Real-Time Imaging **8**, 105–113 (2002)
doi:10.1006/rtim.2001.0259, available online at http://www.idealibrary.com on IDEAL®

Real-time Sub-pixel Cross Bar Position Metrology

Many measurement application fields need to calculate cross bar intersection locations of horizontal and vertical bars. The system we developed and that we present in this paper is an embedded system that measures cross bar positions with sub-pixel accuracy on 1024×1024 pixel images delivered by a camera at a 50 MHz data rate in real time. This is done using an algorithm that looks for intersection areas and then locally calculates two lines representing horizontal and vertical bars. The two line intersection is considered to be the bar intersection. To achieve real time, we developed a hybrid architecture in which low level processes are implemented into FPGAs and others into DSPs. As a result, at the end of the camera scanning (20 ms), all calculations are completed and the results are available.

D. Rivero[1], M. Paindavoine[2,]* and S. Petit[1]

[1]Thomson Tubes and Displays, Avenue du Général de Gaulle, 21110 Genlis, France.
[2]Université de Bourgogne, LE2I, Aile des Sciences de l'Ingénieur, 21011 Dijon, France.

Introduction

In several areas, like human movement analysis, or in our case TV metrology, a cross bar target is displayed and intersection positions are calculated to obtain information about human movements or about TV screen adjustments. This requires highly accurate measurements and most of the time, high-speed processing for feedback effect. In our case, this means using a 1024×1024 pixel CCD camera with a 50 MHz data rate. Using our system we then reach an accuracy of 50 μm in a 1 m wide screen using a sub-pixel algorithm.

Research has been carried out on real-time measurement systems as in astronomic applications [1], in object corner positioning [2] and in human movement analysis [3]. These video systems allow validation of the position measurement algorithm implementation at a maximum of 10 MHz pixel data rate.

In order to process the 1024×1024 images in real time, we selected an algorithm which involves as a first step the object location in the image and as a second step a two dimensional object position measurement in this located area.

The location step processes the whole image and is made using low-level image processing while the second step is achieved on all the pixels in the located areas. Thus step comprises centre of mass calculations and using these results a least mean square line regression is carried out to represent horizontal and vertical bars. The intersection of these two lines is considered as being the bar intersection. Thus, this algorithm requires both low level and high level processing.

**E-mail: paindav@u-bourgogne.fr*

To process data at a 50 MHz data rate, many approaches have been tested. The first approach often met consists of using a Digital Signal Processors (DSPs) board. However, because of speed, this is not easily possible. The second possibility consists of making all the calculations on a dedicated board. The solution we present takes advantages of the two methods and involves implementing one part of the process on programmable electronic components (FPGAs) board and the other one in a DSP board, thereby obtaining an optimum solution.

The algorithm is presented in the next section. The third part of this article describes the algorithm implementation.

Sub-pixel Cross Bar Position Metrology Algorithm

Figure 1 describes the measurement system which uses a 1024×1024 CCD camera in order to accomplish TV screen adjustments. This camera delivers two video outputs working at a data rate of 25 MHz each (the pixel data rate is 50 MHz). The images obtained with the camera have to be processed using a specific algorithm. The algorithm is composed of two entirely independent steps:

1. A low precision intersection area location.
2. A local representation of the two intersection bars using lines with sub-pixel accuracy.

The intersection of the two calculated lines is then considered to correspond to the bar intersection. Figure 2(c) presents a synopsis of this process. The advantage of this decomposition is that the sub-pixel method, requiring a lot of calculations, is not applied to the entire image but only to areas. As interest area windows change from one image to another, each position must be relocated.

Intersection area location

The algorithm consists of sliding a mask to locate interest areas. Because of the simplicity of the observed images and because of the speed at which processes have to be run, only 16 points are tested for each position of the mask. To meet with real time on an embedded system, the window jumps eight pixels horizontally and four pixels vertically between windows tested. This choice has been validated on various images. This is possible because of the width of bars and because the distance between two positions is greater than an intersection area (more than 32 pixels).

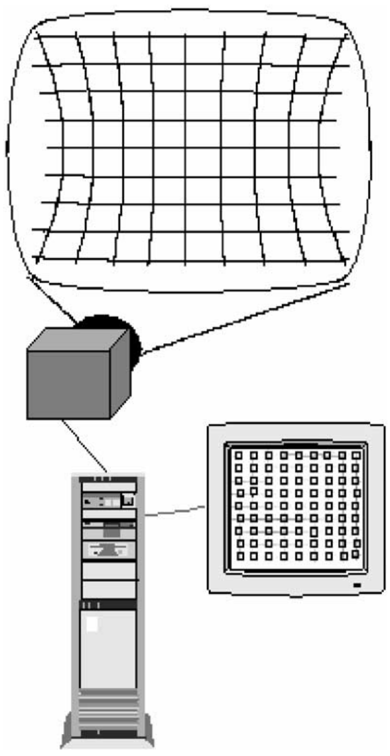
Figure 1. System description.

This method gives good results and allows the detection of all areas when measurement conditions are well suited.

Lines modelling of bars

Considering, in our case, the TV image size (up to 1 m diagonally), accuracy (50 μm) and camera resolution (1024×1024 pixels) a sub-pixel approach is needed.

Many algorithms exist that use a two dimensional approach but most of them require a very high data rate and make use of the straight line edge hypothesis. This is not the case in intersection areas so we do not use them. Examples of these methods are given by [4–6].

We then use a one dimensional sub-pixel approach that involves the approximation of bar positions with a set of points. These points are used for linear regression modeling of horizontal and vertical bars. This is done using the least mean square algorithm. The intersections

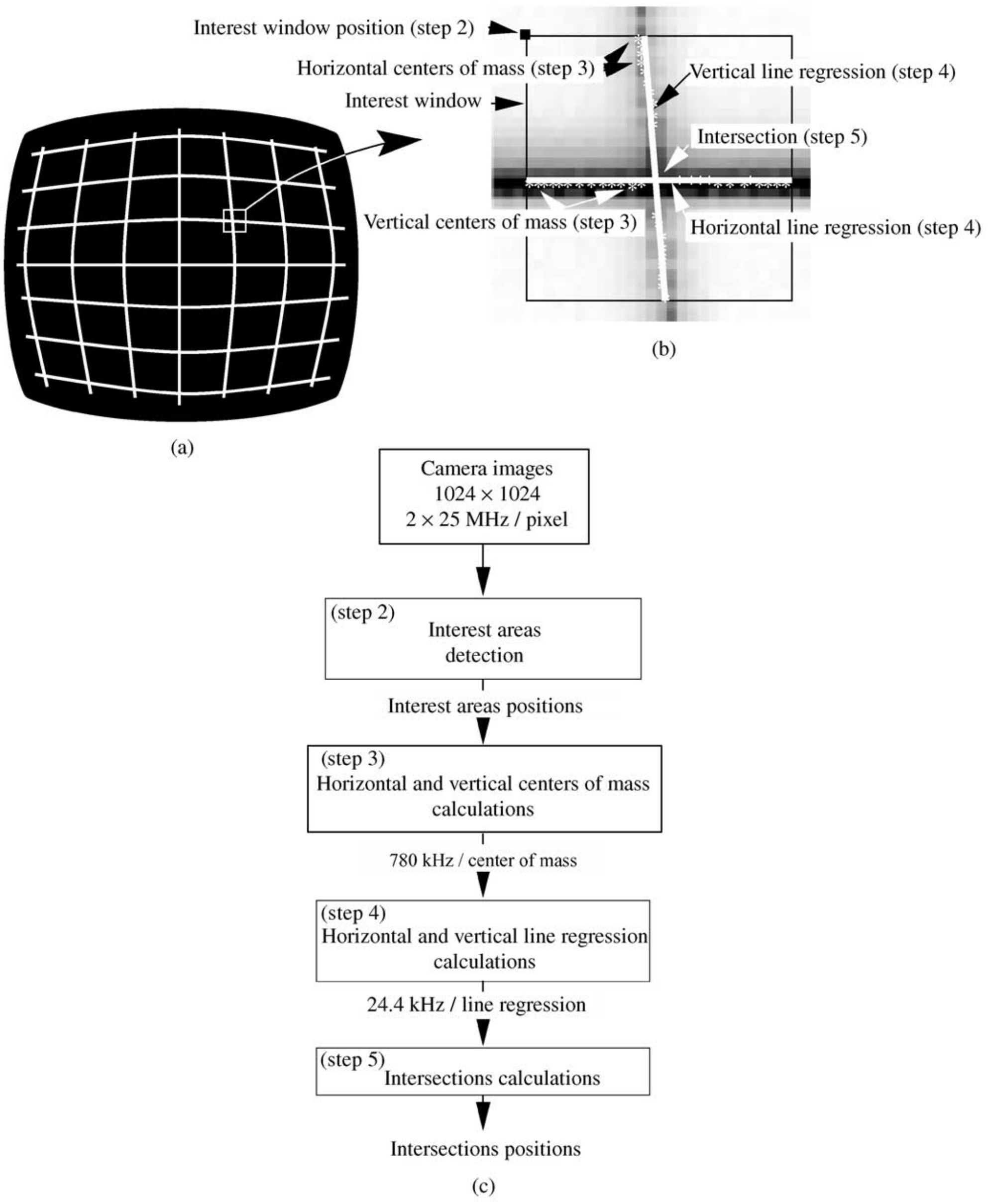
Interest window position (step 2)
Horizontal centers of mass (step 3)
Interest window
Vertical line regression (step 4)
Intersection (step 5)
Vertical centers of mass (step 3)
Horizontal line regression (step 4)
(b)
(a)
Camera images 1024 × 1024 2 × 25 MHz / pixel
(step 2) Interest areas detection
Interest areas positions
(step 3) Horizontal and vertical centers of mass calculations
780 kHz / center of mass
(step 4) Horizontal and vertical line regression calculations
24.4 kHz / line regression
(step 5) Intersections calculations
Intersections positions
(c)

Figure 2. Example of a mask for detecting intersection pattern.

of these lines are considered to correspond to bar intersections as shown on Figures 2(a) and 2(b). The accuracy of the approximation depends on the precision obtained when modeling bars using points. A 1D sub-pixel algorithm [7, 8] comparative study has been made corresponding to the previously mentioned constraints and thus the centre of mass algorithm was selected.

Algorithm implementation

DSP solution

Implementing the previously described algorithm into a DSP based architecture is not possible in real time without using more than six DSPs and an interface board to store signals from the camera. This development is complex and at the end of the camera image scanning, results may not be available.

We can give centre of mass calculation time measured as an example. One complete calculation into two DSPs is achieved in 17.5 µs. In this case, one of the two DSPs calculates the numerator and the denominator and the other one calculates the final division in a pipeline mode. As we need to calculate about 32 centres of mass per intersection with more than 100 intersections per image, the maximum time accorded to one calculation is 7.8 μs. So it is not possible to calculate the centre of mass in two DSPs in real time — we have to consider more DSPs. Moreover, regression line calculations and intersection area location have to be performed and so these operations require more resources.

For all measurements, the chosen DSPs are Texas Instruments TMS320C44 [9]. They have been chosen because they have four communications ports (comports) that permit them to communicate amongst themselves and externally. They also have an internal ALU (Arithmetic Logic Unit), able to process a 40-bit floating point number addition and multiplication in one cycle with a 60 MHz clock. They also incorporate a DMA (Direct Memory Access) coprocessor independent of the ALU and numerous buses that make it possible to access two external memories independently. So when the DMA coprocessor accesses one of these memories the ALU can access the other at the same time. These DSPs also integrate two internal 1 Kword RAMs and a cache.

After various simulations of possible architectures we obtained the results presented in Table 1 [10] and in relation to industrial application, we developed a low cost hybrid architecture. With regard to Table 1, it appears that it would be interesting to implement low-level processing into a dedicated FPGA (Field Programmable Gate Array) board. This is because of all the possibilities they now offer. For example, the Xilinx XC4000 family [11] that we chose for this application incorporates, among other functions, fast integer arithmetic additions, internal RAMs and ROMs.

Finally, the automatic location of windows including intersections and centre of mass calculations are implemented into FPGAs and linear regression calculation is obtained on DSPs. This last process has been kept in the DSPs. A lot of resources would be required for floating point calculations if we tried to implement it into the FPGAs. Moreover, the low data rate is compatible with the DSP time calculation in this step. We now present implementation considerations on centre of mass calculations using FPGAs, and regression line implementation using DSPs.

Centre of mass implementation using FPGAs

The general form of centre of mass is

$$C_k = \frac{\sum_{i=0}^{31} i \cdot p_k(i)}{\sum_{i=0}^{31} p_k(i)} \tag{1}$$

here C_k is the kth centre of mass calculated and where the grey level of pixel i is $p_k(i)$. This step requires numerator and denominator calculations before division calculation. The simplest way to implement this equation is to realize a recursive calculation as shown in Figure 3. This needs an 8×5 multiplier. Many multipliers have been tested (Wallace tree, carry save adder multiplier, ... [12]). Whatever the chosen multiplier, limit time calculation on Xilinx XC4000-5 FPGA is greater than 40 ns (25 MHz pixel data rate). To increase time according to one calculation, we then process two pixels at a time maintaining the same access time and the same number of memory components. Thus, this doubles the time accorded to one calculation although it increases requested resources.

It appears then that simplifications are possible and for example when four pixels are processed at the same time, (1) is transformed into

$$C_k = \frac{N_k}{D_k} = \frac{\sum_{m=0}^{7} \sum_{n=0}^{3} (m+n).p_k(m+n)}{\sum_{m=0}^{7} \sum_{n=0}^{3} p_k(m+n)}. \tag{2}$$

Table 1. Architecture comparison between FPGAs and DSPs solutions considering real-time constraint

Process	FPGA solution	DSP solution
Intersection area seek	1 FPGA - 1 RAM (640 ns time calculation)	>10 DSPs
Center of mass calculation	2 FPGAs - 2 RAMs (1280 ns time calculation)	>6 DSPs
Line calculations	Not evaluated	2 DSPs

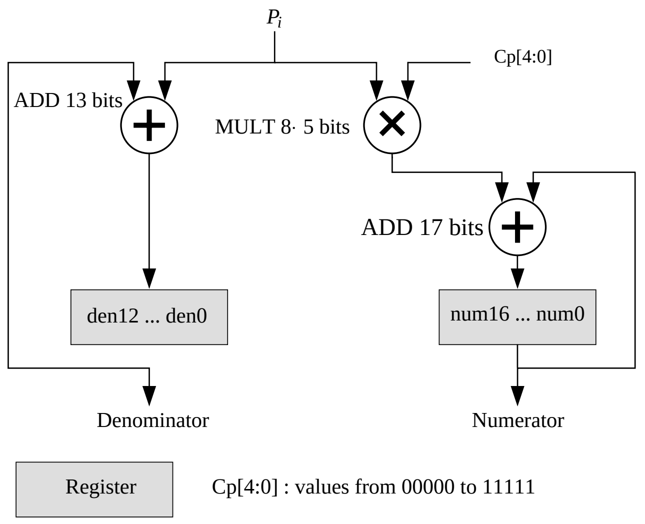

Figure 3. Numerator and denominator implementation (one pixel at a time).

The numerator N_k can then be written

$$N_k = \sum_{m=0}^{7} (m.[p_k(m) + p_k(m+1) + p_k(m+2) + p_k(m+3)] + p_k(m+1) + 2.p_k(m+2) + 3.p_k(m+3)). \quad (3)$$

So,

$$N_k = \sum_{m=0}^{7} \{m.[p_k(m) + p_k(m+1) + p_k(m+2) + p_k(m+3)] + [p_k(m+1) + p_k(m+3)] + 2.[p_k(m+2) + p_k(m+3)])\}. \quad (4)$$

This is interesting because the calculation requires a 10×3 multiplier. Other additions can then be processed in parallel. This solution is developed in Figure 4. This operation has been realized making groups of 1, 2, 4 and 8 pixels. Results are presented in Table 2.

It appears that when grouping more than four pixels at each cycle, the time needed on a Xilinx to process one cycle is less than the maximum time allowed for this cycle and so real-time calculation is possible. Grouping four pixels at each process cycle is the optimal solution because it allows real-time performance using minimum resources. So, the adopted solution consists of grouping 4 pixels at each process cycle and as Figure 4 shows, numerator and denominator calculations are obtained at the same time.

For time reasons, the division is pipelined with previous calculations. The time for one division corresponds to a 32-pixel centre of mass calculation processed with a 25 MHz calculation clock, which means a time of 1280 ns for each division. For this implementation, no timing problem has been encountered and restitution remainder iterative division [12] has been implemented.

We choose to develop a communication through DSPs comports because DSPs are made in such a way that when they wait for data that is not present on the port, they keep waiting until this data is present. This is a simple and powerful way we use to synchronize the DSP with the data that the FPGA sends. This bidirectional communication has been developed integrating the DSP communication protocol on the FPGA and is also used to send data from DSPs to FPGAs at the beginning of the process.

Linear regression implementation on DSPs

Two Texas Instrument TMS320C44 DSPs receive data in the DMA mode. This is done using the DSP characteristic that makes it possible to process data from a memory while storing data in the DMA mode in the other one. At the end of the reception of all the data corresponding to an intersection, a software switch is achieved so that these last data are processed and new data are stored in the other memory. This allows computation during the centre of mass reception. When all centres of mass corresponding to a valid window are received, calculation on these points begins. This is possible because reception time is greater than linear regression calculation time (reception time = time for 32 centre of mass calculations = 32×1280 ns; calculation time: 36.8 μs). This last time includes line parameter calculation, communication time to group results in one DSP and line intersection determination.

For linear regression, the algorithm implemented is least mean square algorithm. This involves a_k and b_k calculations in the equation $y(x) = a_k + b_k.x$:

$$a_k = \frac{A_k}{D_k} = \frac{S_{xx}S_y - S_xS_{xy}}{S_{xx}S - (S_x)^2} \quad (5)$$

$$b_k = \frac{B_k}{D_k} = \frac{S_{xy}S - S_xS_y}{S_{xx}S - (S_x)^2}, \quad (6)$$

where

$$S = \sum_{n=0}^{31} 1 = 32 \quad (7a)$$

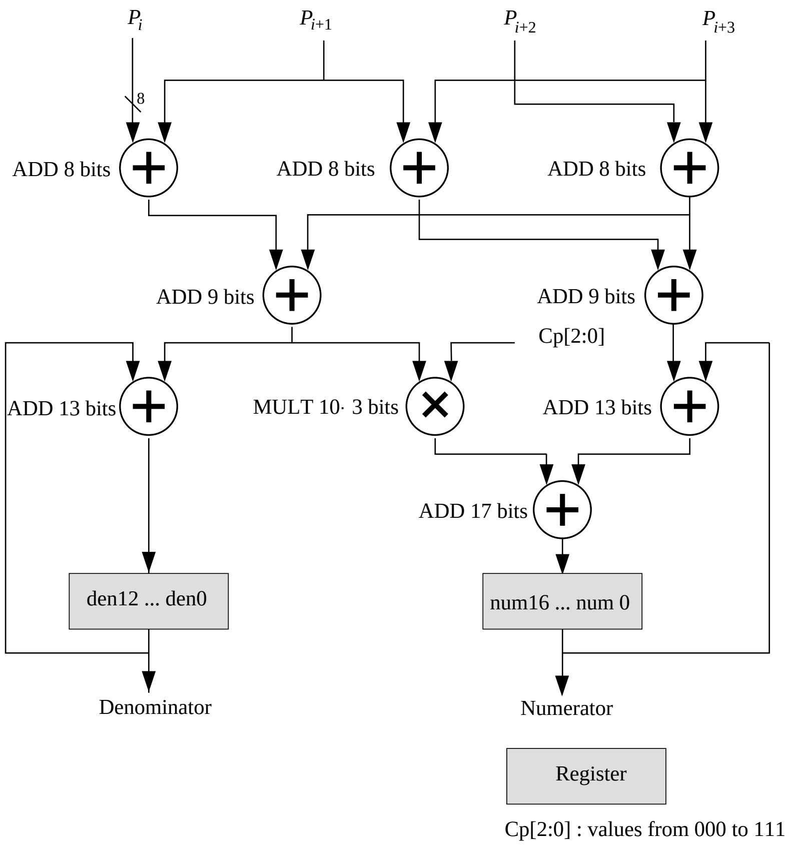

Figure 4. Numerator and denominator implementation (four pixels at a time).

$$S_x \equiv \sum_{n=0}^{31} i = 496 \tag{7b}$$

$$S_{xx} \equiv \sum_{n=0}^{31} i^2 = 10416 \tag{7c}$$

$$S_y \equiv \sum_{n=0}^{31} p_k(i) \tag{7d}$$

$$S_{xy} \equiv \sum_{n=0}^{31} i \cdot p_k(i) \tag{7e}$$

The first step calculates S_y, S_{xy} in the two directions on two DPSs (DSP1 for horizontal direction and DSP2 for vertical direction in Figure 5) in the same way. Then only terms A_k, B_k and D_k are calculated in each direction and DSP2 sends to DSP1 the A_k, B_k and D_k results. Then DSP1 calculates X_0, Y_0, the intersection using the

Table 2. Evolution of time cycle according to the number of pixels grouped

Number of data processed at one cycle	1	2	4	8
Number of cycles for a complete calculation	32	16	8	4
Maximum time allowed	40 ns	80 ns	160 ns	320 ns
Time needed on a Xilinx to process one cycle*	72.2 ns	81.6 ns	87 ns	104.7 ns
Number of function generators used (resources)	301	311	330	389
Number of component input/output needed	28	36	52	84

*Measurements made on a Xilinx XC4013-5 using tool Xdelay.

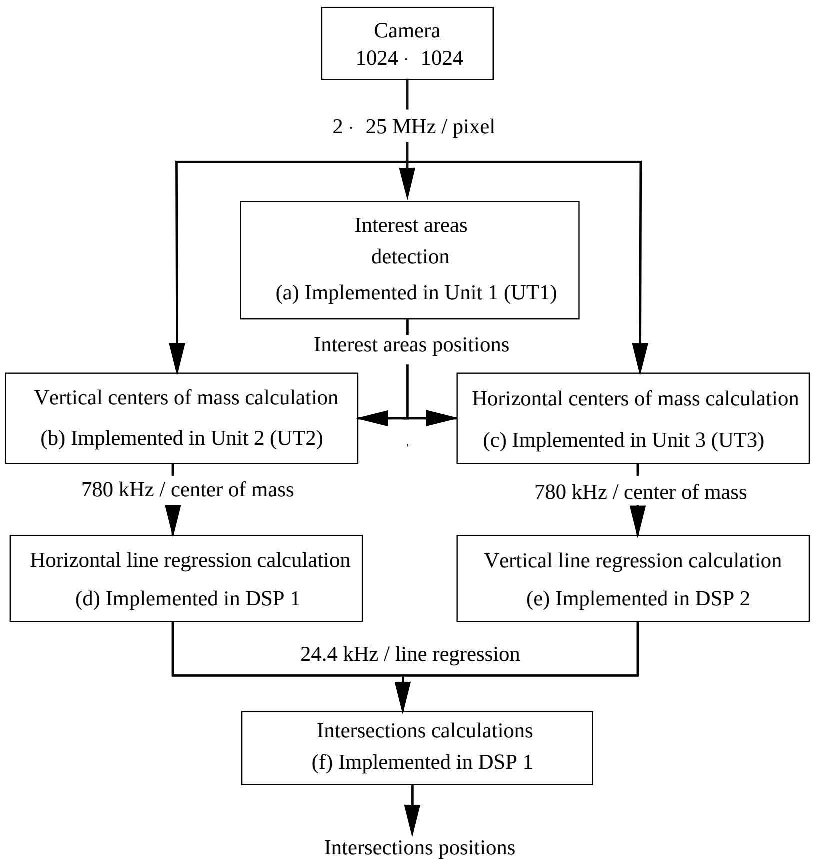
Camera
1024 · 1024
2 · 25 MHz / pixel
Interest areas detection
(a) Implemented in Unit 1 (UT1)
Interest areas positions
Vertical centers of mass calculation
(b) Implemented in Unit 2 (UT2)
Horizontal centers of mass calculation
(c) Implemented in Unit 3 (UT3)
780 kHz / center of mass
780 kHz / center of mass
Horizontal line regression calculation
(d) Implemented in DSP 1
Vertical line regression calculation
(e) Implemented in DSP 2
24.4 kHz / line regression
Intersections calculations
(f) Implemented in DSP 1
Intersections positions

Figure 5. Decomposition of the algorithm.

formula given by (8) and (9).

$$X_0 = \frac{x_0}{d_0} = \frac{A_{v,k}D_{h,k} + B_{v,k}A_{h,k}}{D_{v,k}D_{h,k} - B_{v,k}B_{h,k}} \quad (8)$$

$$Y_0 = \frac{y_0}{d_0} = \frac{A_{v,k}B_{h,k} + D_{v,k}A_{h,k}}{D_{v,k}D_{h,k} - B_{v,k}B_{h,k}} \quad (9)$$

here indices w and h correspond to parameters respectively related to vertical and horizontal lines.

Final architecture and results

The final architecture is shown in Figures 6 and 7 and is composed of 3 FPGAs and 2 DSPs. The process is decomposed as shown in Figure 5.

Unit 1 slides a mask through the image data sent by the camera. When a window is validated, its position is sent to two FPGAs (units 2 and 3) which compute centres of mass for vertical and horizontal bars in pipeline mode so that processes never stop. During these centre of mass calculations, unit 1 continues sliding the mask through the image and detecting intersection windows. All stages are buffered. Centres of mass are then sent to two DSPs via communication ports, which calculate the horizontal and vertical linear regression at the same time as FPGAs continue their calculations.

In Figure 8, an original image to be processed automatically with our system is given. Figure 9 shows the areas detected. This figure shows that all areas are detected in real time (less than 40 ms). Several tests have been carried out using a lot of images and have shown that all measurements are obtained with an accuracy of 50 μm in a 1 m wide screen.

This system (electronic boards in Figures 6 and 7) remains very general and the use of these technologies allows us to easily transpose this system to another

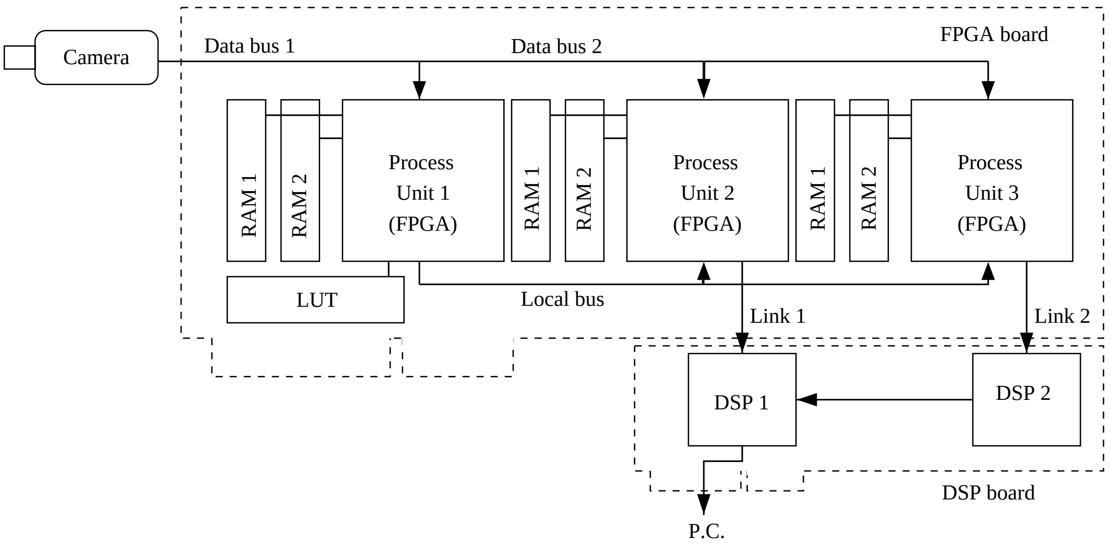

Figure 6. Final architecture.

algorithm that keeps the general block structure presented in Figure 5. It can also be applied to other patterns.

Conclusion

To increase measurement accuracy in cross bar location, we use a 1024 × 1024 pixel camera at a 50 MHz data rate using a sub-pixel algorithm. This algorithm is well suited for these applications but requires a high power calculation and is not easily implemented into real-time applications. To combat these constraints, we realized an embedded hybrid calculation system. In this system, low-level step calculations are implemented into FPGAs and high-level step calculations into DSPs. This allows us to perform a complete calculation (about 100 intersections detections, 6400 centres of mass and 200 least mean square regression line calculations on 32 points) and to obtain all the results during the camera scanning (20 ms) for a retrospective action.

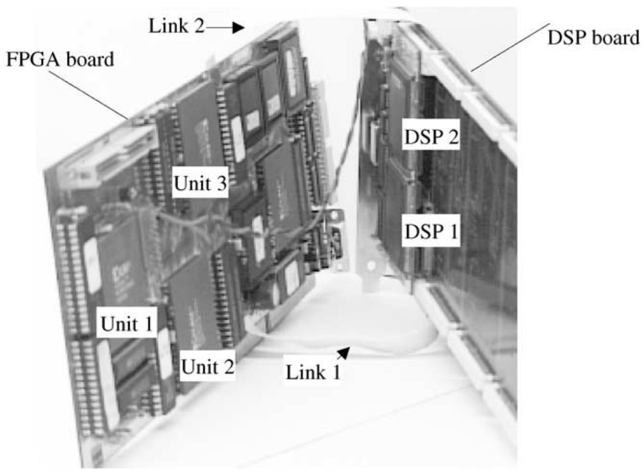

Figure 7. Material realization of the architecture.

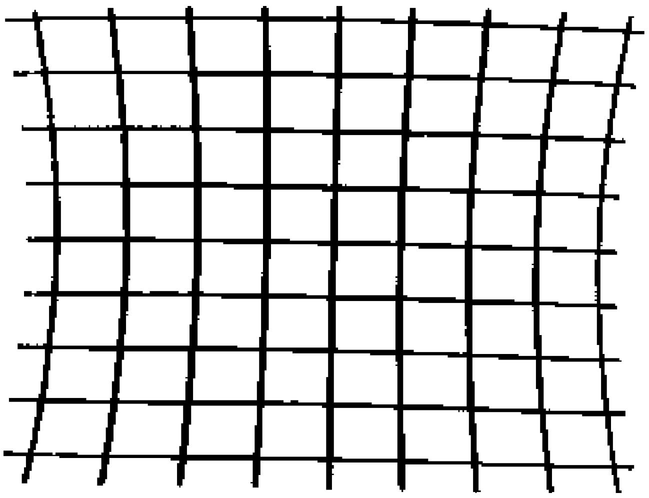

Figure 8. Original image.

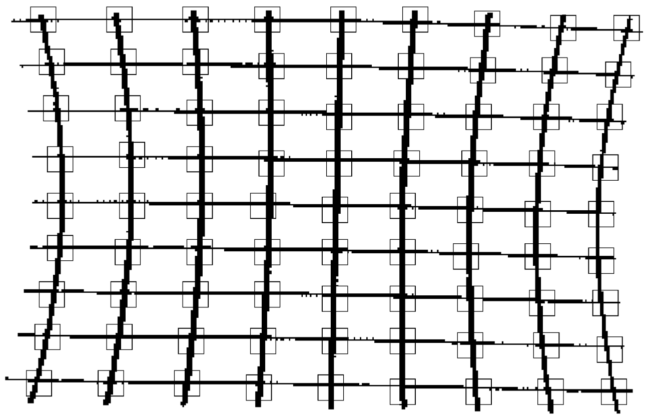

Figure 9. Processed image.

References

1. Fidouh, F. (1993) *Développement d'un détecteur de photocentres de táches lumineuses à base de processeurs de traitement du signal. Applications à l'imagerie astronomique à haute résolution*. PhD Thesis, University of Paris VI, France.
2. Gaiarsa, A.E. & Capson, D.W. (1994) Real-Time Measurement of Corner Position in Binary Images. *IEEE Trans. on Inst. and Meas.* **43**: 567–577.
3. Furnee, H. (1990) PRIMAS: real-time image-based motion measurement system. Image-Based Motion measurement. *Proceedings of the SPIE* **1356**.
4. Marr, D. & Hildreth, E. (1980) Theory of edge detection. *Proc. Roy. Soc. London, Conference on Computing vision.* **B207**.
5. Tabatabai, A.J. & Mitchell, O.R. (1984) Edge location to sub-pixel values in digital imagery. *IEEE Trans. on Pattern Analysis and Machine Intel.* **6**: 188–201.
6. Overrington, I. & Greenway, P. (1987) *Practical first-difference edge detection with sub-pixel accuracy*. In: Image and Vision Computing. Butterworth & Co., volume 5.
7. Kang. J.C. (1993) Sub-pixel Edge Estimation in Machine Vision. *Revue internationale de CFAO et d'infographie* **8**: 371–385.
8. Seitz, P. (1988) Optical superresolution using solid-state cameras and digital signal processing. *Optical Engineering* **27**: 535–540.
9. Texas Instruments (1996) TMS320C4x User's Guide. *Digital Signal Processing Solutions.*
10. Rivero, D. (1999) *Mesure en temps réel de position à précision sub-pixel dans une image*. PhD Thesis, University of Burgundy, France.
11. Xilinx (1994) *The programmable logic data book.*
12. Hennessy, J.L. (1996) *Computer Architecture: A Quantitative Approach.* Paris: Morgan Kaufmann Publishers Inc.