A Low-Power Visual-Horizon Estimation Chip

Timothy K. Horiuchi, *Member, IEEE*

***Abstract*—Recent successes in the construction of micro aerial vehicles ($<$ 15 cm) have highlighted the lack of real-time sensors for flight control. This paper describes a low-power real-time visual-horizon sensor for stabilizing the pitch and roll of miniature aircraft in moderate-to-high-altitude flight. This prototype sensor uses a 12 $\times$ 12 photoreceptor array to find a best-fit horizon line based on image intensity. The sensor includes a "confidence-level" output for signaling poor sensing conditions and can scan out the image. The chip was fabricated in a commercially available 0.5-μm CMOS process and operates on less than 2.5 mW with a 5-V power supply.**

***Index Terms*—Analog VLSI, autonomous flight control, micro aerial vehicles, perceptron, smart sensor.**

I. Introduction

UNMANNED micro aerial vehicles are rapidly being developed for use as a low-cost portable aerial surveillance platform for semiautonomous operation. While they are successfully achieving flight, the sensors needed for autonomous flight (in contrast to long-range navigation) are lacking. Obstacle avoidance and the control of basic flight parameters such as altitude, roll, and pitch remain a problem for such small vehicles with tiny weight and power budgets. Their small size makes them particularly susceptible to tiny wind gusts, making the speed of processing critical for stability.

While many visual-motion approaches to stabilizing aerial vehicles with low-power custom vision chips and systems are in development (e.g., [1]–[6]), these approaches do not provide information about the pitch or roll angle directly. Accelerometers can be used for pitch and roll rates but typically suffer errors in low-amplitude regimes and high-frequency vibration environments and cannot give estimates of altitude. For these reasons, real-time detection of the visual horizon may be desirable for the stabilization of pitch and roll of micro aerial vehicles at medium-to-high altitude where the horizon is likely to be the true horizon and not the shadow of a mountain or building (Fig. 1). The direct measurement of the pitch and roll angle can be used to control banking turns in flight, control altitude changes, or stabilize the aircraft against low-frequency perturbations or drift.

Several low-sensor-count horizon sensing systems have been developed for assisting model aircraft pilots that utilize contrast in the infrared spectrum [7] or visible light [8]; however, these do not provide indications of poor sensing conditions.

Fig. 1. Visual-horizon detection from relatively high altitude can provide valuable sensory data for flight stabilization of micro aerial vehicles, from early dawn to well after dusk.

In recent years, a team from the University of Florida (UF), Gainesville, has demonstrated an automatic visual-horizon-finding algorithm operating on a high-speed computer on the ground that receives a transmitted color video feed from the airplane [9]. In this algorithm, a search is conducted to find a horizon line that best splits the image into two regions (sky and ground) whose pixels are "most like each other." The pixels within each region form a cluster point in three dimensions (red–green–blue or RGB) with a certain mean and variance. By finding the line that minimizes the variance for both clusters, a good horizon solution is found. This approach obviously depends upon the assumption that the sky and ground are mostly different in color. In their implementation, a coarse-to-fine search of the entire parameter space was conducted on each frame of the video. The closed-loop system performance was demonstrated to be more stable than when a trained operator viewed the video feed.

Although the UF algorithm used a broad parameter search to find the horizon anew in each video frame (to avoid problems with video transmission dropouts), we developed a similar algorithm that uses a gradient-descent optimization approach embedded in an continuous-time analog VLSI vision chip to find the best estimate of the visual horizon while providing a measure of confidence signal. The VLSI implementation has the potential for real-time low-power performance and operation over a very wide dynamic range of image intensities.

II. Horizon Detection Algorithm

A. Horizon Vector h

Consider an image where each pixel is assigned a horizontal and vertical coordinate (x, y) with the origin in the center of the image [Fig. 2 (left)]. We can think of this coordinate as the pixel vector p^{μ}, where μ is the index. We introduce the horizon vector

Manuscript received June 30, 2008; revised August 18, 2008. First published December 02, 2008; current version published August 14, 2009. This work was supported in part by the Air Force Office of Scientific Research under Grant FA95500410130 and in part by the National Science Foundation under Grant CCF0347573. This paper was recommended by Associate Editor A. van Schaik.

The author is with the Department of Electrical and Computer Engineering and the Institute for Systems Research, Clark School of Engineering, University of Maryland, College Park, MD 20742 USA (e-mail: timmer@isr.umd.edu).

Digital Object Identifier 10.1109/TCSI.2008.2010097

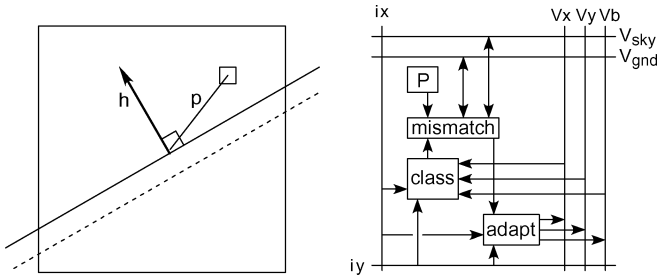


Fig. 2. (a) The sign of the dot product of the horizon vector (h) with the pixel vector (p) plus a bias parameter determines the horizon line. The dotted line represents the horizon line with a positive bias. (b) Block diagram of the processing that occurs in a single pixel. vx, vy, and vb represent the $x-y$ components and bias parameter of the horizon vector, and each pixel's coordinate (x, y) is represented by currents encoded by the *voltages*ix and iy. Vsky and Vgnd represent the average intensities of pixels within the selected sky and ground regions, respectively.

h and define the horizon as the boundary separating the two polarities (or "classes") resulting from the dot product between pixel vectors and the horizon vector (plus a bias parameter b and a fixed threshold θ)

$$\text{class}(p^\mu, h) = \text{sign}\left((p^\mu \cdot h) + b - \theta\right). \tag{1}$$

The two classes represent "sky" and "ground." In the horizon detection algorithm to follow, exactly which class represents sky or ground will not be specified but can easily be determined after the horizon is found by measuring the average image intensity of the sky and ground classes. The horizon *line* is thus perpendicular to the horizon *vector* and is offset from the origin by a distance that is dependent on the horizon vector magnitude, the threshold, and the "bias" parameter b. The horizon vector information is available at each pixel location, and the class assignment is computed in parallel at each pixel.

The goal of finding a visual horizon line requires a working definition of the differences between "sky" and "ground." We use an approach that is similar to that of the UF team [9], noting that a histogram of pixel intensities (in their case, the RGB vector) will, in most cases, show a bimodal distribution with the sky pixels that are bright and the ground pixels that are dark. This is obviously not always true and depends on the particular wavelengths used but describes most situations adequately. The goal is to find the line in the image that best separates the two intensity distributions.

With the horizon vector in some initial state, we begin by computing the average intensity of each class for the current state of the horizon line. At each pixel, the absolute differences between the pixel's intensity and the class intensity averages are computed. We decide that a pixel is "misclassified" if the pixel intensity is closer in value to the opposite class average. The goal of the horizon detection algorithm is to find the horizon vector that minimizes the total number of misclassified pixels. For any realistic image, the horizon will never be perfectly straight due to trees, buildings, canyons, mountains, or lens distortions; by monitoring the total number of misclassified pixels, however, we will have an ongoing estimate of the success or failure to fit a straight line. We may additionally have portions of the image where pixels appear to be misclassified due to their intensity (e.g., bright spots on the ground and dark objects in the sky).

The horizon-line calculation operates as a linear discriminant function over a 2-D input space (i.e., the image). By utilizing well-known neural-network learning algorithms, we can achieve adaptation of the horizon vector to minimize the total number of misclassified pixels.

B. Finding the Best Horizon Vector

In neural-network training, example inputs (pixel vectors) are presented one by one, and the resulting output is compared against a desired output (class match or mismatch). The linear discriminant (horizon) is then moved to minimize a quadratic cost function. In the horizon detection problem, the image represents the distribution to learn, and all of the input examples are presented simultaneously. As our image moves and changes, the horizon vector must quickly adapt to continuously minimize the cost function.

If, instead of $sign(\,)$, we use some sigmoidal activation function $g(x)$ (whose range is 0–1, with $g'(\,) > 0$), each pixel output class can described by

$$O^\mu = g(h \cdot p^\mu + b - \theta) = g\left(\sum_i hi \cdot p_i^\mu + b - \theta\right). \tag{2}$$

If ζ^μ represents the desired class output (0 or 1) for a given pixel, we can define a cost function

$$E(h) = \frac{1}{2}\sum_\mu (\zeta^\mu - O^\mu)^2 \tag{3}$$

and solve for an update rule for each component of the horizon vector

$$\begin{aligned}\Delta hi &= -\eta \frac{\partial E}{\partial hi} \\ &= -\eta\left(-\sum_\mu (\zeta^\mu - O^\mu) g' \right. \\ &\qquad \left. \times \left(\sum_i hi \cdot p_i^\mu + b - \theta\right) p_i^\mu\right) \qquad (4)\end{aligned}$$

$$\Delta hi = \eta \sum_\mu \delta^\mu p_i^\mu \tag{5}$$

where η is the learning rate and

$$\delta^\mu = (\zeta^\mu - O^\mu) g'\left(\sum_i hi \cdot p_i^\mu + b - \theta\right). \tag{6}$$

Since $g'(\,)$ will always be positive and $(\zeta^\mu - O^\mu)$ represents the sign and degree of the mismatch, we approximate this by setting δ^μ equal to $(\zeta^\mu - O^\mu)$, keeping our effective step size η small to avoid overestimating $\partial E/\partial h_i$. The learning rate must be kept small to avoid large oscillations around the solution but must be large enough to find solutions rapidly.

The equations ultimately result in the following. When a pixel determines that it is misclassified as a ground (or sky) pixel, it adds (or subtracts) its own coordinate vector to (or from) the horizon vector, thus rotating the horizon vector slightly. In this way, both the direction and the amplitude of the vector

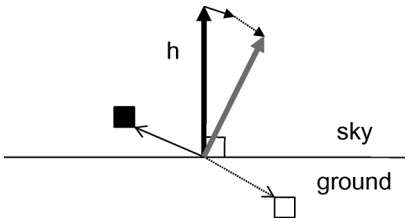


Fig. 3. Example effects of the learning rule. Consider a horizon vector (h) that splits the image into sky (above the horizontal line) and ground regions. When a pixel is found to be misclassified, there are two possible effects on the horizon vector. In the case of a ground pixel found in the sky region, a tiny fraction (exaggerated in the figure) of the ground pixel vector is *subtracted* from the horizon vector, but in the case of a sky pixel found in the ground region, a tiny fraction of the sky pixel vector is *added* to the horizon vector. The resulting vector (gray vector) is closer to properly classifying both pixels.

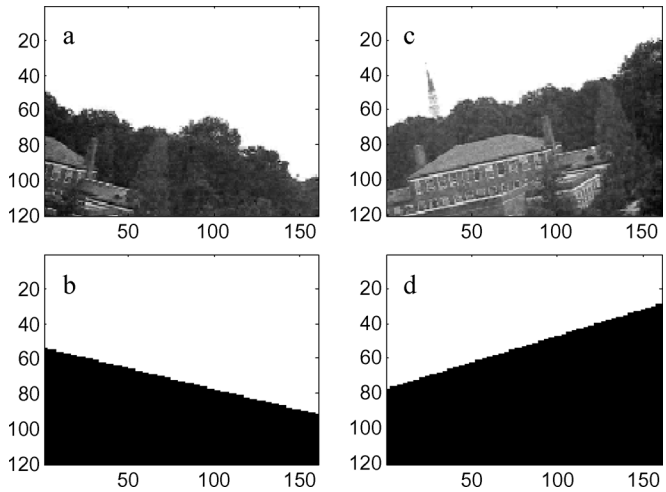


Fig. 4. MATLAB simulation: Two example frames from a movie sequence and the corresponding estimate of the horizon. The movie frames were captured with a digital camera.

are changed (see Fig. 3). Because all pixels perform this operation simultaneously, the change in the horizon vector will be a large vector sum of adaptation vectors. Notice that pixels near the center with tiny vector amplitudes do not have the same weighting as those pixels in the periphery.

The bias variable modification rule operates independently from the rotation and simply increases or decreases the bias parameter (i.e., translates the horizon) to balance the number of misclassified pixels on either side of the horizon line. Thus

$$\Delta b = \lambda \sum_{\mu} \delta^{\mu} \tag{7}$$

where λ is the bias learning rate. Notice that, with a nonzero threshold θ, increasing the horizon vector length has the same *effect* as increasing the bias.

This algorithm was simulated on several different movie sequences captured from the top of a building to simulate a micro aerial vehicle in an urban setting. Following the presentation of each new movie frame, the differential equations were allowed to iterate ten times to create the new estimate of the horizon. A steady-state solution was typically found within four to five iterations (Fig. 4).

In many of these simulations, bright sidewalks or reflections of the sky saturated the digital camera to the same intensity value as the sky, producing a skew in the final horizon solution. In the final chip implementation, however, since we are directly imaging the scene with photodiodes and representing image intensity in the current domain, the sky will typically be measured to be orders of magnitude brighter than the bright sidewalk.

C. Analog VLSI Considerations

The primary motivation for analog VLSI implementation is to achieve real-time performance using very low power. Although analog VLSI implementations typically suffer from transistor mismatch, much of this algorithm works through averaging, minimizing the impact of individual pixel mismatch. Real outdoor scenes contain intensities spanning many orders of magnitude that can commonly overload standard imagers. On this chip, image intensity is represented in the current domain, allowing for many orders of magnitude of image intensity. This can be important since the sky can often be significantly brighter than bright objects on the ground, a situation where conventional imagers would report similar saturated values.

Various practical reasons make it desirable to prevent unbounded growth or shrinkage of the horizon vector amplitude. Large vectors represented by voltages can exceed the dynamic range of a given circuit, while very small vectors can produce outputs close to the computational noise level (i.e., discretization noise, electronic noise, and transistor mismatch). We have introduced the bias term b that allows off-origin horizon lines while keeping the horizon vector magnitude (and, thus, its component voltages) within a desirable range of voltages.

A sophisticated sensor usually needs a method for determining if the conditions for accurate measurement are present. In our application, the sensor does not know *a priori* if the aircraft is looking straight down at the ground or flying at low altitudes where objects may interfere with perception of the horizon. In our algorithm, the total number of misclassified pixels and the average intensities of the two classes can provide an indication of the inability to find a good horizon line or the presence of low-contrast conditions.

III. Circuits

The horizon detection sensor and algorithm were implemented in analog CMOS circuitry, with its transistors operating primarily in the subthreshold region of operation. The measured horizon vector (with bias) is represented by three voltages, and the "total-mismatch" confidence-level measure is reported as a current.

A. System Block Diagram

The horizon detection chip consists of a 12×12 array of pixels, each with a photodiode (approximately 406 μm^2) and horizon detection circuitry [see Fig. 2 (right panel)]. The photodiode current and the computed class assignment for each pixel can be scanned out to produce images. The 2-D array is organized into four quadrants with slightly different cell layouts to allow the use of simpler two-quadrant computational circuits. In the sections to follow, only quadrant-1 (i.e., $x > 0$ and $y > 0$) circuits will be shown. For other quadrants where either the x or y components are negative, the signal voltage (i.e., vx or vy) and $vref$ signals are reversed on the differential pair inputs. Along

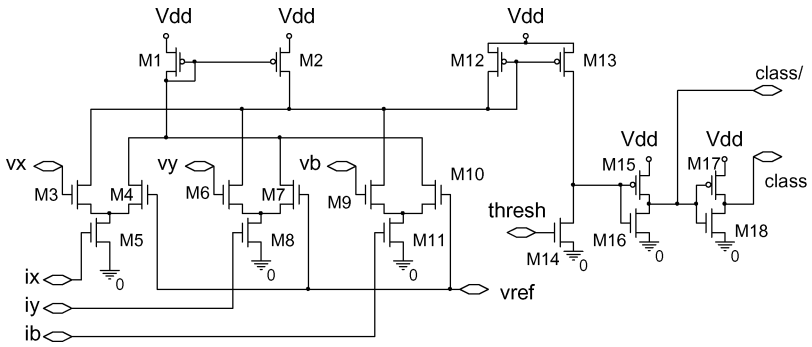


Fig. 5. Quadrant-1 class detector circuit. Pixel coordinates are given by ix and iy, while the horizon vector is given by vx, vy, and vb. (W/L values are given in micrometers: $\mathrm{M1}-\mathrm{M16}=1.8/1.8$; M17 and $\mathrm{M18}=1.8/2.4$).

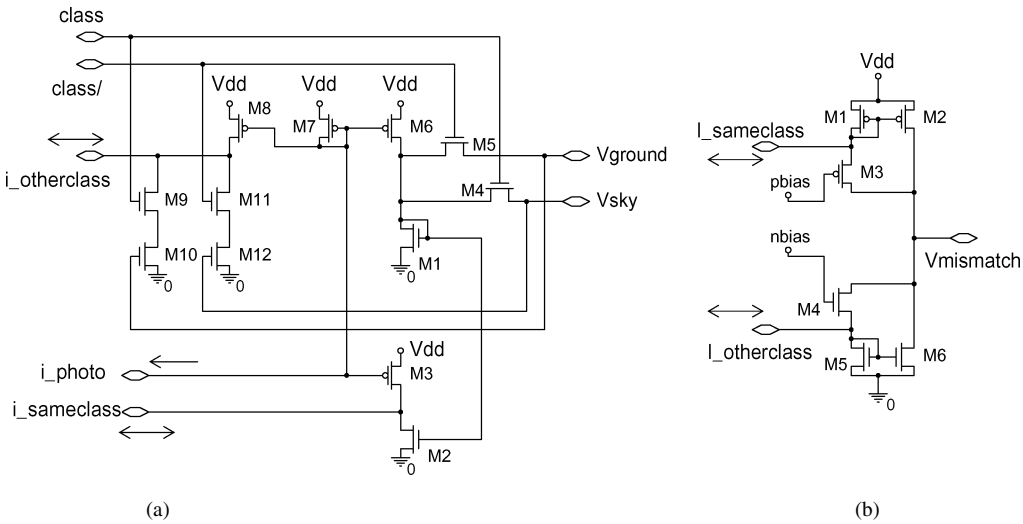


Fig. 6. (a) Average class intensities are computed, and the difference between the local pixel intensity and the two average class intensities is represented as the currents $i\_\mathrm{otherclass}$ and $i\_\mathrm{sameclass}$. (W/L values are given in micrometers: M1, M2, M4, M5, M9, $\mathrm{M11}=1.8/1.8$; M3, M6–M8, M10, $\mathrm{M12}=1.8/2.1$) (b) Absolute-value difference currents are compared to determine if a pixel has been misclassified. (W/L values are given in micrometers: M5, M6, M1, $\mathrm{M2}=1.8/2.1$; M3, $\mathrm{M4}=1.8/1.5$).

the margins of the array, current sources that are proportional to the magnitude $|x|$ and $|y|$ (representing the x-y coordinates of each pixel) are mirrored into each pixel via the voltages ix and iy, respectively (circuit not shown). The parameter ib is a global constant that sets the effective gain of vb in shifting the horizon away from the origin.

B. Class Detector

The circuit schematic for the quadrant-1 class detector ("class" block in Fig. 2) is shown in Fig. 5. Each of the three voltages, namely, vx, vy, and vb, is referenced to the voltage $vref$ allowing negative values. The differential pair currents are summed and compared to a current threshold defined by the voltage parameter $thresh$. This represents the dot product of the pixel vector and the horizon vector compared to the threshold Θ. The resulting digital signal is buffered, and a complementary signal is generated. For other quadrants, negative coordinates are implemented by swapping the output connections of the differential pairs. All pixel class outputs will thus be classified logically as either $\mathrm{class}=1$ or $\mathrm{class}=0$.

C. Mismatch Detector

The two subcircuits of the mismatch detector ("mismatch" block in Fig. 2) are shown in Fig. 6. The class assignment (signals class and $\mathrm{class}/$) of each pixel in the image (generated by the circuit in Fig. 5) is used to compute the average image intensity of each class. Each pixel makes multiple copies of the local photocurrent (i_photo from the photodiode, block P in Fig. 2) via transistor M7 to transistors M3, M6, and M8. The copy from M6 is merged with other photocurrents within its class via the transistor switches (either M5 or M4) and is subsequently averaged by all of the M1 transistors in the dynamically linked class. The class average current is thus mirrored to M2 and is compared to the local photocurrent.

The difference current between the local intensity and the class average is output on the line labeled, namely, $i\_\mathrm{sameclass}$. The difference current between the local intensity and the other class average is output on the line labeled, namely, $i\_\mathrm{otherclass}$. The difference currents are then compared to each other [see Fig. 6(b)] to determine if the pixel was misclassified. Since the difference currents can be either positive or negative, absolute-value circuits are employed. The parameter pbias should be set approximately one threshold voltage below Vdd, and

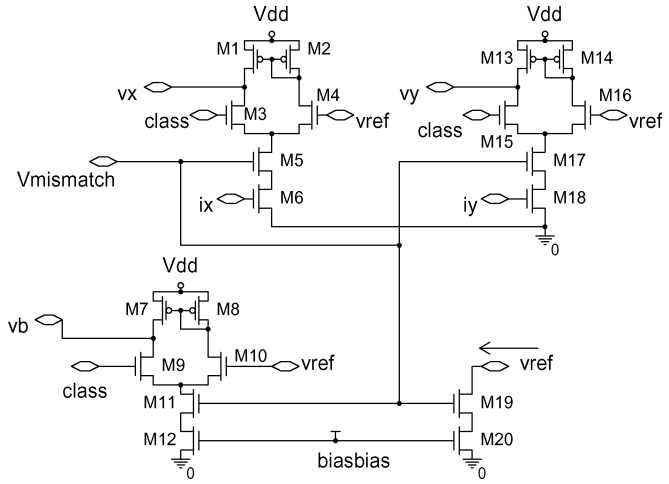


Fig. 7. Quadrant-1 horizon vector adaptation circuit (all transistors have a $W/L = 1.8/1.8$, given in micrometers). The series transistors M19 and M20 sink a fixed current from $vref$ when the pixel is indicating a mismatch condition.

nbias should be set approximately one threshold voltage above ground.

D. Horizon Vector Learning

The circuit schematic for the quadrant-1 adaptation circuit ("adapt" block in Fig. 2) is shown in Fig. 7. If $Vmismatch$ is high, indicating a class mismatch, a current that is proportional to the pixel coordinate is either added or subtracted directly onto or from the vx and vy lines. For adapting the bias value, a fixed current, defined by $biasbias$, is either added or subtracted directly onto or from the vb line, moving the horizon line to change the pixel class. External and parasitic capacitances *integrate* the summed currents from all the pixels to produce changes in vx, vy, and vb. For other quadrants, negative coordinates are represented by swapping the $class$ and $vref$ connections in the differential pair.

E. Confidence Measures and Chip Outputs

The main outputs of the chip are the voltages vx, vy, and vb and the total mismatch current drawn from the $vref$ line (see Fig. 7, M19 and M20) that indicates confidence level. Along the edges of the array, $x-y$ addressing scanners allow access to the photocurrent and the selected class at each pixel. In addition, there are two current outputs derived from $Vsky$ and $Vground$ (see Fig. 6) that mirror the average photocurrent measured in the "sky" and "ground" classes. These two currents are important for distinguishing (off-chip) if the sky class contains the brighter pixels compared to the ground class. Inversion of these two classes is not important for *finding* a good horizon line; it is important, however, for interpreting if the aircraft is upside down or right-side up.

F. Diamond Constraint Circuit

To encode horizon lines that do not pass through the origin, either the horizon vector magnitude (i.e., vx and vy values) or the bias (i.e., vb) can be scaled while leaving vx and vy unchanged. Because the effect is equivalent, the vector adaptation circuit does not bias the solution to any particular combination of vector magnitude and bias. The class detector circuit, however, operates in its linear regime only when vx and vy are within about 100 mV of $vref$. For this reason, we have added another circuit that normalizes the horizon vector to a range of acceptable values and a term to our error function (8) that minimizes the difference between the horizon vector magnitude and a reference magnitude. While a constant magnitude vector (ℓ^2-norm) would be a natural choice (i.e., the vector lies on a unit circle), we have chosen the L^1-norm (i.e., the vector lies on a diamond) for circuit simplicity

$$E(h) = \frac{1}{2}\sum_{\mu}(\zeta^{\mu} - O^{\mu})^2 + \frac{1}{2}\cdot\lambda\left(|h|_1 - C\right)^2 \tag{8}$$

$$\frac{\partial E}{\partial h_i} = \sum_{\mu}\delta^{\mu}p_i^{\mu} + \lambda\left(|h|_1 - C\right)\frac{h_i}{|h_i|}. \tag{9}$$

Although the derivative of the new error function (9) shows a magnitude correction term that is independent of the magnitude of h_i (due to the choice of the L^1-norm), it is desirable to change the magnitude of the horizon vector without modification of the direction of h. To accomplish this, we modify the horizon component correction to include the magnitude of h_i, making the direction of the magnitude correction along the horizon vector

$$\Delta h_i = -\eta\left(\sum_{\mu}\delta^{\mu}p_i^{\mu} + \lambda\left(|h|_1 - C\right)h_i\right). \tag{10}$$

We have designed a circuit (Fig. 8) that sources and sinks current on the vx and vy lines to increase or decrease the vector magnitude in proportion to the x and y components, respectively. This simple circuit drives the sum of the vector components to a constant (i.e., the horizon vector settles onto a diamond-shaped curve). First, the L^1-norm is calculated at the top of the circuit (M1–M16)

$$I_{\mathrm{norm\_L1}} = \alpha\cdot\left(|vx - vref| + |vy - vref|\right) + I_{\mathrm{DCbias}}. \tag{11}$$

The I_{DCbias} is the sum of four small dc currents induced by the transistor pairs M11–M12 and M14–M15 and their mirror currents in the two absolute-value circuits controlled by $b2$. This dc current is unavoidable in this absolute-value circuit, but it is ultimately negated by an adjustment in I_{mag}. α is the transconductance of the two differential amplifiers controlled by abs_gain. The drain voltage of M17 ($V_{\mathrm{mag\_error}}$) indicates whether the vector is larger or smaller than the reference magnitude and "switches" currents that are proportional to x and y onto the vx and vy lines. A lower reference voltage $vref2$ is chosen for the $V_{\mathrm{mag\_error}}$ signal to properly steer the current in the lower differential pair

$$i_{\mathrm{x\_diam}} = \beta\cdot\mathrm{sgn}(I_{\mathrm{norm\_L1}} - I_{\mathrm{mag}})\cdot(vx - vref) \tag{12}$$

$$i_{\mathrm{y\_diam}} = \beta\cdot\mathrm{sgn}(I_{\mathrm{norm\_L1}} - I_{\mathrm{mag}})\cdot(vy - vref) \tag{13}$$

where I_{mag} is set to be the sum of I_{DCbias} and the magnitude reference current defined by the bias voltage $vmag$ (i.e., drain current in M17) and β is the transconductance of the lower two differential pairs controlled by $diam\_gain$. To reduce offset errors that might occur when the constraint error is close to zero and $V_{\mathrm{mag\_error}}$ is not in one of the two extremes (i.e., Vdd or Gnd), the adaptation current is reduced by a "bump" circuit [10]

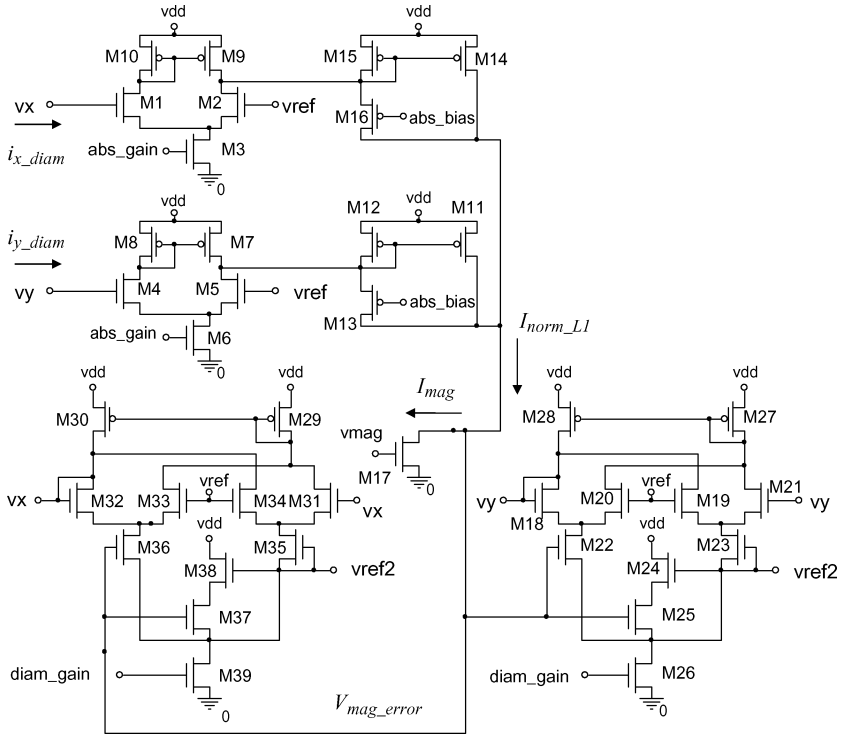
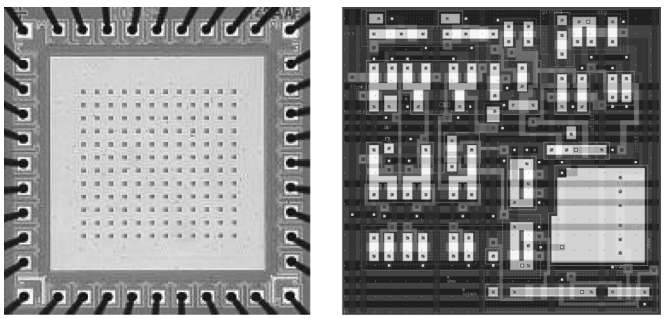


Fig. 8. Diamond constraint circuit. The horizon vector components vx and vy are converted to currents $i_{\mathrm{x\_diam}}$ and $i_{\mathrm{y\_diam}}$ that are the absolute values of the vector components. The components are summed and compared against a current reference defined by M17 and the bias voltage $vmag$. The resulting voltage signal determines whether the magnitude should be increased or decreased. The lower two circuits determine the correct direction of the adaptation current that depends on the sign of each component.



(a) (b)

Fig. 9. (a) Die photograph of the horizon chip with an array of 12 × 12 pixels (the holes in metal 3 are visible). (b) Layout of the quadrant-1 pixel with (green square) the photodiode visible at the bottom right. In this view, the metal-3 layer is hidden to allow a clear view of the design.

(M37/M38 and M24/M25). It should be noted that there is only one instance of this diamond constraint circuit on the chip.

IV. Test Results

A. Horizon Estimator

The chip was fabricated in a commercially available 0.5-μm two-poly three-metal CMOS process using the top metal layer as a light shield with holes over the photodiodes (Fig. 9). This process had an nFET threshold voltage estimated at 0.75 V and a pFET threshold voltage estimated at −0.96 V. The parameter voltages used for the test results in this paper are given in Table I. Images were projected directly onto the chip through a lens mounted on the chip packaging. An overview of the chip characteristics can be found in Table II. The layout graphics for a single pixel (without the metal-3 layer) is shown in Fig. 9(b). For testing purposes, the photocurrent and selected class for each pixel were scanned off using a current-sense amplifier. An example is shown in the top two panels of Fig. 11. The transistor mismatch in the class determination circuit produces a ragged, but clearly discernible, horizon boundary [Fig. 11 (top-right panel)].

TABLE I
Typical Parameter Values

thresh	0.434 V	*vref2*	2.00 V
vref	2.49 V	*abs_bias*	4.00 V
biasbias	0.641 V	*abs_gain*	0.700 V
ib	0.479 V	*diam_gain*	0.650 V
pbias	4.00 V	*vmag*	0.700 V
nbias	0.694 V		

TABLE II
Chip Characteristics

Characteristic	**Value**
Resolution	12x12
Angular Resolution	< 2.6 degrees
Angular Range (roll)	360
Response Time	set by user (0.022μF) Tested: 40ms
Power Consumption	< 2.5mW
Transistors per pixel	63

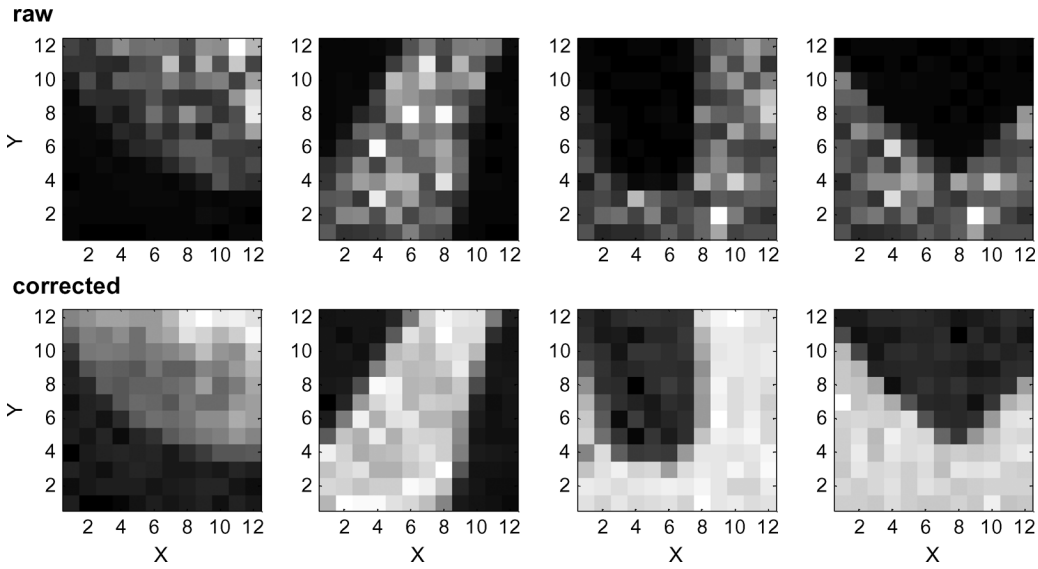


Fig. 10. (Top row) Four example photodiode images and (bottom row) externally offset/gain-corrected versions. Corrections were primarily gain errors.

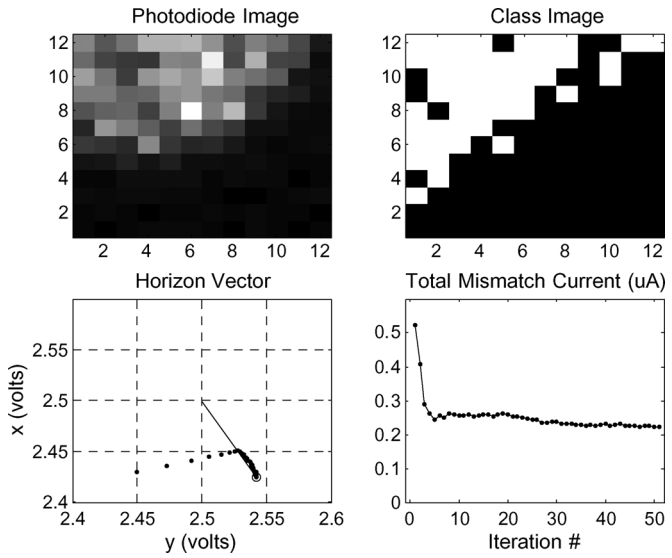


Fig. 11. Time evolution of the horizon estimate (computer-in-the-loop testing). Top left: Static image with nonzero roll and pitch angles. Top right: Final sky/ground class image. Bottom left: Horizon vector endpoints during iteration (the line circle is the final iteration). Bottom right: Total mismatch current during the evolution of the horizon estimate.

Example images from the photodiode array are shown in Fig. 10. Although not designed to be an imaging device, the chip can still be used in this manner.

To observe the adaptation process during testing, the vx, vy, and vb voltages were held externally by computer-controlled digital-to-analog converters, while the total vector adaptation currents on these lines were measured. After reading the photodiode image and class image, the horizon vector voltages were iteratively changed in proportion to the measured adaptation current, simulating the time evolution of the vx, vy, and vb voltages if they had been left floating or connected to external capacitors.

Fig. 11 shows an example image where the horizon vector was initially pointing toward the bottom-left corner, producing an incorrect horizon line [Fig. 11, bottom-left panel]. Initial iterations show a rapid rotation of the horizon vector toward the bottom-right corner, followed by a slow magnitude lengthening. The total mismatch current [Fig. 11, bottom-right panel] that reflects the number of mismatched pixels rapidly decreases, resulting in a stable solution. This final mismatch current is typical for good horizon solutions with the particular parameter settings used. Because this current represents the number of mismatched pixels, this current should be monitored by any system using the sensor's output to evaluate the confidence level of the sensor output.

We then allowed the horizon vector and bias value to freely adapt to produce rapid horizon solutions. Three example images with the resulting class separations are shown in Fig. 12. External 0.022-μF capacitors were attached to the vx, vy, and vb lines for stability, creating a time constant of about 20 ms. The time for settling is dependent on the setting of the adaptation circuits ($biasbias$ and the baseline currents for the coordinates, namely, i_x and i_y).

The roll-angle estimation accuracy is shown in Fig. 13, where horizon images were presented to the chip and the resulting horizon vector was transformed into an angle using

$$\theta = \arctan\left((vy - vref)/(vx - vref)\right). \tag{14}$$

Using the maximum standard deviation observed (0.0453 rad), we conservatively estimate the angular resolution to be 2.6° out of the possible 360° of rotation.

In this system, pitch corresponds to the shift of the horizon line away from the origin along the direction of the horizon vector. This is possible due to the threshold Θ and the bias b (2). Because the magnitude of the horizon vector is not tightly regulated to be constant, pitch is calculated (off-chip) using the following:

$$pitch \propto \frac{\beta \cdot (vb - vref) - \delta(thresh)}{\sqrt{(vx - vref)^2 + (vy - vref)^2}} \tag{15}$$

where β represents the transconductance of the differential pair for vb (see Fig. 5) and δ denotes the transformation of the

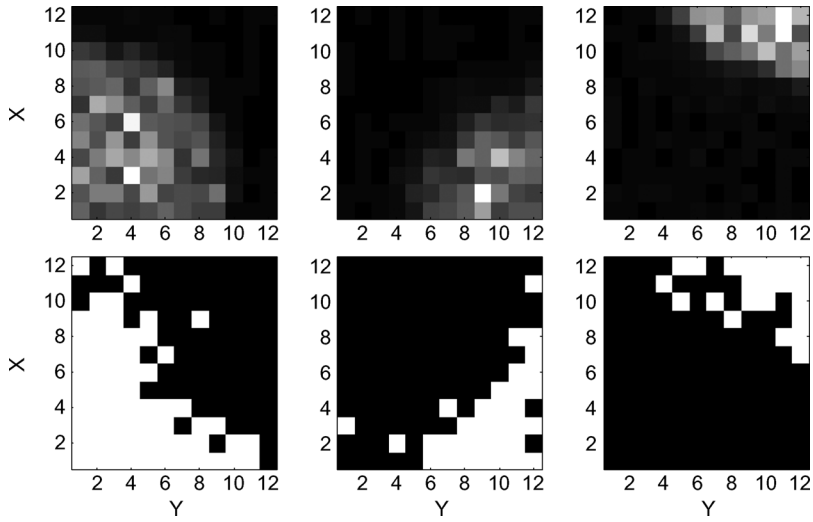


Fig. 12. Three example snapshots of internal state following horizon detection for different roll and pitch angles. Top row: Test images as seen by the photodiode array. Bottom row: Sky/ground class image following settling.

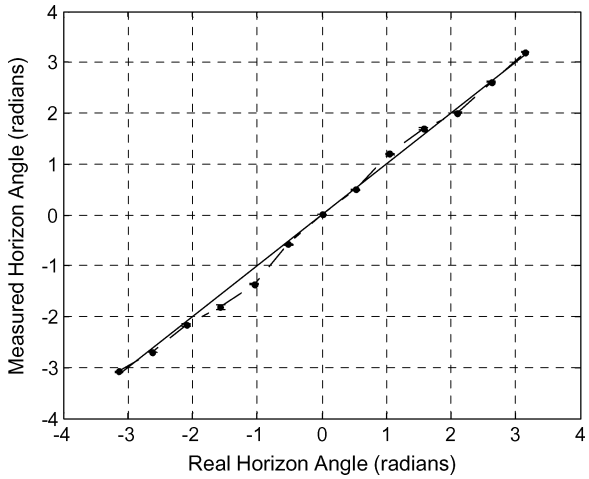


Fig. 13. Measured horizon-line roll angle versus actual horizon roll angle. Error bars represent one standard deviation from 25 measurements.

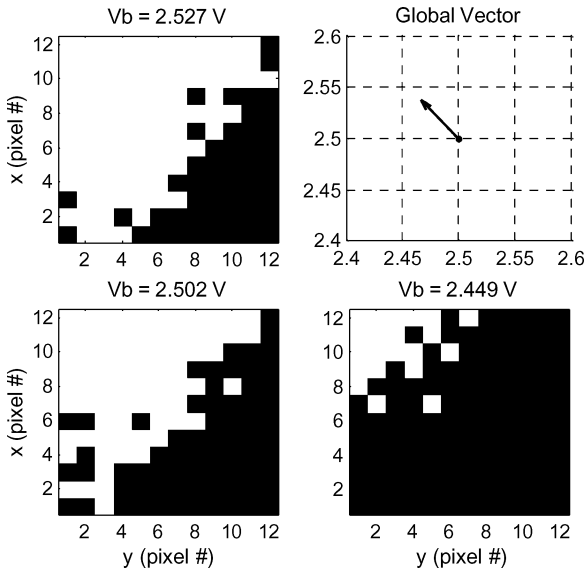


Fig. 14. Changes in the pitch angle without changes in horizon vector. Different vb values correspond to the horizon lines of different pitch angles.

voltage $thresh$ into the threshold current. Notice that if the horizon vector were ℓ^2 normalized, pitch would just be proportional to $(vb - vref)$ minus a constant. Unlike the roll angle, however, this calculation is highly dependent on parameters and device characteristics (e.g., ib, transistor threshold voltage, choice of lens, etc.) and will need to be mapped to specific angles in the final system. Three examples of different pitch (corresponding to different values of vb) are shown in Fig. 14.

B. Rapid Transitions

An important performance measure is the response time of the sensor to sudden changes in orientation. In this circuit, response time is dependent on the speed of the photodiodes (fast) and the charging time of the vx, vy, and vb lines (slow). The charging time depends on both the bias currents used in the adaptation circuit and the combined capacitance of the vx, vy, and vb lines and added external capacitance. Higher bias currents in the adaptation circuits or reduced external capacitance can improve speed. If no capacitors are added, however, oscillations due to the horizon vector adaptation circuit can be large, so a compromise must be made between instability and response time. An example of such a transition is shown in Fig. 15. In this example, the horizon was changed in both roll and pitch, such that both vector rotation (vx and vy) and a shift in the bias (vb) were necessary.

C. Diamond Constraint

We tested the diamond constraint circuit by measuring the 2-D current vector field produced at different $vx-vy$ combinations [upper panel of Fig. 16]. This was performed by applying different vx and vy voltages and measuring the resulting currents on those lines. Because the constraint current-vector magnitude is determined by the $x-y$ coordinate and the constraint error, the vector length drops to zero at the origin and along the

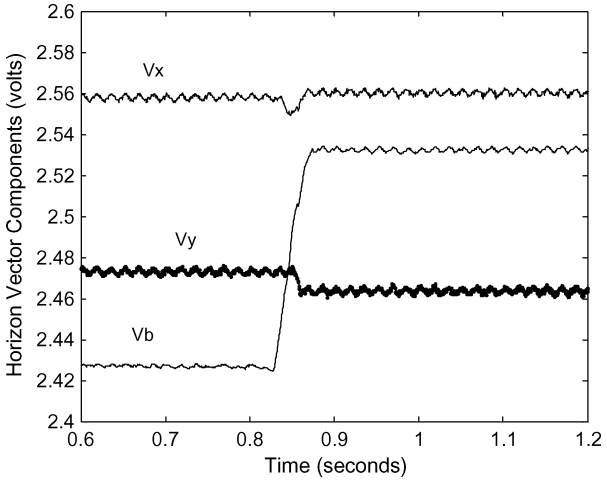


Fig. 15. Sudden image rotation. In this test, a stationary horizon image is suddenly changed to a different orientation. vb transitions in 40 ms. The external capacitors were 0.022 μF and $Vref = 2.50$ V.

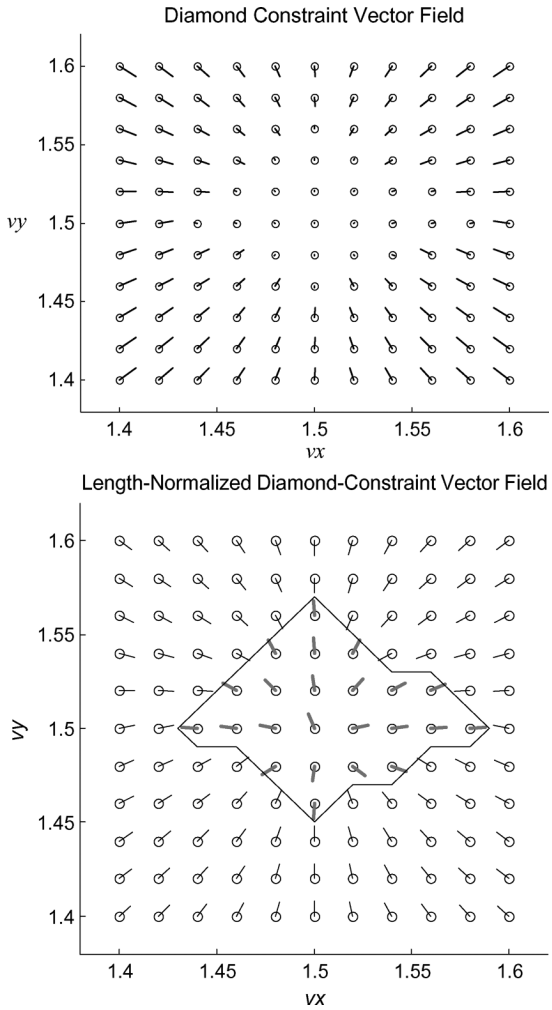


Fig. 16. Vector-field measurement of the diamond constraint circuit. Top: At each vx and vy combination ($vref = 1.5$ V), the current in each wire is measured and displayed as a vector indicating the direction in which the currents "push" the voltages. The black circles indicate the origin of each vector. Bottom: To better see the direction of each vector, the magnitudes from the top graph are normalized with outward-pointing vectors drawn in red (thick) and inward-pointing vectors in black (thin).

constraint contour (i.e., the diamond). Due to the difficulty in observing the vector direction in the small raw vectors, the lower panel shows normalized vector lengths. The diamond pattern can now be discerned by drawing the contour where the vectors flip from pointing inward to pointing outward [lower panel of Fig. 16]. This direction reversal contour represents the equilibrium position for the horizon vector.

D. Power Consumption

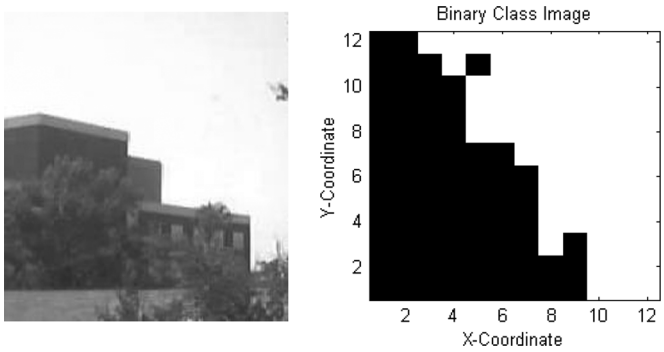
Although power consumption varies dynamically with the image properties (e.g., image brightness, horizon-line classification mismatch, etc.), long-term observations (with the settings used in the presented measurements) show that the power-supply current remains below 500 μA (or 2.5 mW with a 5-V power supply). Due to the 2-D layout of the circuit with the origin in the center, the largest pixel coordinates occur on the margins. Since power consumption in each pixel is based on a fixed dc current for biasing and a coordinate-dependent current used in the calculation of the class, the outer pixels burn the largest amount of power. Power can be saved by scaling down the $x-y$-coordinate currents (ix, iy) or, possibly, by clever elimination of some pixels (e.g., using only a ring of pixels or a sparser density in the periphery).

V. Discussion

In this paper, we describe a low-power visual-horizon detection chip that provides three output signals that can be interpreted to obtain roll and pitch information. The chip also provides a confidence measure signal that indicates when the sensor output is reliable (i.e., whether the sensor seems to be pointed at a horizon line) and has the ability to scan out the photodiode image. Using a novel mixed-mode circuit implementation to solve a cost-minimization problem, this real-time horizon sensor replaces the time- and power-consuming process of horizon detection in software. Integrating the algorithm onto a dedicated silicon chip is intended to facilitate stable attitude control in micro aerial vehicles at higher altitudes in the face of turbulence and wind.

Since the computational goal of the chip is to find the horizon line and not specifically to form an image, a high-resolution system is not necessary, particularly because the angle estimation process can interpolate. A wide field-of-view and broad coverage of the sky is much more important and can be provided by lenses. It should be noted, however, that very wide angle lenses can create image distortion that may create difficulties for the horizon detection algorithm that expects to see straight horizon lines.

The photosensitivity in the current implementation is not very high due to the use of photodiodes instead of phototransistors; this can easily be modified. The current-mode circuits are designed to operate over a very wide range of photocurrents and have been tested and shown to operate properly, even in bright conditions in which the photo-induced carriers in the substrate might be expected to cause problems. The photodiodes have



(a) (b)

Fig. 17. (a) Photograph of a sample outdoor scene used to stimulate the horizon sensor chip. This is an approximate framing of the image projected onto the chip. (b) Resulting class image from sensor exposed to natural daylight illumination.

been shown to be sufficient for most outdoor daylight conditions to properly detect and report the horizon (for example, see Fig. 17).

Although transistor mismatch can be reduced within individual pixel circuits by increasing the size of the transistors, the aggregate calculation is relatively insensitive to pixel mismatch.

In this chip, we have used broad spectral intensity contrast (silicon p-n-junction photodiode) to detect the horizon; however, the contrast between ultraviolet (UV) and green light [11] is known to be a more reliable measure and could be used here by adding optical filters and UV-sensitive photosensors. It should be noted that, although this horizon detection algorithm has been developed as a sensor to be run in parallel with other sensor systems, the algorithm could also be integrated with other vision chips.

In closing, it should be noted that, although the focus of this paper has been on the development of a visual-horizon detection chip, the basic approach of individual sensor circuits nudging the global solution toward minimizing a globally defined error function has the potential for solving problems in many different application domains.

ACKNOWLEDGMENT

The author would like to thank P. S. Krishnaprasad for his encouragement and advice throughout this project.

REFERENCES

[1] T. Netter and N. Franceschini, "Towards UAV Nap-of-the-Earth flight using optical flow," in *Proc. Adv. Artif. Life*, 1999, vol. 1674, pp. 334–338.

[2] W. E. Green, P. Y. Oh, K. Sevcik, and G. L. Barrows, "Autonomous landing for indoor flying robots using optic flow," in *Proc. ASME Int. Mech. Eng. Congr.*, Washington, DC, 2003, pp. 1341–1346.

[3] M. Massie, C. Baxter, J. P. Curzan, P. McCarley, and R. Etienne-Cummings, "Vision chip for navigating and controlling micro unmanned aerial vehicles," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2003, pp. 786–789.

[4] R. Moeckel and S.-C. Liu, "Motion detection circuits for a time-to-travel algorithm," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2007, pp. 3079–3082.

[5] C.-L. Tisse, H. Durant-Whyte, and R. A. Hicks, "An optical navigation sensor for micro aerial vehicles," *Comput. Vis. Image Underst.*, vol. 105, no. 1, pp. 21–29, Jan. 2007.

[6] R. J. Wood, S. Avadhanula, E. Steltz, M. Seeman, J. Entwistle, A. Bachrach, G. Barrows, S. Sanders, and R. S. Fearing, "Design, fabrication and initial results of a 2g autonomous glider," in *Proc. 31st IEEE Ind. Electron. Soc. Conf.*, Raleigh, NC, Nov. 2005, pp. 1870–1877.

[7] B. Taylor, C. Bil, and S. Watkins, "Horizon sensing attitude stabilization: A VMC autopilot," presented at the 18th Int. Unmanned Aerial Vehicles Systems Conf., Bristol, U.K., 2003, unpublished.

[8] Futaba, Futaba(R) PA-2: Pilot Assist Link Auto Pilot System [Online]. Available: http://www.futaba-rc.com/radioaccys/futm0999.html 2004

[9] S. M. Ettinger, M. C. Nechyba, P. G. Ifju, and M. Waszak, "Vision-guided flight stability and control for micro air vehicles," *Adv. Robot.*, vol. 17, no. 7, pp. 617–640, Nov. 2003.

[10] T. Delbruck, ""Bump" circuits for computing similarity and dissimilarity of analog voltages," in *Proc. IJCNN*, Seattle, WA, 1991, pp. 475–479.

[11] R. Moller, "Insects could exploit UV-green contrast for landmark navigation," *J. Theor. Biol.*, vol. 214, no. 4, pp. 619–631, Feb. 2002.

Timothy K. Horiuchi (M'89) received the B.S. degree in electrical engineering and the Ph.D. degree in computation and neural systems from the California Institute of Technology, Pasadena, in 1989 and 1997, respectively.

He was a Postdoctoral Scholar with The Johns Hopkins University, Baltimore, MD, until he moved to the University of Maryland, College Park, in 1999. He is currently an Associate Professor with the Department of Electrical and Computer Engineering and the Institute for Systems Research, Clark School of Engineering, University of Maryland, College Park. His main research interests include the design and fabrication of neuromorphic VLSI circuits and the implementation of neural computation in silicon. His primary focus has been on the modeling of spike-based neural processing models of the auditory system of the echolocating bat.