A Block Matching Approach for Movement Estimation in a CMOS Retina: Principle and Results

David Navarro, Guy Cathébras, Fabrice Gensolen

A Block Matching Approach for Movement Estimation in a CMOS Retina : Principle and Results

D. Navarro, G. Cathébras, F. Gensolen
Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier - France
navarro@lirmm.fr

Abstract

We present in this paper a silicon retina - specific CMOS image sensor – dedicated to movement estimation. This circuit has been fabricated in a standard 0.35µm CMOS process. The pixel is composed with both sensitive elements - a photodiode is used to convert photons to electrons - and electronic components for computations.
Its goal is to extract the optical flow from two acquired images. Movement estimation is a very often used task after image acquisition, which usually needs a sensor, a processing unit (DSP, FPGA) and external memories [5]. Here, the functionality is achieved in a single chip, by an electrical way, giving compactness and low power to this costing task.
The movement estimation algorithm we use comes from [2], its advantages are: robustness with real sequences, its real use in robotics, and its good electrical mapping. The circuit's architecture is divided in two parts, corresponding to the two steps of the original algorithm.

1. Introduction

A silicon retina is a specific image sensor in which an analog and/or digital signal processing circuitry is integrated, in order to compute in a parallel way a low-level image processing.

We generally found electronics close to the photosensitive element [1]. The output of an electronic retina is also not necessarily an image but a higher level information, extracted from the sensed image.

The processing we wish to implement in our CMOS retina is movement estimation in real time. This study is based on an algorithm that is really used in robotics, meanwhile, it requires a software processing via a laptop, or a costly (power consumption) hardware computation, requiring a sensor, an FPGA, and external memories [5].

The architecture of our circuit is characterised with two parts: a coding (non parametric local transform) of the image, close to the sensitive element; followed by a parallel search step, that can be done, in a systolic fashion, at the bottom of the matrix. The output of the circuit is a numeric word that codes, for each pixel of the matrix, its movement from the first acquired image to the second one.

2. Movement estimation

Many algorithms exist to extract movement information from a sequence [3], and it appears that a few are well-suited for real image processing. For example, most of actual retinas, based on the image's derivatives, can't take light variations into account. One of the most efficient way to compute movement is to use algorithms that are based on a block matching method. The well-used MPEG algorithm belongs to that family of algorithms. The problem is that these algorithms use a costly iterative processing of images; so, as far as we know, no such retina exist yet. We also expose the work we did in that way: the first retina using a block matching technique in full 2 dimensions – 8 directions to be precise.

The theoretical optical flow is stemming from a mathematical differential equation [3], we will only focus here on its meaning. The optical flow is a way to represent movement between two images. Figure 1 also shows the optical flow of a scene.

The right picture in figure 1 represents a dense vector field, indicating the movement of each pixel.

Figure 1: An image from the initial sequence and the simulated optical flow

The papers we found on motion computing with block matching algorithms are mainly software computing. [4] proposed a computation on a multi-FPGA card, to obtain

motion computation at video rate. Nowadays, less hardware is needed: [5] proposed a single FPGA implementation, but external memory is still needed. To complete the hardware we must also count an image sensor in addition to the FPGA and the external memory: a power consuming system on board. The solution we propose is single chip, less power consuming, and can process faster than video rate.

The algorithm we choose comes from [2]. Its main advantage is its robustness with real - also highly textured - scenes. Indeed, the authors announced their method was efficient, even with highly textured images, and luminosity changes. We simulated that algorithm in C language, and interesting results appeared on real scenes: about 85% of the pixels are rightly matched, while other algorithms match only 40% of them. The computation time was about 2 seconds with a 400 MHz processor. Then, we have remarked the algorithm is very iterative on each pixel, and by examining it, we concluded it can be strongly fastened with a parallel computing structure, which can be an image matrix array.

That algorithm is divided in two parts: first a coding of blocks of pixels. This coding gives the robustness to the algorithm. Each pixel - considered in a central position - is replaced with a code that corresponds to the binary comparison of its grey level with the eight neighbours' ones. Let's note Ic the grey level of the central pixel, and B_j the related result, we have with its eight neighbours j the coding in Figure 2.

For each neighbour j :

If $I_c > I_j$ then $B_j = 1$

Otherwise $B_j = 0$

Figure 2 : Determination of the block codes

Since the coding involves the eight neighbours, the resulting code is eight bits long and the transformed image has the same size than a "classical" grey level image. Moreover, there is no need for a costly analog to digital converter. After the two images are acquired and coded, a search step starts to match the codes, so each pixel in the first image I_1 is searched in the second one I_2. To do this, the minimal Hamming distance is searched in order to match each code in I_1 with the same code in a search window in I_2.

A movement is also determined for all the pixels. We will see in next section how is composed the core of our retina architecture.

3. The Retina architecture

1.1 Code computation

As we saw previously, the algorithm is composed of two steps to compute the movement. These steps have been mapped into two blocs. The first one computes the pixel's codes, the other one searches the matching pixels.

The codes are created by grey levels comparisons of a central pixel and its neighbours. In our architecture, we compare the luminosity of the pixels. Figure 3 shows the principle of the photodetector structure we choose.

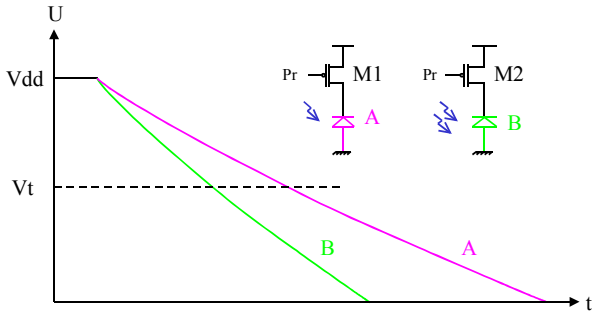


Figure 3 : Photodetector schematic and its discharge

A photodiode converts photons to electrons, creating the also called photocurrent.. At the beginning, the N-diffusion / P-substrate capacitance of the photodiode is pre-charged to Vdd, through the transistors M1 and M2. After that reset state, the photocurrents discharge the capacitances voltages. If we consider that photodiode 'B' is lighter than photodiode 'A', The B discharge is faster. Its discharge curve also crosses first a threshold voltage Vt. The pixel architecture is shown in Figure 4.

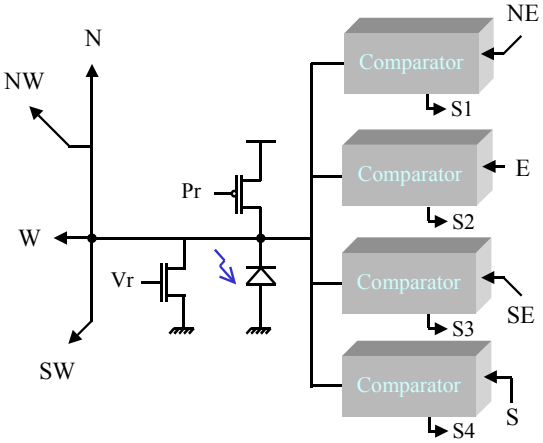


Figure 4: One pixel architecture

Each pixel has 8 contacts with the surrounding pixels: the North West one (NW), the North one (N), the North East one (NE), the West one (W), the East one (E), the South West one (SW), the South one (S), and the South East one (SE). For symmetry reasons, only 4 comparisons per pixel are needed, as the 4 other comparisons occur in the 4 last neighbouring pixels. We also avoid to store redundant information, giving smallest pixels.

The goal of each comparator is to compare the two analog inputs, and to store the binary difference. One half of the comparator is symbolised in figure 5.

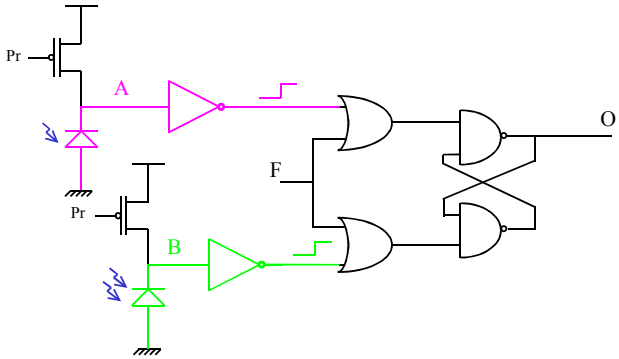


Figure 5: The comparison architecture

The input named F is a forcing input, its level is 1. O is the output. As we saw, pixels A and B are forced to 1, then the light discharges the photodiodes. The lighter pixel (B) will first cause a 0 to 1 edge at the input of the 'RS' latch. The response of the latch is very short, nearly instantaneous. Then, the A pixel edge will not change this state. According to the first incoming edge, O will be set at 0 or 1. Then, the forcing input F is set to 1, the latch is then locked, whatever happens on A and B. The second identical structure of the comparator then stores the code of the second image.

1.2 Pixel matching

The pixel matching is done by comparison of the Hamming distances. A null distance indicates a pixel matching. As a search window is considered, several lines and columns are addressed (depending on the size of the window). The matching step is realised with N elementary parallel processing units.

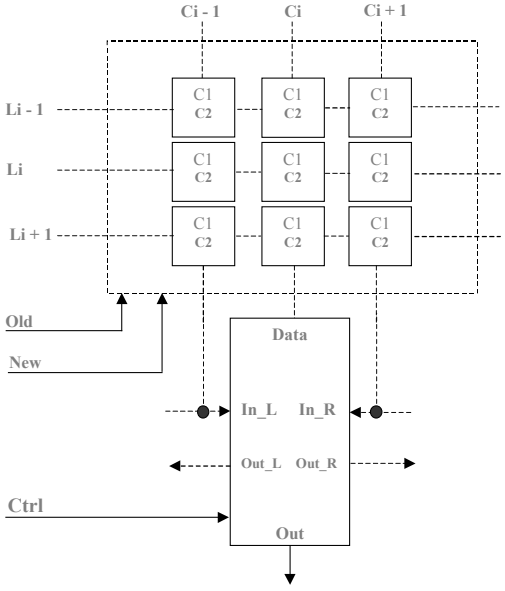


Figure 6: The numeric unit

N is the horizontal resolution of the matrix. That structure uses the natural parallel power of computation of arrays. To parallelize the computations, the units can communicate horizontally. Each processing unit can also give its incoming Data to its neighbours. The synoptic of a slice of this systolic processor is illustrated in figure 6. In_L and In_R are the horizontal – left and right - inputs, Out_L and Out_R are the horizontal outputs to the neigbours. Data is the input of one matrix column, and Out is the movement information for pixels Ci.

This elementary unit has been programmed in VHDL and synthesised with Cadence tools. This block takes place above the 'acquire and coding' array presented in 3.1, and have a very small size.

4. Design and fabrication

For the first hardware implementation of this kind of algorithm, we have chosen to make a 10x10 test matrix in a larger test vehicle that is shown on Figure 7. This chip was realized in the AMS 0.35 µm CMOS process.

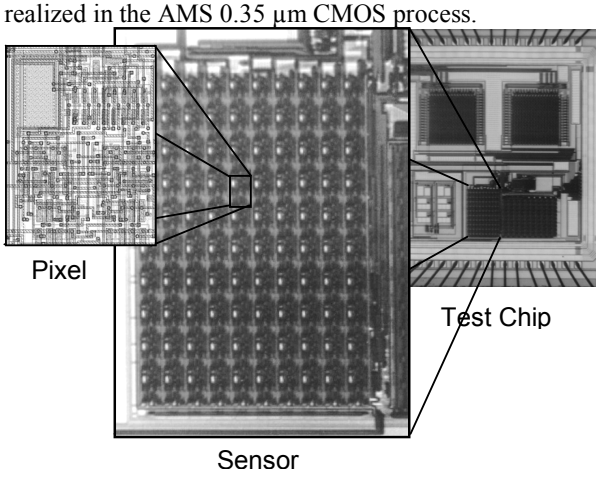


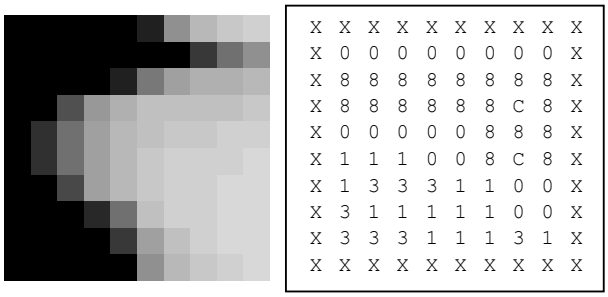
Figure 7: Test matrix microphotograph

The pixel area is 40µm x 50µm, with 12% fill-factor. With the same pixel, a more realistic 100x100 retina would take 20 mm². The digital block takes 50µm x 150µm and can be directly abutted to the matrix. One block per column is required. To have a maximum observability and easier debugging, only the image coding has been integrated in the test chip. Indeed, as the numeric block functionality is sure, we found better to fit it in a programmable device. Moreover, that approach will allows us to test several search window sizes.

5. Results

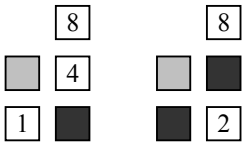
The average power consumption of the coding matrix is about 5 µW (this lead to 5 mW for a 100x100 matrix).

The circuit performs the image coding. Figure 8.a shows the result of an acquisition : a simple image and its codes. Because of borders effects, only 8x8 codes are meaningful. It is possible to match the codes and the pixels luminosity in the image. Each code represents, on four bits, the results of the comparison of the corresponding pixel in the image with its NE, E, SE and S neighbours. Figure 8.b clarifies this notation.



X	X	X	X	X	X	X	X	X	X
X	0	0	0	0	0	0	0	0	X
X	8	8	8	8	8	8	8	8	X
X	8	8	8	8	8	8	C	8	X
X	0	0	0	0	0	8	8	8	X
X	1	1	1	0	0	8	C	8	X
X	1	3	3	3	1	1	0	0	X
X	3	1	1	1	1	1	0	0	X
X	3	3	3	1	1	1	3	1	X
X	X	X	X	X	X	X	X	X	X

(a) image and 4 bits codes (hex)



(b) code meaning
left, value=2 ;
right, value=5

Figure 8: An image and its 4 bits codes

Given that the test chip is a multi-projects vehicle, it uses multiplexed IOs. Also, to facilitate its use, we interfaced it with an Altera-Excalibur prototyping platform (System on a programmable chip). This solution combines the flexibility of a field programmable device with the ease of use and debug of a micro-controller (NIOS embedded processor).

The principle of the comparisons has both advantages and drawbacks. The drawback of this architecture is that the comparison time is not known precisely (see figure 3), as it depends on light conditions. The coding is totally finished after the coding of the darker pixel. If the comparison time is not long enough, the forcing input F (see figure 5) will probably cause a wrong 1. In figure 8, as the timing was respected, the problem is not present. We note that timing problem is in fact inherent in image acquisition. The advantage of our structure is the automatic coding : whatever the light conditions, the comparison will be done. If we consider a small standard deviation in the grey levels of the image, the coding phase is short; and the speed of the computation is controlled by the brightness of the scene. We note that this structure acquires an histogram-equalised image. It is indeed possible to restore a thresholded image, as we know the absolute values of the gradients.

The movement is estimated with 2 acquired images. We take a simple example for a non-moving scene under continuous light. The codes are compared in figure 9. We see that there are mismatches between codes, 23% of them (in bold-italic) are different. This is due to temporal noise (i.e from frame to frame) that DS (double sampling) circuits corrects in standard imagers. In fact, the comparators are so sensitive that they code more than 256 grey levels. This excess of sensitivity causes mistakes because of the low signal / noise ratio in integrating CMOS imaging vision chips. A minimal Hamming distance search will reduce the error to 8%, but yet, we only estimate the required area for a matching (0 Hamming distance) numeric block.

In this configuration, although we use a robust method, we obtain a movement estimation in moving scenes with about 60% of representative vectors. A middle-range sensitivity comparator would avoid those quantification problems.

3	4	E	4	C	D	D	F
0	0	4	D	B	0	3	7
D	C	9	3	2	1	6	7
0	1	3	0	5	A	0	F
0	2	4	9	3	4	A	7
0	4	8	2	2	0	0	F
0	C	D	C	E	C	8	7
9	1	3	1	5	D	D	F

3	4	***C***	***C***	C	D	D	F
0	0	***0***	D	B	0	***1***	7
D	C	9	3	***3***	***0***	***2***	7
0	1	3	0	***7***	***0***	0	F
0	2	4	9	***F***	***0***	A	7
0	***E***	8	2	2	0	0	F
0	***4***	D	C	E	C	8	***F***
9	1	3	1	5	D	***9***	***7***

Figure 9: Codes of 2 acquired images

6. Conclusion

We presented in this paper a silicon retina dedicated to movement detection. This circuit has been fabricated in a standard 0.35µm CMOS process. Its principle is to integrate the computation of a non parametric local transform directly at the pixel level. A digital block, presently implemented on an external FPGA, allows to compute the optical flow between two successive images.

The retina gives a thresholded image, without exposure control, and movement estimation. It appears that the high sensitivity of electronics causes mistakes in results, because of the noise in integrating CMOS imaging products. As DS integration is impossible at pixel level in such a retina, we will then focus on a retina having a middle-range sensitivity to avoid quantification problems.

References

[1] A. Moini, "Vision chips or seeing silicon", Technical Report, Centre for High Performance Integrated Technologies and Systems, University of Adelaide, 1997. http://www.eleceng.adelaide.edu.au/Groups/GAAS/Bugeye/visionchips/

[2] R. Zabih,, J. Woodfill, "Non-Parametric Local Transforms For Computing Visual Correspondence", in proc. of 3rd European Conference on Computer Vision, 1994, pp. 151-158.

[3]. S.S. Beauchemin, J.L. Barron, "The computation of Optical Flow" ACM Computing Surveys, Vol 27, n°3, 1995, pp. 433-467.

[4] J. Woodfill, B. Von Herzen, "Real-Time Stereo Vision on the PARTS Reconfigurable Computer". In proc. of IEEE Symposium on Field-Programmable Custom Computing Machines, 1997, pp. 201-210.

[5] M. Arias-Estrada, E. Rodríguez-Palacios, "An FPGA Co-processor for Real-Time Visual Tracking", in proc. of 12th International Conference on Field-Programmable Logic and Applications (FPL 2002), ISBN 3-540-44108-5, pp. 710-719.