MVA'90 IAPR Workshop on Machine Vision Applications Nov. 28-30,1990, Tokyo

SINGLE-CHIP HIGH-SPEED COMPUTATION OF OPTICAL FLOW

Per-Erik Danielsson*, Pär Emanuelsson*, Keping Chen**, Per Ingelhag**
and Christer Svensson**
*Department of Electrical Engineering and **LSI Design Center
Linköping University
S-581 83 Linköping, Sweden

ABSTRACT

This paper consists of two parts. In the first part we describe how to compute optical flow from second derivatives. In the second part we describe the VIP chip and how this chip can compute a 512x512 optical flow field at nearly full video rate.

INTRODUCTION

Optical flow is a means for the analysis of image sequences. In particular it should be helpful in segmenting rigidly moving objects from a non-moving background or from a background the movement of which could be predicted. The latter case is very often at hand when the image sequence comes from a moving camera in a 3D-world. The potential applications of optical flow and various segmentation algorithms based thereof are to be found in target tracking, automatic inspection, computer vision for autonomous robots and vehicles and related areas.

These applications have been slow in coming because of computation complexity and high cost. In this paper we will address this problem in two ways. Firstly, we advocate the use of second derivatives to solve the optical flow equation explicitly. Secondly, we will present a new chip, called VIP (VIdeo Processor), which comprises 512 bit-serial processors and seems rather ideal for the task.

In the first part of the article we demonstrate that one can obtain a useful optical flow field from the second derivatives using the following computations.

- 25 to 50 operations of type addition or subtraction, the number depending on noise levels which require more or less smoothing in the operators.
- 6 multiplications
- 2 divisions

In the second part of the article we demonstrate how the VIP-chip is able to perform the computation at near-video rate (11-15 frames/second). The VIP-chip is supported by a control unit and a limited set of external memories.

OPTICAL FLOW FROM SECOND DERIVATIVES

The basic equation for optical flow is given by (1).

$$f_x \cdot u + f_y \cdot v + f_t = 0 \qquad (1)$$

Like some other authors [1], [2] we make the smoothness assumption that the derivatives of the flow components (u, v) are zero. Then, the derivative in x and y of (1) gives us two equations from which we explicitly can obtain both u and v.

$$u = \frac{f_{xt}\cdot f_{yy} - f_{yt}\cdot f_{xy}}{f_{xy}^2 - f_{xx}\cdot f_{yy}} = \frac{f_{xt}\cdot f_{yy} - f_{yt}\cdot f_{xy}}{G} \qquad (2)$$

$$v = \frac{f_{yt}\cdot f_{xx} - fxubxt\cdot f_{xy}}{f_{xy}^2 - f_{xx}\cdot f_{yy}} = \frac{f_{yt}\cdot f_{xx} - f_{xt}\cdot f_{xy}}{G} \qquad (3)$$

This method of obtaining the optical flow requires that we compute (estimate) the five second derivatives

$$f_{xx}, f_{xy}, f_{yy}, f_{xt}, f_{yt}$$

throughout the 3D space (x, y, t).

The so called aperture problem for computation of optical flow manifests itself in the denominator $G = f_{xy}^2 - f_{xx}f_{yy}$ which is nothing but the Gaussian curvature for the f(x, y) surface. When the local variation in f(x, y) is one-dimensional only, the Gaussian curvature is zero and in this case we cannot estimate the optical flow. The two equations (2) and (3) will then deliver a result of type 0/0 which is undefined but a quite reasonable response under the given circumstances. In fact, the quantity |G| can be used as a certainty factor in a post-processing procedure to resolve ambiguities and enforce global coherence as will be shown below.

We intend to implement the computation of (2) and (3) in a manner described by Figure 1. The second derivatives are effectively computed from first derivative estimators in two steps using a 3x3x3 kernel of the type shown in Figure 2.

The net result is that the second derivatives are computed from a 5x5x5 neighborhood which make them remarkably noise insensitive. The suggested set of kernels can be seen as generalized Sobel filters and are perfectly decomposable which makes it possible to compute all derivatives using only 17 additions and 8 subtractions between nearest neighbors [3].

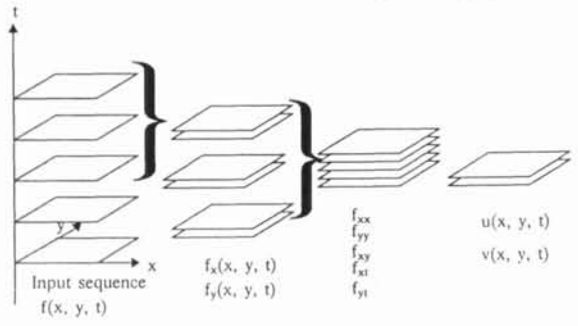

Figure 1

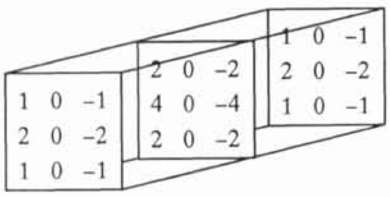

Figure 2

To enforce local coherence (smoothness) we propose the following. Let us introduce the notation for the numerators and denominators of the (u, v)-vector in (4).

$$(u,\ v) \;=\; \left(\frac{u_n}{G},\ \frac{v_n}{G}\right) \qquad (4)$$

A smoothed result (U, V) from a neighborhood Ξ of the (u, v) image using |G| as a weighting factor can then be computed as

$$(U,\ V) \;=\; \frac{\sum_{i\in\Xi}\left(\frac{u_{n_i}}{G_i},\ \frac{v_{n_i}}{G_i}\right)\cdot|G_i|}{\sum_{i\in\Xi}|G_i|} \;=$$

$$=\;\left[\frac{\sum_{i\in\Xi} sgn(G_i)\ u_{n_i}}{\sum_{i\in\Xi}|G_i|},\ \frac{\sum_{i\in\Xi} sgn(G_i)\ v_{n_i}}{\sum_{i\in\Xi}|G_i|}\right] \qquad (5)$$

where the sums are computed over the given neighborhood and the subscript i indicates a pixel position, within the neighborhood. As seen from (5) we manage to simplify this computation into averaging the two numerators and the common denominator G..

The above procedure has been implemented and tested on a number of sequences. The result (U, V) is a vector field which one preferably should present as a color-coded image. In short of this we can only state here that the result seems fully comparable, if not superior, to more computation demanding iterative solutions.

THE VIP CHIP

The VIP (VIdeo rate Processor [4]) has a floor-plan according to Figure 3. A memory array of 512x256 static bit-cells is surrounded on two edges by two 256 linear arrays of processors. Each processor accesses one column of 256 bits of the on-chip 512x256 memory. The 512 processors are identical and each one consists of four major parts, each part forming two 256-element arrays as shown in Figure 3. Functionally, the chip is a 512-processor system organized as a linear array. The two shift registers serve as the necessary means of communication side-wise. The VIP is an SIMD-machine. Hence, there is a common control/address bit-vector which is generated in an off-chip control unit.

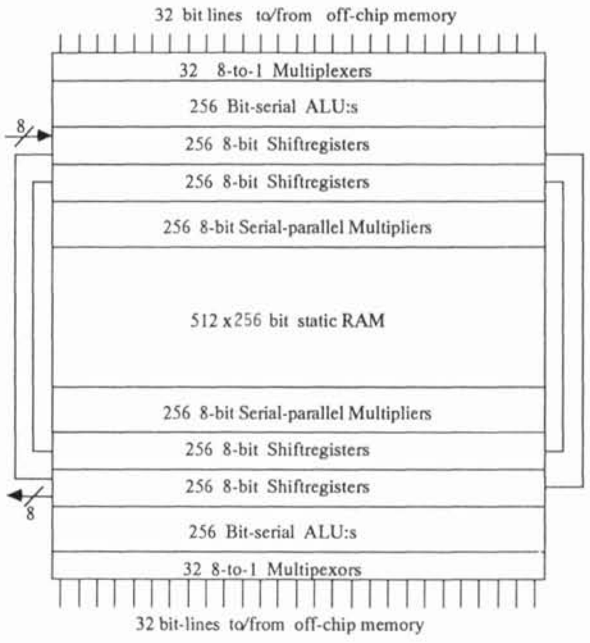

Figure 3

One of the 8-bit shiftregisters are used for input/output of video formatted data. With 20 MHz clock rate the bandwidth is 20 Mbyte/s. Notice that both shiftregisters can transfer data in both directions. For intermediate data the chip communicates over 64 bit-lines to off-chip memory modules. In what follows we assume that these memories are fast static RAM's with a cycle time of 50 ns. Hence, the memory band-width is 160 Mbyte/s.

As was seen above we need four intermediate image frames which amount to at least 4x8x512x512=8Mbit (more if we don't truncate some results). To be on the safe side we assign 16Mbit=eight 8x256k SRAM's. The on-chip memory of just 256 kbit will then buffer lines rather than full images. The **major cycle** of the processing is to produce one or several new lines of intermediate and final results. 512 such major cycles produce one image. If the algorithm, as in our case requires four intermediate images and produces two output images (U, V) we have to do the following in each major cycle. See Figure 4.

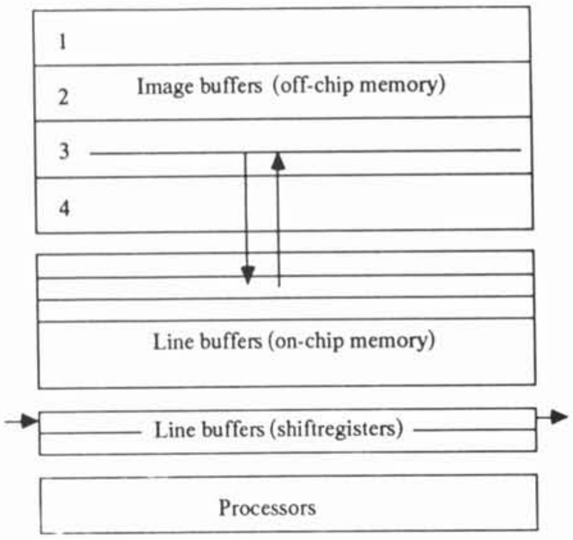
1
2
Image buffers (off-chip memory)
3
4
Line buffers (on-chip memory)
Line buffers (shiftregisters)
Processors

Figure 4

Load one line from I1, I2, I3, I4
Store one line to I1, I2, I3, I4
Shift in one line from input image I_0
Shift out two lines to output images (U, V)

Assume as a worst case that we assign 16 bit/pixel to all images I1, I2, I3, I4. Then, the total data transport in one major cycle is 2x4x512x16 bit=8 kbyte and in one image cycle 4 Mbyte. Thus, the available bandwidth of 160 Mbyte/s maximizes the frame rate to 40 frames/s.

The "video" input and output images are assumed to consist of one byte/pixel. Therefore they contain 256 kbyte each. One line of input data can possibly be shifted in at the same time as one of the output data lines but we don't count on this. Thus, the total amount of data per frame to be shifted in or out is 0.75 Mbyte, which limits the frame rate to 20 Mbyte/s / 0.75 Mbyte = 27 frames/sec.

The architecture of the 512 processing elements is shown in Figure 5. All components are arranged vertically around a single bit bus. The micro instruction has six fields, each one controlling the six following units in Figure 5.

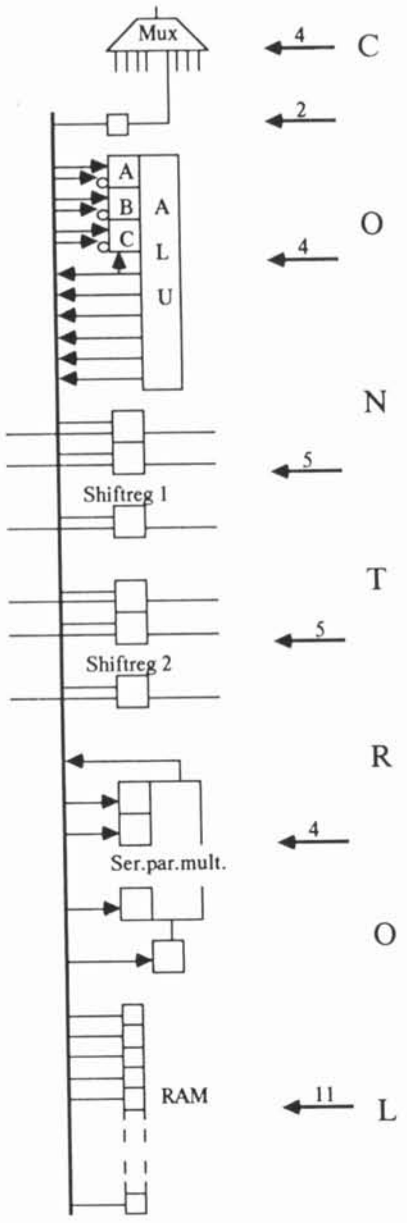

Figure 5

Off-chip memory interface
Arithmetic/Logical Unit
Shiftregister 1 I/O connected
Shiftregister 2
Serial-parallel multiplier
On-chip RAM

The ALU is a bit-serial device which has been used also for the PASIC chip. It has a general purpose capability which has been demonstrated for various algorithms [5], [6]. In the present application we intend to use it mainly for addition, subtraction and division operations. Hence, we will only use the SUM output and carry feedback of ALU-functions. However, the ability to load the three input registers A, B, C with inverted inputs and the various logical function outputs makes it possible to generate practically all three-input Boolean functions [6].

It is easy to see that addition and subtraction require three cycles per bit. The two input bits can be fetched over the bus from any of the three sources, shiftregister 1, shiftregister 2 and RAM. The output bit can be stored at any of these places or the serial-parallel multiplier or the off-chip memory interface. An 8-bit addition producing a 9-bit result takes 25 clock cycles.

Suppose we can match and interleave the I/O-operations transparent (or close to transparent) to the processing and that we on the average use 12 bits per pixel. The 25-50 add/sub operations in the optical flow computation would then require a minimum of

$$36 \cdot 25 \cdot 512 \ = \ 460,800 \ \textit{cycles/frame}$$

When using larger convolution kernels for the derivative estimation and larger smoothing kernels for the post-processing this number increases to approximately

$$36 \cdot 50 \cdot 512 \ = \ 921,600 \ \textit{cycles/frame}$$

The bit-serial multiplier is described in some detail in Figure 6. One of the operands in the multiplication (a_0, a_1, ..., a_7) is fixed to eight bit. This bit vector is loaded into eight latches in eight initial cycles. The arbitrarily long operand b_0, b_1,....., b_{n-1} is then serially, MSB first, furnished to the remaining input in every second clock interval. In every other second interval the output bit is gated to the bus and stored in registers or RAM.

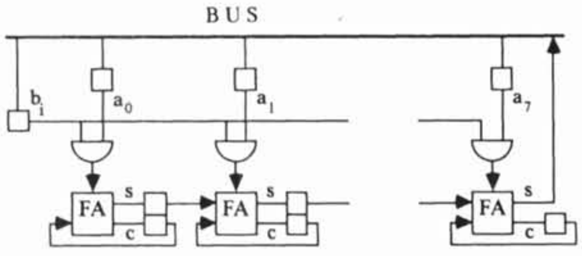

Figure 6

The cascaded full adders perform addition up to the point when the sign bit b_0 arrives. This is known to the microprogram which then changes the function to subtraction via a special control signal. The multiplication of a signed 8-bit number with a signed n-bit number produces a signed 8+n-1 bit product and takes

$$8 + n + 8 + n - 1 \ = \ 2n + 15 \ \textit{cycles}$$

Multiplication of two eight bit numbers then takes 31 cycles. Multiplication with a longer operand than 8 bit requires two separate multiplications followed by a separate addition process. Assuming truncation to a 16 bit result, a 16x16 bit multiplication then takes

$$(8 + 16 + 8) + (8 + 16 + 16) + 3 \cdot 16 \ = \ 120 \ \textit{cycles}$$

The six multiplications in the optical flow formulas using 16 bit precision then take at most

$$6 \cdot 120 \cdot 512 \ = \ 368,640 \ \textit{cycles/frame}$$

The last step in the optical flow computation is the two divisions yielding the final result (U, V).Using the present bit-serial ALU in VIP a division takes approximately 8 n^2 cycles producing an n bit quotient from an n bit divisor. With n=8 this step then requires

$$2 \cdot 8 \cdot 8 \cdot 8 \cdot 512 \ = \ 524,288 \ \textit{cycles/frame}$$

By introducing a data controlled selector latch at the A-register input a division can be made in 3 n^2 cycles, which would reduce the cycle count for division with more than 60 %.

Using the above numbers the total execution time is

$$460,800 + 368,640 + 524,288 \ \approx \ 1.35 \cdot 10^6 \ \textit{cycles/frame}$$

and

$$921,600 + 368,640 + 524,288 \ \approx \ 1.81 \cdot 10^6 \ \textit{cycles/frame}$$

respectively.

Thus, provided with the 20 MHz clock rate the VIP should be able to sustain the following frame rates.

$$20 \cdot 10^6 / 1.35 \cdot 10^6 \ = \ 14.8 \ \textit{frames/sec}$$
$$20 \cdot 10^6 / 1.81 \cdot 10^6 \ = \ 11.0 \ \textit{frames/sec}$$

Since the frame rates for maximum I/O communication is higher, the total procedure is processor bounded to these numbers.

CONCLUSIONS

Optical flow can be computed in a straight-forward fashion using second derivatives. The VIP-chip (which exists in a laboratory version) seems to be capable of computing a 512x512 optical flow result 10-15 times per second depending on noise and precision requirements. The computation is processor bounded but the I/O communication and the memory bandwidth have been shown to be in good balance with available processing power.

References

[1] Tretiak O., Pastor L., *"Velocity estimation from image sequences with second order differential operators"*, Proc. "7th Int. Conf. Pattern Recognition", pp 16-19, 1984.

[2] Uras S., Girosi F., Verri A., Torre V., *"A Computational Approach to Motion Perception"*, "Biol. Cybern." No 60, pp 79-87, 1988.

[3] Danielsson P-E., *"Generalized and Separable Sobel operators"*, in "Machine Vision. Acquiring and Interpreting the 3D Scene", H. Freeman (ed.), Academic Press, 1990.

[4] Chen K., Svensson C., *"A 512-processor array chip for video/image processing"*, Proc. of "From Pixels to Features II," Bonas, France, Aug. 27 - Sept. 1, 1990, pp 349-361.

[5] Chen K., Danielsson P.E., Åström A., *"PASIC. A Sensor/Processor Array for Computer Vision"*, Proc. of the IEEE Conference on Application Specific Array Processors, Princeton, N.J., 1990.

[6] Åström A., *"Design and Evaluation of a Smart Image Sensor"*, Licentiate thesis, Dept. of EE, Linköping University, 1990.