection. The EMD map can be used to visualize the detection ranges for any arrangement of EMDs. For target tracking, a foveal array is expected to be most useful because for such an array the periphery covers high relative velocities and low spatial frequencies while the center covers low relative velocities and high spatial frequencies. The gross overall pattern of the target can be initially detected by the periphery and then as the target and the array become more aligned the fine details of the target can be detected precisely by the center of the array. Now what is needed is a way to systematically incorporate a space-variant motion detection array into a tracking system. Chapter 4 describes such a tracking system and a method by which to design it.

4

Proposed Target Tracking System and a Method of Analysis and Design

The objective of the work of this thesis is to design an active visual target tracking system that can track targets both precisely and over a wide range of speeds in a computationally efficient manner. Chapter 3 described how space-variant imaging can be used to create an array of EMDs that are tuned to detect different velocity ranges and spatial frequencies. The detectable ranges across the array can be visualized with an EMD map. Now what is needed is a way to incorporate a space-variant motion detection array into a complete target tracking system. In this chapter we propose such a tracking system and describe how it can be modeled. We also propose a method of analysis and design that can be used to systematically design the parameters of the tracking system to meet

performance criteria such as overshoot and settling time. This is a challenging task because it is difficult to visualize how the motion detection array interacts with the control system and the dynamics of actuation. The methods proposed in this chapter help to visualize this interaction in order to facilitate the design of an effective and computationally efficient tracking system.

4.1 Description of the System

A diagram of a basic target tracking task is shown in Fig. 4.1. The target is assumed to be a rigid object of length, l, moving in a plane at some fixed distance, d, from a motion detection array which can pan left/right in order to keep its optical axis aligned with the target. For simplicity the motion detection array is assumed to be a 1-D array of EMDs like the one shown previously in Fig. 3.6 (a). The target's position is $X_t(t)$ and its

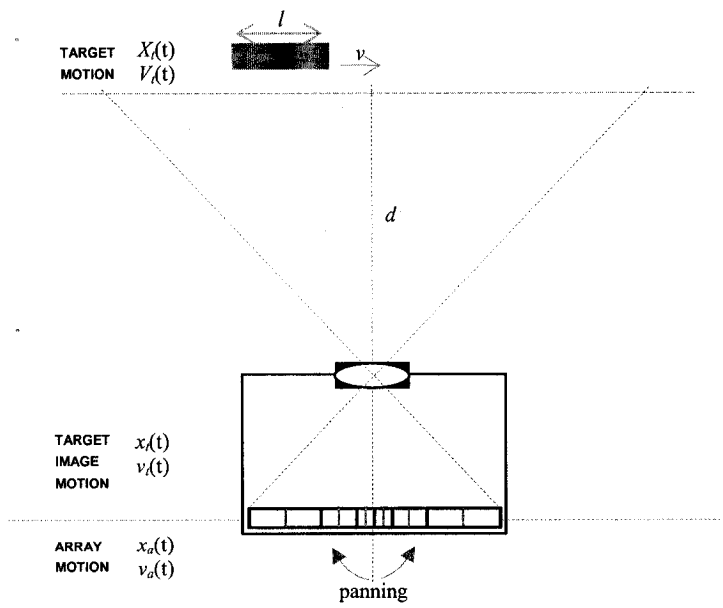


Fig. 4.1. Top-view diagram of a basic active visual target tracking system that is based on a 1-D space-variant motion detection array.

velocity is $V_t(t)$ in the same coordinate frame as the array's position $x_a(t)$ and velocity $v_a(t)$. When projected onto the array, the target's image position is $x_t(t)$ and its velocity is $v_t(t)$. It is assumed that the projection of the target onto the array is approximately orthographic and that when the array pans left/right it is approximately the same as if it has translated left/right. These approximations are good when $d >> l$. The result is that changes in target position and camera position are related by a constant scale factor.

The target's appearance is assumed to consist of a range of spatial frequencies that fall within the bandwidths of the different EMDs in the array. Thus, if the target moves such that its image passes completely over an individual EMD and their relative velocity is within the velocity range of that EMD, then the EMD's response will be sufficient to indicate a detection of the target's motion. It is possible that the target's image actually spans more than one EMD so that multiple EMDs may simultaneously make a detection. For simplicity it is assumed that the response of one of these EMDs will be dominant and will be considered to be the only detection made.

The array of EMDs can follow any progression suitable for the application. It is expected that for target tracking, a foveal array will be most useful because the target can be initially detected by the periphery and then as target and array become more aligned the fine details of the target can be detected precisely by the center of the array. The exact dimensions used for the array depend on the needs of the application. If knowledge of the target's appearance in terms of its spatial frequency content is known *a priori* then this can be used to draw the EMD map for the application. The presence of such a space-variant array in a feedback control loop leads to a nonlinear system. The nature of this nonlinearity is what leads to efficient target tracking, as discussed in the next section.

4.2 Modeling the Tracking System

As the target passes through the field-of-view of the array, a detection will be made only when the velocity of the target's image, relative to the array, is within the range of at least one of the EMDs. Initially the relative velocity may be high and detections will only be possible at the periphery. Each detection results in a new estimate of the target's position and velocity relative to the array. This information can then be used to update the control signal used to drive the motion of the array. Through the controlled movement of the array, the relative velocity will be decreased and detections can be made closer to the center. Each new detection that is made is a *discrete event* that marks a change in the control signal. The arrangement of the EMDs in the array acts as a switching mechanism that effectively 'switches' the system between different control signals depending on the particular detections that occur. The tracking performance of such a system depends on the interaction between the discrete events of detection and the continuous-time dynamics of the actuator/plant. This type of system is often referred to as a switched system or a hybrid control system [53][54]. This type of system is commonly found in a diverse group of application areas including power systems [55], robotics [56], manufacturing [57] and air traffic control [58].

Since detections and therefore 'switching' will occur at the boundaries of individual EMDs, an EMD map as shown before in Fig. 3.6 (b) describes those switching boundaries for a 1-D array. In order to be able to refer to particular detections and their locations, an indexing scheme for the map is needed. Fig. 4.2 shows the same EMD map as in Fig. 3.6 (b) but now each rectangle is indexed with a quadrant number $q \in \{I,II,III,IV\}$ and an EMD number $n \in \{1, 2,...N\}$. As before, the y-axis represents the

relative velocity v_r and the x-axis represents the relative position x_r of the target with respect to the array. Recall that the plot represents a 1-D array of EMDs so that quadrants I and II represent positive relative velocities while quadrants III and IV represent negative relative velocities for the same EMDs. When a detection is made, the relative position and velocity of a target can be estimated resulting in a quantized measure of the actual values. The EMD map is essentially a non-uniform quantizer that partitions x_r-v_r space into a non-exhaustive set of disjoint rectangles, each associated with an estimate of the relative position and velocity. One must be careful not to confuse the various spaces that are being sampled in the system. First, the sensing plane is being non-uniformly sampled, and second, the target is being detected at a non-uniform sampling frequency as well.

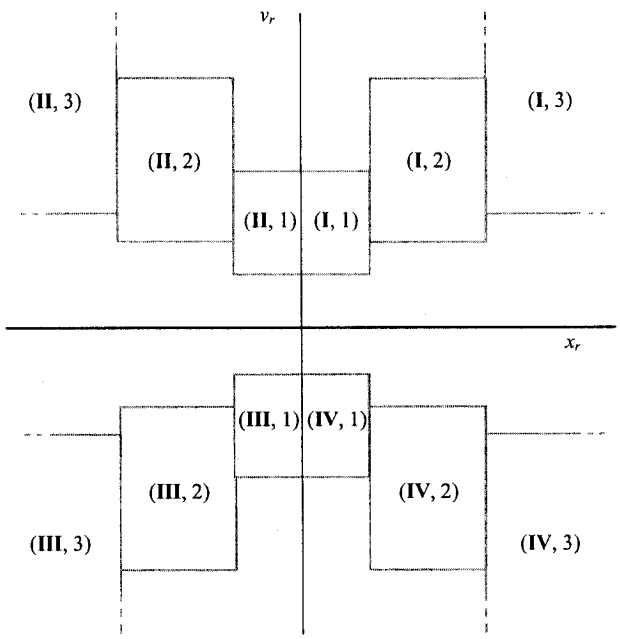


Fig. 4.2. Indexing of the EMD map's rectangles or quantization regions in v_r-x_r space showing quadrant numbers $q \in \{\text{I,II,III,IV}\}$ and EMD numbers $n \in \{1, 2,...N\}$.

The complete tracking system can be modeled as a basic closed-loop system with three components: the motion detection array, the controller and the actuator/plant. A block diagram of such a system is shown in Fig. 4.3. The target state vector is:

$$\mathbf{x}_t(t) = \begin{bmatrix} x_t(t) \\ v_t(t) \end{bmatrix} \tag{4.1}$$

and the array state vector is:

$$\mathbf{x}_a(t) = \begin{bmatrix} x_a(t) \\ v_a(t) \end{bmatrix} \tag{4.2}$$

The relative state vector is thus:

$$\mathbf{x}_r(t) = \mathbf{x}_t(t) - \mathbf{x}_a(t) \tag{4.3}$$

When a detection is made at a particular (q,n) in the EMD map, only an estimate of the relative state vector can be made:

$$\hat{\mathbf{x}}_r(q,n) = \begin{bmatrix} \hat{x}_r(q,n) \\ \hat{v}_r(q,n) \end{bmatrix} \tag{4.4}$$

A control signal $u(q,n)$ is then computed according to a control law. A very simple proportional control law computes $u(q,n)$ as the product of a gain vector $\mathbf{K}(q,n)$ and the estimate of the relative state vector, both of which depend on which EMD has made a detection:

$$u(q,n) = \mathbf{K}(q,n) \cdot \hat{\mathbf{x}}_r(q,n) = \begin{bmatrix} k_x(q,n) & k_v(q,n) \end{bmatrix} \cdot \begin{bmatrix} \hat{x}_r(q,n) \\ \hat{v}_r(q,n) \end{bmatrix} \tag{4.5}$$

The value of the gain vector $\mathbf{K}(q,n)$ can be set differently for each rectangle in the EMD map. For each detection a new control signal will result from a new gain vector and a

new estimate of the relative state vector. During the interval between detections the signal is unchanged resulting in a piecewise constant signal that drives the actuator/plant. The continuous-time dynamics of the actuator/plant can be described with a set of linear systems that are parameterized according to the quadrant and EMD number from the EMD map:

$$\dot{\mathbf{x}}_a(t) = \mathbf{A}\mathbf{x}_a(t) + \mathbf{B}u(q,n) \tag{4.6}$$

The state matrix **A** and the input matrix **B** depend on the dynamic characteristics of the actuator/plant. The combination of these continuous-time dynamics with the nonlinearity of the motion detection array makes for a difficult system to analyze. On the other hand, this is the feature that enables the system to effectively compute improved estimates of the target's position and velocity without the need for extra computational blocks. This is discussed further in section 4.5.

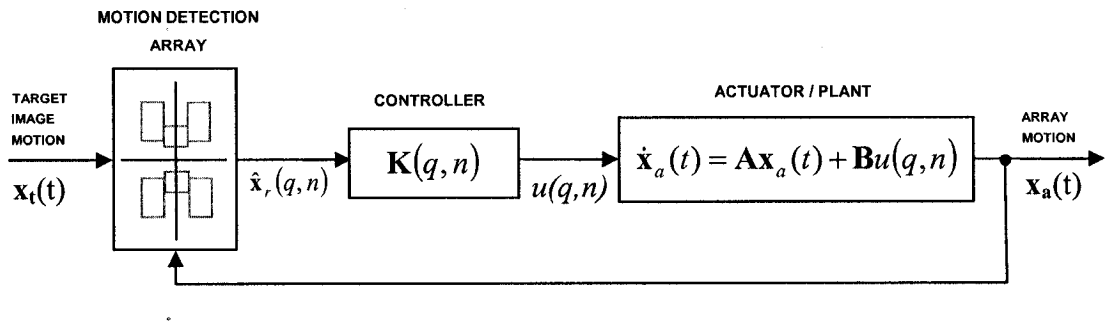


Fig. 4.3. Block diagram of a target tracking control loop with a space-variant motion detection array.

4.3 Phase-Plane Analysis and the Sensory-Motor Map

The analysis and design of such a system falls outside the scope of traditional control system approaches. Phase-plane methods and numerical analysis have mostly been used to study switched systems [54]. Some approaches to hybrid control systems with quantization in the loop [60] might be useful in this case but the non-uniform and non-exhaustive nature of the EMD map makes analysis difficult. A way to visualize the interaction between the detection events and the continuous-time dynamics is needed to acquire insight into how the various parameters of the system will affect tracking performance. Using a simple example, this section describes such a method based on combining the EMD map with a phase-plane analysis of the system.

One way to visualize the tracking dynamics of the system in Fig. 4.3 is to plot the relative state vector, $\mathbf{x}_r(t)$, in the phase-plane. For a 1-D array we can plot relative velocity on the y-axis and relative position on the x-axis. In the case of successful tracking we expect both relative velocity and position to approach zero in a cyclic trajectory about the origin. The exact trajectory would depend on the sequence of detections made by the EMDs in the array. An ordinary phase-plane plot, however, does not show the interaction between the detections made by the array and the changes in the trajectory. What is needed is a way to combine the information in the EMD map with the phase-plane trajectory in order to visualize the interaction so that we may design the system parameters for better tracking performance. To see how this can be done we use a simple example of the system in Fig. 4.3. Consider a 1-degree-of-freedom tracking system that pans left/right with continuous-time dynamics described by a first-order system:

$$\begin{bmatrix} \dot{x}_a \\ \dot{v}_a \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -a \end{bmatrix} \begin{bmatrix} x_a \\ v_a \end{bmatrix} + \begin{bmatrix} 0 \\ b \end{bmatrix} u(q,n) \tag{4.7}$$

that has a single time-constant, $\tau = 1/a$. Such a system could serve as a simple model for a dc motor or for eye movement control. Assume that a 1-D array of EMDs with an EMD map as shown previously in Fig. 3.6 (b) is being used. A good way to visualize the interaction between the detections that are made by the EMDs and the dynamics of (4.7) would be to transform the coordinates of (4.7), which are radians and radians/sec., into the coordinates and units of the EMD map, which are δx and $\delta x/\delta t$. By making this simple transformation it becomes possible to overlay a phase-plane plot of the relative state vector, $\mathbf{x}_r(t)$ onto the EMD map. Fig. 4.4 shows a Simulink block diagram that was used to compute and plot $\mathbf{x}_r(t)$. The light grey triangles represent the transformation from rad./sec. to $\delta x/\delta t$ and the dark grey triangles represent the transformation from $\delta x/\delta t$ to rad./sec. The plot of the resulting trajectory in the phase-plane x_r-v_r can then be superimposed onto the EMD map.

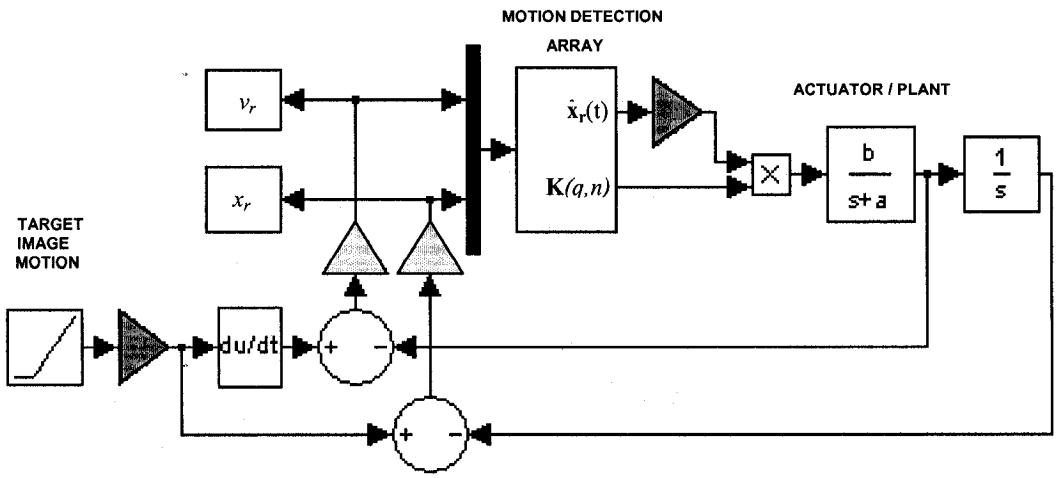


Fig.4.4. Block diagram for simulation of the tracking system with a tool like Matlab/Simulink. Light grey triangles represent transformation from rad./sec. to $\delta x/\delta t$. Dark grey triangles represent transformation from $\delta x/\delta t$ to rad./sec. Relative position and velocity (x_r, v_r) can be plotted on top of the system's EMD map.

For this particular example, Fig. 4.5 (a) shows a plot of the phase-plane trajectory of $\mathbf{x}_r(t)$. Fig. 4.5 (b) shows the same trajectory superimposed onto a plot of the EMD map. The combination produces a 'sensory-motor map' that reveals the interaction between the motion detection capabilities of the array and the tracking dynamics of the system. We assume that the motion detection array is initially stationary and the target is entering its field-of-view from the left at an image plane speed of 3 $\delta x/\delta t$. The y-axis of the sensory-motor map corresponds with zero position error, the $x_r = 0$ line, and the x-axis corresponds with zero velocity error, the $v_r = 0$ line. The origin represents zero tracking error. The diagram shows ideal results obtained after manually tuning the value of the gain vector. A detection is assumed to be made if the trajectory enters a rectangle on one side and leaves from the other side.

Referring to Fig. 4.5 (b), when the first detection at (II,3) is made, one possibility is to compute a control signal and use it to move the array to the left to reduce the position error. This would, however, raise the velocity error beyond anything that could be detected. Instead the correct reaction is for the array to move to the right to reduce the velocity error. The result of this motion is shown in the plot as the trajectory gradually slopes downwards towards zero velocity error. The trajectory eventually crosses the y-axis overshooting the $x_r = 0$ line. The next detection is made at (I,2), indicating that the target is moving toward the right edge of the field-of-view and so the array should continue to move to the right to decrease both position and velocity error. The trajectory eventually crosses the x-axis overshooting the $v_r = 0$ line. This means that the array is now moving strongly to the right to 'catch' the target and the relative velocity now becomes negative.

When a detection is made at (IV,2) it means that the velocity error is now negative and the array should slow its rightward motion gradually so as to reduce velocity error. This is shown as the trajectory is pushed closer into the origin as it crosses the negative y-axis. This process continues until the trajectory enters a periodic cycle in which it orbits the origin at a distance that is limited by the size of the smallest EMDs. The time

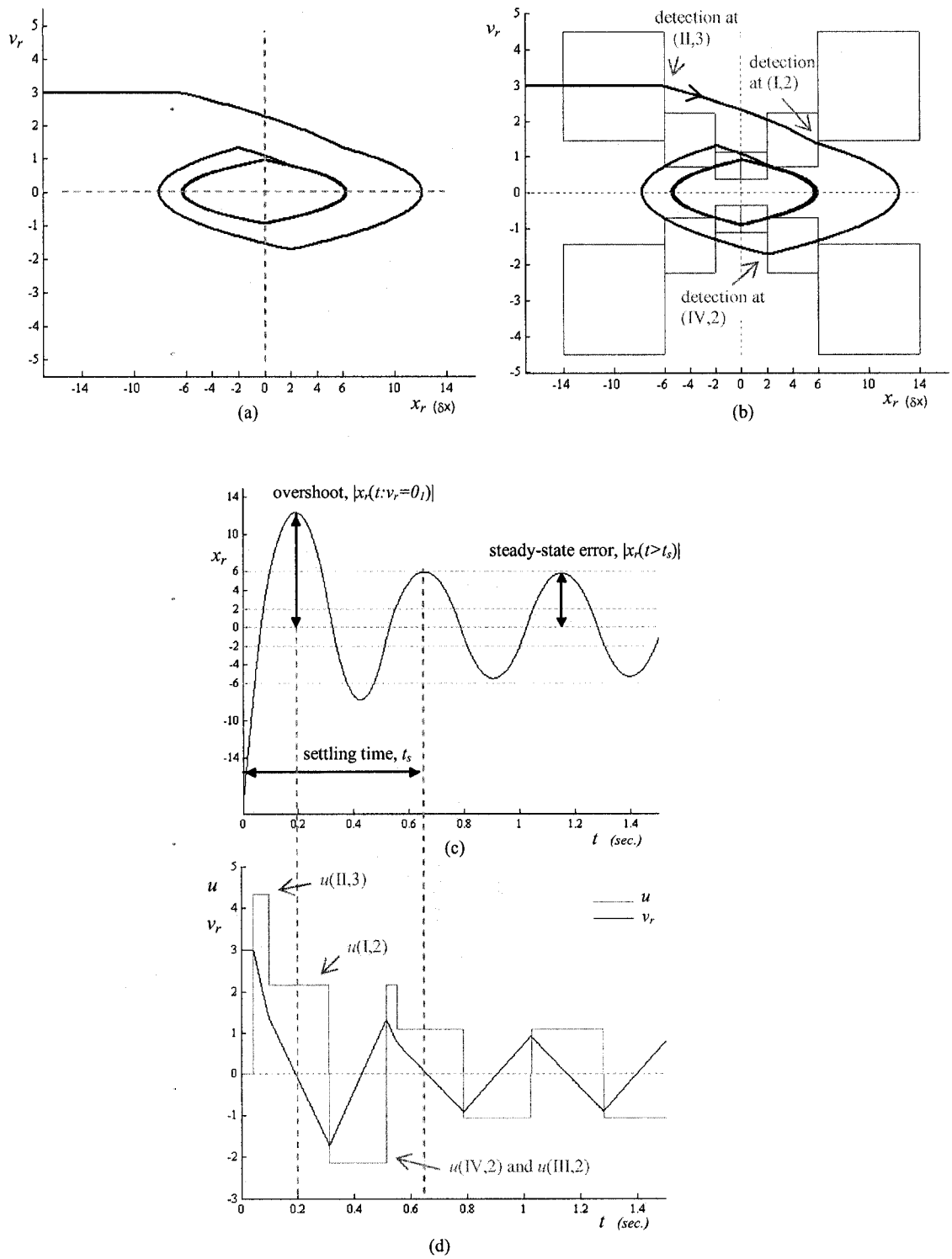


Fig. 4.5. (a) The phase-plane trajectory of $\mathbf{x_r}(t)$ and (b) the sensory-motor map showing the trajectory $\mathbf{x_r}(t)$ superimposed onto EMD map. (c) The relative position as a function of time and (d) the relative velocity and control signal as functions of time.

response is plotted in Fig. 4.5 (c). It shows the correspondence between the shape of the trajectory and such time-domain metrics as overshoot, settling time and steady-state error. The settling time is the time needed for the trajectory to reach a periodic cycle and the steady-state error is given by the width of the cycle in the phase-plane.

By experimenting with the model it was found that achieving the type of trajectory shown in Fig. 4.5 (b) was only possible for an actuator/plant with a relatively large time constant. If the time constant $\tau = 1/a$ was too small then no amount of tuning of the gain vectors $\mathbf{K}(q,n)$ would lead to this trajectory. The trajectory would not cross the x and y-axes in a way that is needed for periodic orbiting. With a larger time constant and sufficiently large gain the array would be able to overshoot the target in a smooth and gradual way that is controllable with the gain. This is because the velocity of the plant should continue to increase/decrease between any two detections in the lower/upper half of the map. Its response to the control signal should be like that of an integrator. This means that the time constant should be several times larger than the time between any two detections. If L is the total length of the array in pixels and v_{rmin} is the smallest relative velocity of interest, in units of $\delta x/\delta t$, then as a guideline:

$$\tau >> \frac{L}{v_{r\min}} \cdot \Delta t \tag{4.8}$$

For the array and trajectory of Fig. 4.5 (b), $L = 144\delta x$, $\Delta t = 0.01$ sec. and if we set $v_{rmin} = 0.5\ \delta x/\delta t$ then we need $\tau >> 2.9$ sec. The actual value used for the plot was $a = 0.01$ or $\tau = 100$ sec. Fig. 4.6 shows a plot of v_r and u over time for $\tau = 0.1$ sec. The velocity of the array matches that of the target instead of continuing to decrease as shown by the dashed line. The result is that v_r goes to zero asymptotically while x_r increases without bound. A

physical actuator/plant will have its own dynamics that may be very different from what is needed for good tracking performance. In that case, compensation can be used to reduce the effects of those dynamics and introduce a sufficiently large time constant into the system.

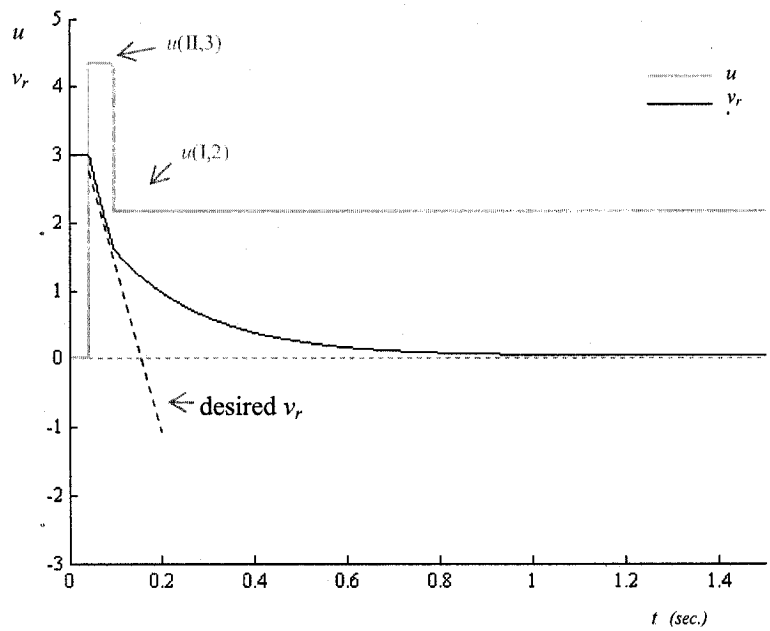


Fig.4.6. Plot of v_r and u over time for $\tau = 0.1$ sec. showing how the velocity of the array now matches that of the target instead of continuing to increase, as shown by the dashed line.

The sensory-motor map makes it possible to explore the influence of the system parameters on the tracking performance by showing both the EMD map and the relative state trajectory together. From Fig. 4.5 (b) it can be observed that the array must be controlled to move in such a way that the trajectory will spiral inward gradually and enter into a cycle close to the smallest EMDs. As long as the relative velocity is positive the array should move to the right. When the relative velocity becomes negative then detections in quadrants III and IV should cause the array to slow its rightward motion. The control signal, $u(q,n)$, should be positive for detections in quadrants I and II and negative for detections in quadrants III and IV as shown in Fig. 4.5 (d). The control signal is shown in grey and the relative velocity is shown in black. The vertical dashed line shows the correspondence between $v_r(t) = 0$ and the peaks of $x_r(t)$. The shape of the plot in (d) resembles the same type of plot for a simple nonlinear oscillator in which a relay with hysteresis and a linear system are connected in a feedback loop [59]. The relay is an odd function and therefore guarantees zero-mean cyclic signals. For our tracking system, $u(q,n)$ should also be an odd function symmetric about the x-axis. This is shown graphically in Fig. 4.7 (a). The values of $\mathbf{K}(q,n)$ must be set appropriately to achieve such a $u(q,n)$.

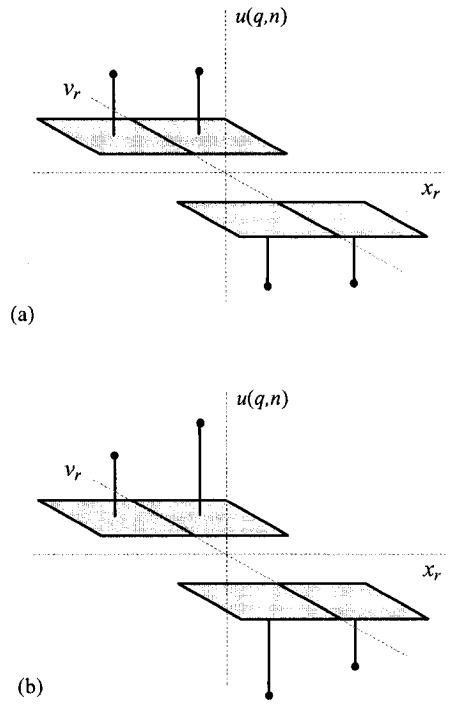


Fig. 4.7. (a) A simplified EMD map showing that $u(q,n)$ should be an odd function about the x_r axis, resembling the action of a relay in a simple nonlinear oscillator. Only the rectangles for the smallest EMDs are shown to highlight the shape of the control signal. (b) Quadrants I and III are adjusted for less overshoot.

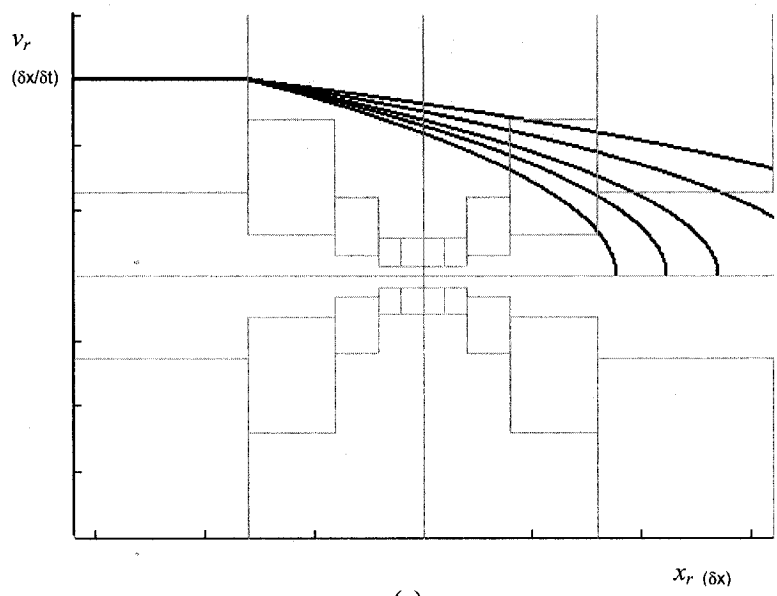
Looking back at Fig. 4.5 (b) it can be seen that a detection at (I,2) causes the trajectory to flatten out somewhat and thereby increase the overshoot. To reduce the overshoot it appears that all that is needed is to increase the value of the control signal, and therefore $\mathbf{K}(q,n)$, for quadrants I and III as shown in Fig. 4.7 (b). Increasing $\mathbf{K}(q,n)$ in any case must be done incrementally because if the slope of the trajectory becomes too sharp then it will not pass left/right through a rectangle which means that the target's

motion is not detected. Without detections the trajectory will move outward away from the origin. and the target will be lost. What is needed is a trajectory that spirals inward gradually but not so gradually that the settling time and overshoot are prohibitively large.

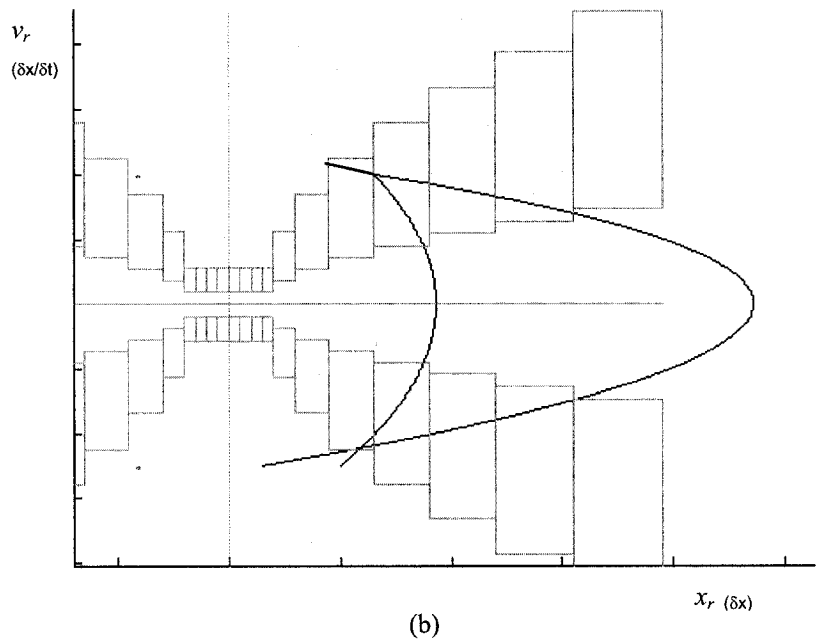
4.4 A Design Example Using the Sensory-Motor Map

Designing the tracking system requires a systematic way or algorithm to determine the values $\mathbf{K}(q,n)$ that will minimize settling time, steady-state error and overshoot for any combination of motion detection array and actuator/plant. Insights provided by the sensory-motor map make this possible. Experiments with the simulation have shown us that large values of $\mathbf{K}(q,n)$ result in trajectories that slope sharply toward the origin, reducing the settling time. Examples are shown in Fig. 4.8. But if the slope is too sharp then detections will be missed and the target will be lost. The lower the relative velocity, the sharper the slope will be for a given $\mathbf{K}(q,n)$. Thus, if we consider a target entering the field-of-view of the array on the left or right, we should use the minimum relative velocity detectable by the outermost EMD, call it $v_{min}(q,N)$, when maximizing $\mathbf{K}(q,n)$. A simple algorithm would initialize $\mathbf{K}(q,n)$ to zero for all (q,n) and iteratively increment all of the values by some step size. A good step size can be quickly found manually by trying a few values to see the effect on the shape of the trajectory.

A simple two-stage algorithm is used in which the first stage attempts to minimize settling time and steady-state error while the second stage attempts to minimize overshoot. For the first stage, $\mathbf{K}(q,n)$ is iteratively incremented causing the trajectory to slope more sharply toward the origin. Eventually the slope of the trajectory will be too sharp to be detected and the trajectory will move out of range. The values of $\mathbf{K}(q,n)$ for



(a)



(b)

Fig. 4.8. The effect of different values of **K**(q,n) on the trajectory. In (a) the values of **K**(q,n) for q = II are incremented causing the trajectory to slope more downward. In (b) the values of **K**(q,n) for q = I are incremented causing the trajectory to slope more downward until it cannot be detected.

the previous iteration are those that will minimize the settling time and the steady-state error associated with that settling time. This simple procedure is shown as Stage 1 of the algorithm in Table I.

For higher relative velocities the trajectory will begin to have a larger value of overshoot. To reduce this overshoot, a second stage of optimization can be used. In this stage the values of $\mathbf{K}(q,n)$ for quadrants I and III only are iteratively incremented to increase the rate at which the trajectory will slope downward to cross the x-axis. The iterations continue until the trajectory suddenly diverges, indicating that a detection was missed. This is Stage 2 of the algorithm in Table I.

TABLE I
ALGORITHM TO SET $\mathbf{K}(q,n)$

Stage 1: Minimize settling time (t_s) and steady-state error $|x_r(t>t_s)|$

initialize: $\mathbf{K}(q,n) = 0$ for $\begin{cases} q = \text{I,II,III,IV} \\ n = 1, 2, \ldots N \end{cases}$

do
- increment $\mathbf{K}(q,n) \leftarrow \mathbf{K}(q,n) + \text{step}^*$ for $\begin{cases} q = \text{I,II,III,IV} \\ n = 1, 2, \ldots N \end{cases}$
- solve system for $v_t(t) = v_{min}(q,N)$

while $|x_r(t>t_s)| < |x_r(t>t_s)|_{prev}$

Stage 2: Minimize overshoot

do
- increment $\mathbf{K}(q,n) \leftarrow \mathbf{K}(q,n) + \text{step}^*$ for $\begin{cases} q = \text{I,III} \\ n = 1, 2, \ldots N \end{cases}$
- solve system for $v_t(t) = v_{max}(q,N)$

while $|x_r(t>t_s)| < |x_r(t>t_s)|_{prev}$

* step size graphically determined

The algorithm is demonstrated with an example array of EMDs with values of Δx in a progression of 1:2:3:4:5:6:7:8, resulting in the most overlap of velocity range for a system based on a standard camera. The plots in Fig. 4.9 (a)-(c) show the resulting sensory-motor diagrams after the first stage of the algorithm for target velocities of 4 $\delta x/\delta t$, 6$\delta x/\delta t$ and 8 $\delta x/\delta t$. As the target velocity is increased from (a) to (c) the trajectory approaches the origin less sharply and with more overshoot. The plots in Fig. 4.9 (d)-(f) show how the trajectories look after the second stage of adjustment for the same relative velocities as used in (a)-(c). The overshoot has been improved but the settling time has gotten worse. It is possible to continue in this way to tune the gain vectors individually to achieve the performance that a particular application calls for.

The sensory-motor diagram is a visualization tool that helps in exploring the behavior of tracking systems that are based on space-variant motion detection arrays. It would be difficult to design such a tracking system without being able to make the observations that were made with the help of the diagram. The approach easily extends to larger arrays with an endless variety of arrangements of EMDs. More sophisticated algorithms to find optimal values of $\mathbf{K}(q,n)$ are possible and can easily be explored within this framework.

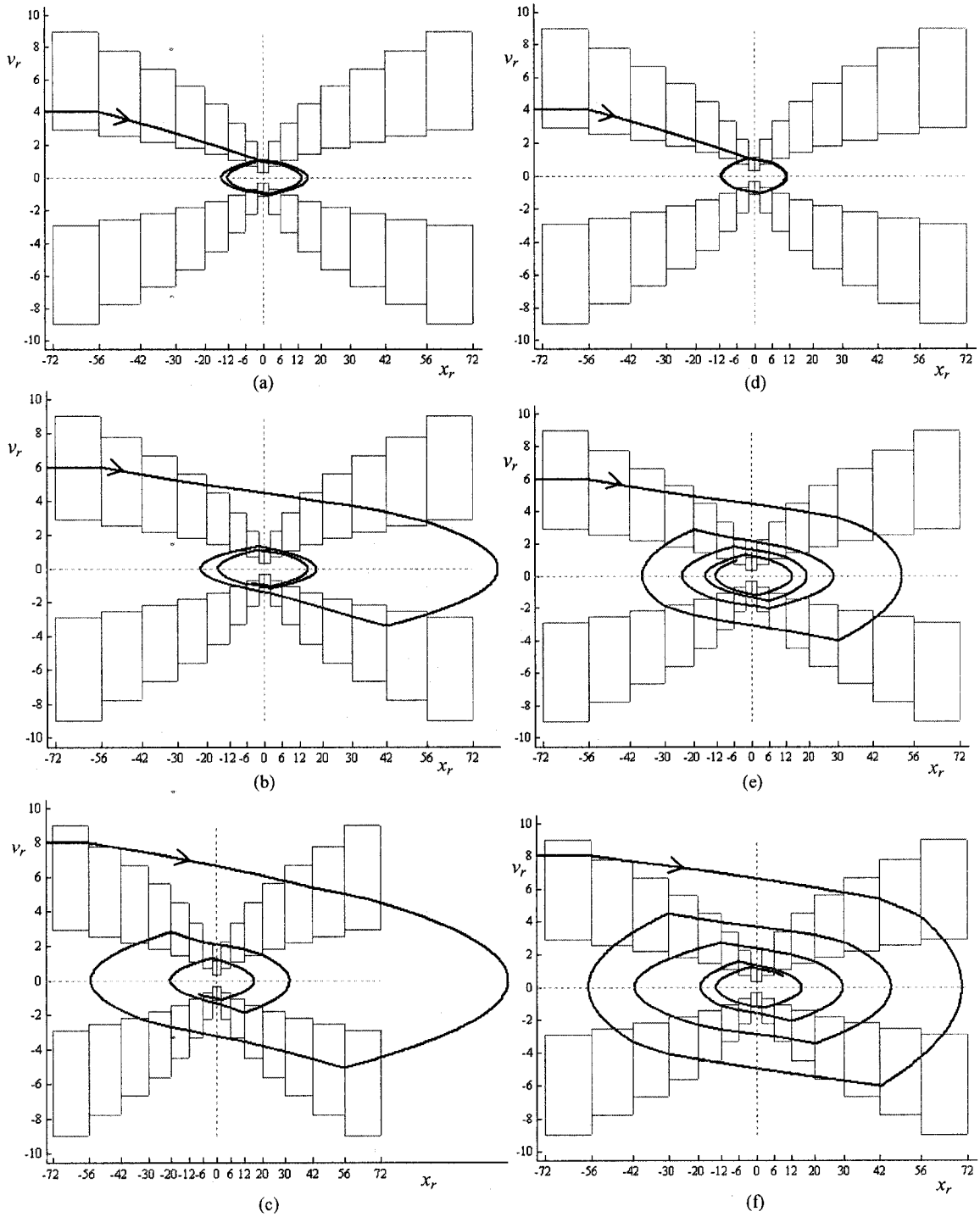


Fig. 4.9. Sensory-motor maps produced using the algorithm in Table I to set $\mathbf{K}(q,n)$. The first stage of the algorithm was used for (a)-(c) and the second stage was used for (d)-(f). The initial relative velocities are 4 $\delta x/\delta t$, 6$\delta x/\delta t$ and 8 $\delta x/\delta t$. The trajectories of (e) and (f) show much less overshoot but longer settling times than the corresponding trajectories in (b) and (c). For all plots x_r is in units of δx and v_r is in units of $\delta x/\delta t$.

4.5 Discussion

The model for the active visual target tracking system, as presented in sections 4.1 and 4.2, consisted of a set of linear systems to determine the position and velocity of the sensor array, $\mathbf{x}_a(t)$:

$$\dot{\mathbf{x}}_a(t) = \mathbf{A}\mathbf{x}_a(t) + \mathbf{B}u(q,n) \tag{4.9}$$

in which:

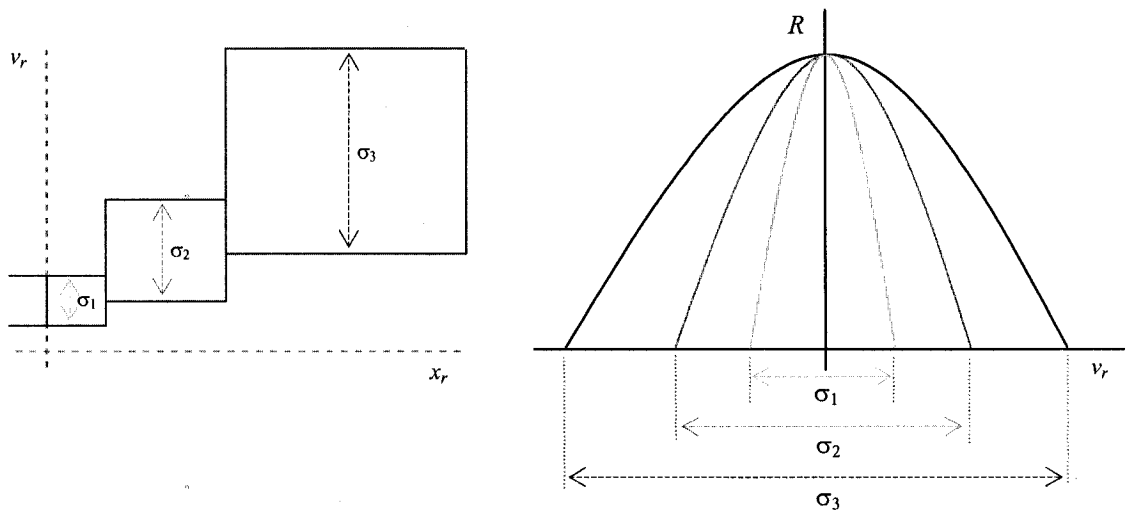
$$u(q,n) = \mathbf{K}(q,n)\cdot \hat{\mathbf{x}}_r(q,n). \tag{4.10}$$

$\mathbf{K}(q,n)$, is a gain vector assigned to each rectangle (q,n) in the EMD map and $\hat{\mathbf{x}}_r(t)$ is the estimate of the relative state of the target to the array. This estimate is produced by the motion detection array each time a detection is made. The values of $\mathbf{K}(q,n)$ can be computed, as shown in section 4.3, such that each detection of a constant velocity target will lead to a relative state trajectory, $\mathbf{x}_r(t)$, that converges to the origin. With this convergence the accuracy of each estimate should be increasing because the range of possible velocities and positions are getting smaller, as shown in Fig. 4.10 (a). If the ranges of the possible velocities for $n = 1,..,N$ in any quadrant of an EMD map are denoted with $\sigma_1,...,\sigma_N$, then:

$$\sigma_1 \leq \sigma_2 \leq ... \leq \sigma_N \tag{4.11}$$

This means that the estimates are always improving. Fig. 4.10 (b) shows the response of each EMD and how the 'spread' or accuracy of the estimates will improve as detections are made closer to the center of the array. This is an important result of the space-variant array. In systems that use uniform sensing there must be some other way to improve the quality of the estimate. Often, recursive computations are used to improve the estimate,

most notably algorithms such as the Kalman filter [61]. These recursive algorithms are usually computationally expensive. A space-variant motion detection array, on the other hand, will produce improved estimates without any extra computation.



(a) (b)

Fig. 4.10. (a) One quadrant of an EMD map showing the ranges of velocities for which each EMD is sensitive. (b) A plot of the response of each of the EMDs in (a) showing how the 'spread' of each estimate is narrower for detections closer to the center.

An active visual target tracking system is a closed-loop system with a sensor and an actuator. Typically, for such a closed-loop control system, the issue of stability is important. Stability can be defined in several different ways depending on the type of system being considered and the way it is modeled. For linear time-invariant (LTI) systems, the most common definition is bounded-input, bounded-output (BIBO) stability

and the criterion is that the impulse response is absolutely integrable or that the transfer function has all of its poles in the open left-half plane.

The tracking system described in this thesis is not an LTI system. It has been described as a hybrid system. Such a system is characterized by the interaction between its continuous-time dynamics and the discrete events, in this case produced by the detections of target motion. Many hybrid systems, including the tracking system in this thesis, exhibit periodic behavior. The system is designed with the goal of trapping the evolving system state within a constrained region of state space defined by the smallest EMDs closest to the origin. This periodic behavior is usually called a limit cycle. A limit cycle can be stable (attracting), unstable (repelling) or non-stable (saddle). The stability of a limit cycle can be explored in a number of different ways. One interesting way involves determining the characteristic or Floquet multipliers [61]. These multipliers are a generalization of the eigenvalues at an equilibrium point. A limit cycle, or periodic solution, corresponds to a fixed point on a Poincare map. The stability of the periodic solution is the same as the stability of the fixed point. The multipliers are the eigenvalues of the Poincare map linearized about the fixed point. While interesting, this approach and others are not applicable to the system in this thesis because of some particular features of the way motion is detected. The EMD map captures this, showing that the motion detection is essentially described as a non-uniform quantizer that partitions x_r-v_r space into a non-exhaustive set of disjoint rectangles, each associated with an estimate of the relative position and velocity. As such, it provides an incomplete measurement of the system state. Fig. 4.11 shows how this can affect the tracking. Two trajectories are shown. The lower trajectory shows the system converging to a cycle very close to the

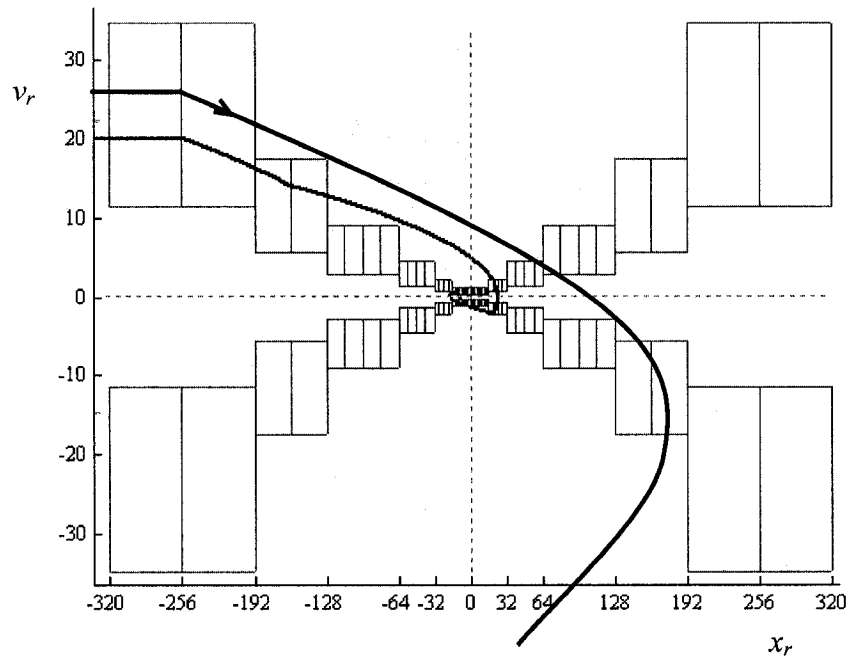


Fig. 4.11. Sensory-motor map showing how a target can be lost because of the non-exhaustive nature of the EMD map.

origin. The upper trajectory shows how a slight change in initial velocity of the target and the system parameters results in the system being unable to detect the target's motion and the trajectory diverges. The system is not observable and therefore is not stabilizable since it is possible for the system to evolve in ways that are not detectable by the sensor. The system would need more EMDs distributed in such a way as to better cover the x_r-v_r space.

Although a formal statement regarding the stability of the tracking system cannot be made, the system can be designed to tend toward a stable limit cycle as described in section 4.3. Through the use of the sensory-motor map it is possible to set the system parameters so that the trajectory cycles inward to the origin. Of course, if a detection is

not made when needed, then the target will likely be lost. This can only be helped with a denser and more redundant array of EMDs.

4.6 Summary

This chapter has presented a complete target tracking system based on a space-variant motion detection array. The system is based on a foveal arrangement of EMDs making it possible to detect a target moving at high speed and to gradually reduce the tracking error while increasing the precision of the tracking. It does not require any special computational resources beyond the basic mechanism of motion detection. We have proposed a model for such a tracking system and we have shown how the model can be used as the basis for a method of analysis and design to systematically design the parameters of the system to meet performance goals. A visualization tool, called the sensory-motor map, makes it possible to design the tracking system by showing both the EMD map and the relative state trajectory together. Overall, the proposed method of design of a space-variant tracking system involves two major steps:

i) First, the EMD map, as described in chapter 3, must be defined to determine the distribution and tuning of the EMDs across the array. This is based on the needs of the application such as the range of velocities that must be detected and the size of the minimum spatial feature that must be resolved and tracked.

ii) Second, a sensory-motor diagram of the tracking system is used to plot the behavior of the tracking system for a range of target velocities. This can be done with any suitable numerical package such as MATLAB/Simulink. An optimization

algorithm can then be used in conjunction with the plots to determine the control gain for each EMD.

The result is an effective and computationally efficient tracking system that combines both range and precision in a single array. Designing such a system without the aid of these visualization tools would be extremely difficult. The approach is general and can be applied to different motion detection arrays for a variety of applications.

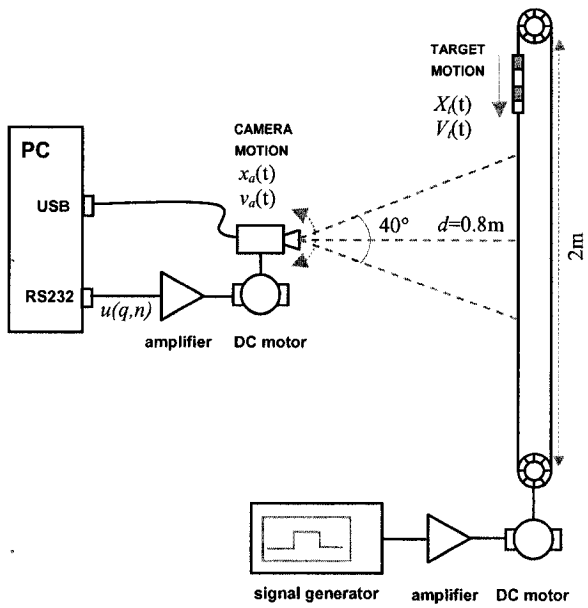
5

Experiments

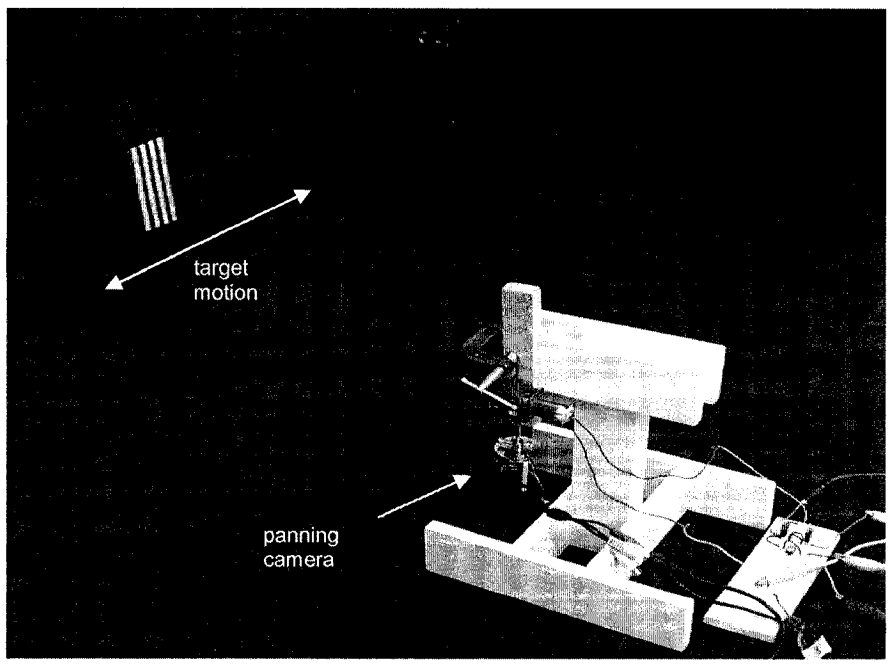
A complete target tracking system based on a space-variant motion detection array has been proposed. The system is based on a foveal arrangement of EMDs making it possible to detect a target over a wide range of velocities and with high precision. A method of design and analysis for such a system has also been proposed. The purpose of this chapter is to demonstrate how these ideas can be used to implement an experimental active vision system. The implementation is used to explore the effectiveness of the proposed architecture in an experimental setting. A set of target tracking experiments are conducted with the goal of verifying the usefulness of the architecture by comparing the results with the predictions of the system model.

5.1 Experimental Setup for Target Tracking

Fig. 5.1 (a) shows a schematic diagram of the active tracking system and the experimental setup that was used to perform the target tracking experiments. Fig. 5.1 (b) shows a photograph of the setup. For all of the experiments the target is a rectangular paper card with a spatial frequency pattern printed on it. The card is attached to a 2m long horizontal conveyor belt that is driven by a DC motor. A signal generator and amplifier are used to control the speed of the left-to-right movement of the target. Images of the moving target are captured by a standard video camera. The camera is mounted on a pan-head at a distance of $d = 0.8$m from the target plane. The video output of the camera is connected to a desktop PC via a USB interface. A program written in C and the OpenCV library [62] processes the video. Fig. 5.2 shows the graphical interface of the program while a target is passing from left to right in front of the camera. The plot at the bottom of the picture shows the response of the EMDs to the target's motion. The video is processed in two stages. First, each frame's resolution is modified to create a foveal image according to a pre-specified EMD plot. This is done by combining pixels into groups and computing their average value. Second, the output of each EMD operation is computed using the previous frame for the delayed inputs. This means that the Δt is fixed by the system's achievable frame rate, which in this case was about 15 frames per second. A separate thread in the program computes the control signal $u(q,n)$ at each frame interval and outputs a pulse-width modulated (PWM) signal to the RS232 serial port. That signal is then amplified in order to drive a DC motor on the pan-head to pan the camera horizontally.



(a)



(b)

Fig. 5.1. Schematic diagram (a) of experimental setup along with a photograph (b) of the setup.

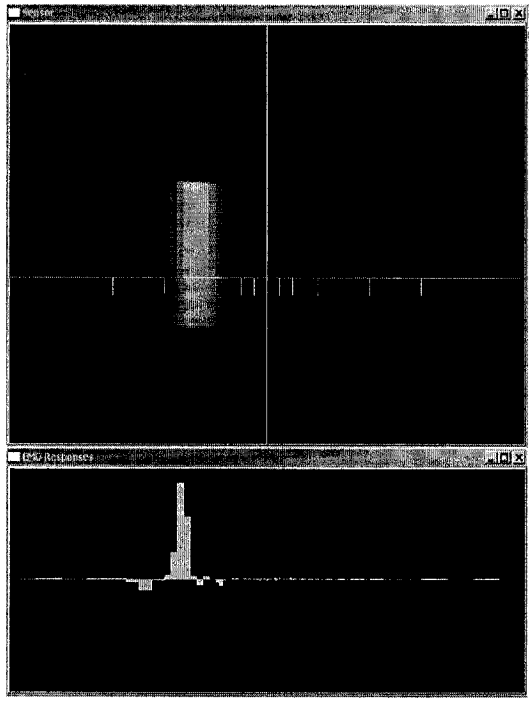


Fig. 5.2. Interface of tracking software showing the image captured by the camera (top) and the response of the EMDs across the array (bottom).

The video camera used in the experiments has a focal length of f = 5mm. Its imaging area is 4.69mm×3.54mm and it captures images at a resolution of 640×480. This means that each pixel has a width of approximately 7.3μm which is the δx for this system. For each experiment the same EMD plot was used. Fig. 5.3 shows a sensory-motor diagram for the experiment with a target initially moving to the right at 20 $\delta x/\delta t$. The algorithm of Table I was used to compute the values of $\mathbf{K}(q, n)$.

Assuming perspective projection, distances in the image plane are related to distances in the target plane by:

$$x_t \frac{d}{f} = X_t \tag{5.1}$$

Speeds in the image plane are related to speeds in the target plane by:

$$1\frac{\delta x}{\delta t}\cdot\frac{d}{f}=0.0175\,m/s \tag{5.2}$$

in which $\delta x = 7.3\times10^{-6}$ m and δt = 1/15 sec. Of course the depth, d, actually changes with time because the camera is panning but the target is translating. This alters the target's image size and speed slightly but the approximation of a constant d was found to be acceptable when the camera only pans through an angle of less than 90°. Equation (5.2) can be used to calculate the speed of the target's motion needed for the experiments. For example, 20 $\delta x/\delta t$ in the image plane is 35cm/s in the target plane, which can be achieved by adjusting the signal generator that drives the target's conveyor belt.

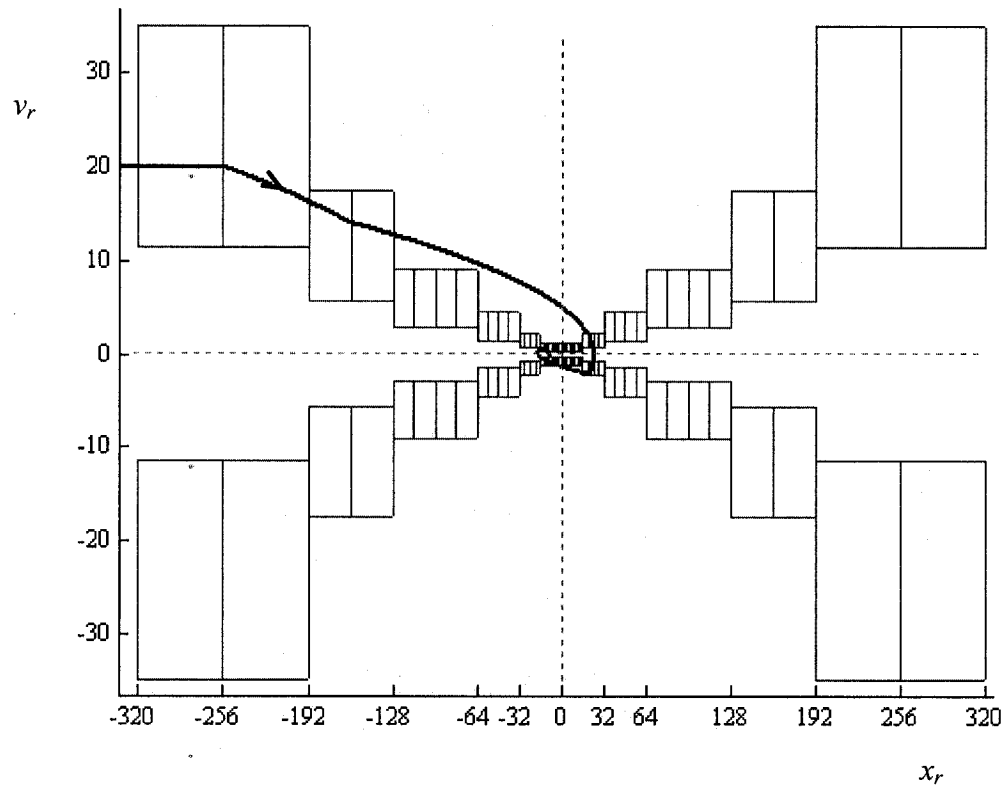


Fig. 5.3. Sensory-motor map for the experimental system.

5.2 Experiments and Results

Each experimental run consisted of initially positioning the target at the left extreme of the conveyor belt and pointing the camera such that the target was just outside the left edge of its field-of-view. The target was then driven to the right at one of three pre-selected speeds. Fig. 5.4 shows the tracking results for speeds of (a) 20 $\delta x/\delta t$, (b) 25 $\delta x/\delta t$ and (c) 30 $\delta x/\delta t$. Each plot shows the estimate of the relative position, x_r, as a function of time. The values were measured by the program for each frame processed.

For each run the relative position converged to the expected oscillation about the zero error axis. As expected, the settling time, t_s, increased with target speed as shown in the plots. The general form and tendency of each plot agrees well with the predictions of the model. The experiments confirm that it is possible to design a single array that can track targets over a wide range of velocities and with precision.

There were, however, some deviations from the expected results. The overshoot was expected to increase with target speed but the results in Fig. 5.4 (a) and (b) show, with dashed circles, times at which the tracking system reacted too quickly and then produced overshoots when compensating for the overreaction. This occurs twice in (a) likely due to the slow speed of the target at 20 $\delta x/\delta t$, but only once for (b) when the target was moving at 25 $\delta x/\delta t$. Because of the way the system was implemented it was difficult to control the time constant of the tracking dynamics. The use of a software generated PWM signal and the PC's serial port undoubtedly led to unpredictable delays in the feedback loop and made it difficult to control the system's performance precisely. Nonetheless, the experiments showed that even a crude implementation of the proposed approach enables target tracking over a wide range of speeds while still being able to achieve precise

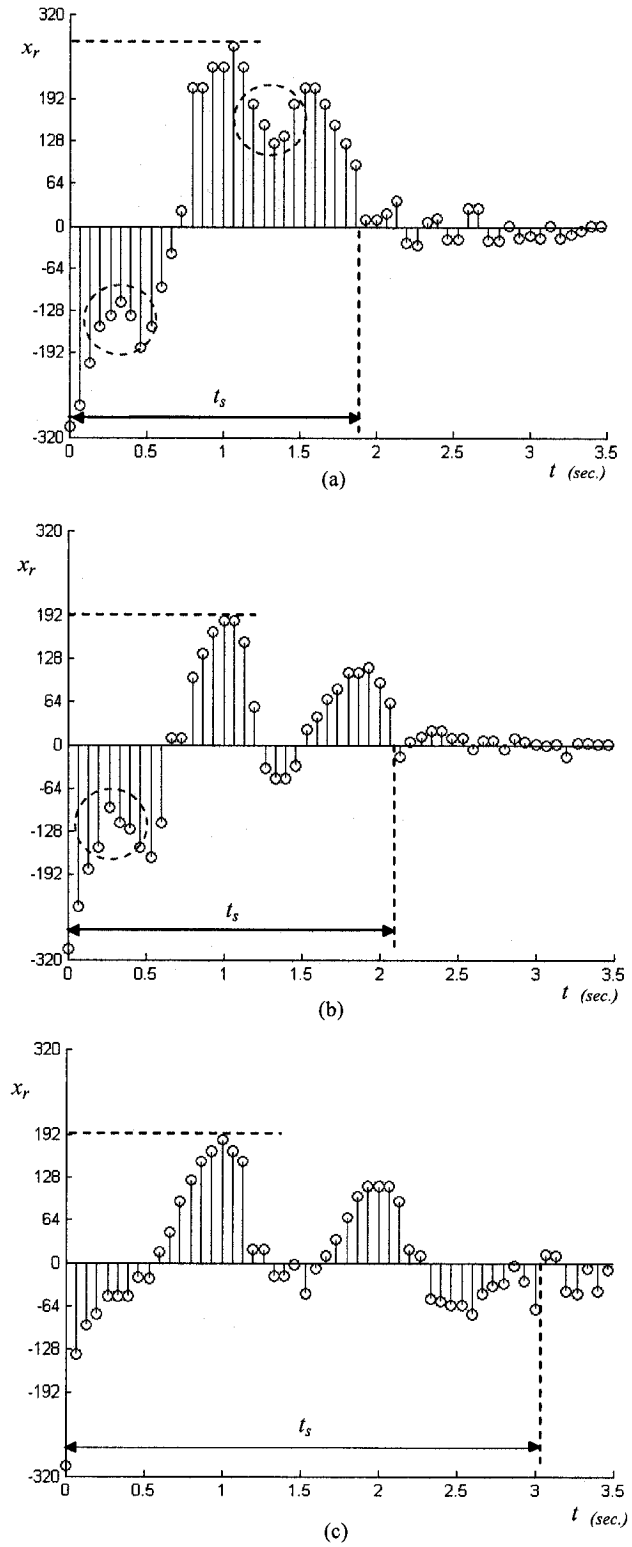


Fig. 5.4. Tracking results for target speeds of (a) 20 $\delta x/\delta t$, (b) 25 $\delta x/\delta t$ and (c) 30 $\delta x/\delta t$.

tracking. It should be emphasized that no special calibration of the camera or of the experimental setup was required. The camera and platform could be oriented in a very approximate way without significantly affecting the results.

5.3 Summary

A complete active vision system has been implemented based on the architecture proposed in this thesis. The system has been designed using the proposed methods including an EMD map, a sensory-motor diagram and a simple optimization algorithm. A set of simple target tracking experiments have confirmed the effectiveness of the architecture and the design method. The results show that even a crude implementation of such a system will enable target tracking over a wide range of speeds while still being able to achieve precise tracking.

6

Conclusion and Future Work

6.1 Conclusion

An active visual target tracking system is a type of automatic feedback control system that can track a moving target by controlling the movement of a camera or sensor array. This kind of system finds use in application areas that include automatic surveillance, human-computer interaction and mobile robotics. The design of an effective target tracking system is challenging because the system should be able to precisely detect the fine movements of a target while still being able to detect a large range of target velocities. Achieving this in a computationally efficient manner is difficult. Conventional system architectures usually consist of a standard video camera connected to a digital

computer. That approach generally requires high-speed computational resources to process the image signal in real time. The algorithms needed to extract the target's motion information and to produce a control signal are usually very complex.

The design and implementation of an effective visual tracking system involves techniques from fields such as computer vision and control systems. At the core of visual tracking is the process of extracting information about the target's motion from the image signal. In this thesis we have reviewed three general categories of motion detection techniques: i) techniques based on matching, ii) gradient-based techniques and iii) spatio-temporal frequency-based techniques. Spatio-temporal frequency-based techniques compute the spatio-temporal frequency content of local image regions. They use filters which are selectively activated by a range of spatio-temporal frequency patterns produced by the motion of a target. A simple and commonly used example is the Reichardt EMD. Because of its simplicity and efficiency, the work of this thesis is based on the Reichardt EMD, however the results are applicable to any tunable spatio-temporal frequency-based technique.

The spatio-temporal frequency-based techniques are very useful for target tracking but an individual detector of this type is limited to a particular range of spatial and temporal frequency and therefore to a particular range of target velocity. Researchers have attempted to work around this by modifying the basic detector at the computational level or by incorporating the basic detector into an architecture that supports a wider range of motion detection. While these ideas have led to improvements, they have not achieved visual tracking in a computationally efficient manner.

This thesis has presented an architecture for an active visual target tracking system based on the idea of space-variant motion detection. Here the word architecture has been used to describe the space-variant 'sensing scheme' that is used. In general, space-variant imaging involves the use of a non-uniform distribution of sensing elements across a sensor array, similar to how the photoreceptors in the human eye are not evenly distributed. In the architecture proposed in this thesis, space-variant imaging is used to design an array of elementary motion detectors (EMDs) that are tuned in such a way as to make it possible to detect motion both precisely and over a wide range of velocities in a computationally efficient manner.

Each EMD is a bandpass spatio-temporal filter. It can be tuned to detect different spatial frequency ranges and different velocity ranges by varying the inter-receptor distance, Δx. This can be achieved by varying the size and spacing of the photoreceptors across the array. Such a space-variant array can be designed to provide sensitivity to different velocity and spatial frequency ranges. The increased ranges are achieved without additional computational costs beyond the basic mechanism of motion detection. The technique is general in that it can be used with different motion detection mechanisms and the overall space-variant structure can be varied to suit a particular application.

The basic idea of the architecture is that a foveal array is expected to be ideal for target tracking because the low-resolution periphery covers high relative velocities and low spatial frequencies while the high-resolution center covers low relative velocities and high spatial frequencies. The gross overall pattern of the target can be initially detected

by the periphery and then as the target and the array become more aligned the fine details of the target can be detected precisely by the center of the array.

A visualization tool called the EMD map was proposed as a way to show how the velocity and spatial frequency ranges of the EMDs vary across the array. The EMD map can be used to visualize the detection ranges for any arrangement of EMDs whether they are foveal or some other custom arrangement. It facilitates the design of these arrays by helping the designer to see the how the spatial frequency and velocity ranges of detection are distributed across the array in the v_r-x_r plane.

A complete target tracking system based on a space-variant motion detection array has been proposed. Based on a foveal arrangement of EMDs, it combines both range and precision while not requiring any special computational resources beyond the basic mechanism of motion detection. Estimates of the relative position and velocity error improve as the system converges on the target. This improvement arises automatically from the foveal arrangement of the array without the need for complex state estimation algorithms.

Because of the unconventional nature of the tracking system, we have proposed a model for the system. We have shown how the model can be used as the basis for a method of analysis and design to systematically design the parameters of the system to meet performance goals such as overshoot, settling time and steady-state error. A visualization tool, called the sensory-motor map, makes it possible to design the tracking system by showing both the EMD map and the relative state trajectory together.

The proposed method of design of a space-variant tracking system involves two major steps. First, the EMD map must be defined to determine the distribution and tuning

of the EMDs across the array. This is based on the needs of the application such as the range of velocities that must be detected and the size of the minimum spatial feature that must be resolved and tracked. Second, a sensory-motor diagram of the tracking system is used to plot the behavior of the tracking system for a range of target velocities. This can be done with any suitable numerical package such as MATLAB/Simulink. An optimization algorithm can then be used in conjunction with the plots to determine the control gain for each EMD. The result is an effective and computationally efficient tracking system that combines both range and precision in a single array. Different implementations for such a system are possible, including a VLSI smart sensor, because of the system's modest computational needs.

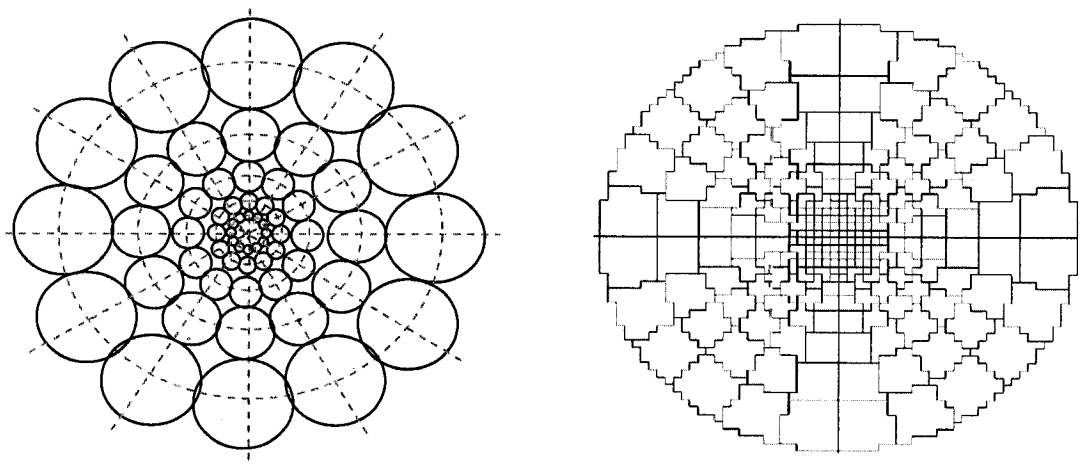
Several simulations have shown how the method can be used to control tracking performance and to meet tracking specifications. Experiments with a real active vision system have shown that the basic idea and the design method can easily improve tracking performance.

Overall the approach represents an effort toward a more general treatment of sensory-motor systems. It may be possible to use the proposed sensory-motor map in other sensory-motor contexts such as in haptic devices and auditory systems. Being able to visualize both sensing and motor actions on the same plot makes it possible to design effective systems that require fewer computational resources. When the sensing is well matched with the motor activities, and both are well-matched with the task, then the entire system can operate more efficiently. In addition, this approach may help researchers to model real biological systems and gain insight into biological mechanisms of sensory-motor control.

6.2 Future Work

It has been shown that a space-variant motion detection architecture can be used to design an active visual target tracking system that exhibits both range and precision. In particular this thesis has focused on a foveal arrangement for a 1-D array. Future work will involve an exploration of other space-variant arrangements and how they can be used in different applications. The spacing of the photoreceptors can be done in many different ways and it is possible that other visual tasks, such as obstacle avoidance, may benefit from specific arrangements that have not been explored here.

Another area for future work involves the extension of this architecture to 2-D arrays. A simple way to extend a 1-D array to a 2-D array is to arrange a set of 1-D arrays radially as shown in Fig. 6.1 (a). Each circle represents a photoreceptor. The photoreceptors along each dashed straight line form a 1-D array and the EMDs can be



(a) (b)

Fig. 6.1. 2-D arrays of photoreceptors. In (a) the photoreceptors are individually sized. In (b) the array is based on a standard uniform resolution camera.

arranged between adjacent photoreceptors as in the thesis. They can also be arranged between adjacent photoreceptors on the same dashed circle. These can be used to detect rotational motion. The same type of arrangement can be achieved using a standard camera with a uniform resolution, as shown in Fig. 6.1 (b). Each enclosed area represents a group of pixels that are averaged to create a foveal array. This approach can easily be implemented with field programmable gate array (FPGA) hardware to accelerate the averaging. Other 2-D schemes are possible, for example motion can be measured in the x and y directions with a rectangular array of EMDs. Often x,y motion information is sufficient.

Another area of future work involves an exploration of how the architecture can best deal with real-world targets that have complex appearances and complex motion patterns. It may be possible to design a space-variant array that will be more suitable to certain real-world situations depending on the spatial frequency content and motion characteristics of the targets and the surroundings.

References

[1] P. Guha, D. Palai, D. Goswami, A. Mukerjee, "DynaTracker: Target tracking in active video surveillance systems," *Proceedings of the 12th International Conference on Advanced Robotics*, pp. 621- 627, July 18-20, 2005.

[2] N. T. Siebel and S. Maybank, "Real-Time Tracking of Pedestrians and Vehicles," *2nd IEEE Workshop on Performance Evaluation of Tracking and Surveillance (PETS'2001)*, Hawaii, December 2001.

[3] T. Starner, B. Leibe, D. Minnen, T. Westyn, A. Hurst, and J. Weeks, "The perceptive workbench: Computervision-based gesture tracking, object tracking, and 3D reconstruction of augmented desks," *Machine Vision and Applications*, 14. pp. 59-71, 2003.

[4] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-Time Tracking of the Human Body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, No. 7, July 1997.

[5] M. Maimone, I. Nesnas, and H. Das, "Autonomous Rock Tracking and Acquisition from a Mars Rover," *1999 International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS'99)*, June, 1999.

[6] S. Feyrer and A. Zell, "Tracking and Pursuing Persons with a Mobile Robot," *International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*, p. 83, 1999.

[7] G. Metta, F. Panerai and G. Sandini, "Babybot: A biologically inspired developing robotic agent," *Proceedings of SPIE*, Boston (USA) November 2000.

[8] C. Koch and H. Lui, "Vision Chips: Implementing Vision Algorithms with Analog VLSI Circuits," IEEE Computer Society Press, 1995.

[9] S. Venkatesh and M. V. Srinivasan (eds), From living eyes to seeing machines, Oxford Univesity Press, Oxford, p174-198, 1997.

[10] R. Mudra, R. Hahnloser, and R. J. Douglas, "Neuromorphic Active Vision Used in Simple Navigation Behavior for a Robot," *Proceedings of the 7th international Conference on Microelectronics For Neural, Fuzzy and Bio-inspired Systems*, IEEE Computer Society, Washington, DC, April 07 - 09, 1999.

[11] J. L. Barron, S.S. Beauchemin and D.J. Fleet, "On Optical Flow,". *6th Int. Conf. on Artificial Intelligence and Information-Control Systems of Robots,* Bratislava, Slovakia, Sept 12-16, pp3-14, 1994.

[12] E. H. Adelson and J. R. Bergen, "The extraction of spatio-temporal energy in human and machine vision", *Proceedings from the workshop on motion: Representation and Analysis*, May 1986, pp. 151-155.

[13] S. Moskowitz, "Terminal Guidance by Pattern Recognition- A New Approach," *IEEE Transactions on Aerospace and Navigational Electronics*, December, 1964.

[14] A. L. Gilbert, "Video Data Conversion and Real-Time Tracking," *IEEE Computer*, pp. 50–56, August, 1981.

[15] A. E. Hunt and A. C. Sanderson, "Vision-based predictive tracking of a moving target," Camegie Mellon Univ., The Robotics Inst., Tech. Rep. CMU-RI-TR-82-15, Jan. 1982.

[16] D. Marr and E. Hilreth, "Theory of edge detection," *Proc. Royal Soc. Lond.*, vol. B 207, pp. 187-217, 1980.

[17] C. Harris and M. J. Stephens, "A combined corner and edge detector," *Alvey Vision Conference*, pages 147–152, 1988.

[18] S. T. Barnard and W. B. Thompson, "Disparity analysis of images," *IEEE Pattern Analysis and Machine Intelligence*, vol. 2, no. 4, pp. 333-340, 1980.

[19] J. R. Jain and A. K. Jain, "Displacement measurement and its applications in intraframe image coding," *IEEE Transactions on Communications*, pp. 1799-1808, 1981.

[20] D. Murray and A. Basu, "Motion Tracking with an Active Camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, May 1994.

[21] J.O. Limb and J. A. Murphy, "Estimating the Velocity of Moving Objects in Television Signals," *Computer Graphics and Image Processing*, Vol. 4, pp. 311-327, 1975.

[22] C. L. Fennema and W. B. Thompson, "Velocity Determination in Scenes Containing Several Moving Objects," *Computer Graphics and Image Processing*, Vol. 9, 1979, pp. 301-315.

[23] B. K. P. Horn and B. G. Schunk, "Determining Optical Flow," MIT A.I. Memo 572, MIT, Cambridge, Mass., 1980.

[24] J. Tanner and C. Mead, "An integrated analog optical motion sensor," in *VLSI Signal Processing, II*, S.Y. Kung, Ed. New York: IEEE Press, 1986, pp. 59-76.

[25] A. Stocker, "An improved 2D optical flow sensor for motion segmentation," in *Proceedings IEEE Int. Symposium on Circuits and Systems ISCAS*, pages II-332-II-335, phoenix, U.S.A., 2002.

[26] W. Reichardt, "Autocorrelation, a principle for the evaluation of sensory information by the central nervous system," in Sensory Communication, ed. W.A. Rosenblith, pp. 303-317, MIT Press, Cambridge, Massachusetts, 1961.

[27] P. H. Schiller, B. L. Finlay, and S. F. Volman, "Qualitative studies of single-cell properties in monkey striate cortex. I. Spatiotemporal organization of receptive fields," *J. Neurophysiology*, vol. 39. no. 6, pp. 1288-1319, 1976.

[28] E. H. Adelson and J.R. Bergen, "Spatiotemporal energy models for the perception of motion," *Journal of the Optical Society of America A.*, vol. 2, no. 2, pp. 284--299, Feb 1985.

[29] J. S. Bendat, Principles and Applications of Random Noise Theory, John Wiley and Sons, Inc., 1958.

[30] R. O. Dror, D. C. O'Carroll, and S. B. Laughlin, "Accuracy of velocity estimation by Reichardt correlators," *Journal of the Optical Society of America A* 18, pp. 241–252, February 2001.

[31] T. Delbruck, "Silicon retina with correlation-based, velocity-tuned pixels," *IEEE Trans. Neural Networks*, vol. 4, pp. 529-541, May 1993.

[32] J. Kramer, R. Sarpeshkar, and C. Koch, "Pulse-based analog VLSI velocity sensors," *IEEE Trans. Circuits Syst. II*, vol. 44, pp. 86-101, 1997.

[33] R.R. Harrison and C. Koch, "A silicon implementation of the fly's optomotor control system," *Neural Computation*, 12: 2291-2304, 2000.

[34] H. C. Jiang and C. Y. Wu, "A 2-D velocity and direction selective sensor with BJT-based silicon retina and temporal zero-crossing detector," *IEEE Journal of Solid-State Circuits*, 34(2) pp. 241-247, Feb. 1999.

[35] C. M. Higgins, and S. A. Shams, "A Biologically-Inspired Modular VLSI System for Visual Measurement of Self-Motion," *IEEE Sensors Journal*, pp. 508--528, December 2002.

[36] G. Walter, "A machine that learns," *Scientific American* (1951) 185(2): 60—63.

[37] C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley Longman Publishing Co., Inc, 1989.

[38] J. Y. Aloimonos, I. Weiss and A. Bandopadhay, "Active Vision", *International Journal on Computer Vision*, pp. 333-356, 1987.

[39] R. Bajcsy, "Active Perception", *IEEE Proceedings*, vol. 76, no. 8, pp. 996-1006, August 1988.

[40] D. Ballard, "Animate Vision", *Artificial Intelligence*, vol. 48, no. 1, pp. 1-27, February 1991.

[41] G. Osterberg, "Topography of the layer of rods and cones in the human retina," *Acta Ophthal.*, 6, pp. 1-103, 1935

[42] R. Wallace, P. W. Ong, B. Bederson, E. L. Schwartz, "Space variant image processing," *International Journal of Computer Vision*, 13(1): 71–90, 1994.

[43] G. Sandini and V. Tagliasco, "An anthropomorphic retina-like structure for scene analysis," *CGIP*, vol.14, pp. 365-372, 1980.

[44] M. Tistarelli, and G. Sandini, "On the advantages of polar ad log-polar mapping for direct estimation of time-to-impact from optical flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 401-410, 1993.

[45] C. F. R. Weiman and R. Juday, "Tracking algorithms using log-polar mapped image co-ordinates," *Intelligent Robots and Computer Vision VIII: SPIE Proceedings on Intelligent Robots and Computer Vision*, pp. 843-853, 1998.

[46] R. Manzotti, E. Grosso, R. Tiso, G. Sandini, "A space-variant approach to oculomotor control," *proceedings of IEEE Int. Symposium on Computer Vision*, Coral Gables, Florida, November 1995.

[47] P. M. Blough, "Functional implications of the pigeon's peculiar retinal structure," in Neural Mechanisms of Behavior in the Pigeon, pp. 71–88, New York, NY: Plenum Press, 1979.

[48] D. Claveau and C. Wang, "An Architecture for a VLSI Sensory-motor System for Autonomous Robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, Switzerland.

[49] R. Etienne-Cummings, J. Van der Spiegel, P. Mueller and M.Z. Zhang "A foveated silicon retina for two-dimensional tracking", *IEEE Trans. Circuits and Systems II*, Vol. 47, pp. 504-517, June 2000.

[50] B. E. Stein and M. A. Meredith, *The Merging of the Senses*, Cambridge, MA: The MIT Press, 1993.

[51] D. Claveau and C. Wang, "A space-variant sensor structure for complex target detection and tracking," *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2006 Proceedings*, Volume: 3, Page(s): III-896- III-899, 21-24 May 2006.

[52] D. Claveau and C. Wang, "A spatial variance approach to target tracking with sensor arrays," *IEEE International Symposium on Circuits and Systems, 2005, ISCAS 2005 Proceedings*, Page(s):4759-4762 Vol. 5, May 2005.

[53] A. van der Schaft and H. Schumacher, *An Introduction to Hybrid Dynamical Systems*, Springer-Verlag, London, 2000.

[54] D. Liberzon, *Switching in Systems and Control*, Birkhauser, Boston, 2003.

[55] I. A. Hiskens and M. A. Pai, "Hybrid systems view of power system modelling," in *Proceedings of the IEEE International Symposium on Circuits and Systems,* Geneva, Switzerland, May 2000.

[56] M. H. Raibert, *Legged Robots That Balance,* MIT Press, Cambridge, MA, 1986.

[57] S. Pettersson, "Analysis and design of hybrid systems," Ph.D. Thesis, Department of Signals and Systems, Chalmers University of Technology, Goteborg, Sweden, 1999.

[58] C. Tomlin, G. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A study in multiagent hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 509-521, April 1998.

[59] W. Richard Kolk and Robert A. Lerman, *Nonlinear System Dynamics*, New York : Van Nostrand Reinhold, 1992.

[60] R. W. Brockett and D. Liberzon. "Quantized feedback stabilization of linear systems," *IEEE Transactions on Automatic Control*, 45:1279--1289, July 2000.

[61] D. Simon, Optimal state estimation: Kalman, H-infinity, and nonlinear approaches, John Wiley & Sons, 2006.

[62] Open Source Computer Vision Library,
http://www.intel.com/technology/computing/opencv/

Appendix A

Here we present the complete derivation, as found in [30], of the basic Reichardt EMD's response to a sinusoidal grating with amplitude C, spatial frequency f_x, temporal frequency f_t, and a constant DC component K:

$$I(x,t) = C\cos(2\pi \cdot f_t t - 2\pi \cdot f_x x) + K \tag{A.1}$$

A block diagram of the basic Reichardt EMD is shown again in Fig. A.1.

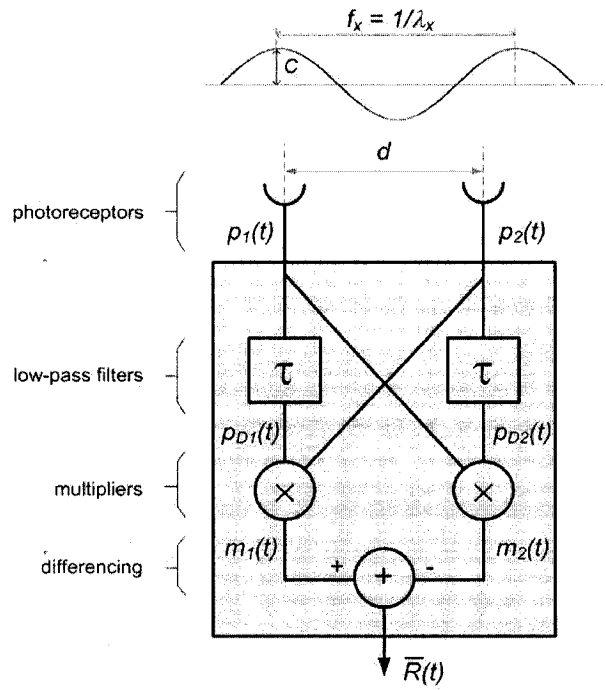


Fig. A.1. Block diagram of a simplified Reichardt EMD.

If the photoreceptors are separated by a distance, d, then the outputs of the photoreceptors are:

$$p_1(x,t) = C\cos(2\pi \cdot f_t t) + K \tag{A.2}$$

$$p_2(x,t) = C\cos(2\pi \cdot f_t t - 2\pi f_x d) + K \tag{A.3}$$

If the delay is achieved with a first-order lowpass filter:

$$D(f_t) = A(f_t)e^{-j\Theta(f_t)} \tag{A.4}$$

then the delayed signals are:

$$p_{D1}(x,t) = C \cdot A(f_t)\cos(2\pi \cdot f_t t - \Theta(f_t)) + K \cdot A(0) \tag{A.5}$$
$$p_{D2}(x,t) = C \cdot A(f_t)\cos(2\pi \cdot f_t t - 2\pi \cdot f_x d - \Theta(f_t)) + K \cdot A(0) \tag{A.6}$$

The output of the multipliers is:

$$m_1(t) = p_{D1}(t) \cdot p_2(t) \tag{A.7}$$
$$m_2(t) = p_{D2}(t) \cdot p_1(t) \tag{A.8}$$

and:

$$R(t) = p_{D1}(t) \cdot p_2(t) - p_{D2}(t) \cdot p_1(t) \tag{A.9}$$

The first term is:

$$\begin{aligned} p_{D1}(t) \cdot p_2(t) &= \left[C \cdot A(f_t)\cos(2\pi f_t t - \Theta(f_t)) + K \cdot A(0)\right] \cdot \left[C \cdot \cos(2\pi f_t t - 2\pi f_x d) + K\right] \\ &= C^2 \cdot A(f_t)\cos(2\pi f_t t - \Theta(f_t))\cos(2\pi f_t t - 2\pi f_x d) + \\ &\quad C \cdot K \cdot A(f_t)\cos(2\pi f_t t - \Theta(f_t)) + \\ &\quad C \cdot K \cdot A(0)\cos(2\pi f_t t - 2\pi f_x d) + \\ &\quad K^2 \cdot A(0) \end{aligned} \tag{A.10}$$

Using the identity:

$$\cos(\alpha)\cos(\beta) = \frac{1}{2}\left[\cos(\alpha + \beta) + \cos(\alpha - \beta)\right] \tag{A.11}$$

equation (7) becomes:

$$\begin{aligned} p_{D1}(t) \cdot p_2(t) &= C^2 \cdot A(f_t)\left[\cos(4\pi f_t t - \Theta(f_t) - 2\pi f_x d) + \cos(2\pi f_x d - \Theta(f_t))\right] + \\ &\quad C \cdot K \cdot A(f_t)\cos(2\pi f_t t - \Theta(f_t)) + \\ &\quad C \cdot K \cdot A(0)\cos(2\pi f_t t - 2\pi f_x d) + \\ &\quad K^2 \cdot A(0) \end{aligned} \tag{A.12}$$

and similarly for the second term of equation (9) we have:

$$\begin{aligned} p_{D2}(t)\cdot p_1(t) &= \left[C\cdot A(f_t)\cos(2\pi f_t t - 2\pi f_x d - \Theta(f_t)) + K\cdot A(0)\right]\cdot\left[C\cdot\cos(2\pi f_t t) + K\right] \\ &= C^2\cdot A(f_t)\cos(2\pi f_t t - 2\pi f_x d - \Theta(f_t))\cos(2\pi f_t t) + \\ &\quad C\cdot K\cdot A(f_t)\cos(2\pi f_t t - 2\pi f_x d - \Theta(f_t)) + \\ &\quad C\cdot K\cdot A(0)\cos(2\pi f_t t) + \\ &\quad K^2\cdot A(0) \\ &= C^2\cdot A(f_t)\left[\cos(4\pi f_t t - \Theta(f_t) - 2\pi f_x d) + \cos(-2\pi f_x d - \Theta(f_t))\right] + \\ &\quad C\cdot K\cdot A(f_t)\cos(2\pi f_t t - 2\pi f_x d - \Theta(f_t)) + \\ &\quad C\cdot K\cdot A(0)\cos(2\pi f_t t) + \\ &\quad K^2\cdot A(0) \end{aligned} \tag{A.13}$$

Combining the two terms yields:

$$\begin{aligned} R(t) &= p_{D1}(t)\cdot p_2(t) - p_{D2}(t)\cdot p_1(t) \\ &= \frac{C^2}{2}\cdot A(f_t)\left[\cos(2\pi f_x d - \Theta(f_t)) - \cos(-2\pi f_x d - \Theta(f_t))\right] + \\ &\quad C\cdot K\cdot A(f_t)\left[\cos(2\pi f_t t - \Theta(f_t)) - \cos(2\pi f_t t - 2\pi f_x d - \Theta(f_t))\right] + \\ &\quad C\cdot K\cdot A(0)\left[\cos(2\pi f_t t - 2\pi f_x d) - \cos(2\pi f_t t)\right] \end{aligned} \tag{A.14}$$

Using the identity:

$$\cos(\alpha+\beta) = \cos(\alpha)\cdot\cos(\beta) - \sin(\alpha)\sin(\beta) \tag{A.15}$$

gives us:

$$
\begin{aligned}
R(t) = \frac{C^2}{2} \cdot A(f_t) \Big[& \cos(2\pi f_x d)\cos(-\Theta(f_t)) - \sin(2\pi f_x d)\sin(-\Theta(f_t)) - \\
& \cos(-2\pi f_x d)\cos(-\Theta(f_t)) + \sin(-2\pi f_x d)\sin(-\Theta(f_t)) \Big] + \\
& C \cdot K \cdot A(f_t)\left[\cos(2\pi f_t t - \Theta(f_t)) - \cos(2\pi f_t t - 2\pi f_x d - \Theta(f_t))\right] + \\
& C \cdot K \cdot A(0)\left[\cos(2\pi f_t t - 2\pi f_x d) - \cos(2\pi f_t t)\right] \\
= & C^2 \cdot A(f_t)\sin(2\pi f_x d)\sin(\Theta(f_t)) + \\
& C \cdot K \cdot A(f_t)\left[\cos(2\pi f_t t - \Theta(f_t)) - \cos(2\pi f_t t - 2\pi f_x d - \Theta(f_t))\right] + \\
& C \cdot K \cdot A(0)\left[\cos(2\pi f_t t - 2\pi f_x d) - \cos(2\pi f_t t)\right]
\end{aligned}
\tag{A.16}
$$

Now, using the identity:

$$
\cos(\alpha) - \cos(\beta) = -2\sin(\frac{\alpha+\beta}{2})\sin(\frac{\alpha-\beta}{2}) \tag{A.17}
$$

gives us:

$$
\begin{aligned}
R(t) = & C^2 \cdot A(f_t)\sin(\Theta(f_t))\sin(2\pi f_x d) + \\
& C \cdot K \cdot A(f_t)\left[-2\sin(2\pi f_t t - \pi f_x d - \Theta(f_t))\sin(\pi f_x d)\right] + \\
& C \cdot K \cdot A(0)\left[-2\sin(2\pi f_t t - \pi f_x d)\sin(-\pi f_x d)\right] \\
= & C^2 \cdot A(f_t)\sin(\Theta(f_t))\sin(2\pi f_x d) + \\
& 2 \cdot C \cdot K \cdot \sin(\pi f_x d)\left[-A(f_t)\sin(2\pi f_t t - \pi f_x d - \Theta(f_t)) + A(0)\sin(2\pi f_t t - \pi f_x d)\right]
\end{aligned}
\tag{A.18}
$$

If the frequency response of the filter is:

$$
D(f_t) = A(f_t)e^{-j\Theta(f_t)} = \frac{1}{\sqrt{1+(2\pi \cdot f_t \tau)^2}} e^{-j\arctan(2\pi \cdot f_t \tau)}, \tag{A.19}
$$

then:

$$
\sin[\Theta(f_t)] = \frac{2\pi f_t \tau}{\sqrt{1+(2\pi \cdot f_t \tau)^2}}, \tag{A.20}
$$

which gives us:

$$R(t) = \frac{C^2}{2\pi\tau}\frac{f_t}{f_t^2 + 1/(2\pi\tau)^2}\sin(2\pi f_x d) + 2 \cdot C \cdot K \sin(\pi f_x d) \cdot \frac{1}{\sqrt{1+(2\pi \cdot f_t \tau)^2}} \sin(2\pi f_t t - \Psi(f_t)) \tag{A.21}$$

Appendix B

Here we present the entire C++ program used in the implementation of the active vision system described in Chapter 5.

```
#include "cv.h"
#include "highgui.h"
#include <iostream>
#include <fstream>
#include <ctype.h>
//#include "ImageIterator.h"
#include "modetect.h"
#include <time.h>

#include <windows.h>

using namespace std;

int GetKeyCode(char c);


bool program_done = false;

float u = 0.0;
float sigma_u = 0.0;
int idle_count = 0;
bool left_fov = true;
bool left_motion = false;

//=========================================================================
// serial port thread
//=========================================================================

HANDLE hSerial;

DWORD WINAPI
SerialPortThread(LPVOID lpParameter)
{
        OVERLAPPED osWrite = {0};
        DWORD dwWritten;
        DWORD dwToWrite = 8;

        while (!program_done)
        {
                // integrate the command, u

                //sigma_u = sigma_u + u;

                // quantize the result

                //int tcode = sigma_u*27;
                int tcode = u*27;

                code = left_motion ? 27 - tcode : 27 + tcode;

                //cout << code << endl;
                //if (code > 48)
```

```
                //          code = 48;
                //else
                //if (code < 10)
                //          code = 10;

                //if (sigma_u == 0.0)
                if (u == 0.0)
                          code = 27;

                // output to serial port

                if (!WriteFile(hSerial, &motor_cmd2[code], dwToWrite, &dwWritten, &osWrite))
                          cout << "serial write failed" << endl;
        }

  return 0;
}


//=============================================================================================
// Motion Detection thread
//=============================================================================================
DWORD WINAPI
MotionDetectionThread(LPVOID lpParameter)
{

        //---- output file -------------------------------------------

        ofstream mfile("C:\\Program Files\\MATLAB71\\work\\mfile.txt");
        ofstream nfile("C:\\Program Files\\MATLAB71\\work\\nfile.txt");

        //---- timer -------------------------------------------------

        LARGE_INTEGER ticksPerSecond, start_ticks, end_ticks, cputime;

        // counter's accuracy
        if (!QueryPerformanceFrequency(&ticksPerSecond))
                  printf("QueryPerformance not present");
        printf ("freq test:   %I64Ld ticks/sec \n\n",ticksPerSecond    );

        //---- OpenCV initialization ---------------------------------

        IplImage *image = 0, *grey = 0, *prev_grey = 0;
  CvCapture* capture = 0;

  capture = cvCaptureFromCAM( 0 );

  cvNamedWindow( "Sensor", CV_WINDOW_AUTOSIZE );
  cvNamedWindow( "EMD Responses", CV_WINDOW_AUTOSIZE );

  IplImage* frame = 0;

  frame = cvQueryFrame( capture );

  image = cvCreateImage( cvGetSize(frame), 8, 3 );
  image->origin = frame->origin;
  grey = cvCreateImage( cvGetSize(frame), 8, 1 );
  prev_grey = cvCreateImage( cvGetSize(frame), 8, 1 );

        CvSize sz = cvGetSize(frame);
        cout << sz.width << ' ' << sz.height << endl;
```

```
        CvSize test_size = {640,256};
IplImage* testimage = cvCreateImage( test_size, 8, 1 );

        //---- foveation array initialization ----------------------

        int full_foveation[NUM_RINGS];
        for (int idx = 0, fidx = 0; idx < NUM_LEVELS; idx++)
                for (int jdx = 0; jdx < foveation[idx].num; jdx++)
                        full_foveation[fidx++] = foveation[idx].res;

        //---- start timer and initialize frame count ----------------

        QueryPerformanceCounter(&start_ticks);
        int count = 0;

        //---- grab frames and process them --------------------------

        unsigned char line_buf[NUM_RINGS];
        unsigned char line_buf_prev[NUM_RINGS];
        float motion_array[NUM_RINGS-1];

        for(;;)
        {
                count++;
  frame = cvQueryFrame( capture );
  if( !frame )
    break;

                cvFlip (frame, NULL, 0);

  cvCopy( frame, image, 0 );
  cvCvtColor( image, grey, CV_BGR2GRAY );

                //---- foveation begin ---------------------------------
                int x = 50*640;
                for (int r = 0; r < 240; r++)
                {
                        int num, res;
                        char* image = grey->imageData;
                        float ave;
                        unsigned long im_total;
                        int buf_idx = 0;
                        for (int i = 0; i < 12; i++)
                        {
                                num = foveation[i].num;
                                res = foveation[i].res;
                                for (int j = 0; j < num; j++)
                                {
                                        ave = 0.0;
                                        im_total = 0;
                                        for (int k = 0; k < res; k++)
                                           im_total = im_total + static_cast<unsigned char>(image[x+k]);
                                        ave = static_cast<float>(im_total)/static_cast<float>(res);
                                        for (int m = 0; m < res; m++)
                                                image[x++] = static_cast<unsigned char>(ave);
                                        if (r == 239)
                                                line_buf[buf_idx++] = static_cast<unsigned char>(ave);
                                }
                        }
                }
```

```
// draw white center lines on image
for (int i = 0; i < 640; i++)
        grey->imageData[x++] = static_cast<unsigned char>(0xFF);

for( int j = 0; j < 20; j++)
        for (i = 0; i < NUM_LEVELS; i++)
        {
                x = x + foveation[i].num * foveation[i].res;
                grey->imageData[x] = static_cast<unsigned char>(0xFF);

        }

x = sz.width/2;
for (i = 0; i < 480; i++)
{
        grey->imageData[x] = static_cast<unsigned char>(0xFF);
        x = x + sz.width;
}

//---- motion detection begin ----------------------------------
if (count > 1)
{
        //---- clear bar chart
        CvPoint ptleft = {0,0};
        CvPoint ptright = {639,255};
        cvRectangle( testimage, ptleft, ptright, CV_RGB(0,0,0), CV_FILLED);

        //---- motion detection proper
        int max_idx = 48;
        float max_motion = 0.0;

        //---- go through EMDs
        for (int i = 0, pos = 0; i < NUM_RINGS-1; i++)
        {
                //---- EMD calculation
                float left_current = static_cast<float>(line_buf[i])/255;
                float left_delayed = static_cast<float>(line_buf_prev[i])/255;
                float right_current = static_cast<float>(line_buf[i+1])/255;
                float right_delayed = static_cast<float>(line_buf_prev[i+1])/255;

                motion_array[i] = right_current*left_delayed - left_current*right_delayed;

                //---- keep maximum response
                if (fabs(motion_array[i]) > fabs(motion_array[max_idx]))
                        max_idx = i;

                //----- draw bar for this EMD
                float motion = motion_array[i]*255;  // motion_array: 0-0.5
                CvPoint pt1 = {pos,127};
                pos += full_foveation[i];
                CvPoint pt2 = {pos,127-motion};
                cvRectangle( testimage, pt1, pt2, CV_RGB(220,220,220), CV_FILLED);
        }

        //---- get pixel and velocity error from table
        left_fov = (max_idx-48)<0 ? true:false;
        left_motion = motion_array[max_idx] > 0 ? false:true;
        int fov_index = left_fov ? abs(max_idx-48)-1:abs(max_idx-48);
        float pixel_error = detectors[fov_index].pos/304.0;     //normalized: 0-1
        float velocity_error = detectors[fov_index].vel/0.6;  //normalized: 0-1
        float Kp = detectors[fov_index].Kp;
        float Kv = detectors[fov_index].Kv;
```

```
//---- threshold
if (motion_array[max_idx] > 0.1 || motion_array[max_idx] < -0.1)
{
        //float u;
        /*
        if ((left_fov && left_motion) || (!left_fov && !left_motion))  // mving out FOV
                //u = Kp*pixel_error + Kv*velocity_error;
                u = Kv*velocity_error;
        else
        if ((left_fov && !left_motion) || (!left_fov && left_motion))  // mving in FOV
                //u = Kp*pixel_error - Kv*velocity_error;
                u = Kv*velocity_error;
        */

        if ( !left_motion )
                if (left_fov)
                        u = Kv*velocity_error*0.6;
                else
                        u = Kv*velocity_error*0.85;

        else
                if (left_fov)
                        u = Kv*velocity_error*0.85;
                else
                        u = Kv*velocity_error*0.6;
                        //u = -Kv*velocity_error*0.02;

        //cout << velocity_error;

        // quantization
        /*
        int tcode = u*27;

        code = left_fov ? 27 - tcode : 27 + tcode;

        if (code > 48)
                code = 48;
        else
        if (code < 10)
                code = 10;
        */

        //---- output pixel error to file
}
else
{
        idle_count++;
        if (idle_count == 20)
        {
                u = 0.0;
                idle_count = 0;
                cout << u << endl;
        }
        //u = 0.0;
        //code = 27;
        //mfile << 0 << endl;
}
        if (left_fov)
                mfile << -detectors[fov_index].pos << endl;
        else
                mfile << detectors[fov_index].pos << endl;
```

```
                                        //nfile << motion_array[max_idx];


                                //cout << code << endl;
                        }

                        for (int lb = 0; lb < NUM_RINGS; lb++)
                                line_buf_prev[lb] = line_buf[lb];

                        //---- show the images ---------------------------------------------

                        cvShowImage( "Sensor", grey );
                        cvShowImage( "EMD Responses", testimage );

                         int c = cvWaitKey(1);
                        //code = GetKeyCode(static_cast<char>(c));
                        if( c == 27 )
                        {
                                program_done = true;
                                break;
                        }
  }

        QueryPerformanceCounter(&end_ticks);
        //-------------------------------------------------------------

        cputime.QuadPart = end_ticks.QuadPart- start_ticks.QuadPart;
        float etime = (float)cputime.QuadPart/(float)ticksPerSecond.QuadPart;
        float fps = float(count)/etime;

        cout << "fps = " << fps << endl;

         cvReleaseCapture( &capture );
         cvDestroyWindow("Sensor");
         cvDestroyWindow("EMD Responses");

        mfile.close();
        nfile.close();


  return 0;
}


//=============================================================================================
// main
//=============================================================================================
void
main( int argc, char** argv )
{

        //---- serial port ------------------------------------------------------

        hSerial = CreateFile("COM1", GENERIC_READ | GENERIC_WRITE, 0, 0,
                                                        OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);

        if(hSerial==INVALID_HANDLE_VALUE)
                if(GetLastError()==ERROR_FILE_NOT_FOUND)
                        cout << "serial port does not exist" << endl;
                else
                        cout << "some other error occurred" << endl;
```

```
        //---- serial port thread -------------------------------------------------

    DWORD dwThreadId1;

    HANDLE hThread1 = CreateThread(
      NULL,      // pointer to security attributes
      0,         // initial thread stack size
      SerialPortThread, // pointer to thread function
      0,         // argument for new thread
      0,         // creation flags (immediate)
      &dwThreadId1 // pointer to receive thread ID
    );

    if (NULL == hThread1) {
      cout << "problem creating thread" << endl;
      exit(1);
    }

        //---- motion detection thread -------------------------------------------

    DWORD dwThreadId2;

    HANDLE hThread2 = CreateThread(
      NULL,      // pointer to security attributes
      0,         // initial thread stack size
      MotionDetectionThread, // pointer to thread function
      0,         // argument for new thread
      0,         // creation flags (immediate)
      &dwThreadId2 // pointer to receive thread ID
    );

    if (NULL == hThread2) {
      cout << "problem creating thread" << endl;
      exit(1);
    }

        //---- wait for motion detection thread to end ----------------------------

        if (WaitForSingleObject(hThread2, INFINITE) != WAIT_OBJECT_0)
        {
         perror("Thread join failed");
        exit(EXIT_FAILURE);
          }

        //---- close everything ---------------------------------------------------

        CloseHandle(hThread1);
        CloseHandle(hThread2);
        CloseHandle(hSerial);
}

//-------------------------

#ifdef WIN32
#pragma warning( disable : 4305)
#endif

const int RES = 640;
const int RES_HALF = RES/2;

int code = 16;
```

```
typedef struct
{
   int num;
   int res;
} Fov;

const int NUM_LEVELS = 12;
const int NUM_RINGS = 96;

Fov foveation[NUM_LEVELS] =
{
            {4,32},
            {4,16},
            {8,8},
            {8,4},
            {8,2},
            {16,1},
            {16,1},
            {8,2},
            {8,4},
            {8,8},
            {4,16},
            {4,32},
};

typedef struct
{
            int pos;
            float vel;
            float Kp;
            float Kv;
} PosVel;

PosVel detectors[NUM_RINGS/2] =
{
            {1,0.1,0.8,0.2},
            {2,0.1,0.8,0.2},
            {3,0.1,0.8,0.2},
            {4,0.1,0.8,0.2},
            {5,0.1,0.8,0.2},
            {6,0.1,0.8,0.2},
            {7,0.1,0.8,0.3},
            {8,0.1,0.8,0.3},
            {9,0.1,0.8,0.3},
            {10,0.1,0.8,0.3},
            {11,0.1,0.8,0.3},
            {12,0.1,0.8,0.3},
            {13,0.1,0.8,0.3},
            {14,0.1,0.8,0.3},
            {15,0.1,0.8,0.4},
            {16,0.1,0.8,0.4}, //--
            {17,0.2,0.7,0.4},
            {19,0.2,0.7,0.4},
            {21,0.2,0.7,0.4},
            {23,0.2,0.7,0.5},
            {25,0.2,0.7,0.5},
            {27,0.2,0.7,0.5},
            {29,0.2,0.7,0.5},
            {31,0.2,0.7,0.5}, //--
            {34,0.3,0.6,0.5},
            {38,0.3,0.6,0.6},
```

```
        {42,0.3,0.6,0.6},
        {46,0.3,0.6,0.6},
        {50,0.3,0.6,0.6},
        {54,0.3,0.6,0.7},
        {58,0.3,0.6,0.7},
        {62,0.3,0.6,0.7}, //--
        {68,0.4,0.5,0.7},
        {76,0.4,0.5,0.7},
        {84,0.4,0.5,0.7},
        {92,0.4,0.5,0.8},
        {100,0.4,0.4,0.8},
        {108,0.4,0.4,0.8},
        {116,0.4,0.4,0.8},
        {124,0.4,0.4,0.8}, //--
        {136,0.5,0.4,0.8},
        {152,0.5,0.4,0.8},
        {168,0.5,0.4,0.8},
        {184,0.5,0.4,0.8}, //--
        {208,0.6,0.3,0.8},
        {240,0.6,0.3,0.8},
        {272,0.6,0.3,0.8},
        {304,0.6,0.3,0.8}

};


__int64 motor_cmd2[57] =
        {
                0x0000000000000000,             //0  -8.57
                0x0000000000000001,             //1  -8.27
                0x0000000000000003,             //2  -7.97
                0x0000000000000007,             //3  -7.67
                0x000000000000000f,             //4  -7.37
                0x000000000000001f,             //5  -7.07
                0x000000000000003f,             //6  -6.77
                0x000000000000007f,             //7  -6.47================ ~ -4.3
                //0x00000000000000FF,
                0x00000000000001FF,             //8  -6.17
                0x00000000000003FF,             //9  -5.87
                0x00000000000007FF,             //10 -5.57
                0x0000000000000fFF,             //11 -5.27
                0x0000000000001fFF,             //12 -4.97
                0x0000000000003fFF,             //13 -4.67
                0x0000000000007fFF,             //14 -4.37
                //0x000000000000FfFF,
                0x000000000001ffFF,             //15 -4.07
                0x000000000003ffFF,             //16 -3.77
                0x000000000007ffFF,             //17 -3.47
                0x00000000000FffFF,     //18 -3.17
                0x00000000001FffFF,             //19 -2.87
                0x00000000003FffFF,             //20 -2.57
                0x00000000007FffFF,             //21 -2.27
                //0x0000000000FFffFF,
                0x0000000001FFffFF,             //22 -1.97
                0x0000000003FFffFF,             //23 -1.67----------
                0x0000000007FFffFF,             //24 -1.37
                0x000000000fFFffFF,             //25 -1.07
                0x000000001fFFffFF,             //26 -0.77
                0x000000003fFFffFF,             //27 -0.47----------
                0x000000007fFFffFF,             //28 -0.17
                //0x00000000ffFFffFF,
```

```
0x00000001ffFFffFF,          //29 +0.13
0x00000003ffFFffFF,          //30 +0.43
0x00000007ffFFffFF,          //31 +0.73<<<<<<<<<<<<<<<<<<<<<<
0x0000000FffFFffFF,          //32 +1.03<<<<<<<<<<<<<<<<<<<<<<
0x0000001FffFFffFF,          //33 +1.33
0x0000003FffFFffFF,          //34 +1.63
0x0000007FffFFffFF,          //35 +1.93
//0x000000FFffFFffFF,
0x000001FFffFFffFF,          //36 +2.23
0x000003FFffFFffFF,          //37 +2.53----------
0x000007FFffFFffFF,          //38 +2.83
0x00000fFFffFFffFF,          //39 +3.13
0x00001fFFffFFffFF,          //40 +3.43
0x00003fFFffFFffFF,          //41 +3.73----------
0x00007fFFffFFffFF,          //42 +4.03
//0x0000ffFFffFFffFF,
0x0001ffFFffFFffFF,          //43 +4.33
0x0003ffFFffFFffFF,          //44 +4.63
0x0007ffFFffFFffFF,          //45 +4.93
0x000FffFFffFFffFF,          //46 +5.23
0x001FffFFffFFffFF,          //47 +5.53
0x003FffFFffFFffFF,          //48 +5.83
0x007FffFFffFFffFF,          //49 +6.13
//0x00FFffFFffFFffFF,
0x01FFffFFffFFffFF,          //50 +6.43
0x03FFffFFffFFffFF,          //51 +6.73
0x07FFffFFffFFffFF,          //52 +7.03
0x0fFFffFFffFFffFF,          //53 +7.33
0x1fFFffFFffFFffFF,          //54 +7.63
0x3fFFffFFffFFffFF,          //55 +7.93
0x7fFFffFFffFFffFF,          //56 +8.23================== ~+4.3
//0xffFFffFFffFFffFF         //64
};
```