13—36

FPGA Implementation of a Sub-Pixel Correction Algorithm for Active Laser Range Finders

Hakim Khali[1]
Faculty of Computer Science & Engineering
Ajman University of Science and Technology

Yvon Savaria[2]
Department of Electrical Engineering
Ecole Polytechnique of Montreal

Abstract

This paper presents a FPGA implementation of a Sub-Pixel correction algorithm for active laser range finders. It shows how to replace complex CPU operations by an efficient use of arithmetic functional units and lookup tables LUTs. This leads to a less complex architecture and an increase in performance. The architecture of a processor element, its complexity and performance on a Xilinx FPGA device are presented.

1 Introduction

Active optical triangulation systems project light towards an object. The light is then reflected by the surface and collected by a detector. The collected energy can provide information about the object under analysis, for example its range [1]. The data to be processed is a 3-D surface map [2] which is captured by scanning a laser beam onto the scene. Measurements are made on the illuminated points in the scene and are mapped onto (x, y, z) rectangular coordinates by applying a set of trigonometric transformations. Figure 1 shows an example of an active laser range finder based on an autosynchronized camera. When the reflectance of the scene under analysis is uniform, the intensity profile of the image spot is Gaussian and its centroid is correctly detected assuming an accurate peak position detector. However, when a change of reflectance occurs on the scene, the intensity profile of the image spot is no longer gaussian. This change introduces a deviation Δp on the detected centroid p, which will lead to erroneous coordinates. To correct this deviation, an iterative correction algorithm [3] has been developed. As explained in the next sections, this algorithm is characterized by a high complexity which limits the throughput of high-speed optical systems. In this paper we present a VLSI architecture that is suitable to accelerate the iterative correction method and to improve the throughput of the corresponding optical system. This paper contains the following sections. Section 2 and 3 respectively presents and analyses the iterative correction algorithm. Section 4 and 5 respectively presents the architecture of the processor element and its corresponding results. Section 6 concludes the paper.

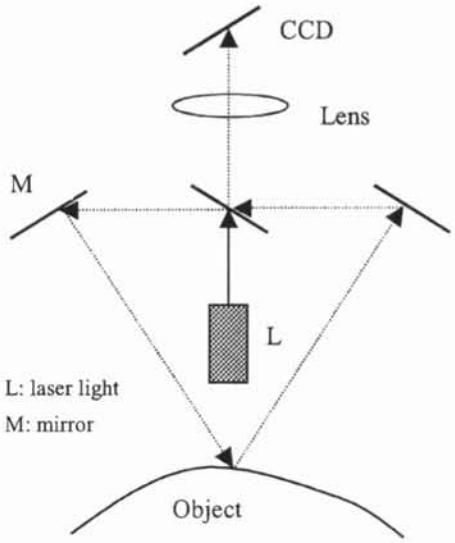

Figure 1. General view of an autosynchronized system.

2 Iterative Correction Method

The iterative correction method is based on a two-loops algorithm that minimizes the distance between a computed image spot position and a predicted image spot position. This algorithm is described in details in [3]. A summarized version is given below.

For each magnification factor
For each position of the reference spot
1- Compute the energy of each pixel of the current reference spot.
2- Compute the reflectance value corresponding to each pixel of the detected spot.
3- Compute the finite differences of first and second order using the reflectance values found in step 2.
4- Compute the maximum value of the finite differences of second order.
End
End
Compute the minimum of all maximum values found in step 4.
End.

[1] Address: PO.BOX 346 Ajman, UAE. E-mail: ajac.hakim@ajman.ac.ae
[2] Address: PO.BOX 6079, Station Downtown, Mtl, Que, H3C 3A7, Canada. E-mail: savaria@vlsi.polymtl.ca

As explained in [3], the number of magnification factors and positions of the reference spot are respectively equal to $\lceil W/R \rceil$ and $\lceil W/2R \rceil$, where W and R are respectively the pixel width and the system resolution. These values show that the iterative correction algorithm has a quadratic complexity $O((W/R)^2)$. Depending on the camera used to perform the 3D measurements, this complexity may lead to a high number of CPU operations required to achieve the desired correction. As an example, a typical camera has the following parameters:

- $W = 50\mu m, \quad R = 0.78\mu m$.
- Average spot size = 10 pixels.
- Average number of reference spot samples per pixel = 64.

Based on the above values, the total number of CPU operations required to perform the desired correction can be estimated as:

$$N_{opr} = \left(\frac{50}{0.78}\right)\left(\frac{50}{2*0.78}\right)(10*64+73) \approx 1382700$$

Assuming that each operation requires one clock cycle to execute on a 100-MHz processor, a sequential execution of the iterative correction algorithm leads to a throughput of 72 corrections per second, which is considered as a very low throughput. To increase this throughput, we propose to exploit the parallelism available in the iterative correction method. This parallel version will be implemented by a dedicated VLSI architecture.

3 Analysis of Proposed Method

The iterative correction algorithm features characteristics which make it suitable for a parallel VLSI implementation. These characteristics can be summarized as follows:

1- All loop iterations are independent from each others.

2- Specific computations can be achieved by using look-up tables (LUTs).

The first characteristic allows to increase the degree of parallelism by assigning loop iterations to dedicated processor elements (PE)[4]. The second characteristic greatly contributes to ease the design of the corresponding VLSI architecture and to increase the system throughput, measured as the number of corrections performed by second. LUTs can be very effective to replace complex computations by a memory access [5]. As an example, step 2 in the iterative correction algorithm is achieved by computing the following equation:

$R(i) = I(i)/I_{ref}(i)$, where $R(i), I(i), I_{ref}(i)$ are respectively the reflectance, the current and reference intensities of pixel i. The computation of the inverse value $1/I_{ref}(i)$ can be avoided by using a LUT containing all inverse values for all pixels. Based on the above considerations, we developed the architecture of the processor element that is described in the next section.

4 The Processor Element PE

Each processor aims to implement the computations involved in one inner-loop iteration. The corresponding VLSI architecture is presented in figure 2.

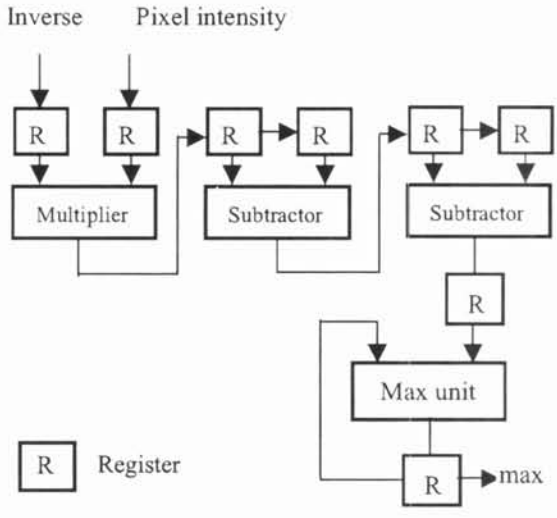

Figure 2. Basic architecture of a processor element.

As shown in figure 2, each step in the algorithm is implemented by a set of arithmetic functional units. These functional units are connected in a pipeline-like manner using registers, which allow the architecture to receive one new input data every clock cycle. Figure 3 shows the proposed architecture with several PEs.

As mentioned earlier, each processor element computes one inner-loop iteration. Depending on the number of PEs, a given PE may be assigned several iterations. All Max results are compared together in a reduction-like manner and the Min value is retained. This Min value corresponds to one magnification factor (outer loop) and is kept locally. The same process is repeated for all remaining magnification factors. After processing all iterations, The final MIN value is transmitted to the host. The proposed VLSI architecture relies on the ability to use several LUTs in parallel. The total amount of memory needed to perform all computations becomes a key factor and must be estimated. This amount of memory is a function of:

- N_{mag} : number of magnification factor (outer loop).
- N_{pos} : number of ositions of the reference spot to analyze.
- $S_{\min}$: smallest size (in pixel unit) of reference spot.
- $S_{\max}$: largest size (in pixel unit) of the reference spot.
- k : number of bytes required to represent a result from LUT.

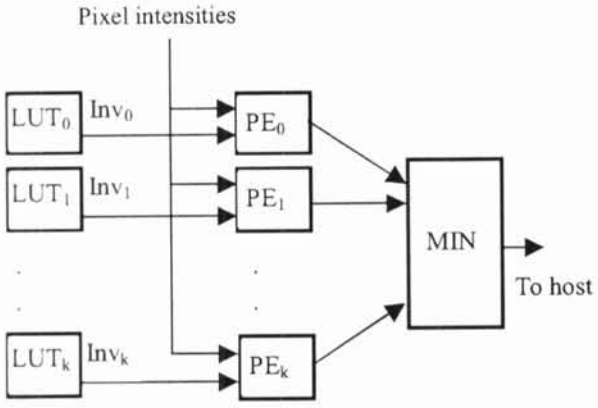

Figure 3. Parallel structure of the iterative correction algorithm.

The total amount of memory, Q_{mem}, needed to perform the targeted computations, can be estimated by:

$$Q_{mem} = N_{mag} * N_{pos} * k * \sum_{i=S_{min}}^{S_{max}} i.$$

Recall that $N_{mag} = \lceil W / R \rceil$ and $N_{pos} = \lceil W / 2R \rceil$.

Figure 4 shows the variation of Q_{mem} for $W = 50\mu m$, $R = 0.78\mu m$, and $k = 1$. Figure 4 shows that for optical system where the ratio W / R is high, the cost of using LUTs can be very high, which suggests the use of other approaches to implement the iterative correction algorithm.

5 Results

The architecture of a processor element has been implemented on a Xilinx XC4036 FPGA. All functional units (multiplier, subtractor, compare unit) has been generated using Xilinx software tools [6]. A first implementation of the processor element showed a complexity of 170 CLBs (configuration logic block), which allowed to integrate 7 processor elements. A timing analysis performed by Xilinx software tools showed that the design can run at 20MHz on a XC4036-6 version without optimization. The speed of the processor element can be dramatically increased on Xilinx Virtex and Virtex II architectures which offer high-performance hardware resources, high-speed arithmetic logic and a huge amount of I/O pins on FG packages to implement multiple LUTs [7], [8]. The communication interface between the processor elements and the host can be implemented using the RAM blocks which can be configured as a bi-directional FIFO with variable width and depth sizes, depending on application requirements.

6 Conclusion

In this paper we presented a FPGA implementation of a sub-pixel correction algorithm for active laser range finders. The proposed implementation combines the use of pipeline-like computations and lookup tables to increase the system throughput. The high degree of parallelism featured by the algorithm makes it a good candidate for implementation on high-density FPGA devices, like Xilinx Virtex and Virtex II FPGA families. Depending on the number of processor elements and the ratio W / R, the amount of memory and I/O pins required by LUTs may be high, which suggest, in this case, the use of other approaches to implement the iterative correction algorithm. As a future work, we intend to perform a hardware-software analysis of the iterative correction algorithm based on application requirements, type of FPGA and the value . This analysis should lead to a hardware-software implementation of the iterative correction algorithm.

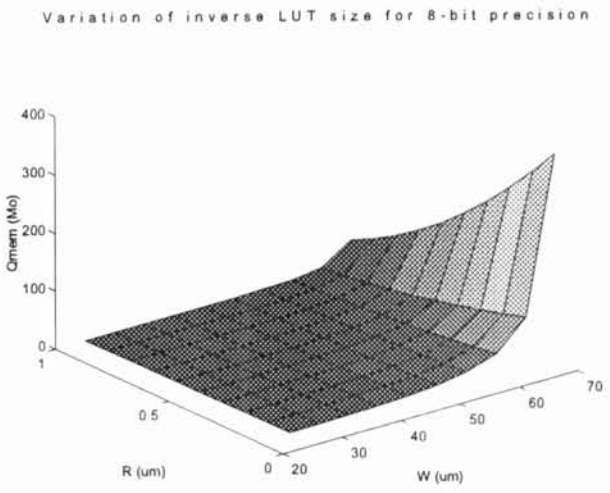

Figure 4. Variation of Q_{mem} with 8-bit-LUT precision.

References

[1] Rioux, M. Laser Range Finder Based on Synchronized Scanners. Appl. Opt., 23, 1984, pp. 3837-3844.

[2] Rioux, M. Applications of Digital 3-D Imaging. Canadian Conf. on Electr. and Comp. Eng., Ottawa, Sept. 1990, pp. 37.3.1-37.3.10.

[3] Khali, H., *et al.* Iterative Model for Accurate Centroid Correction Applied to Laser Range Finders in the Case of a Reflectance Gradient. *Submitted for publication in IEEE Transactions on Instrumentation and Measurements.*

[4] Savaria, Y., Bois, G., Popovic, P. et Wayne, A. Computational Acceleration Methodologies: Advantages of Reconfigurable Acceleration Subsystems. SPIE Proceedings, 2914, 1996, pp.195-205.

[5] Khali, H., Y. Savaria et Houle, J.L Computational Limits of Homogeneous Acceleration Using Lookup Tables. High Performance Computing Systems (HPCS97), 1997, pp. 345-351.

[6] Xilinx, COREGEN, V1.4, 1998.

[7] Xilinx Virtex datasheet.

[8] Xilinx Virtex II datasheet, DS031-2, 2001.