A Low-Power Visual Horizon Estimation Chip

Timothy K. Horiuchi,
Electrical and Computer Engineering Department,
Institute for Systems Research & the Neuroscience and Cognitive Science Program
University of Maryland, College Park, MD 20742, timmer@isr.umd.edu

***Abstract*—Recent successes in micro-aerial vehicles (< 15cm length, wingspan, height), have highlighted the lack of real-time sensors for flight control. In this paper we describe a low-power, real-time visual horizon sensor for use in stabilizing miniature aircraft with respect to pitch and roll in moderate-to-high altitude flight. This prototype sensor incorporates a 12x12 photoreceptor array and finds a best-fit horizon line based on image intensity. The sensor includes a "confidence-level" output for a flight control system to detect poor sensing conditions. The chip was fabricated in a commercially-available 0.5μm CMOS process and operates on less than 2.5 milliwatts with a 5V power supply.**

***Index Terms*—analog VLSI, smart sensor, autonomous flight control, micro-aerial vehicles.**

I. Introduction

Unmanned micro-aerial vehicles are rapidly being developed for use as a low-cost, portable, aerial surveillance platform for semi-autonomous operation. While they are successfully achieving flight, the sensors needed for autonomous flight (in contrast to long-range navigation) are lacking. Obstacle avoidance and flight stability remain a problem for such small vehicles with tiny weight and power budgets. Their small size makes them 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 chips are being pursued (e.g. [1, 2]), detection of the horizon is also desirable for high-altitude pitch and roll stabilization. Several low sensor-count horizon sensing systems have been developed for assisting aircraft pilots that utilize contrast in the infrared spectrum [3] or visible light [4]. In recent years, a team from the University of Florida (UF) 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 [5]. We have devised a similar algorithm that uses an optimization approach embedded in an analog VLSI vision chip to find the best visual horizon and provide a measure of confidence. The VLSI implementation has the potential for real-time, low-power performance and operation over a wider dynamic range of image intensities.

This work was supported in part by the Air Force Office of Scientific Research (FA95500410130) and in part by the National Science Foundation under (CCF0347573)

II. Horizon Detection Algorithm

A. The 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 (Figure 1, left). We can think of this coordinate as the pixel vector, p^{μ}, where μ is the index. We introduce the horizon vector, 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 fixed threshold, θ).

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

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 be determined after the horizon is found by measuring the average 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 defined by the 'bias' parameter, b. The horizon vector is available at each pixel location and the class assignment is computed in parallel at each pixel.

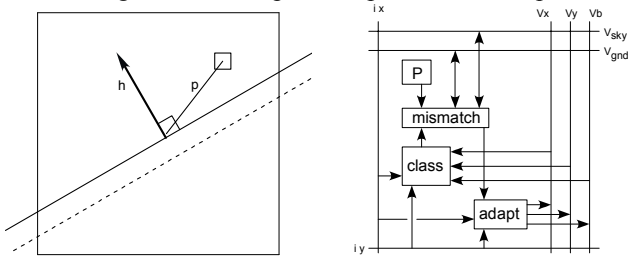

Fig. 1. Left: The sign of the dot-product of a horizon vector (h) and the pixel vector (p) plus a bias determines the horizon line. The dotted line represents the horizon line with a positive bias. Right: Block diagram of processing in a single pixel. vx, vy, and vb represent the horizon vector and each pixel's coordinate (x, y) is determined by ix and iy.

The goal of finding a visual horizon requires a working definition of the differences between 'sky' and 'ground'. We use an approach similar to the UF team [5], noting that a histogram of pixel intensities (in their case, the RGB vector) will show a bimodal distribution with the sky pixels bright and the ground pixels dark. This is obviously not always true, but describes most situations adequately. The goal is to find the line that best separates the two intensity distributions.

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 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 not 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.

The horizon line calculation operates as a linear discriminant function over a two-dimensional input space. By utilizing 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 to 1 with $g'() > 0$) each pixel output class can described by,

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

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

$E(h) = \frac{1}{2}\sum_{\mu}\left(\zeta^{\mu} - O^{\mu}\right)^2$ and solve for an update rule for each component of the horizon vector:

$$\Delta h_i = -\eta\frac{\partial E}{\partial h_i} = -\eta\left(-\sum_{\mu}\left(\zeta^{\mu} - O^{\mu}\right)g'\left(\sum_i h_i \cdot p_i^{\mu} + b - \theta\right)p_i^{\mu}\right)$$

$\Delta h_i = \eta\sum_{\mu}\delta^{\mu}p_i^{\mu}$, where

$\delta^{\mu} = \left(\zeta^{\mu} - O^{\mu}\right)g'\left(\sum_i h_i \cdot p_i^{\mu} + b - \theta\right)$ and η is the learning rate. 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 $\frac{\partial E}{\partial h_i}$.

When a pixel determines itself to be misclassified as a ground (or sky) pixel, it adds (or subtracts) its own coordinate vector to (or from) the horizon vector, thus rotating the vector slightly. In this way, both the direction and the amplitude of the vector are changed. 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, $\Delta b = \lambda\sum_{\mu}\delta^{\mu}$, where λ is the bias learning rate. Notice that with a non-zero threshold, θ, increasing the horizon vector length has the same *effect* as increasing the bias.

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 which can commonly overload standard cameras. On this chip, image intensity is represented in the current domain, allowing for many orders of magnitude of image intensity.

Various practical reasons make it desirable to prevent unbounded growth or shrinkage of the horizon vector amplitude. Large vectors can exceed the dynamic range of a given circuit and very small vectors can produce outputs close to the computational noise level (i.e., discretization noise, electronic noise, and transistor mismatch). The bias term, *b*, works to prevent large horizon vector amplitudes.

III. Circuits

The horizon detection algorithm was implemented in analog CMOS circuitry operating 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 12x12 array of pixels each with a photodiode and horizon detection circuitry (see Figure 1, right). The photodiode current and the class assignment for each pixel can be scanned out to produce images. The array is organized into four quadrants with different cell layouts to allow the use of simpler two-quadrant computational circuits. In the sections to follow, only quadrant-one (i.e. x>0, y>0) circuits will be shown. Along

the margins of the array, current sources of magnitude $|x|$ and $|y|$ are mirrored into each pixel via the voltages *ix* and *iy* respectively. The parameter *ib* is a global constant.

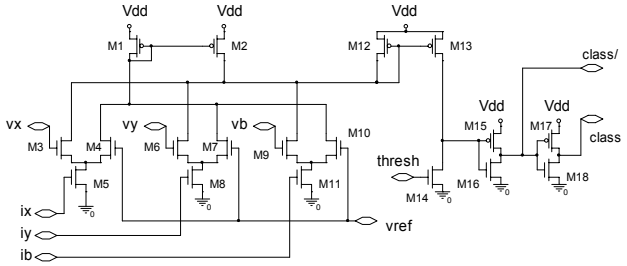

Fig. 2. The quadrant-one class detector circuit. Pixel coordinates are given by *ix* and *iy* while the horizon vector is given by *vx*, *vy*, and *vb*. (W/L values are given in microns: M1-M16 = 1.8/1.8; M17 and M18 = 1.8/2.4)

B. Class Detector

The circuit schematic for the quadrant-one class detector is shown in Figure 2. Each of the three voltages, *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*. 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 class = 1 or class = 0.

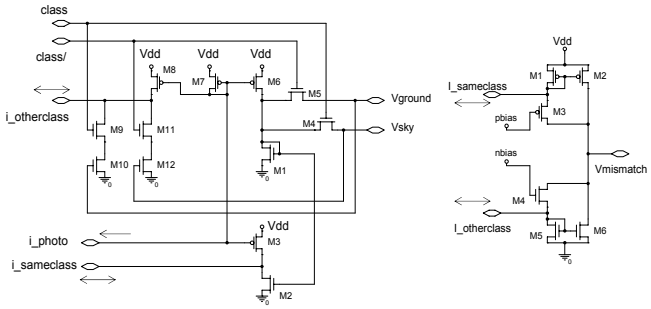

Fig. 3. Left: The average class intensities are computed and the **difference** between the local pixel intensity and the two average class intensities are represented as the currents *i_otherclass* and *i_sameclass*. (W/L values are given in microns: M1, M2, M4, M5, M9, M11 = 1.8/1.8; M3, M6-M8, M10, M12 = 1.8/2.1) Right: The absolute-value difference currents are compared to determine if a pixel has been misclassified. (W/L values are given in microns: M5, M6, M1, M2 = 1.8/2.1; M3, M4 = 1.8/1.5)

C. Mismatch Detector

The two subcircuits of the mismatch detector are shown in Figure 3. Once a class has been assigned to each pixel in the image, it is possible to compute the average image intensity of each class (see Figure 3). Each pixel makes a copy of the local photocurrent and mirrors this current (M1 and M2) back into the pixel for comparison against the local photocurrent. By coupling together the M1 transistors of all pixels of a given class, the mirrored current becomes the class average.

The difference current between the local intensity and the class average is output on the line labeled, *i_sameclass*. The difference current between the local intensity and the other class average is output on the line labeled, *i_otherclass*. The difference currents are then compared to each other (see Figure 3, right) to determine if the pixel was misclassified.

D. Horizon Vector Learning

The circuit schematic for the quadrant-one adaptation circuit is shown in Figure 4. If *Vmismatch* is high, indicating a class mismatch, a current proportional to the pixel coordinate is either added (subtracted) directly onto (from) the *vx* and *vy* lines. For adapting the bias value, a fixed current, defined by *biasbias*, is added (subtracted) directly onto (from) the *vb* line, moving the horizon line to change the pixel class. For other quadrants, negative coordinates are represented by swapping the *class* and *Vref* connections in the differential pair.

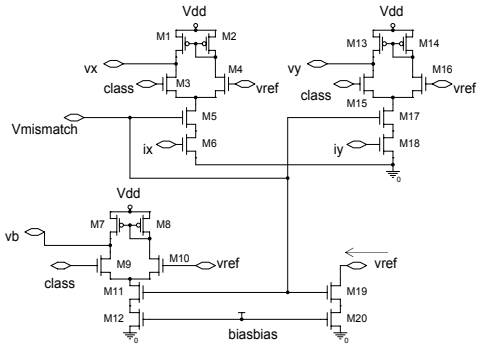

Fig. 4. The quadrant-one horizon vector adaptation circuit. (All transistors have a W/L = 1.8/1.8, given in microns)

E. Confidence Measures and Chip Outputs

The main outputs of the chip are the voltages *vx*, *vy*, *vb,* and the total mismatch current drawn from the v*ref* line (see Figure 4) indicating confidence level. An x-y scanner allows access to the photocurrent and selected class at each pixel. In addition, there are two current outputs that mirror the average photocurrent measured in the 'sky' and 'ground' classes. These two currents are important for distinguishing if the sky class contains the brighter pixels compared to the ground class.

IV. Testing Results

The chip was fabricated in a commercially-available 0.5 μm, 2-poly, 3-metal CMOS process using the top metal layer as a light shield with holes over the photodiodes. Horizon-like images were projected onto the chip through a lens mounted on the chip package. 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 Figure 5. Transistor mismatch and light-leakage currents in the class determination circuit produce a ragged horizon boundary.

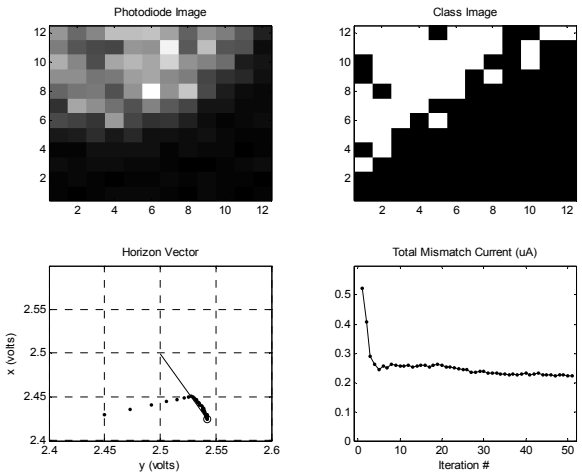

Fig. 5. Time-evolution of the horizon estimate. Top-Left: Static image, Top-right: Final sky/ground class image, Bottom-Left: Horizon vector end-points during iteration (line-circle is the final iteration), Bottom-Right: Total mismatch current during the evolution of the horizon estimate.

To observe the adaptation process, the *vx*, *vy*, and *vb* voltages were held externally by computer-controlled digital-to-analog converters (DAC) 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.

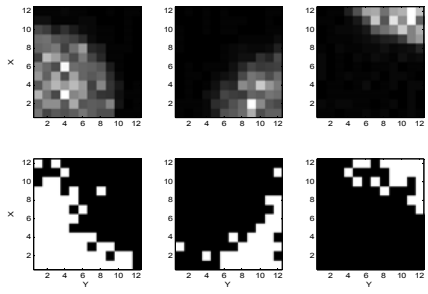

Fig. 6. Three example snapshots of internal state following horizon detection. Top Row: Photodiode image. Bottom Row: sky/ground class image following settling.

Figure 5 shows an example image where the horizon vector was initially pointing towards the bottom-left corner, producing an incorrect horizon line (Fig. 5, bottom-left panel). Initial iterations show a rapid rotation of the horizon vector followed by a slow lengthening. The total mismatch current (Fig. 5, bottom-right panel) which 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.

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 Figure 6. External 0.022 μF capacitors are attached to the *vx*, *vy*, and *vb* lines for stability, creating a time constant of about 20 ms.

Roll angle estimation accuracy is shown in Figure 7 where horizon images were presented to the chip and the resulting horizon vector was transformed into an angle using: $\theta = \arctan\left((vy - vref)/(vx - vref)\right)$.

While the power consumption varies dynamically with the image, long-term observations show the power supply current to be less than 500μA (or 2.5mW with a 5V power supply).

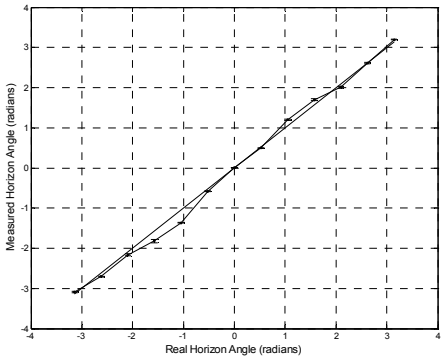

Fig. 7. Measured horizon line roll angle vs. actual horizon roll angle. Error bars represent one standard deviation from 25 measurements.

V. Discussion

While this chip satisfies the basic goals of the project, there are many places for improvement. In particular, the sensitivity of the sensor can be improved by using phototransistors and transistor mismatch can be reduced by using larger transistors. The contrast between ultraviolet (UV) and green light [6] is known to be a more reliable measure and could be used here by adding optical filters and UV-sensitive photosensors.

Acknowledgment

The author thanks 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," *Advances in Artificial Life, Proceedings*, vol. 1674, pp. 334-338, 1999.

[2] W. E. Green, P. Y. Oh, K. Sevcik, and G. L. Barrows, "Autonomous Landing for Indoor Flying Robots Using Optic Flow," presented at ASME Intl Mech. Engr. Congress, Washington D.C., 2003.

[3] B. Taylor, C. Bil, and S. Watkins, "Horizon Sensing Attitude Stabilization: A VMC Autopilot," presented at 18th Intl. UAV Systems Conference, Bristol, UK, 2003.

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

[5] S. M. Ettinger, M. C. Nechyba, P. G. Ifju, and M. Waszak, "Vision-guided flight stability and control for micro air vehicles," *Advanced Robotics*, vol. 17, pp. 617-640, 2003.

[6] R. Möller, "Insects could exploit UV-green contrast for Landmark navigation," *J Theor Biol*, vol. 214, pp. 619-31, 2002.