Chapter 8

Motion Detection Chips for Robotic Platforms

Rico Moeckel and Shih-Chii Liu

Abstract The on-board requirements for small, light, low-power sensors, and electronics on autonomous micro-aerial vehicles limit the computational power and speed available for processing sensory signals. The sensory processing on these platforms is usually inspired by the sensory information extracted by insects from their world, in particular optic flow. This information is also useful for distance estimation of the vehicle from objects in its path. Custom Very Large Scale Integrated (VLSI) sensor chips which perform focal-plane motion estimation are beneficial for such platforms because of properties including compactness, continuous-time operation, and low-power dissipation. This chapter gives an overview of the various monolithic analog VLSI motion detection/optic flow chips that have been designed over the last 2 decades. We contrast the pros and cons of the different algorithms that have been implemented and we identify promising chip architectures that are suitable for flying platforms.

8.1 Introduction

When considering sensors and electronics for small mobile platforms including microflyers, one has to face the limited power and payload constraints of such platforms. The on-board electronic devices and sensors have to be small, low power, with real-time responses that match the speed of the platform. Custom very large-scale integrated (VLSI) chips which perform focal-plane sensory processing like motion detection match the requirements of these platforms. The chips incorporate networks with a parallel architecture; the transistors are operated in subthreshold leading to low-power dissipation; and the computation is continuous time, thus leading to possibly fast response times. The silicon technology allows a large number of transistors to be placed within a small area, thus leading to small devices in the order of millimeters.

VLSI chips that extract motion information are useful for autonomous mobile platforms that navigate in uncontrolled surroundings. This is especially true of micro-aerial vehicles which depend on optic flow information to avoid obstacles in their world similar to their biological counterparts (insects) where optic flow estimates are extracted by the direction-selective cells in the visual neuropile [19, 10, 27] (see Chap. 4).

Over the past 20 years, various analog VLSI (aVLSI) motion detection chips have been reported in the literature. These chips implement various algorithms; many of them are based on biological models of motion processing, for example, the optic flow estimate extracted by cells in the insect visual pathway [10]. The implementation of these algorithms should be contrasted with machine vision implementations of motion algorithms on many mobile platforms. The latter use primarily outputs of clocked frame-based imagers from which motion is extracted. Either this computation is performed on the on-board microcontroller or the imager data are transmitted to a remote computer on which the motion computation is done. By using an aVLSI motion chip, the microcontroller is freed from this processing and on-board sensory-motor controllers can be considered. The frame-based

R. Moeckel (✉)
Institute of Neuroinformatics, University of Zürich and ETH Zürich, Zürich, Switzerland
e-mail: moeckel@ini.phys.ethz.ch

motion computation methods lead to motion outputs that are only available at discrete sampling times in contrast to the continuous-time analog motion outputs that are available from the analog VLSI motion chips. These chips have been used to guide robotic platforms [17, 37] in a similar spirit to the fly robotic platform setup by Franceschini et al. [13].

In the following sections, we will expound on the various algorithms implemented on the analog VLSI motion chips and the trade-offs that have to be made in these implementations. We briefly touch on three main issues that designers face when considering the aVLSI circuits for the implemented forms – fill-factor, circuit mismatch, and fidelity.

Fill factor: Because a motion pixel includes both the phototransduction circuits and the motion computing circuits, the pixel fill-factor (percentage of area occupied by the phototransduction site in the pixel) of a motion chip depends on the complexity of the algorithm and the subsequent circuits to implement the algorithm.

A complex motion algorithm can result in a pixel with a small fill-factor which then limits the number of pixels that can be placed in a fixed chip area; and a lower spatial sampling frequency of the motion array. The pixel fill-factor of these motion chips can vary between 2.5 and 10%.

Circuit mismatch: Analog circuit designers also have to tackle the effect of transistor mismatch on the motion pixel responses. Unlike computer simulations, any two pixels on the same chip will not produce exactly the same analog output response to the same stimulus. This transistor mismatch phenomenon is due to the fabrication process of the transistors. This mismatch is one source of the observed differences between the outputs of the hardware-implemented algorithm and the software simulation of the same algorithm. For a chip to have a small output variance (low mismatch) across the pixels of the array, the designer has to trade-off the sizes of transistors in analog circuits against the amount of mismatch that can be tolerated [41].

Fidelity: A mathematical operation in an algorithm like a multiplication is not always easily converted in the analog VLSI form. In many cases, the operation is precise only for a limited range of input voltages. Hence the selection of circuits to ensure the closest fidelity to the algorithm is important.

Because of the different circuit techniques used by the individual circuit designers and the fact that the reported motion chips have been fabricated in a variety of processes, we have refrained from making explicit comparisons between the different implementations in Fig. 8.1. We will rather concentrate on a comparison between the implemented motion algorithms and show their corresponding limitations in the implemented aVLSI form.

8.2 Algorithms for aVLSI Motion Detection Chips and Their Implementations

Since the first aVLSI motion chip which was designed for use in the Xerox optical mouse in 1981 [30], more than 20 different motion chip implementations have been published. Many of them were designed with the goal of using them to guide the pilotage of robots in uncontrolled lighting environments. Some major contributions are shown in the timeline chart of Fig. 8.1. This chapter focuses only on aVLSI motion processors that implement both the phototransduction circuits and the motion detection circuits on a single chip. As shown in Fig. 8.2, the algorithms implemented on-chip fall into two primary groups: (1) intensity-based algorithms and (2) token-based algorithms.

(1) **Intensity-based algorithms** estimate optic flow directly from the image brightness. Motion is estimated based on either (a) the gradient of the image brightness or (b) the correlation of the image brightness of neighboring pixels. The image brightness in the implementations could be the equivalent of the output of some pre-processing on the visual input: for example, the absolute intensity, the log intensity, or the output of a retina.

(2) **Token-based algorithms** estimate motion by tracking detected features or tokens across space and time. The term *token* comes from the field of computer vision where tokens are defined either as low-level features like edges or corners or high-level features like objects [47]. On the chips, the token usually consists of a binary pulse which is generated when the contrast of an edge exceeds a threshold. These algorithms can be further divided into two major sub-categories:

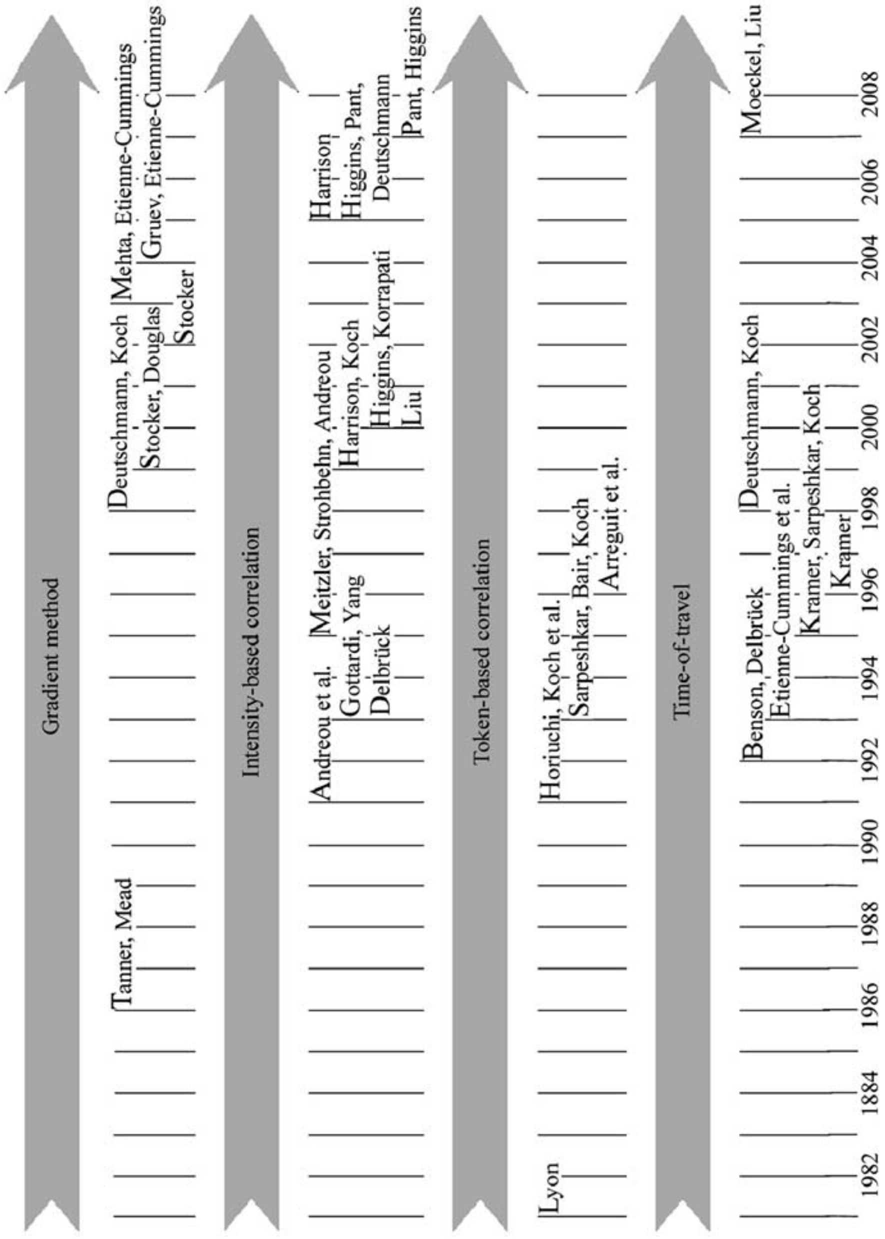

Fig. 8.1 Timeline of single-chip VLSI motion implementations. We show only major contributions of different authors. For example, publications based on one design are represented using the date of the first publication

(a) motion is estimated through the correlation of edge patterns and (b) motion is estimated by measuring the time taken for a feature to travel between two adjacent pixels. The latter is also called *time-of-travel* algorithms.

Intensity-based versus token-based: The advantage of intensity-based algorithms is that the motion computation is continuous whereas token-based algorithms update the motion values only if a predefined feature is detected in the image. The different update dynamics means that the host system which regularly samples the motion values will always receive the instantaneous motion value from an intensity-based motion chip but will most likely receive a motion value that was computed sometime in the past from a token-based motion chip. However, the readout of token-based motion chips could be an advantage if the feature detection signal or token is

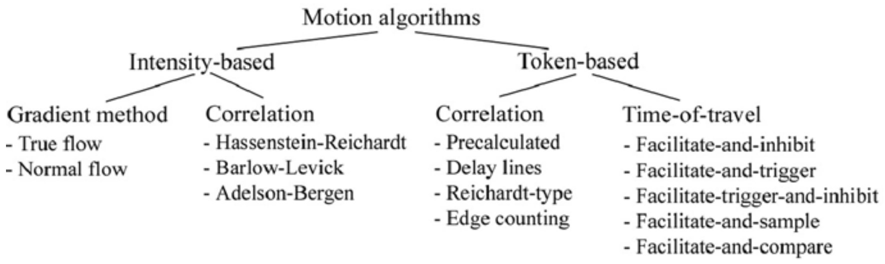

Fig. 8.2 Overview of motion algorithms implemented in aVLSI

used to signal the host system when a new value is computed.

Token-based algorithms can be more robust against noise since only reliable detected features are used to determine the local image speeds. Their disadvantage lies in the fact that the accuracy of the computed motion is determined by the accuracy of the feature detectors. For example, if a feature is detected in a particular pixel but not detected in the neighboring pixel, those algorithms will produce an incorrect motion value. Hence, the interpretation of the motion value by the host system is more challenging. The last updated motion value can either be allowed to decay slowly over time at a predefined rate or is reset after a set time interval.

Token-based algorithms also have the advantage during implementation that the algorithms allow easy partitioning into the phototransduction, feature detection, and motion detection blocks. This partitioning can simplify the design of the subsequent aVLSI circuits since each circuit can be optimized for its particular task.

Aperture problem: Algorithms that detect two-dimensional velocities have to address the inherent ambiguity in determining the correct motion from the measured motions along the two axes at a single pixel which has only a limited field of view or aperture. The local motion values are compatible with a family of possible velocities of an edge through a pixel thus setting up a constraint line in velocity space that has the same orientation as this edge. Through the intersections of the constraint lines from different pixels at the boundaries of a moving object, a unique velocity can be determined for this object.

We describe in some detail the intensity-based algorithms in Sects. 2.1 and 2.2 and the token-based algorithms in Sects. 2.3 and 2.4.

8.2.1 Gradient-Based Intensity Algorithm

The gradient-based algorithm for computing optical flow was one of the first algorithms implemented on an aVLSI chip. This algorithm is also often used in the computer/machine vision community. The optical flow equation is derived from the brightness constancy assumption that the image brightness, I, at a point (x,y) and at a time t stays constant. Taking the derivative of I with respect to time leads to $\frac{d}{dt}I(x,y,t) = 0$. Using the chain rule for differentiation we get an expression that relates the image flow components in the x- and y-directions $\nu_x = dx/dt$ and $\nu_y = dy/dt$ and the partial change of the brightness ∂I:

$$I_x \nu_x + I_y \nu_y + I_t = 0 \tag{8.1}$$

where $I_x = \partial I/\partial x$, $I_y = \partial I/\partial y$, and $I_t = \partial I/\partial t$.

Equation (8.1) is underconstrained since only one independent measure of the image brightness $I(x,y,t)$ at a point in time and space is available while the optical flow velocity has two components ν_x and ν_y. By combining estimated flows spatially across pixels, one can arrive at a unique solution of the equation. Two main implementations of optical flow models have been proposed in the literature:

True flow: Horn and Schunck proposed a smoothness constraint for the optical flow field [23] such that the flow field varies smoothly across space. Using both constraints, the local optical flow vectors $\nu = (\nu_x, \nu_y)$ can be combined to arrive at an optimal solution for the optical flow estimate by minimizing a cost function:

$$\begin{aligned} H_{OF}(\nu) = \sum_{ij} \Big[\left(I_x \nu_x + I_y \nu_y + I_t\right)^2 + \rho \left(\left(\Delta^x \nu_x\right)^2 \right. \\ \left. + \left(\Delta^x \nu_y\right)^2 \right) + \rho \left(\left(\Delta^y \nu_x\right)^2 + \left(\Delta^y \nu_y\right)^2 \right) \Big] \end{aligned} \tag{8.2}$$

where the global constant ρ describes the amount of spatial smoothing and Δ^x and Δ^y represent the discrete derivative operators in the x and y directions, respectively, for each i,j th pixel in the array.

The first optical flow chip that implemented an energy minimization equation was proposed by Tanner and Mead [46] with a subsequent higher performance version described by Stocker and Douglas [44]. The Tanner and Mead chip is one of the first aVLSI motion detection chips and is often cited as the first single-chip motion processor which did not depend on a specific visual pattern environment. The chip contained an array of 8 × 8 pixels and produced a global velocity value over a limited range of input contrasts. It did not implement the smoothness constraint term in Eq. (8.2). The chip by Stocker and Douglas implements this term and provides smooth optical flow values. However, the smoothness constraint term does not take into account object boundaries, thus it blurs the motion values around the object boundaries. To overcome the constant smoothing across the whole chip, a second chip by Stocker [45] implements a network which locally breaks the smoothness constraint at object boundaries. Motion segmentation is included in the computation by minimizing a modified cost function:

$$H_{OF}(v) = \sum_{ij} \Bigg[\left(I_x v_x + I_y v_y + I_t\right)^2 + \rho_{ij}^x \left(\left(\Delta^x v_x\right)^2 + \left(\Delta^x v_y\right)^2 \right) + \rho_{ij}^y \left(\left(\Delta^y v_x\right)^2 + \left(\Delta^y v_y\right)^2 \right) + \sigma \left(\left(v_x - v_{ref}^x\right)^2 + \left(v_y - v_{ref}^y\right)^2 \right) \Bigg] \tag{8.3}$$

where ρ_{ij}^x and ρ_{ij}^y are local variables that set the amount of smoothing. These variables are allowed to adapt locally or are set to zero in the case of an object boundary. The additional bias term in Eq. (8.3) allows components of the local optical flow vectors v_x and v_y to be compared against known motion priors v_{ref}^x and v_{ref}^y, respectively.

The advantage of the combination of the brightness constancy and spatial smoothness contraints lies in the robustness of the algorithm against noisy perturbations in the image. The smoothing operation across local motion outputs can average out this noise.

The minimization of energy equations is attractive for aVLSI implementation because they do not require division circuits and no threshold operation is needed to avoid the zero-division case. However, they do require high-precision multipliers with a wide input linear range as well as linear resistive networks with variable resistance that perform the smoothing operation. Since these circuits are difficult to implement in VLSI, both the implementations in [46, 44] showed a strong dependence of the motion output on image contrast and can only produce reliable measurements with high-contrast stimuli.

Normal optical flow: Yet another way of estimating optical flow is to compute normal flow. This computation is based on the assumption that of all possible flow vectors measured at a pixel, the correct flow is the one that is perpendicular to the edge orientation at each pixel. Circuits for implementing the normal flow model are described in [9, 31, 15].

Mathematically, this constraint is set up as follows:

$$\frac{I_x}{v_x} = \frac{I_y}{v_y}. \tag{8.4}$$

Substituting this constraint equation into Eq. (8.1) leads to

$$v_x = -\frac{I_x I_t}{I_x^2 + I_y^2}, \quad v_y = -\frac{I_y I_t}{I_x^2 + I_y^2}. \tag{8.5}$$

The implementation of these equations in aVLSI circuits is challenging because the denominator in Eq. (8.5) includes partial derivatives of local brightness which are highly dependent on contrast. Especially for low-contrast images one has to divide a small value in the numerator by another small number in the denominator. This division process is very susceptible to noisy estimates of the numerator and the denominator. Many aVLSI motion detection chips include a threshold for I_x or I_y so that the motion is not computed when I_x or I_y are below this threshold. To avoid the zero-division problem, one implementation did not include the denominator term [9]. Although this simplification reduces the complexity of their circuits, the optical flow output of the chip can be ambiguous and is highly dependent on contrast. Implementations of simplified forms of the normal optical flow equations $v_x = -\frac{I_t}{I_x}$, $v_y = -\frac{I_t}{I_y}$ decrease the complexity of the necessary circuits [9, 31]. However, this simplification does not solve the zero-division problem. Another approach is to compute only the local spatial and temporal derivatives I_x, I_y, and I_t and to perform the division off-chip [15].

8.2.2 Intensity-Based Correlation

The intensity-based correlation algorithms are based on spatio-temporal frequency-based models of motion processing in flies [36] and primates [40]. The primary spatio-temporal frequency-based motion algorithms that have been implemented in aVLSI are the *Hassenstein–Reichardt model* [18], the *Barlow–Levick model* [3], and the *Adelson–Bergen motion energy model* [1].

Many implementations of spatio-temporal frequency motion algorithms in single-chip aVLSI systems have been reported, for example, by Andreou et al. [2], Gottardi and Yang [14], Delbrück [6], Meitzler et al. [33], Harrison and Koch [16, 17], Liu [29], and Higgins et al. [20, 21, 35].

Intensity-based correlation algorithms work off the image brightness values unlike token-based algorithms that compute motion from detected features in the image. The motion outputs of the former algorithms have a profile that depends on the spatial and temporal frequency components of the local image patches in the field of view. Although the motion output amplitude is dependent on the square of the contrast of the input signals, these algorithms provide a motion output even for low-contrast inputs that fall below the threshold of a token-based algorithm.

The fact that the outputs of these models are dependent on both contrast and temporal frequencies means that the readout is ambiguous, thus it becomes impossible to determine the speed of a visual patch from a single filter. To solve this ambiguity problem, elaborated versions of these models use a bank of filters with different time constants [38]. These versions use a *place code* where the activation of a filter output codes for the presence of a particular spatio-temporal frequency component in the image patch instead of the value code used for gradient and time-of-travel algorithms. The place code helps, for example, in situations where transparent objects are moving on top of one another, thus allowing several spatio-temporal frequency filters to be activated at the same time. A value code would have to decide for a particular movement or would just give an average response to the movements in the scene.

The place coding of motion using several filter banks is an advantage of spatio-temporal frequency-based motion algorithms but is also a drawback if implemented in aVLSI because of the following reasons: (a) since several filters are needed for each motion pixel, the pixel fill-factor will be greatly reduced and (b) the readout and integration of the filter outputs is non-trivial even without considering mismatch. Mismatch between the filter circuits in different pixels makes the readout very difficult because the time constants of the filters across pixels will not match.

8.2.3 Token-Based Correlation

Token-based correlation algorithms start with an initial step of detecting a particular token or feature in the image. Common to all implementations is the use of a non-linear thresholding circuit for extracting local temporal contrast edges and/or spatial contrast edges. Contrast edges exceeding a threshold produce binary pulses. We describe only two of the many described correlation algorithms here: (a) the correlation is performed between a detected edge at one pixel with a delayed version of this edge at an adjacent pixel with the help of delay lines and (b) the correlation is performed between adjacent pixels using binary correlation filters that were inspired by the original Reichardt correlator.

Delay line correlation: The motion detection chips that perform correlation based on delay lines [22] are inspired by the coincidence detector model in the auditory system of the barn owl. A temporal ON edge triggers a binary pulse of fixed width at each pixel. By propagating the pulses from neighboring pixels through two parallel delay lines from opposite directions the image velocity can be determined from the intersection point of the pulses. This motion chip highlights the following concerns: (a) aVLSI implementations of delay lines are generally costly in silicon area and (b) only a limited range of velocities can be detected for a particular delay setting.

Binary Reichardt correlator: The binary motion correlator chip by Sarpeshkar et al. [39] is inspired by the Hassenstein–Reichardt correlator. Each edge that is detected by a motion pixel triggers a pulse of fixed width. This pulse is correlated with the delayed pulse of the neighboring pixel. The correlation circuit is very simple since the output is determined by the overlap of the two pulses. However, the pulse width determines the range of detected speeds, and the edge detection circuit is not sensitive to low-contrast stimuli.

Similar to the intensity-based correlation algorithms, the performance of the token-based implementations is limited by the fixed time constants of the correlation filters. To detect a wider range of stimulus velocities, filter banks with different time constants or correlation filters with self-adjustable delays would be necessary. These solutions would lead to even less compact designs and lower pixel fill-factors. The clear advantage of token-based over intensity-based correlation algorithms lies in the non-linear edge detection circuits that make them less dependent on contrast because each edge is simply represented by a binary event independent of contrast. However, where token-based correlation algorithms fail because the signal is below the threshold, intensity-based correlation implementations are often still able to report at least the correct direction of motion.

8.2.4 Time-of-Travel

Time-of-travel algorithms directly measure the time taken by a contrast edge to travel between two adjacent pixels. Five major algorithms have been reported in the literature: facilitate-and-inhibit, facilitate-and-trigger, facilitate-trigger-and-inhibit, facilitate-and-sample, and facilitate-and-compare. An overview of the different algorithms is given in Fig. 8.3.

One general advantage of time-of-travel motion detection chips is that they do not require any high-precision dividers, multipliers, and differentiators. Thus they do not suffer from zero-division problems like the gradient algorithms and allow for more compact designs than many correlation algorithms. The time of travel of a contrast edge is usually measured by using voltage or current pulses which are robust against noise pertusbations. Due to the non-linear high-gain stages in the edge detection circuits, the motion outputs are dependent only on the stimulus speed and independent of the stimulus contrast down to very low-contrast values.

All time-of-travel algorithms except the facilitate-and-sample algorithm that uses a log code (Fig. 8.3e) show a linear relationship between the time-of-travel t_{tof} of a contrast edge and its speed. Thus the computed outputs are inversely proportional to the edge velocity v, that is, $v = \frac{\Delta x}{t_{tof}}$, where Δ_x corresponds to the interpixel distance. This inverse relationship means that the sensitivity of the algorithm to different speed ranges is non-uniform. This property is common to all time-of-travel algorithms. It does not necessarily mean that these algorithms are at a disadvantage because the chips can be tuned so that the sensitivity is approximately linear in the expected optical flow range.

Facilitate-and-inhibit: In the facilitate-and-inhibit (FI) algorithm (Fig. 8.3a), an edge detected at a pixel triggers an output pulse which is reset when the edge reaches the neighboring pixel. The width of this pulse linearly increases with the time of travel of the stimulus.

The FI algorithm has its roots in the neural direction selectivity model of the rabbit retina. The first silicon design that implemented this model [4] uses temporal ON contrast edges to trigger pulses that are terminated by inhibition from neighboring photoreceptor pixels. In contrast to the motion chips that are described in the rest of this section, the implementation in [4] did not include a circuit that generates a binary pulse when an edge is detected. The output pulse amplitude of this circuit is dependent on the contrast of the temporal edge, thus making the chip useable only for limited contrast and velocity ranges.

An implementation that uses pulses generated from the detection of a spatial edge using a spatial difference of Gaussian filter was proposed by Etienne-Cummings et. al [11]. The time of travel is determined by measuring the time that an edge took to appear at a pixel and disappear at the neighboring pixel.

Facilitate-and-trigger: In the facilitate-and-trigger (FT) algorithm (Fig. 8.3b) contrast edges trigger pulses of a fixed width. When an edge travels across the image plane it triggers two pulses at neighboring pixels that are delayed by the time of travel. Thus a simple boolean AND operation is sufficient to determine the temporal overlap of the pulses which linearly decreases with the image velocity. The FT implementation by Kramer et al. [26] can detect speeds from 2 to 120 mm/s, measured through image projections onto the chip. However, as shown in Fig. 8.3b the motion output saturates for higher image speeds. The general problem of the FT algorithm lies in the fixed pulse width of the triggered pulses. If the pulse width is too small, slow stimuli cannot be detected since the pulses will not overlap. On the other hand, a large pulse width

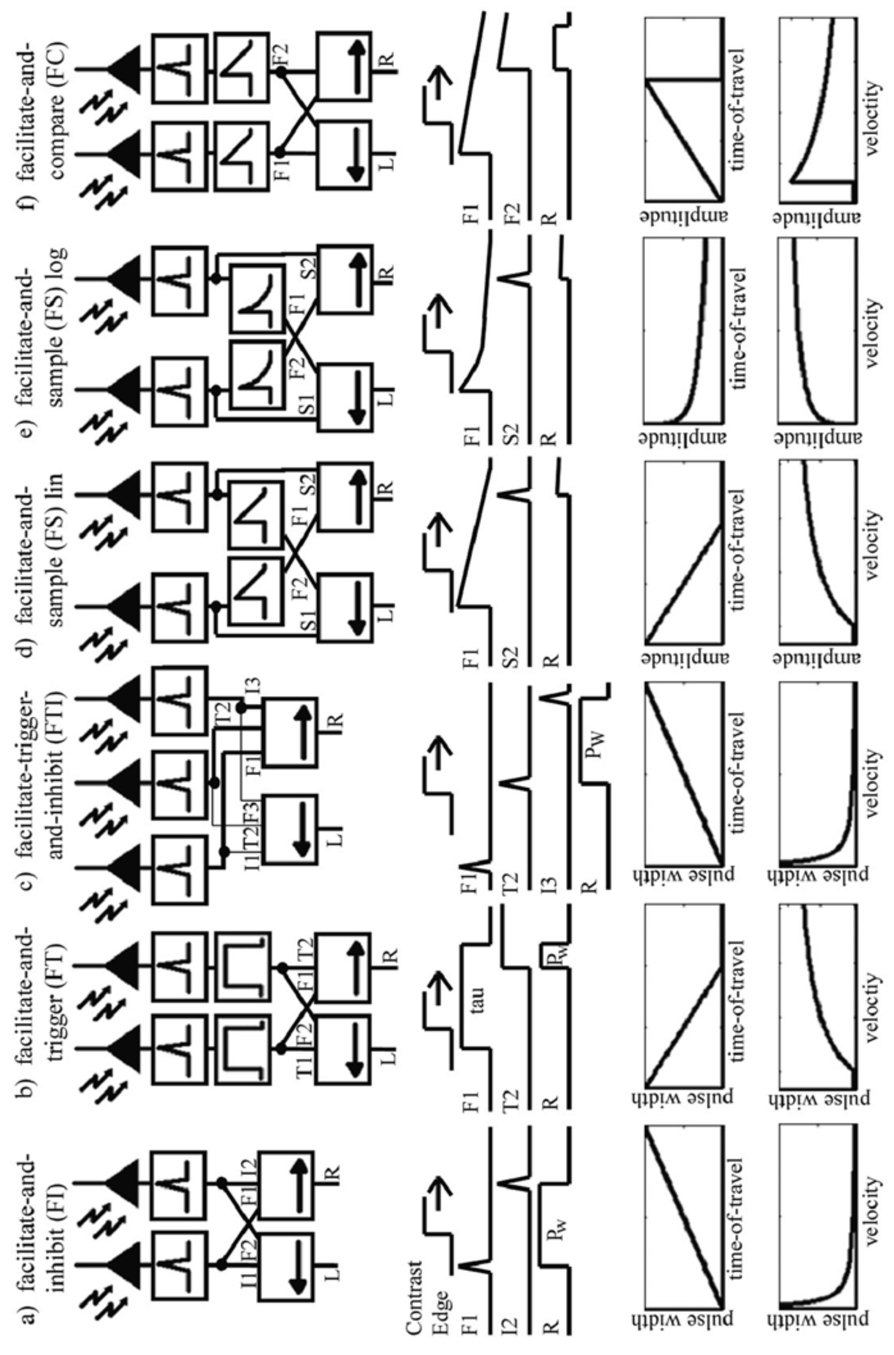
a) facilitate-and-inhibit (FI)
b) facilitate-and-trigger (FT)
c) facilitate-trigger-and-inhibit (FTI)
d) facilitate-and-sample (FS) lin
e) facilitate-and-sample (FS) log
f) facilitate-and-compare (FC)
Contrast Edge
F1
I2
R
P_W
tau
T2
I3
S2
F2
L
time-of-travel
velocity
pulse width
amplitude

Fig. 8.3 Comparison between time-of-travel motion algorithms. We show the block diagrams of circuitry (*top*), the time traces for the internal signals, and the motion output that prefers a stimulus traveling from a photoreceptor at the left to its right neighbor (*middle*) as well as the dependence of the output signal on the time of travel and the velocity of the traveling stimulus (*bottom*). Note that while the FI, FT, and FTI algorithms' use encode the stimulus velocity in the width of an output pulse the FS and FC algorithms output a value code. F: facilitate, T: trigger, I: inhibit, L: motion output that prefers traveling edges from right to the left, R: motion output that prefers traveling edges from the left to the right, P_w: pulse width, tau: time constant of pulse with fixed width

can lead to an output that constantly remains "high," for example, when pulses are triggered periodically due to a highly textured and fast stimulus. This means that no output pulse is detected by the system.

Facilitate-trigger-and-inhibit: To overcome the fixed pulse width limitation in the FT algorithm, Kramer introduced the facilitate-trigger-and-inhibit (FTI) [24]. The FTI algorithm (Fig. 8.3c) integrates signals from three adjacent pixels and outputs a binary pulse whose width is inversely related to the velocity of contrast edges traveling across the image plane. A contrast edge moving from left to right causes the edge detection circuit at pixel 1 to output a pulse, called the facilitate signal F1. The signal F1 enables the motion detection circuit to generate the trigger signal T2 that is caused by the same edge traveling across pixel 2. The signal T2 causes the motion circuit to output a pulse R that lasts until the edge reaches pixel 3. Once the edge is detected by pixel 3, it sends an inhibition signal I3 which terminates the output pulse R. The detectable image speed reported in [24] ranges between 0.034 and 60 mm/s. For velocities above 60 mm/s the circuit also shows a motion output in the non-preferred direction due to the limited rise time of the inhibition signal.

Facilitate-and-sample: Facilitate-and-sample (FS) aVLSI motion circuits first described by Kramer et al. [25, 26] use (Fig. 8.3d,e) a contrast edge detected by a pixel to trigger both a small sampling pulse (S1, S2) and a facilitate signal (F1, F2) that logarithmically decay over time. The time to travel across two pixels is determined by the sampled value of F1 by S2 for stimuli moving from left to right and by the sampled value of F2 by S1 for stimuli moving in the opposite direction. A similar token-based algorithm implemented using discrete components was reported earlier by Franceschini et al. [5, 12].

The FS motion detection chip of Kramer et al. uses a logarithmic encoding for the time of travel (Fig. 8.3e). This encoding allows the motion circuit to represent a time-of-travel range of more than seven orders of magnitude for fixed time constants. This chip performance is the best reported in the literature so far. However, these motion measurements were done by using electronically generated pulses which bypassed the outputs of the pixel edge detection circuits. In addition, due to the highly compressed encoding of the motion output, where a change of 200 mV in the output corresponds to one order of change in the time of travel of an edge, 20 mV of noise will lead to a relative error of 10% in the decoding of the time of travel.

The FS algorithm is advantageous over the FI, FT, and FTI algorithms because its motion output can be held on a capacitor for a few milliseconds, while the latter algorithms produce temporal-coded motion outputs that need to be constantly monitored.

Facilitate-and-compare: The facilitate-and-compare (FC) algorithm is inspired by the FS algorithm [8]. The implemented algorithm (Fig. 8.3f) does not require sample-and-hold circuits to store the sampled value of the facilitate signals. The motion output is computed continuously by subtracting the linearly decaying facilitate signals from adjacent pixels. This difference codes the speed of the stimulus. This algorithm leads to less complexity for the FC circuits but the implemented circuits suffer from the trade-off between high sensitivity settings (the facilitate signals will decay quickly to ground) and from mismatch (the motion output varies over time if the decay rates of the facilitate signals are different). The FC implementation in [8] can measure image velocities in the range from 0.3 to 40 mm/s.

8.3 Promising Motion Chip Architectures

In our opinion, the most interesting motion detection chips so far are the normal flow implementations [32], the gradient-based implementations [44], and our time-of-travel implementation [34].

The **normal optic flow implementations** [31, 32] make use of front-end APS pixels (similar to pixels in imagers) for sensing. These clock-based APS circuits allow one to sequentially read out the local intensity values of a row of pixels over a global data bus. The computation of the motion values is done on the periphery of the APS array, so very dense two-dimensional motion arrays can be achieved. The authors report an array size of 120 $\times$ 80 APS pixels, a fill-factor of 10%, and an equivalent frame rate of 20 frames per second with a power consumption of 34.5 mW. Note that the normal flow estimation in this implementation is done in a line-sequential way. The mismatch between the outputs of the different APS pixels (also called fixed pattern noise in imaging literature) is removed by the correlated double sampling (CDS) technique and since the motion values are computed sequentially row by row, the circuits that perform the motion computation can be shared thus eliminating another source of mismatch.

However, this implementation has drawbacks that are caused in part by the APS implementation and by the normal flow algorithm itself. Since the operation of the APS requires a clock, this implementation is susceptible to temporal aliasing unlike the continuous-time motion detection chips. In addition, the sequential readout of the APS array for computing the motion can be a bottleneck and prevents the detection of fast moving stimuli.

A general problem with the normal optic flow implementation is the need for high-precision division circuits whose output is imprecise for low-contrast stimuli and especially in the presence of mismatch and noise. These chips cannot produce reliable measurements in low-contrast environments and a thresholding circuit is needed to eliminate low-contrast signals from the flow computation. Moreover, APS pixels do not have local gain control properties like the adaptive photoreceptor circuit [7] and the one with selectable adaptation time constants [28] which allow circuit operation over a wide range of background lighting conditions. These photoreceptor circuits are widely used as the front end in many motion detection chips including the remaining two implementations to be described in this section.

The *gradient-based motion implementation* in [45] provides local motion values and uses the spatial integration of the local values in addressing the aperture problem. It allows the local adaptation of the smoothing across motion pixels, thus supporting motion segmentation. However, the need for linear multiplication circuits makes the chip vulnerable to noise and mismatch. Due to the strong dependence of the motion output on contrast for values below 50% the chip is not usable in low-contrast environments. The smoothing and multiplication operation circuits also require a lot of transistors leading to a reported pixel fill-factor of 4%. However, the smoothing and segmentation properties on the motion computation makes this an interesting network with cooperation and competition properties.

In our *time-to-travel implementation*, we combined the FS algorithm together with a novel implementation of a front-end two-threshold edge detection circuit that allows the robust detection of features with contrast down to 2.5%. In contrast to the logarithmic encoding of the motion output used by Kramer et al., we use a linear encoding for the motion values [34]. This linear code is obtained by letting the facilitate signals F1 and F2 decay linearly over time (Fig. 8.3d). Figure 8.4a shows the measured motion output for all 24 motion pixels in response to a drifting sinusoidal grating with a contrast C of 10% using a passive LCD display. The stimulus contrast is defined as $C = \frac{I_{max}-I_{min}}{I_{max}+I_{min}} * 100\%$. As predicted theoretically, the linear encoding of the speeds leads to a saturation of the signal for high stimulus speeds. The chip is sensitive for speed values in a range of approximately two orders of magnitude (approximately 0.1–10 mm/s) for a given time constant. The global decay rate of the facilitate-signals can be adjusted in real time, for example, through an off-chip controller to match the motion speeds in the image. Figure 8.4a also shows the mismatch profile of the motion detection pixels. This mismatch is generated during the fabrication of the chip. However, the offset and decay rate differences in the motion output across pixels due to this mismatch can be easily removed off-chip.

As mentioned above, this implementation uses the adaptive photoreceptor circuit by Delbrück and

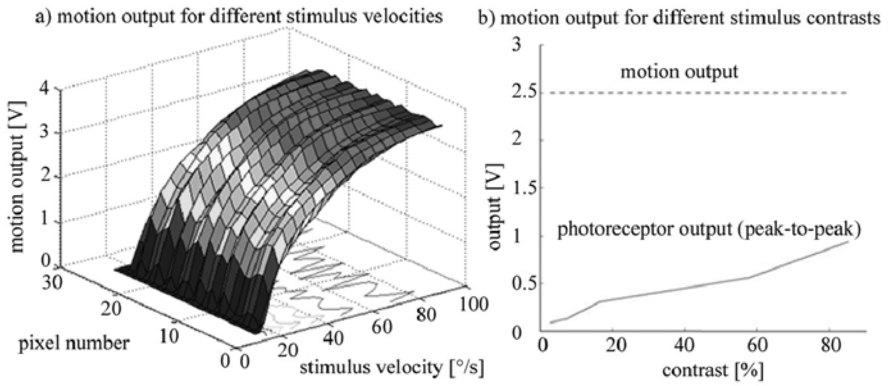

Fig. 8.4 Motion output of our facilitate-and-sample motion detection chip [34]. (**a**) Average motion output for all 24 motion pixels for different stimulus velocities for a particular setting of the decay rate of the facilitate signal. ($100°/s$ is equivalent to 10 mm/s for a 6 mm focal length lens.) For this particular decay rate, the range of velocities detected is less than the maximum possible range of two orders achievable with a slower decay rate. The motion outputs across the array for a set of fixed stimulus velocities (see projected lines on the x – y plane) show the typical sinusoidal mismatch pattern caused by the aVLSI fabrication process. (**b**) Motion output for different contrast values of the stimuli for a single pixel. While the peak-to-peak value of the photoreceptor output decreases for lower contrast stimuli, the average motion output remains constant for motion values down to 2.5%

Mead [7]. This circuit adapts locally over 6 decades of background light intensity, thus allowing a reliable coding of the motion outputs under conditions ranging from moonlight to sunlight. This adaptation property is of great benefit when the chip is operated in natural surroundings where there could be a wide spatial distribution of light intensities, for example, when sunlight streams into part of a room and the remaining areas are in shadow.

To operate reliably in low-contrast environments and to provide a motion output that is invariant to a large range of spatial and temporal frequencies, we combined the circuits originally proposed by Liu [29] to model the laminar monopolar cell (LMC) in the fly visual system together with a novel two-threshold edge detection circuit. Instead of a single threshold, where a signal above this threshold is considered as a contrast edge, we require that the signal moves across two thresholds which are about 100 mV apart, thus ensuring that contrast edges are not randomly generated in the case of a noisy signal that is riding around threshold. The output of the LMC circuit which is a high-gain, high-pass filtered version of the photoreceptor output is compared against two adjustable thresholds. By adjusting these thresholds, we are able to reliably detect low-contrast edges and to filter out noise from, for example, light flicker. The adjustable distance between the thresholds allows us to trade-off between the detection of low-contrast stimuli and the filtering out of noise. A small distance between the thresholds will support the detection of low-contrast stimuli but is more affected by noise while a bigger distance will result in a system that is more robust against noise but is less sensitive to low-contrast stimuli. In our measurements, we placed the thresholds just far enough apart to ensure that noise alone does not trigger an output pulse at the edge detection circuit.

Figure 8.4b shows both the peak-to-peak photoreceptor output and the motion output of a single motion pixel in response to a sinusoidal drifting grating of constant speed and different contrast values presented on an LCD screen. While the peak-to-peak photoreceptor output monotonically decreases with lower stimulus contrast the motion output stays constant. Due to the limitations of the LCD screen, we could not reliably decrease the stimulus contrast below 2.5%.

The chip provides real-time computation of one-dimensional optical flow, and has high temporal and spatial resolutions. The printed circuit board holding this chip weighs less than a gram and consumes only a few milliwatts of power. Similar to many of the motion chip implementations, a drawback of the current implementation is that it is a linear array which means that the extracted motion values are susceptible to the aperture problem. However, this one-dimensional optical flow can be sufficient for steering a micro-aerial vehicle [48].

8.4 Summary

Over the last 2 decades, more than 20 different aVLSI monolithic motion detection chips have been reported. The implemented algorithms on these chips can be divided into more than 14 sub-categories. Because of the wide range of algorithms and implementations, an unbiased comparison of the best motion chip to use on a robotic platform is difficult. However, there are several things that one can consider when designing or using these chips as a sensor on a mobile robot: (a) to operate in natural environments and without needing high-contrast textures, the chip should respond to low-contrast stimuli reliably and unambiguously; (b) the motion chip outputs should have low mismatch or the mismatch must at least be easily removed in the readout, for example, through calibration; (c) the chip should operate reliably in the presence of noise, for example, noise on the power supply lines as well as noise in the visual stimuli like light flicker; (d) the pixel design should be compact and the design should support a large two-dimensional spatial array; and (e) finally, the chip should work robustly under different background light intensities.

To decide on a particular motion algorithm and its subsequent implementation, one has to clearly identify the advantages and disadvantages that come from the motion algorithm itself and those that come from the VLSI circuit implementations. For example most of the motion detection chips reported so far show a high dependence of the motion output on the stimulus contrast. While this *contrast sensitivity* can be easily explained by the algorithm itself for intensity-based correlation implementations, there is no theoretical reason why an FT implementation should respond more ambiguosly to low-contrast features than an FS implementation. In this case, the ambiguous response of the FT motion chip lies in the implementation of the edge detection circuits.

The *compactness* of a motion detection chip depends on both the underlying algorithm and the implementation. For most analog implementations a trade-off has to be made between design compactness and mismatch. By increasing the area of transistors the matching can be improved because the circuits are less affected by noise in the fabrication process. On the other hand an increase in transistor area will lead to a reduction of the pixel fill-factor and a less compact design. In general it is very difficult to compare the circuit compactness of different motion detection chips. In the literature specifications of the fill-factor, pixel area and transistor count of a design are usually used for comparison. However, these numbers have to be taken with great care since the transistor sizes can vary greatly between different implementations not only because of matching considerations but also because the chips were fabricated in different processes. The comparison of the designs is more meaningful if the chips are fabricated in the same process.

A better measure for the comparison of compactness might be the number of circuit elements that are needed for the implementation. However, even the number of transistors and capacitors can only give a rough basis for comparison since the size of the transistors varies considerably depending on how the transistors are operated, that is, whether they are "digital" or "analog" transistors; and the capacitors greatly vary in size depending on the required time constant and the process by which they are fabricated. Furthermore, there are a number of circuits that can implement the same basic function but with different dynamic range for the input and/or output. As an example, a circuit designer can implement a voltage buffer as a 2-transistor source follower circuit, a 5-transistor operational amplifier circuit, or a 10-transistor wide range linear amplifier which has the largest input linear range of the three possibilities.

A comparison of the *mismatch* characteristics in the motion outputs of different chips in the literature is often not possible because these data are usually not available. Hence, one can only speculate on the performance of a chip in the presence of fixed pattern noise. In general, one prefers motion chip implementations with low mismatch which can be achieved by using, for example, on-chip compensation techniques. Another approach around mismatch is to use circuits where the mismatch is only seen as an offset or gain shift in the output transfer function of the motion circuits. This offset and gain mismatch can then be easily compensated off-chip.

To facilitate *robustness to noise*, a circuit designer has to consider the possible different noise sources. For example, we use the low-pass characteristics of the photoreceptor circuit to remove lighting flicker noise. The presence of noise also affects the performance of motion algorithms that require high-precision analog computations like division, multiplication, and taking derivatives. Implementations that compute motion based on pulses, like time-of-travel algorithms and token-based correlation algorithms, have shown good robustness against noise. Based on the considerations discussed above, we identified the three promising monolithic motion chip architectures for flying platforms in Sect. 8.3.

Although we have focused only on the monolithic chip implementations in this chapter, some of the motion algorithms themselves can also be implemented using a two-chip system consisting of an imager and a processor albeit at the cost of speed and weight, for example, systems on the robotic platforms in Chaps. 3, 5, 6, and 9. As mentioned before, many of the implemented chips so far usually have a linear architecture which makes the motion outputs susceptible to the aperture problem. Future solutions like the use of three-dimensional technology, a two-chip system, or implementations of alternate methods like the image interpolation methods proposed by Srinivasan [42, 43] can help to relieve this problem. We believe that the aVLSI motion chips have important properties that allow them to provide useful optic flow information even under low-contrast and noisy conditions for the navigation of autonomous flying vehicles. The technology for the circuits is improving fast and we will likely soon see these chips used on micro-aerial vehicles leading to a new generation of small autonomously flying robots with vision.

References

1. Adelson, E., Bergen, J.: Spatio-temporal energy models for the perception of motion. Journal of the Optical Society of America A **2**, 284–299 (1985)
2. Andreou, A., Strohbehn, K., Jenkins, R.: Silicon retina for motion computation. IEEE International Symposium on Circuits and Systems pp. 1373–1376 (1991)

3. Barlow, H.B., Levick, W.R.: The mechanism of directionally selective units in rabbit's retina. Journal of Physiology **178**, 477–504 (1965)
4. Benson, R., Delbruck, T.: Direction selective silicon retina that uses null inhibition. Advances in Neural Information Processing Systems **4**, 756–763 (1992)
5. Blanes, C.: Appareil visuel pour la navigation vue d'un robot mobile. Ms thesis in neuroscience, Univ. of Aix-Marseille II, Marseille, France (1986)
6. Debruck, T.: Silicon retina with correlation-based, velocity tuned pixels. IEEE Transactions on Neural Networks **4**, 529–541 (1993)
7. Delbruck, T., Mead, C.: Adaptive photoreceptor circuit with wide dynamic range. IEEE International Symposium on Circuits and Systems **IV**, 339–342 (1994)
8. Deutschmann, R.A., Koch, C.: Compact analog VLSI 2-D velocity sensor. IEEE International Conference on Intelligent Vehicles **1**, 359–364 (1998)
9. Deutschmann, R.A., Koch, C.: Compact real-time 2-D gradient based analog VLSI motion sensor. Proceedings of SPIE International Conference on Advanced Focal Plan Array and Electronic Cameras II **3410**, 98–108 (1998)
10. Egelhaaf, M., Borst, A.: Motion computation and visual orientation in flies. Comparative Biochemistry and Physiology **104A**, 659–673 (1993)
11. Etienne-Cummings, R., Fernando, S., Takahashi, N., Shtonov, V., Van der Spiegel, J., Mueller, P.: A new temporal domain optical flow measurement technique for focal plane VLSI implementation. IEEE Proceedings on Computer Architectures for Machine Perception pp. 241–250 (1993)
12. Franceschini, N., Blanes, C., Oufar, L.: Appareil de mesure passif et sans contact de la vitesse d'un objet quelconque. Dossier technique Nb 51549, ANVAR/DVAR, Paris (1986)
13. Franceschini, N., Pichon, J.M., Blanes, C.: From insect vision to robot vision. Philosophical Transactions of the Royal Society of London. Series B **337**, 238–294 (1992)
14. Gottardi, M., Yang, W.: A CCD/CMOS image motion sensor. International Solid State Circuits Conference pp. 194–195 (1993)
15. Gruev, V., Etienne-Cummings, R.: Active pixel sensor with on-chip normal flow computation on the read out. IEEE International Conference on Electronics, Circuits, and Systems pp. 215–218 (2004)
16. Harrison, R.R.: A biologically-inspired analog IC for visual collision detection. IEEE Transactions on Circuits and Systems I **52**, 2308–2318 (2005)
17. Harrison, R.R., Koch, C.: A robust analog VLSI motion sensor. Autonomous Robots **7**, 211–224 (1999)
18. Hassenstein, B., Reichardt, W.: Systemtheoretische Analyse der Zeit-, Reihenfolgen- und Vorzeichenauswertung bei der Bewegungsperzeption des Russelkafers Chlorophanus. Zeitschrift fur Naturforschung **11b**, 513–524 (1956)
19. Hausen, K.: Motion sensitive interneurons in the optomotor system of the fly. II. The horizontal cells – Receptive-field organization and response characteristics. Biological Cybernetics **46**(1), 67–79 (1982)
20. Higgins, C., Korrapti, S.: An analog VLSI motion energy sensor based on the Adelson-Bergen algorithm. International ICSC Symposium on Biologically-Inspired Systems (2000)
21. Higgins, C.M., Pant, V., Deutschmann, R.: Analog VLSI implementation of spatio-temporal frequency tuned visual motion algorithms. IEEE Transactions on Circuits and Systems – I **52**(3), 489–502 (2005)
22. Horiuchi, T., Lazzaro, J., Moore, A., Koch, C.: A delay line based motion detection chip. Advances in Neural Information Processing Systems **3**, 406–412 (1991)
23. Horn, B., Schunk, B.: Determining optical flow. Artificial Intelligence **17**, 185–203 (1981)
24. Kramer, J.: Compact integrated motion sensor with three-pixel interaction. IEEE Transactions on Pattern Analysis and Machine Intelligence **18**, 455–460 (1996)
25. Kramer, J., Sarpeshkar, R., Koch, C.: An analog VLSI velocity sensor. IEEE International Symposium on Circuits and Systems pp. 413–416 (1995)
26. Kramer, J., Sarpeshkar, R., Koch, C.: Pulse-based analog VLSI velocity sensors. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing **44 (2)**, 86–101 (1997)
27. Krapp, H., Hengstenberg, R.: Estimation of self-motion by optic flow processing in single visual interneurons. Letters to Nature **384**, 463–466 (1996)
28. Liu, S.C.: Silicon photoreceptors with controllable adaptive filtering properties. In: C. G, M. Bayoumi (eds.) Learning on Silicon, pp. 67–78. Kluwer Academic Publishers, Norwell, MA (1999)
29. Liu, S.C.: A neuromorphic aVLSI model of global motion processing in the fly. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing **47**(12), 1458–1467 (2000)
30. Lyon, R.: The optical mouse, and an architectural methodology for smart digital sensors. CMU Conference on VLSI Structures and Computations pp. 1–19 (1981)
31. Mehta, S., Etienne-Cummings, R.: Normal optical flow chip. IEEE International Symposium on Circuits and Systems **4**, 784–787 (2003)
32. Mehta, S., Etienne-Cummings, R.: A simplified normal optical flow CMOS camera. IEEE Transactions on Circuits and Systems I **53**(6), 1223–1234 (2006)
33. Meitzler, R.C., Strohbehn, K., Andreou, A.G.: A silicon retina for 2-D position and motion computation. IEEE International Symposium on Circuits and Systems **III**, 2096–2099 (1995)
34. Moeckel, R., Liu, S.C.: Motion detection circuits for a time-to-travel algorithm. IEEE International Symposium on Circuits and Systems pp. 3079–3082 (2007)
35. Pant, V., Higgins, C.M.: A biomimetic focal plane speed computation architecture. Proceedings of the Computational Optical Sensing and Imaging Conference (2007)
36. Reichardt, W.: Autocorrelation, a principle for the evaluation of sensory information by the central nervous system. Sensory Communication pp. 303–317 (1961)
37. Reichel, L., Liechti, D., Presser, K., Liu, S.C.: Range estimation on a robot using neuromorphic motion sensors. Robotics and Autonomous Systems **51**(2–3), 167–174 (2005)
38. Santen, v.J., Sperling, G.: A temporal covariance model of motion perception. Journal of the Optical Society of America A **1**, 451–473 (1984)

39. Sarpeshkar, R., Bair, W., Koch, C.: Visual motion computation in analog VLSI using pulses. Advances in Neural Information Processing Systems **5**, 781–788 (1993)
40. Schiller, P.H., Finlay, B.L., Volman, S.F.: Qualitative studies of single-cell properties in monkey striate cortex. I. Spatiotemporal organization of receptive fields. Journal of Neurophysiology **39**, 1288–1319 (1976)
41. Serrano-Gotarredona, T., Linares-Barranco, B.: A new 5-parameter MOS transistor mismatch model. IEEE Electron Device Letters **21**(1), 37–39 (2000)
42. Srinivasan, M.V.: Generalized gradient schemes for the measurement of two dimensional image motion. Biological Cybernetics **63**, 421–431 (1990)
43. Srinivasan, M.V.: An image-interpolation technique for the computation of optic flow and egomotion. Biological Cybernetics **71**, 401–416 (1994)
44. Stocker, A., Douglas, R.: Computation of smooth optical flow in a feedback connected analog network. Advances in Neural Information Processing Systems **11**, 706–712 (1999)
45. Stocker, A.A.: An improved 2D optical flow sensor for motion segmentation. IEEE International Symposium on Circuits and Systems **2**, 332–335 (2002)
46. Tanner, J., Mead, C.: An integrated analog optical motion sensor. VLSI Signal Processing **2**, 59–76 (1986)
47. Ullman, S.: Analysis of visual motion by biological and computer systems. IEEE Computer **14**, 57–69 (1981)
48. Zufferey, J., Klaptocz, A., Beyeler, A., Nicoud, J., Floreano, D.: A 10-gram microflyer for vision-based indoor navigation. IEEE/RSJ International Conference on Intelligent Robots and Systems pp. 3267–3272 (2006)