Invited Paper

Orientation-Selective VLSI Retina

Tim Allen, Carver Mead, Federico Faggin, Glenn Gribble

Synaptics, Inc.
2860 Zanker Rd. suite 105, San Jose, CA 95134

ABSTRACT

In both biological and artificial pattern-recognition systems, the detection of oriented light-intensity edges is an important preprocessing step. We have constructed a silicon VLSI device containing an array of photoreceptors with additional hardware for computing center-surround (edge-enhanced) response as well as edge orientation at every point in the receptor lattice. Because computing the edge orientations in the array local to each photoreceptor would have made each pixel-computation unit too large (thereby reducing the resolution of the device), we devised a novel technique for computing the orientations outside of the array. All the transducers and computational elements are analog circuits made with a conventional CMOS process.

1. INTRODUCTION

Our design of hardware to perform orientation pre-processing meets the needs of a feature-based pattern recognition system and is inspired by the structure of animal visual systems. We use a simple, pixel-wise local operation for producing three analog values that encode the magnitude and direction of the intensity gradient at each point in our hexagonal pixel lattice. The lattice itself is a silicon retina, based largely on work done at the California Institute of Technology by Carver Mead. The retina produces an edge-enhanced image, which then passes through an additional stage, located outside the array, that computes the orientation. The outputs of this stage provide an enriched feature set to the next level of our pattern recognition system.

2. WHY ORIENTATIONS?

One plausible way to make a pattern recognition system is to design a device that will identify salient features in an image and will compare the intensity and position of these features with known patterns. A question which naturally arises, then: What constitutes a salient feature? Intensity edges are often considered salient features, and many systems have been built for edge enhancement. Higher-level, more specific features might include oriented edge vectors. One could imagine constructing other simple features (corners, bars, Ts etc.) using these oriented edges as an input representation.

We duplicated the work of many researchers [1] in the image processing field by feeding small patches of digitized images to a simulated self-organizing learning network (in our case, a Kohonen net [4]). We presented our network with a large number of small subregions from both raw intensity and edge-enhanced images. We found that many of the Kohonen units adapted their weights to become feature detectors that respond to bars and edges in various orientations and positions. This result led us to believe that oriented bars and edges are, in the statistical vector-quantized sense, salient features.

The familiar results that Hubel and Wiesel obtained from their analysis of the mammalian visual cortex [2] underscore the importance of oriented-edge detection. These researchers found many cells that respond to oriented edges and bars in the input image. These simple cells do not take their inputs directly from the photoreceptors; considerable preprocessing is done before the information is passed to the simple cells in the cortex. The best-understood processing is center-surround (or lateral inhibition), wherein a strong stimulus in the center of a unit's receptive field tends to activate, whereas strong stimulus towards the periphery tends to inhibit the cell. Units with an opposite response pattern (off-center) also exist. The result of this local

operation is that units tend to respond to edges in the illumination rather than to the absolute level of the light. In short, animal vision systems perform a lateral-inhibition edge-enhancement step *before* computing edge orientations.

3. COMPUTATION AND ENCODING OF ORIENTATIONS

Our chip uses analog electronic elements to compute local edge orientations and to pass the information off-chip as analog-valued current signals. It is based on a simple algorithm for computing orientations, and on a scheme for encoding those orientations as signal levels. The computation chosen falls out neatly from the geometry of the pixel lattice, which is hexagonal to facilitate neighborhood operations used in the lateral-inhibition step. In such an array, each pixel has six nearest neighbors, and there are three principal axes of symmetry. We chose to encode the direction and magnitude of a gradient vector as three analog signed signals corresponding to its projection along each of these three principal axes.

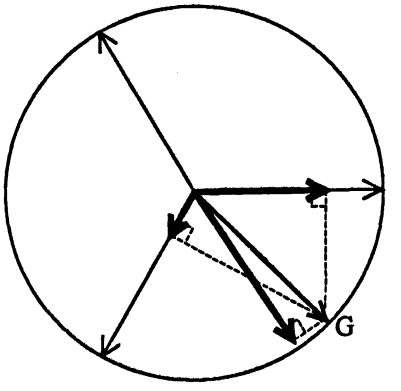


Fig. 1: Three-axis projection of gradient vector.

Figure 1 shows the local gradient vector (G) and its projections along the three major axes of the array. Notice that one of the components is negative. A simple unit that computes approximately the projection of the gradient along one of the axes is shown in Fig. 2(a).

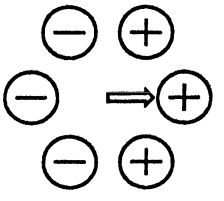
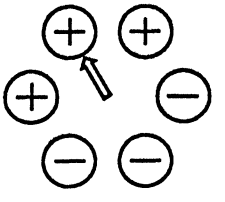
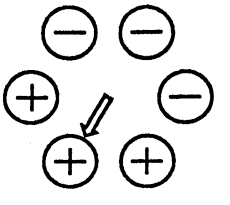


Fig. 2: (a) Type 1.



(b) Type 2.



(c) Type 3.

By adding the response of the neighbors to the right, and subtracting the response of the neighbors to the left, the unit will return a large response to a left-right gradient in its receptive field. There are three varieties of this unit, one for each principal axis of the array—see Fig. 2. Computing the components using this algorithm ignores the response of the unit in the center of the field. So be it. To encode the gradient vector everywhere in the image, we must compute all three components for every lattice point. From these three components, we can reconstruct the magnitude and direction of the local edges.

4. IMPLEMENTATION AS A SILICON RETINA

The silicon-retina paradigm, a VLSI chip with an array of photodetectors and local preprocessing electronics, has been described in detail by Carver Mead et al. [3]. In constructing our orientation chip, called RET1, we used as a starting point a center-surround retina, which implements lateral-inhibition processing with a resistive smoothing network and a differential amplifier in each pixel. The circuit uses 26 transistors per pixel to perform the center-surround computation. The basic array element, consisting of a pixel and its associated local processing, is 110 by 99 microns (in a 2-micron minimum-feature-size technology); a 100 by 100 array of these elements could fit on the largest die size commonly available.

In deciding how to implement the orientation computation, we drew heavily on results from our experiments with an earlier chip. We had built a previous orientation retina, called DIR34b, which uses a number of shortcuts to fit the computation into the pixel array. First, the pixel at the center of every hexagon is removed and is replaced by a processor of the type described in Section 3.

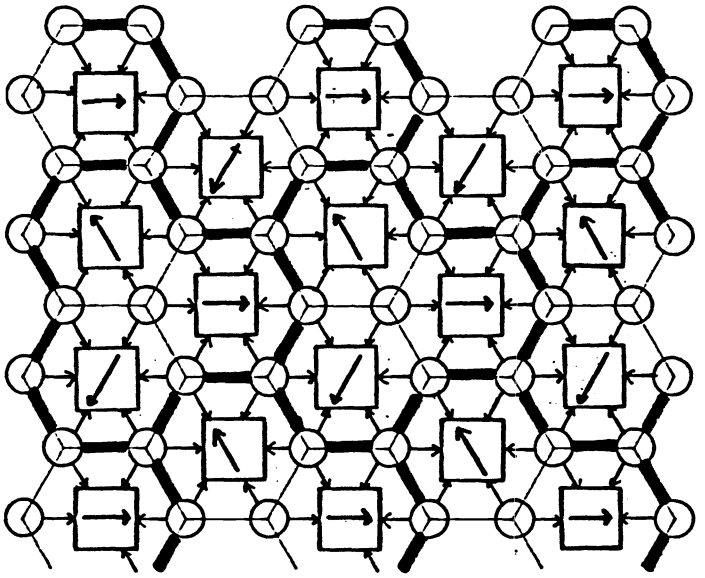


Fig. 3: DIR34b—Tiling with three types of processors.

Each processor computes only one of the three components of the local gradient, so we devised a tiling scheme in which all three types of processor are distributed evenly in the array (Fig. 3). The processors operate *not* on a center-surround preprocessed image, but instead directly on the intensity data as recorded from the neighboring pixels. Only one component is computed at any point in the array, so the resultant edge-orientation vectors have to be reconstructed from three components computed at three (slightly) different positions in the array. Because the gradient need not be uniform over this three-processor neighborhood, gross errors in the reconstruction can result. When we looked at images generated by the DIR34b chip, we noticed that many of the responses were directions different from the gradient, but that, for the most part, responses only occurred near edges in the image. We realized that we could obtain much better results if all three orientations were computed at *every* point in the array. We also learned to avoid deleting pixels from the array and replacing them with orientation processors, an approach that leads to a lower-resolution image and introduces inhomogeneities into the array; the latter interfere with the operation of the resistive smoothing network used in the center-surround computation.

The obvious lesson learned from DIR34b, then, was to include in each pixel-processor unit not only the circuitry neccessary for center-surround response, but also additional elements for implementing all three types of orientation detectors. Even a minimal implementation of these orientation detectors requires at least six transistors each, adding 18 more transistors to every cell. There is an even higher price to be paid

in wiring overhead for this computation. Not only does each pixel-processor need to communicate with its six neighbors via the resistive smoothing network, but also the center-surround response now needs to be passed out for each neighbor's orientation computations, and, conversely, each pixel-processor needs to recieve input from all six neighbors for its own computation. Furthermore, each pixel-processor unit now has four outputs (one for each orientation response, plus the original center-surround output), which implies four more wires in each column to scan the information out of the array. Trial layouts of such a scheme yielded unacceptably large pixel-processor units. We needed to find an alternative.

5. SCAN-TIME COMPUTATION

Our solution is to compute all three orientations at every pixel position, but *not* to do so by adding additional hardware to each unit. THe feasibility of our solution rests on two properties of the orientation computation being done: (1), we must necessarily scan information out of the pixel-processor array sequentially, row by row, and (2), the orientation computation involves only the six nearest neighbors of a pixel, and, therefore, involves only three rows (the resistive net response at each pixel, on the other hand, is affected by *all other pixels* in the array).

RET1 is implemented as an array of center-surround pixel-processor units, which perform no orientation-selective function at all. There is an additional processing and storage block along one edge of the array, roughly as large as one row of pixels for computing the orientation response.

To understand the operation of this system, we must first understand how information is scanned out of the array.

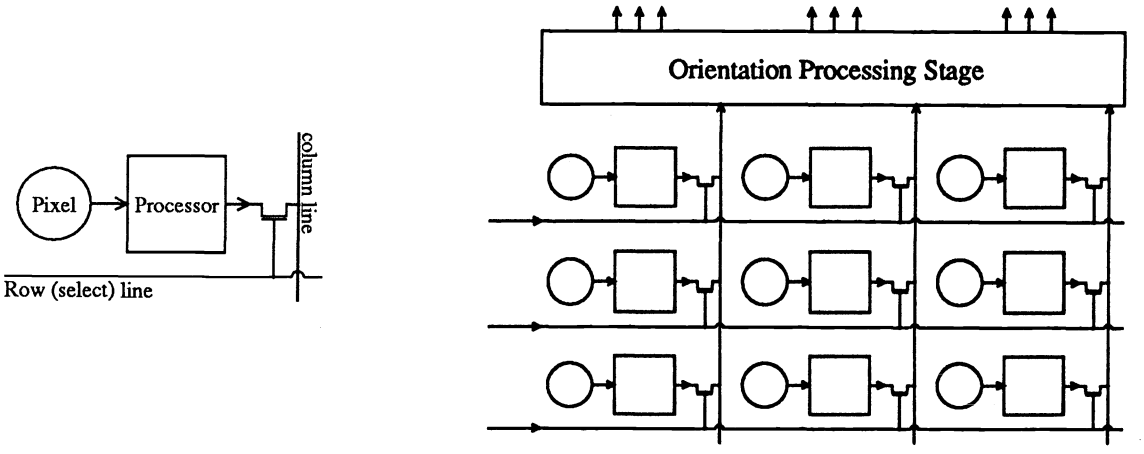


Fig. 4: (a) Array element. *(b) Array section.*

Figure 4(a) is a simplified schematic of one of the processor elements. The output of the unit is connected to the vertical column line when the unit is selected by an "active" signal on the horizontal row line. Fig. 4(b) shows an array of these units. Whenever one of the horizontal row (select) lines is active, all the pixel-processors in that row are connected to the vertical column lines. Only one select line is active at a time; otherwise, more than one processor's output would be connected to each vertical column line. The column lines, then, are connected to the output of the currently selected row. We scan the rows in order (top to bottom, say) by connecting the select lines to the stages of a shift register. One bit is entered into the register; as the bit moves along, the select lines are activated in order until the register is empty, whereupon another bit must be entered. Each time a bit is entered and passes through the register, one full image (frame) is scanned out in parallel via the column lines. The orientation-processing stage takes as input the vertical column lines. This stage contains a two-level analog sample-and-hold chain for every vertical column line. Having taken one sample per selected row, the orientation processing stage contains a memory of the last two rows scanned (Fig. 5). These last two rows, along with the row currently selected, provide the stage with enough information to compute one horizontal slice of the orientation image. The circuitry for performing the three orientation sums is nestled between the sample-and-hold chains atop each column. As

information is scanned out of the retina, passing along the column lines and through the sample-and-hold chains, the three-valued orientation image is computed one row at a time and is passed off-chip.

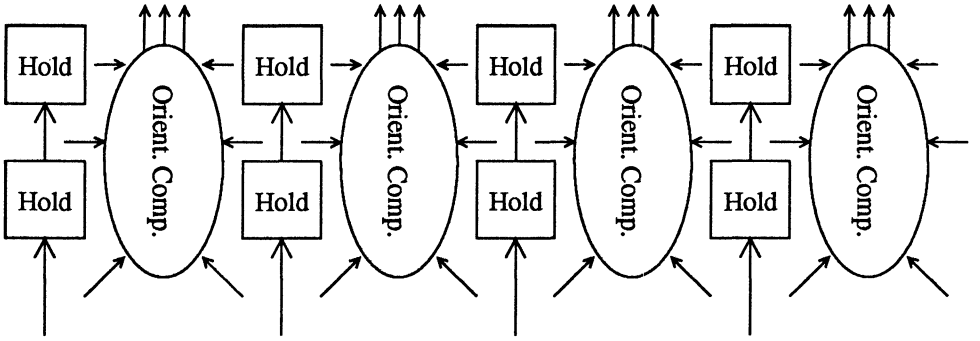


Fig. 5: Orientation-processing stage.

The processing stage is roughly the same size as a row of pixels. If hardware to perform this computation were added to every pixel, the array size would grow enormously. Notice that we could do virtually any limited-neighborhood computation in such a scan-through stage by replacing the orientation-sum devices with different circuits. As an added benefit, since the orientation computer is connected to one end of the column lines (say, the top), we can still have access to the raw center-surround information just by looking at the *bottom* of the column lines as we scan the retina, giving us access to both center-surround and orientation-processed images.

At first glance, we appear to have sacrificed one of the greatest benefits of the silicon retina by making this scan-time computation stage. The orientations are no longer computed continuously and in parallel everywhere in the array; they are computed sequentially and discretely. But even if the orientations were computed instantaneously and in parallel inside each pixel-processor, we would still need to scan the information out of the array. In practice, scanning is the bandwidth-limiting step. As long as the sample-and-hold chains and the orientation processors are able to operate as fast as does the scanner, the loss of parallelism will not matter. Although we constructed RET1 as a proof-of-concept experiment rather than a high-bandwidth imaging device, we have built other separate experimental scan, sample-and-hold, and computation circuits that can process one line per microsecond (giving a frame rate of 10 kHz for a 100-pixel tall orientation retina). If we assume that each orientation computation requires six integer operations, then a 100 by 100 pixel chip has an equivalent computation rate of 1800 MIPS. This estimate excludes the center-surround processing, which we conservatively estimate to be 10,000 MIPS. Images taken from RET1 are presented in Figs. 6 through 8. These results, combined with the promise of great computational speed, make this technology attractive for fast, compact, low-power image preprocessing.

6. LIMITATIONS AND FUTURE WORK

The image presented in Fig. 6 shows how this chip can accurately compute the direction and magnitude of edges in an input image. The data for this image were read from the chip as analog voltage levels, were encoded by an analog-to-digital converter, and were passed to a computer for processing and display. The image needed to be modified before display for only one reason: The center-surround and orientation responses for any point in the lattice had large, fixed-pattern offsets. Because the processors associated with the pixels need to be compact, they are fairly unsophisticated circuits made from the smallest transistors possible. When an array of such simple circuits is made from minimum-sized devices in a standard CMOS process, large response differences from cell to cell can occur due to variations in lithography, dopant densities, and other process parameters. Such variations can be large indeed (as much as a factor of two in output current between small amplifier stages), but can be kept to a minimum with careful circuit layout. For our

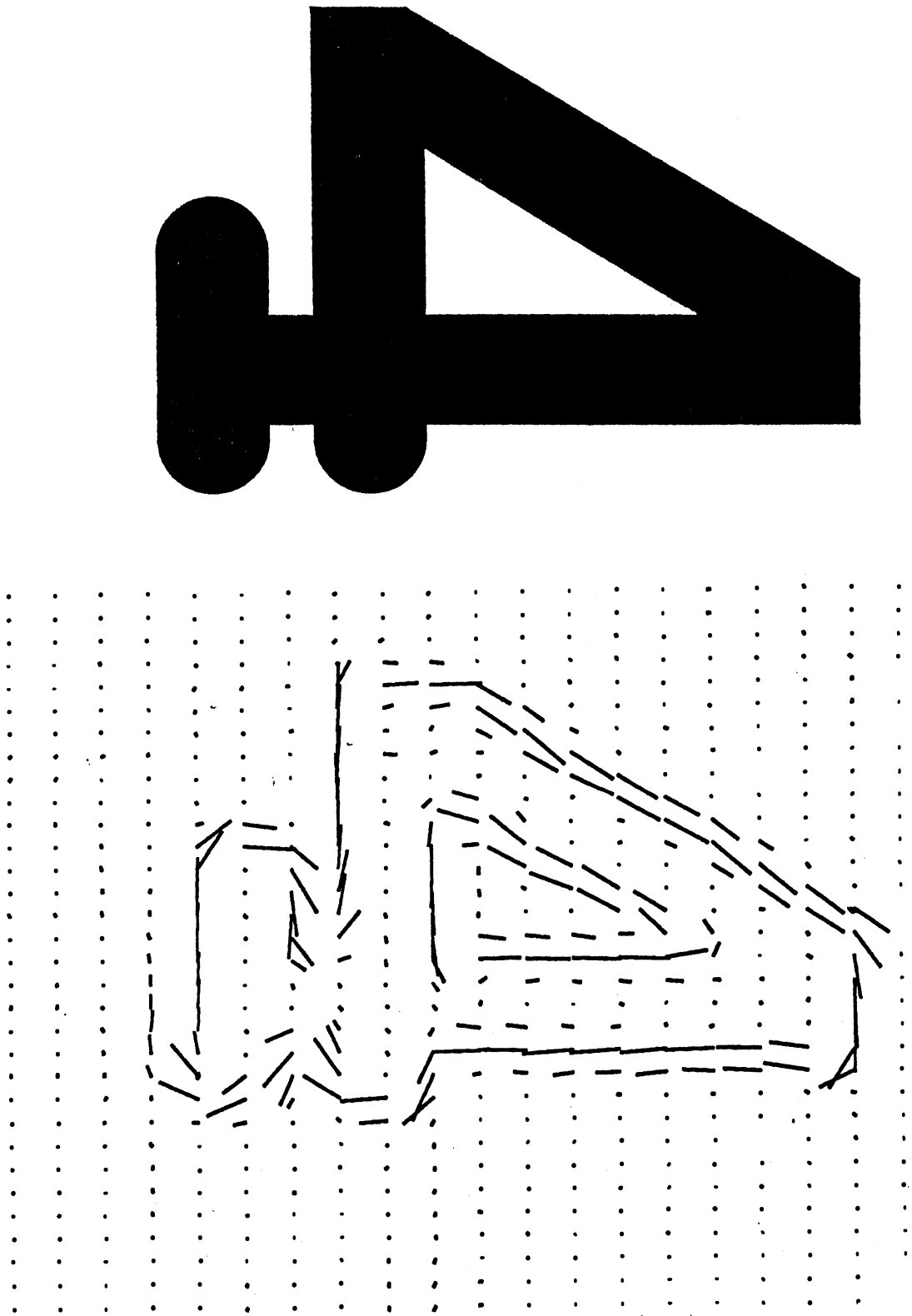


Fig. 6: RET1–stimulus (solid) and output (lines).
(32 by 20 pixel array)

chip, the offset differences were roughly as large as the range of the response (in short, the noise was as large as the signal). At first, this would seem to be a hopeless situation; however, the noise is a *fixed pattern*, and we can be eliminate it entirely by subtracting an image of the offsets, pixel by pixel, from the output image of the chip. We obtain an image of the offsets by providing a uniform input stimulus (a blank white page), which, for a perfect device, would give a zero signal level for the center-surround and orientation outputs. Any signal that does appear at each point in the image under this reference condition is taken to be the offset at that pixel. To remember the image of these offsets and subtract it from all subsequent data, we used a digital computer. This was the only type of processing we did off-chip to display the image shown in Fig. 6.

These offsets are an obstacle to widespread commercial application of this retina chip unless the outputs are fed to a system that could correct for the fixed-pattern noise (in our case, by storing the noise pattern and subtracting it from each frame). We are currently working on techniques for canceling the offsets on-chip.

7. REFERENCES

1. Barrow, Harry G. "Learning of Receptive Fields." Schlumberger Palo Alto Research, Palo Alto, CA 94303 (Personal Communication to Carver Mead, June 1987)
2. Levine, Martin D. *Vision in Man and Machine,* pp. 151–210 (McGraw-Hill: New York, 1985)
3. Mead, Carver A. and Mahowald, Misha A. "A Silicon Model of Early Visual Processing." *Neural Networks* [Vol. 1]:p. 91 (Pergammon 1988)
4. Kohonen, T. *Self-Organization and Associative Memory* pp. 119–157 (Springer-Verlag: Berlin 1984)