Real-time Geometric Distortion Correction and Image Processing on the 1D SIMD architecture IVIP

Anders Åström[1], Folke Isaksson[2]
[1]Image Processing Laboratory, Dept of Electrical Engineering
Linköping University, S-581 83 LINKÖPING, Sweden
[2]Saab Missiles AB, S-581 88 LINKÖPING, Sweden
email: andersa@isy.liu.se

Abstract

This paper describes how geometric distortion correction and image processing operation can be performed in real-time on an IVIP system. The IVIP is a modified version of RVIP. RVIP is currently being manufactured and it is going to be used for radar signal processing. The IVP is targeted to perform the time consuming first stages of image processing. This includes geometric distortion correction and low level image processing. The material shown in this paper shows that very little time is spent by IVIP on image correction and linear interpolation (6 ms/frame in the projection case). Given a frame-rate of 25 frames/s, less than 15% of the available processing time is spent on correction and interpolation.

Introduction

Many image processing applications require geometric corrections on the acquired image before the actual image processing can take place. This is the case when we want to have a constant view of the scene and the camera is mounted on, for example, a robot arm, a car, or a plane. Figure 1 shows a schematic operation flow for this type of processing.

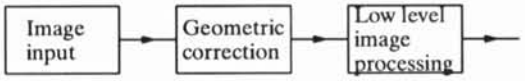

Figure 1

A very typical geometric distortion is shown in Figure 2. The image that we get from the input is the one to the left, i.e. the (x,y) image. The image that we want to perform our image processing operation on is to the right, i.e. the (u,v) image. In this case we have both a rotation and a projection of the (u,v) image in the (x,y) image.

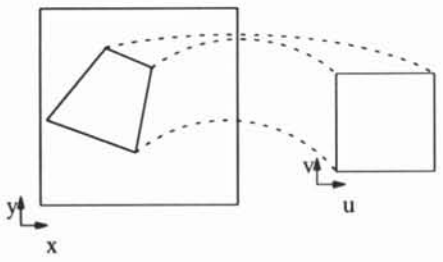

Figure 2

This means that each pixel in the (u,v) image has to compute its coordinate in the (x,y) image and obtain the pixel value at that position. In the projection and rotation case the mapping function can be described as

$$\begin{aligned} x &= \frac{a_0u + a_1v + a_2}{b_0u + b_1v + b_2} \\ y &= \frac{c_0u + c_1v + c_2}{d_0u + d_1v + d_2} \end{aligned} \qquad (1)$$

where a_i, b_i, c_i, and d_i are constant over the whole array.

It is also possible to think of an application where we have a number of small parts in the (x,y) image which we want to assemble to a single image as in Figure 3. This is the case if we want to perform the same image processing operation on a number of sub images. However, we must be aware of the edge problem.

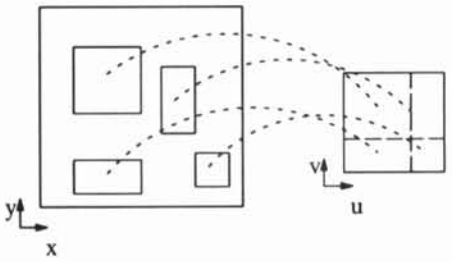

Figure 3

This type of mapping procedure is normally considered to be unsuitable for an SIMD array. The reason is that two neighboring pixels in the (u,v) image are not necessary

neighbors in the (x,y) image. This is especial true if we have generalized functions which can be expressed as

$$(x,y) = f(u,v) \qquad (2)$$

In this paper we describe an SIMD system which both compute the (x,y)-addresses for each (u,v) and perform the low level image processing. The SIMD processor used in this application, called IVIP (Infra-red Video Image Processor), is a modified version of the 1-D SIMD array processor RVIP [1]. RVIP, Radar Video Image Processing, has been designed at Linköping University and it is currently being manufactured at Ericsson Radar Electronics in Mölndal, Sweden, and it istargeted for radar signal processing.

IVIP, Architecture

The intention is to build the IVIP, Infra-red VIP, as an MCM. The size of each module would be 2"*2", and it will contain 512 processor elements as shown in Figure 4. This is the same strategy as RVIP which is already being built. Several IVIP chip can easily be joined together to form a larger array.

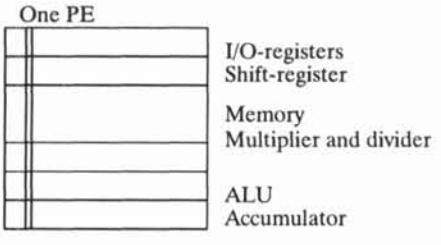

Figure 4, A 512-IVIP chip

Each PE is bit-serial with a common 1-bit bus to which all the units are hooked on as shown in Figure 5.
At the top of the bus we have four I/O banks of 32 bits each. Each register is dual-ported which means that, at the same time as the input data is written into the register the previous content is sent to the output.
The bi-directional shift register is used for internal communication between PEs in the array.
There are 2048 bits of static memory.
The ALU is a 1-bit logic unit with three input registers. The ALU performs the logical operations and the addition and subtraction.
The serial parallel multiplier and divider is added to each PE to speed up the multiply-and-accumulate (MAC) and the division operations.
The accumulator is used to speed up the arithmetic operations as well as to store temporary data.

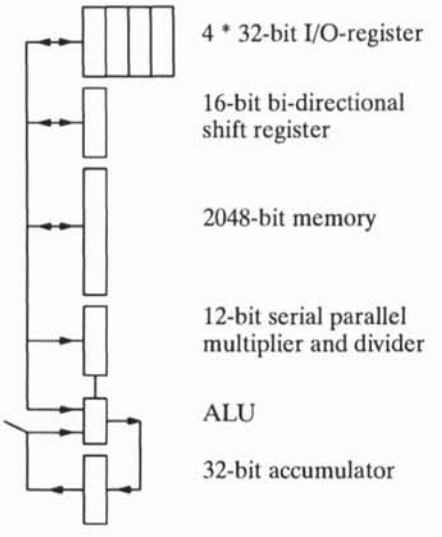

Figure 5, One PE

At this stage the IVIP is clocked with 50 MHz. A MAC operation takes **2b** cycles to complete. For 12-bit input data that is **24** cycles or 0.5 μs. However, in this time this operation has been perform by 512 PEs which means that a MAC operation is performed every 0.05 cycle, or 1 ns. The division takes **6b** cycles due to the carry propagation.

Address computation

To perform a geometric distortion correction the IVIP works in a two-stage pipe-lined fashion. First, the (x,y) coordinate for each PE are computed within the array, i.e. all (u,v) for a given row v. The computed (x,y) coordinates are placed in the I/O-register. Then, the image processing operations are performed on the image data from the previous line at the same time as the (x,y)-addresses are read out from the processing array and the image data are read in. This is shown in Figure 6.

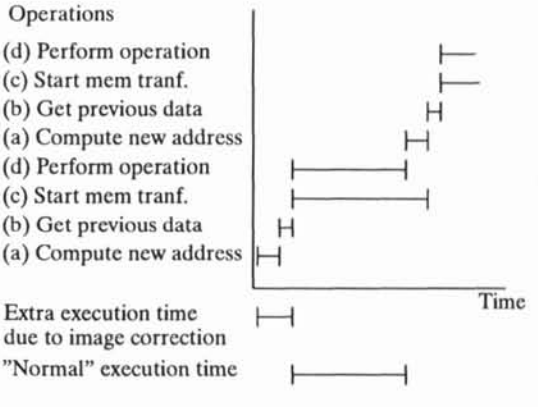

Figure 6

When computing the (x,y) coordinates we assume that each PE has its u-coordinate stored in its internal memory,

i.e. its position in the array. From the program flow we will then obtain the v-coordinate.

Figure 7 shows the IVIP system with its external data communications. If we interpret [x] as the round value of x we will have a nearest neighbor selection in the image memory. The image memory is dual ported.

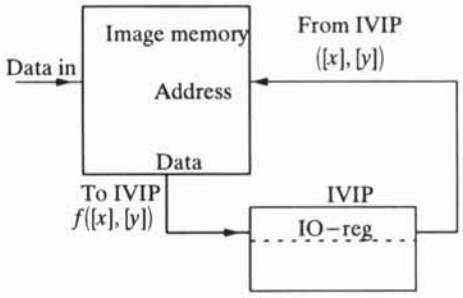

Figure 7, IVIP system overview

For the case in Figure 2, the address computation consists of 4 MAC operations, 2 divisions, and some data movement. This takes 8 μs for each line given a 12-bit input data. If we do not perform any other image processing operation we are able to sustain a frame rate, with 512^2 images, of 250 frames/s.

Interpolation

In some cases it is not sufficient just to select the value belonging to the nearest neighbor. An alternative method is then to perform a linear interpolation using the four nearest neighbors. Figure 8 shows a case when we want to compute the f(x,y) value given that we know the values of the four nearest neighbors.

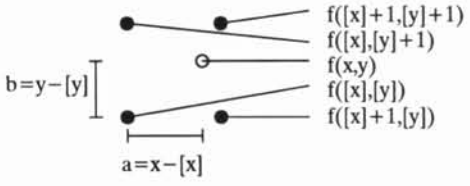

Figure 8, Linear interpolation

An approximation of $f(x,y)=f_{ip}$ can be computed as a weight linear sum of the four neighbors

$$f(x,y) \approx f_{ip} = \big((1-a)f_{00} + af_{10}\big)(1-b) + b\big((1-a)f_{01} + af_{11}\big) \quad (3)$$

To perform the operation in (3) on an image line in IVIP, given that we have the four values f_{00}, f_{01}, f_{10}, and f_{11}, takes 12μs for both the interpolation and the address computation which totals 6ms per 512^2 frame. It corresponds to a frame rate of 160 frames/s.

Figure 9 shows when different processes occur in time. Here we can see that the data transfer from the image memory can be performed concurrently with the "normal" image operations. This is true as long as the time for the "normal" execution (d) and the time to compute the new address (a) is longer than the data transfer time (c).

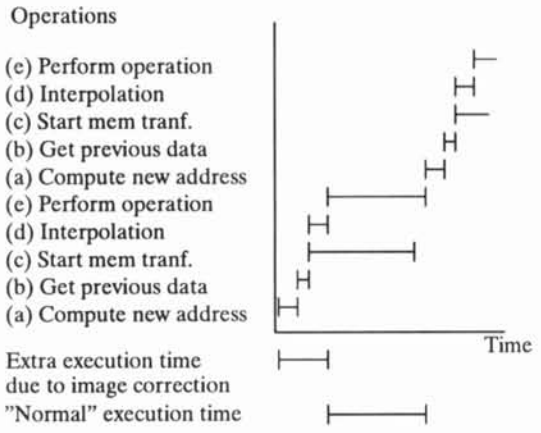

Figure 9, Operation flow with interpolation

Figure 9 shows that the interpolation computation only add time to the total execution time if

$$t_{(c)} < t_{(a)} + t_{(d)} + t_{(e)} \quad (4)$$

The requirement of hardware to perform linear interpolation is actually very small which is shown in Figure 10. We need four identical image memories. This means that the input image should be fed to all these memories. Given an address (a,b) from the IVIP, the four identical image memories are addressed with (a,b), (a+1,b), (a,b+1), and (a+1,b+1) respectively. The four output result, f_{ij}, are then fed back to the corresponding PE. The I/O-register in IVIP is well suited for this since it consists of 32=4*8 bits.

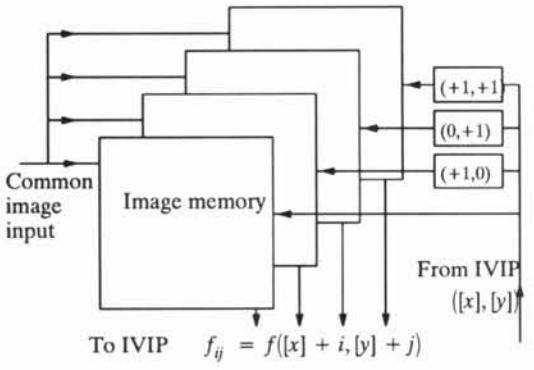

Figure 10, A straight forward interpolation solution

An alternative solution, in order to save memory, is to store the image in four different image memory, each four times as small as the original image, according to the pixel pattern in Figure 11.

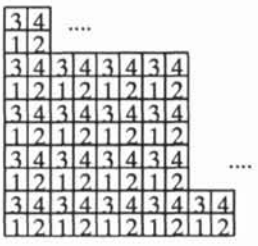

Figure 11, The division of the image into four memories

We can then change the system in Figure 10 by replacing the simple increment units of the case address, by a slightly more sophisticated address calculator as in Figure 12. This unit keeps track on which image memory is which according to Figure 11, given a base address [x],[y].

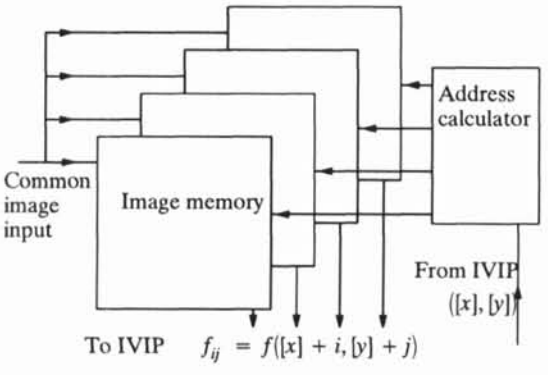

Figure 12, A reduced memory interpolation solution

Conclusions

This work is part of a preliminary study to see if the IVIP architecture is suitable for this type of application. The material shown in this paper shows that very little time is spent by IVIP on image correction and interpolation (6ms/frame in the projection case). Given a frame-rate of 25 frames/s, less than 15% of the available processing time is spent on correction and interpolation.

Acknowledgement

The authors would like to thank the Swedish Board for Technical Development (NUTEK) for financial support.

References

[1] Johanneson M., Åström A, Ingelhag P., *The RVIP Image Processing Array*, Proc of CAMP93, New Orleans, USA.

[2] Åström Anders, Forchheimer Robert, *A Global Arithmetic Unit for Linear Arrays of Processing Elements*, submitted to IEEE trans on Computer.

[3] Åström A, Mattias Johannesson, Anders Edman, Tor Ehlersson, Ulf Näsström, Bo Lyckegård, *An Implementation Study of Airborne Medium PRF Doppler Radar Signal Processing on a Massively Parallel SIMD processor architecture*, Submitted for publication.