Real-time Component Labelling with Centre of Gravity Calculation on FPGA

Abdul Waheed Malik, Benny Thörnberg
Department of Information Technology and Media
Mid Sweden University,
Sundsvall, Sweden
{waheed.malik | benny.thornberg}@miun.se

Xin Cheng, Najeem Lawal
Department of information Technology and Media
Mid Sweden University,
Sundsvall, Sweden
{xin.cheng | najeem.lawal}@miun.se

Abstract—**In this paper, we present a hardware unit for real time component labelling with Centre of Gravity calculation. The main targeted application area is light spots, used as references for robotic navigation. Centre of Gravity calculation can be done in parallel with a single pass component labelling unit without first having to resolve merged labels. We present a hardware architecture suitable for implementation of this Centre of Gravity unit on Field Programmable Gate Arrays. As a result, we get high frame speed, low in power and low in latency design. The device utilization and estimated power dissipation are reported for the Xilinx Virtex II pro device simulated at 86, Video Graphics Adaptor (VGA), sized frames per second. Maximum speed is 410 frames per second at 126 MHz clock.**

Keywords- Centre of Gravity (COG); Component Labelling; Position Measurement

I. INTRODUCTION

Object detection and measurement of an object's position are important aspects for many image processing problems and machine vision systems. In medical image processing, we can see it in marker recognition and leukocyte tracking [1]. We can see the same requirement for automatic target recognition delineation and for light spots used as references for robot navigation [2]. In all those applications, it is really important to measure the object's positions correctly. When using light spots for robotic navigation, it is also necessary for the processing of video frames to be fast. High frame speed and low latency becomes important performance measures. Field Programmable Gate Arrays (FPGAs) are the preferred computational platform to reach this high performance. FPGAs offer massive parallelism, on-chip memories and arithmetic units and are therefore found to be most suitable for front end video processing [5]. This has motivated us to develop a FPGA based hardware unit for component labelling and COG calculation.

This is the most aggressive implementation of component labelling with feature calculation with minimum latency. It is shown that calculation of COG can be done without first resolving the complex chains of labels. The resolving of labels is done after the whole frame is labelled, avoiding any dependency on horizontal synchronization as described in [6], [10], thus maximizing the frame speed. Only one pass labelling is used as compared to classical two pass labelling, and no need to store image as described in [9]. Thus, the latency for our implementation is much lower than the previous implementations.

The developed architecture is suitable for smart cameras with a built in computational platform and having a low bandwidth output communication channel [3].

The smart camera only sends the processed and refined information, rather than sending large amount of video data. Machine vision algorithms are often divided into the following steps [4]. Video is acquired from the image sensor at Image acquisition. Image objects are extracted from the pre-processed video data at Segmentation, as shown in Figure 1A. During labelling, pixels belonging to the same image component are assigned a unique label. At Feature extraction an image component is described, for example in terms of region features such as area, ellipse, square or circle parameters. Components can also be described in terms of gray value features such as mean gray value or position. This feature information can then be used for Classification of image components. Information about recognized objects in the camera's observation area can be transmitted to the camera output using typically a very low bandwidth.

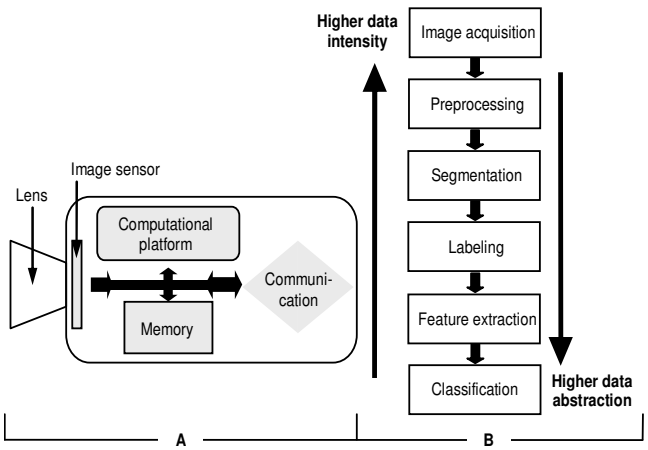

Figure 1. A) Smart camera. B) Fundamental steps of machine vision

 ISBN:978-1-61208-114-4

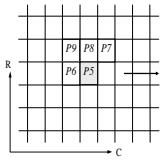

Figure 2. Neighbourhood for eight connectivity labelling.

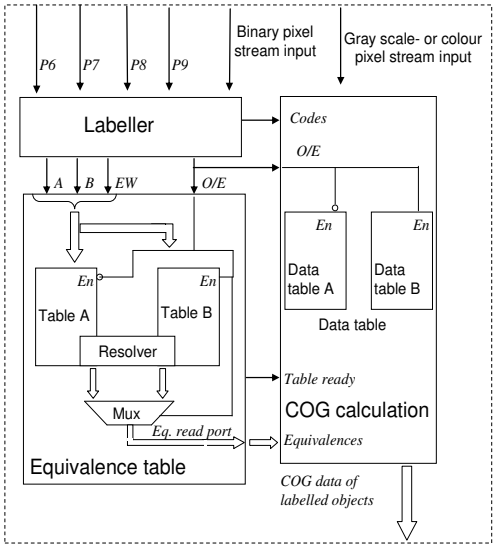

Figure 3. Kernel for labelling and COG computation

In this work, we focus on the COG calculation along with the image component labelling. Eight connectivity labelling process has been used, and labels are resolved after every frame. As shown in Figure 2, P5 is the current pixel for labelling, and it is labelled depending on labels P6 to P9. If the labeller finds two different labels in the neighborhood, it assigns the smaller label to the current pixel and marks that those two labels are equivalent and should be merged.

The equivalent labelled pair (A, B) is sent to the Equivalence table for merging. Label merging is targeted to either Table A or B, depending on odd or even frames (O/E) [7]. Resolving of linked lists of labels is thus frame interleaved with the labelling process. The architecture is shown in Figure 3. We assign a label to the current binary pixel depending on the previously labelled pixels from P6 to P9. At the same time as we are assigning a label to the current pixel, we are also accumulating pixel data in Data table A or B for the COG calculation without first having to resolve linked lists of labels. This method has previously been analyzed for COG in [7]. We have found published results on high speed, low latency hardware component labeller [6], but no results on how such a labeller can be efficiently combined with calculation on image component features such as COG.

The COG calculation method will be discussed in Section II. We discuss hardware architecture in Section III, while memory requirement is discussed in Section IV. Performance and device utilization are discussed in Section VI.

II COG CALCULATION

The COG for an image component *O* in an image is calculated as

$$(r_o, c_o) = \frac{\sum_{r_i,c_i \in O} r_i I(r_i,c_i)}{\sum_{r_i,c_i \in O} I(r_i,c_i)}, \frac{\sum_{r_i,c_i \in O} c_i I(r_i,c_i)}{\sum_{r_i,c_i \in O} I(r_i,c_i)} \quad (1)$$

In the Equation 1, (r_o,c_o) is the mass centre of the object. $I(r_i,c_i)$ is intensity of pixels and r_i, c_i are the row and column count respectively. Let us assume that an object region is divided into two sub regions, *S* and *T,* as a result of the first pass of labelling. *S* and *T* belonging to the single object *SUT* are resolved at the end of first pass and in parallel with labelling of the next frame. The numerator and denominator according to Equation 2 are accumulated in to *Data table A* or *B,* at the same time as the first labelling pass. At the end of the first labelling pass, when the codes are resolved in equivalence table, the data stored in *Data table A* or *B* will be merged into numerator and denominator for the region *SUT*. This data merging for regions *S* and *T* is illustrated in Equation 2 for row the dimension. Thus we add the numerator data from different codes of same image component, and the same is true for the denominator. The subsequent step will be to perform the final division to conclude the COG computation. Since, COG is computed in parallel with the first pass of the image component labelling, there is no need for a second labelling pass [6][7].

$$r_o = \frac{\sum_{r_i,c_i \in S} r_i I(r_i,c_i) + \sum_{r_i,c_i \in T} r_i I(r_i,c_i)}{\sum_{r_i,c_i \in S} I(r_i,c_i) + \sum_{r_i,c_i \in T} I(r_i,c_i)} = \frac{N}{D} \quad (2)$$

It can be seen that we perform the multiply and accumulate (MAC) operation along with the labelling process depending on different codes. Before applying the COG algorithm to a video frame, objects of interest must first be separated from the background at an image segmentation step [4]. This image segmentation is at its simplest form a threshold applied globally on the grey levels of the image.

III. PROPOSED ARCHITECTURE

In this section, we will describe the hardware architecture for calculating the COG from labelled pixel data. The *COG calculation* shown in Figure 3 is further divided into two main modules: *Sequencer* and *Multiply & Accumulate (MAC) unit*. As shown in Figure 4, The *Sequencer* is controlling all the data merging and computation and is also responsible for sending computed COG data to an arbitrary communication controller. The *MAC unit* accumulates the

nominators and denominators and performs serial divisions for final COG output. This accumulation of pixel data in the *MAC unit* is done in parallel with the labelling process. When the equivalence table is resolved, the *Table ready* signal becomes active and the sequencer starts merging data from regions belonging to same image objects. The sequencer is able to do this data merging based on the resolved equivalences read from the equivalence table, as shown in Figures 3 and 4. After merging, the numerators and denominators are ready for the division. At completion of the division, a *Compute done* signal is sent back to *Sequencer,* which then enables *Feature Strobe* to send the COG value to the communication device. In our experiment, we used an asynchronous serial port. The sequencer scans the equivalence table for all image objects until all objects are computed and transmitted. Two data tables, *A* and *B*, are used for numerators and denominators, as shown in Figure 5.

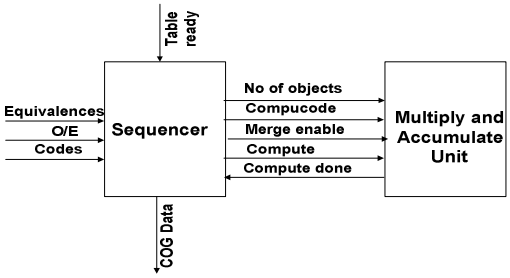

Figure 4. Architecture for COG computation.

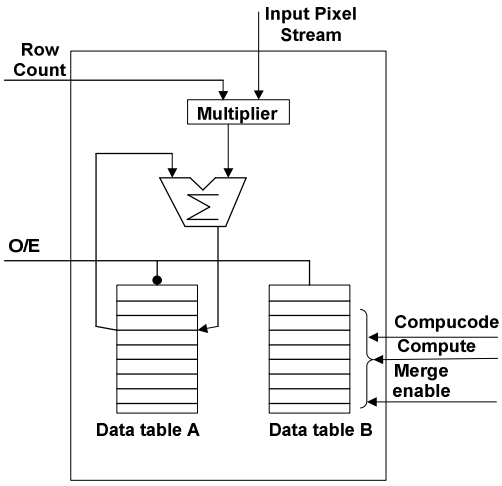

Figure 5. MAC unit for row centre computation.

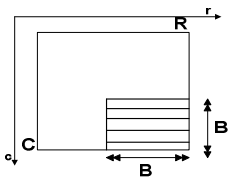

Figure 6. Worst case scenario for storage requirement.

One table accumulates the values after MAC operations and at the same time the other table is used for merging the data used for COG calculation. It will be shown in the next section, the memory storage requirement for the data tables are dependent on the maximum allowable object size, the maximum number of objects, and maximum pixel intensity.

IV. STORAGE REQUIREMENT

In this section, we will analyse the memory storage requirement for the COG calculation. According to Section II and Equation 2, numerators N and denominators D are accumulated in memory storage. The worst case scenario will be when a large image component of size BxB pixels is present at the right down most corner of the video frame, as shown in Figure 6. The frame size is R, C number of rows and columns. The maximum pixel intensity Imax is assumed to be a power of two, and we assume that the worst case object has maximum intensity for all its pixels. First, we calculate the maximum integer value I_{Num} for numerator row centre assuming that the row dimension is the largest. From Equation 2 we can conclude that,

$$I_{Num} \le \sum_{r=R-B+1}^{R} \sum_{c=C-B+1}^{C} rI_{\max} = BI_{\max} \sum_{r=R-B+1}^{R} r \quad (3)$$

,

$$\sum_{r=R-B+1}^{R} r = (R-B+1+R-B+2+\ldots+R-B+B) \quad (4)$$

$$\sum_{r=R-B+1}^{R} r = B(R-B) + \sum_{r=1}^{B} r \quad (5)$$

$$\sum_{r=R-B+1}^{R} r = B(R-B) + \frac{B\cdot(B+1)}{2} \quad (6)$$

Substituting Equation (6) into log2 of Equation (3) gives,

$$S_{Num} = \log_2\left(B^2 I_{\max}\left(R + \frac{1-B}{2}\right)\right) \quad (7)$$

S_{Num} is thus the number of bits required to store one single numerator. For our experiments, we have used the following values for R, C, B and Imax: R=640, C=480, B=170, Imax=255. For these values, $S_{Num} \le 32$. This means that for the maximum image component of size 170x170, we need 32 bits for accumulating the numerator in one single memory cell. For the maximum allowable number of labels, L=1024, we then conclude that two block RAMs with word length of sixteen bits and depth of memory equal to L are needed. As we use interleaving for memory access, a total of four block RAMs are required. The above expression for row centre is also valid for column centre, so four block RAMs also required for column centre. Equation 9 shows the storage requirement for the denominators.

 ISBN:978-1-61208-114-4

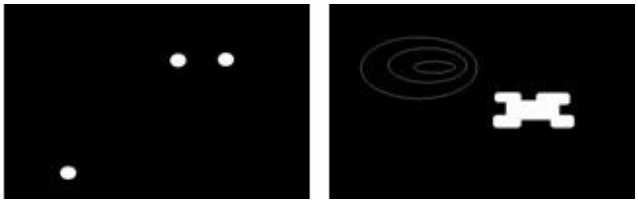

Figure 7. Input stimuli

TABLE I. DEVICE UTILIZATION AND POWER CALCULATION

Slices	**RAMB16**	**Slice flip-flop**	**4 input LUT**
3876 (28%)	17 (12%)	936 (3%)	7589 (27%)

a)

Dynamic power(mW)	**Clock**	**Signals**	**Logic**	**IOs**	**Mult**
	30	0.5	0.2	0.6	0.7
Dynamic Power	32.0 mW (at 27Mhz clock)				
Quiescent Power	103 mW				
Total Power	135 mW (at 27MHz clock)				
Frequency	Max 126 MHz				
Latency	313584 clock cycles 11.6 ms (at 27 MHz clock and 86 fps)				

b)

$$I_{Den} \le \sum_{r=R-B+1}^{R} \sum_{c=C-B+1}^{C} I_{\max} = BI_{\max} \sum_{r=R-B+1}^{R} \quad (8)$$

$$S_{Den} = \log_2\left(B^2 I_{\max}\right) \quad (9)$$

For the values of B and Imax chosen above, S_{Den}=23 so four block rams of 1024x16 is required for frame interleaved storage of L number of denominators. These denominators are the same for both row and column dimensions. Four additional block RAMs are used in the labelling process for maintaining equivalence tables A and B in Figure 3. One block ram is allocated for the line buffer used to maintain the neighbourhood shown in Figure 2. We now summarize the number of required block RAMs used for storage of numerators in both row and column dimensions (8), denominators (4), equivalence tables (4) and line buffer (1). This means that a total of (17) block RAMs are required for the combined labeller and COG calculator presented in this paper.

V. FUNCTIONAL VERIFICATION

The proposed hardware architecture for the calculation of COG was captured in VHDL. The model was simulated at register transfer level using two different input stimuli as shown in Figure 7. A frame size of R=640 and C=480 pixels was used for the experiment. Simulation output showed the correct number of objects and correct COG values for all objects shown in Figure 7.

VI. PERFORMANCE AND DEVICE UTILIZATION

The COG computation along with labelling as explained in section III was captured in VHDL and synthesized for implementation on the Xilinx VirtexII Pro device having speed grade 6. Post route and placement simulations were performed and design files were analyzed by the Xpower tool included in the Xilinx ISE Foundation toolset [8]. The pixel clock frequency was set to 27 MHz and at a frame speed of 86 fps. The input stimuli are shown in Figure 7. The power consumption, maximum clock frequency, latency and device utilization are results as reported by the toolset after synthesis and simulation, as shown in Table 1. We define latency as the time from the first pixel in a frame arriving at the input of the hardware unit until COG data of the first image object in the same frame starts to transmit on the output.

VII. DISCUSSION

The work presented in this paper shows the most efficient parallelization of the first pass of labelling along with COG calculation.

The performance of the labeller along with COG calculation is shown in Table 1a and 1b. One block RAM is used for the delay of previously assigned labels. Two triple port memories are used for the equivalence table, two read ports and one write port. The synthesis tool duplicates the ram in order to implement single write and dual read port memory. Twelve block RAMs are used for storing the numerators and denominators used for COG calculation. The static power dissipation is dominant, and we can see that only 28% of the available slices are active. The Maximum clock frequency reported by the toolset is 126 MHz. This clock frequency corresponds to 410 frames per second for a video format of 640 by 480 pixels, assuming no synchronisation overhead. This is a relevant assumption for a FPGA based smart camera having a high speed CMOS sensor connected directly to it. The latency is only 11.6 ms for the simulation at 86 fps. This latency is almost exactly the time of one frame with addition of the clock cycles needed for the serial divisors to conclude COG computation for the first object.

From Section IV, it is obvious that the memory storage requirement depends on the maximum allowable image component size BxB, as well as maximum number of labels L. These parameters must be set at system synthesis time and with a margin with respect to the expected video input. More efficient use of the block RAMs will thus require a hardware centric dynamic memory management.

VIII. CONCLUSION

In this paper, we presented a hardware architecture for computation of connected component labelling along with

 ISBN:978-1-61208-114-4

COG calculation. This implementation is suitable for embedded machine vision systems and smart camera applications having high demands on frame speed, power and latency. We have reason to believe that this work can be extended with computation of additional image object features such as area, bounding box or ellipse parameters.

REFERENCES

[1] H. C. Van Assen, H. A. Vrooman, M. Egmont-Petersen, J. G. Bosch, G. Koning, E. L. Van Der Linden, B. Goedhart, and J. H. C. Reiber, "*Automated calibration in vascular X-ray images using the accurate localization of catheter marker bands*", *invest. Radiol*, vol. 35, no. 4, pp 219-226, April 2000.

[2] X. Cheng, B.Thörnberg, A.W. Malik and N. Lawal, "Hardware centric machine vision for high precision center of gravity calculation", *Proc. of world academy of science, engineering and technology*, Vol. 64, pp. 736-743, Rome, Italy, 2010.

[3] W.WOLF, B.Ozer and T. Lv, "*Smart cameras as embedded systems*", *IEE computer*, vol.35, No. 9, pp. 48-53, Sept, 2002.

[4] C. Steger, M. Ulrich and C. Wiedemann, Machine vision algorithems and applications, Wiley-VCH 2008.

[5] M. Wnuk, "*Remarks on hardware implementation of image processing algorithms*". *Journal of applied mathematics and computer science*. Vol. 18, No. 1, pp105--110 (2008).

[6] C.T. Johnston and D.G. Bailey, "*FPGA implementation of a single pass connected component algorithm*", *Proc. Of 4th IEEE symp. On electronic design test & applications*, pp 228-231, Hong Kong, China 2008.

[7] B. Thörnberg and N. Lawal, "*Real-time component labelling and feature extraction on FPGA*", *Proc. of International Symposium on Signals, Circuits and Systems*, pp1-4, Iasi, Romania 2009.

[8] www.xilinx.com, last accessed, 14 November 2010.

[9] D.K. Kim, D.R. Lee, T.C. Pham, T.T. Nguyen and J.W. Jeon, "*Real- time component labeling and boundary tracing system based on FPGA*", *Proc.2007 IEEE Int. Conf. on Robotics and Biomimetics* (ROBIO '07), pp 189-194, 15-18 Dec. 2007 , Sanya, China

[10] D.G. Bailey and C.T. Johnston, "*Connected component analysis of streamed images*", *Proc. 2008 Int. Conf. on Field Programmable Logic and Applications* (FPL), pp 679-682, 8-10 Sept. 2008, Heidelberg, Germany.

 ISBN:978-1-61208-114-4