J. Vis. Commun. Image R. 21 (2010) 9–16

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

A real-time FPGA architecture for 3D reconstruction from integral images ☆

D. Chaikalis *, N.P. Sgouros, D. Maroulis

Department of Informatics and Telecommuncations, National and Kapodistrian University of Athens, Ilisia 15784 Athens, Greece

ARTICLE INFO

Article history:
Received 4 March 2009
Accepted 6 September 2009
Available online 12 September 2009

Keywords:
Three-dimensional
Integral image
Image reconstruction
Image representation
Autostereoscopy
Architecture
Real-time
Hardware
FPGA

ABSTRACT

In this paper, we present a hardware architecture for real-time three-dimensional (3D) surface model reconstruction from Integral Images (InIms). The proposed parallel digital system realizes a number of computational-heavy calculations in order to achieve real-time operation. The processing elements are deployed in a systolic architecture and operate on multiple image areas simultaneously. Moreover, memory organization allows random access to image data and copes with the increased processing throughput of the system. Operating results reveal that the proposed architecture is able to process 3D data at a real-time rate. The proposed system can handle large sized InIms in real time and outputs 3D scenes of enhanced depth and detailed texture, which apply to emerging 3D applications.

1. Introduction

The demand for 3D imaging applications is continuously rising, covering a wide variety of both specialized and everyday visual communications. A great number of 3D capturing and display systems have been developed [1], which target in acquiring real world objects or scenes and provide users with their 3D representations. In general these systems can be divided in stereoscopic and autostereoscopic systems [1]. In stereoscopic systems, the viewers need glasses or other special viewing devices in order to experience the 3D effect, while most of the stereoscopic capturing setups use two cameras in order to provide the correct input to each of the viewer's eyes. In autostereoscopic setups [2] the optics needed in order to provide the 3D effect are embedded in the display and in most cases a large number of cameras is used during the pickup process in order to provide the user with the correct stereoscopic image as the viewer moves in front of the display. The capturing systems in these setups are usually bulky and in most cases prone to calibration errors after use. In recent years depth cameras are used in order to capture the 3D objects and provide the necessary input for stereoscopic or autostereoscopic displays [3]. However, current cameras are too expensive for everyday and broad commercial use. A complementary technique that provides high-quality autostereoscopic content with 2D continuous parallax, reduced complexity and significantly lower cost is Integral Photography (IP) which was proposed by Lippmann [4] in 1908. Its digital counterpart, Integral Imaging (InIm) has recently evolved as a robust alternative to the afore-mentioned autostereoscopic setups as the Charged Coupled Devices (CCDs) and Liquid Crystal Displays (LCDs) reached adequate resolutions.

The operational principle of an InIm capturing setup is based on the acquisition of images of small portions of an object through a lens array (LA) placed in front of a CCD as shown in Fig. 1(a). Each of these small images recorded on the CCD is called an Elemental Image (EI). In the reconstruction stage, the EIs that form the InIm are back-projected using an LCD, through an appropriate LA, to provide a 3D representation of the original object, as shown in Fig. 1(b).

One significant issue in a 3D system's pipeline is the suitability of the acquired data for other applications such as 3D object reconstruction, recognition and 3D tracking of moving objects in presence of occlusions. The afore-mentioned characteristics designate InIm as a promising candidate for all the above applications [2]. The robustness of an InIm system relates with the fact that none or elementary calibration issues arise during use, in contrast with multi-camera setups. In addition the technique provides both texture and shape reconstruction of an object without the use of structured light illumination as used in many 3D cameras. However, as in all 3D cameras, the 3D object reconstruction or tracking algorithms use estimations of projections of the object in a large

☆ This work was realized under the framework 8.3 of the Reinforcement Programme of Human Research Manpower ("PENED 2003"-03ED656), co-funded 25% by the General Secretariat for Research and Technology, Greece, 75% by the European Social Fund, and by the private sector.

* Corresponding author. Fax: +30 2107275333.
E-mail addresses: dhaik@di.uoa.gr, dio.chaik@gmail.com (D. Chaikalis), nsg@di.uoa.gr (N.P. Sgouros), dmarou@di.uoa.gr (D. Maroulis).

1047-3203/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2009.09.004

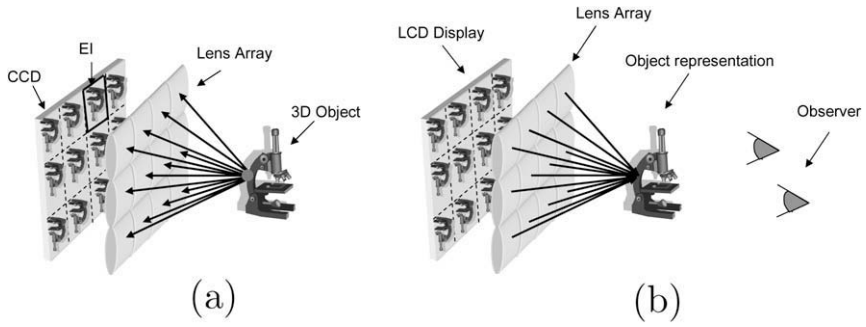

Fig. 1. InIm (a) capturing and (b) display setup.

number of images, which usually causes delays in the processing pipeline. The potential of creating high-quality 3D object reconstructions from InIms leads to hardware implementation of time-consuming algorithms in an effort to provide real-time characteristics for the processing pipeline.

Several attempts for 3D object reconstruction using InIms have been reported in the literature [5–8]. In some works, such as Shin et al. [9] the term 3D object reconstruction is used to describe the generation of 2D images from multiple views and focus depths from a single InIm. Most of the proposed InIm reconstruction algorithms are targeted to small object InIms, that are represented on a limited number of EIs. This is mostly done to avoid stitching problems during the reconstruction stage caused by abrupt discontinuities due to depth estimation errors [9]. Moreover EI modification techniques are proposed in an effort to increase depth accuracy [6].

A method that is focused on the reconstruction of a fully 3D surface model is proposed in [10], where 3D shape and texture of real-life objects are reconstructed using the InIm technique. The method addresses the classic stereo correspondence problem, where a set of points in one view must be identified as the same points in another view, using horizontal and vertical parallax and multiple correspondences. The output of the process is the 3D polygonal representation of the object's shape and texture. This is shown in Fig. 2. To achieve this, the method utilizes a two-step strategy: it initially computes a rough grid and subsequently refines it. It has unique properties compared to previous works, such as configurable depth accuracy, direct and seamless triangulation and the ability of handling cases where the field of view of the EIs is less than the acquired object's size at a certain distance.

Several attempts to accelerate 3D reconstruction applications utilize dedicated platforms, most notably FPGA devices for the implementation of the digital architecture [11] while other researchers use clustering for boosting performance [12]. However, all these implementations target typical two-view stereoscopic systems and there is no implementation for accelerating a full 3D surface model reconstruction method.

In this paper, a robust, parallel digital system for 3D object reconstruction acceleration is presented. By efficiently exploiting the properties of the reconstruction algorithm, the implemented architecture demonstrates extensive processing capability. The Processing Elements (PEs) operate simultaneously on multiple image areas, thus maximizing processing throughput and reducing idle PE time. Memory reads are minimized by reutilizing EI data when appropriate. Data reutilization has a positive effect on processing time, since consecutive calculations can proceed simultaneously. Timing results reveal the real-time capabilities of the architecture, which can be integrated in a robust contemporary 3D reconstruction system in order to target a wide range of applications.

2. 3D reconstruction from InIm algorithm outline

The method proposed in [10] estimates the 3D shape and texture of an object from a single InIm. The reconstruction process consists of the computation of 3D points (vertices) on the surface of the object and their connection in a polygonal (e.g. triangular) mesh. Based on this process, the reconstruction algorithm can be summarized to the following three steps: vertex grid computation, grid refinement and triangulation, and post-processing.

The computational core of the first two steps is based on pixel distance calculations, which are used to determine the best correspondence among several candidates. The distance $D(p_1, p_2)$ between two pixels p_1 and p_2 from different EIs is defined using a simple but effective metric:

$$D(p_1, p_2) = \sum_{j=-W}^{W} \sum_{i=-W}^{W} |E_1(u_1 + i, v_1 + j) - E_2(u_2 + i, v_2 + j)| \quad (1)$$

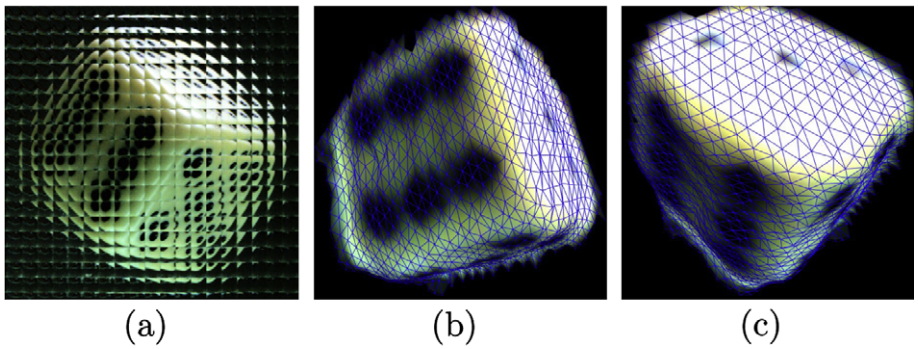

Fig. 2. Reconstruction of a dice: (a) integral Image with $f = 3.3$ mm. (b and c) Reconstructed 3D object rendered with triangulation superimposed.

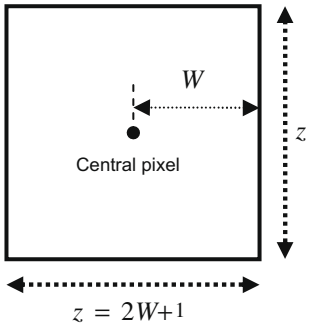

Fig. 3. Definition of the comparison window size.

In the above equation, E_1 and E_2 are the two EIs, u and v are the corresponding pixel coordinates and W defines the size of the $z \times z$ comparison window, where $z = 2W + 1$, as depicted in Fig. 3.

The above distance metric is subsequently extended to more than two EIs. In practice, $2N + 1$ neighboring EIs per direction are used, thus forming a symmetrical neighborhood area of radius N around each EI (see Fig. 4).

The first algorithmic step involves the computation of the 3D vertices that correspond to the central pixels of each EI. These vertices form a regularly sampled grid that is refined in the following step of the algorithm. The best correspondence has the minimum sum of the distances over all neighbors. Specifically, for the central pixel $p_{k,l}$ of each EI $E_{k,l}$, the 3D vertex P that minimizes the expression:

$$D_{\text{total}} = \sum_{j=-N}^{N} \sum_{i=-N}^{N} D(p_{k,l}, p_{k+i,l+j}) \tag{2}$$

is determined, where $p_{k+i,l+j}$ is the projection of P in EI $E_{k+i,l+j}$.

Fig. 5 outlines the search areas used in the algorithm for $N = 1$ and EI size of $L \times L$ pixels. These areas are defined by the search method [10], and their sizes are summarized to Table 1. It should be noted that only the $z \times z$ windows that are entirely inside the EIs are used for the comparison.

In order to refine the vertex grid, we introduce a subdivision parameter S, which defines how many vertices we will compute between the computed central vertices. The reason for separating this step from the previous one is to allow seamless triangulation. As seen in Fig. 4 we can project the central pixels from neighboring EIs computed in the previous step onto a specific EI. In this way, additional vertices that are used for refinement will be derived only from pixels between the central pixels of the EI and the correspondences of the central pixels of the neighboring EIs. After the additional pixels are determined in this step, the above distance metric is used in order to compute the reconstructed vertex and add it to the grid.

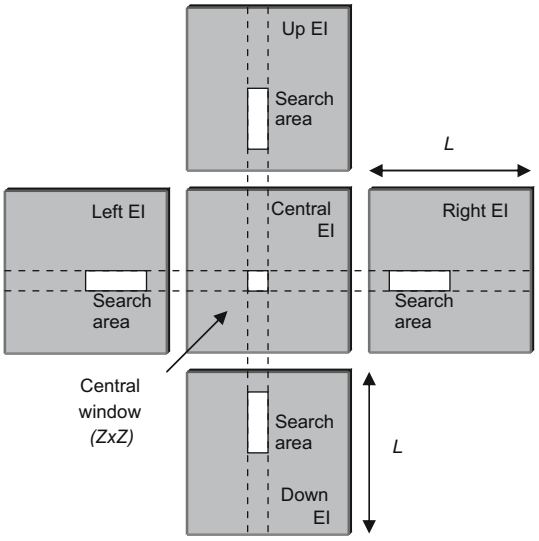

Fig. 5. Algorithm search area outline. For depiction clarity, $N = 1$.

Table 1
Search area sizes according to central EI location.

Cental EI pixel	(x, y)
Left neighbors	$(x+3, y)$ to $(L-W, y)$
Right neighbors	$(x-3, y)$ to (W, y)
Up neighbors	$(x, y+3)$ to $(x, L-W)$
Down neighbors	$(x, y-3)$ to (x, W)

Finally, to improve the quality of the final reconstruction, a series of post-processing filters are applied on the grid of vertices, in order to remove possible spikes and white noise. A sub-sampling filter can also be applied if the object is reconstructed at a higher resolution than required.

3. FPGA implementation

As it is evident from the algorithm description, the pixel distance metric $D(p_1, p_2)$ imposes the most significant processing strain, since it is used for every pixel comparison of every window

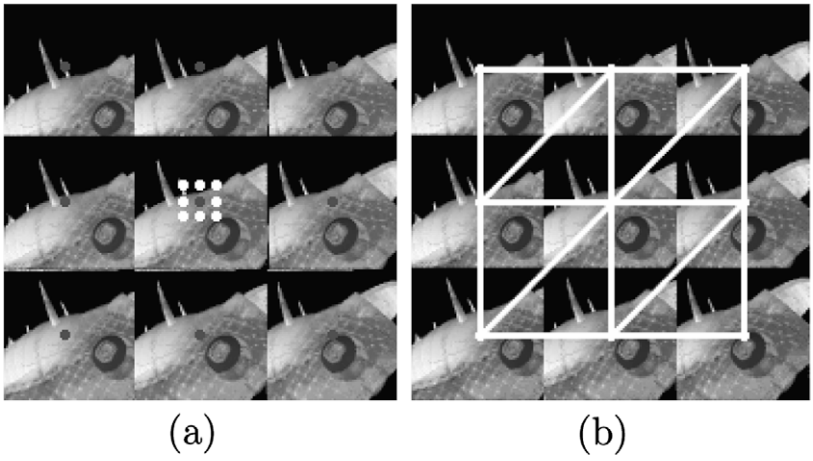

Fig. 4. Part of an InIm forming a neighborhood of EIs ($N = 1$): (a) the central pixels for each EI and their correspondences in the central EI are dotted. (b) Triangulation of the same neighborhood using only central pixels, superimposed over the image.

in every neighborhood. Moreover, software calculation of such a metric is time-consuming due to the complex nature of the absolute value calculation and the subsequent multitude of additions. Addressing these concerns, hardware implementation of the calculation can significantly improve performance. The repetitive nature of the metric computations favors a parallel architecture where many PEs can operate simultaneously. In addition, an optimized implementation can sufficiently minimize the redundancy in data access, imposed by most exhaustive or detailed search methods that aim for high-quality results.

In the proposed hardware system that targets real-time InIm reconstruction, the Sum of Absolute Differences (SADs) is used as the distance metric. A detailed description of the SAD architecture can be found in [13,14]. The SAD metric involves summation of pixel intensity values. This way, its hardware implementation causes no resolution loss, since all its intermediate and final products are integer values. Immediate access to arbitrary comparison windows is achieved by using look-up tables for their locations and optimized memory organization architecture.

3.1. Implementation considerations

The SAD is a very effective distance metric which has been shown to be efficiently implemented in hardware [13,15]. Aiming for a practical system, attention must be drawn to the integration feasibility of such a component in a contemporary yet reasonably low-cost device, in coordination with all peripheral systems. Thus, processing speed must be combined with relatively low data throughput and reasonable gate count. The low data throughput allows for low pin-out count and the limited gate count in an FPGA device is translated to limited area coverage, both parameters determining the feasibility of the design.

Taking the above into concern, a completely parallel SAD Unit is ruled out mostly because of its high memory bandwidth demands [15], which would require more than one of modern FPGA devices. Since the calculations are performed on 2D data, it is straightforward to decompose the SAD calculation in two steps: performing the calculation in one dimension, so many times as the next dimension imposes, and summing up the intermediate results until the final SAD value is produced.

In the proposed implementation, a z-sized SAD Unit is designed as the core PE, in order to target a single FPGA device. This unit can perform pixel comparisons and additions on a row or a column of the comparison windows (blocks) in every clock cycle. The intermediate results are accumulated and the final SAD value for the $z \times z$ block comparison is available after z clock cycles. The outline of this SAD Unit is depicted in Fig. 6.

In order to perform the Absolute Difference (AD) calculation of two operands, the smaller is determined and inverted, then both operands are passed to an adder tree. Finally, a correction term is added, in order to compensate for the initial inversion error. This architecture favors parallel AD implementation, because it avoids dealing with 2's complement values. The subtraction operation and 2's complement calculation are replaced by one addition and two inversions, as shown in Fig. 7. The specific components in Fig. 7(b) have more hardware cost for a single PE, but this cost is inverted by using the 2's complement correction term only at the final stage of the value additions, rather than after every value couple subtraction.

The structure of the SAD Unit can be exploited in an one-dimensional systolic array [16] of M units as depicted in Fig. 8, where M is equal to the number of block comparisons in the search area of the EIs. If the input pixel lines are set to be perpendicular to the direction of the search (e.g. for a horizontal search area, set the block columns as the input), then each unit in the array can start its operation one clock cycle after the previous one, when the next line of

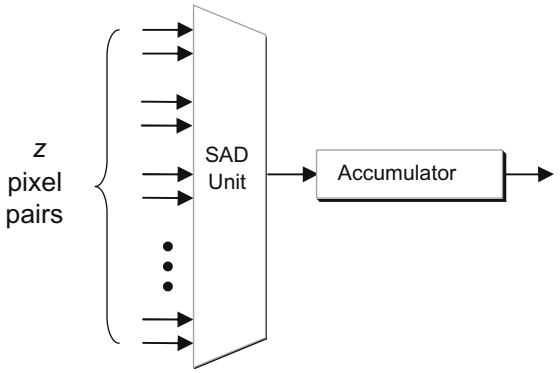

Fig. 6. The implemented SAD Unit outline.

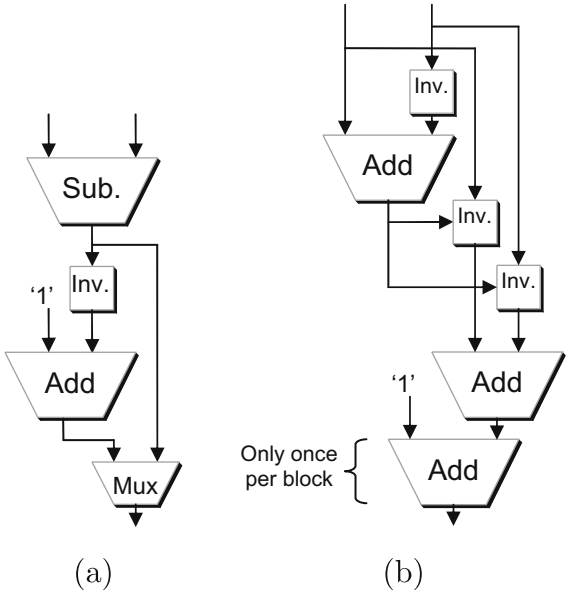

Fig. 7. Example of the absolute difference calculation architecture using (a) subtraction (2's complement) and (b) addition (1's complement). Note that in this case the final addition is performed only once per block comparison.

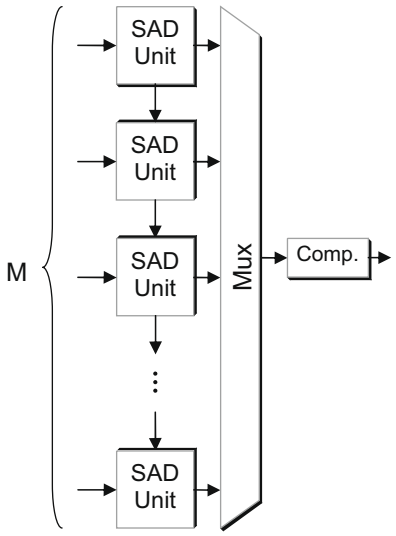

Fig. 8. Depiction of the M-unit array and the sequential comparison.

pixels is read from the search area memory. In this manner, by propagating the pixels of the search area through the array, memory access is significantly minimized.

The successive operation of the SAD Units in an M-unit systolic array also removes the need for a parallel comparison unit for the outputs, which would aggravate area coverage and operation speed of the system. The SAD values of such an array are available at the outputs of the SAD Units in successive clock cycles, and they are compared using a sequential comparator in order to determine the smallest value. The comparisons impose only one clock cycle delay on the process, since they begin when the first unit of the array outputs its SAD value, and end one clock cycle after the last unit of the array outputs its SAD value. The total delay of this design is $z + M + 1$.

The design of the FPGA memory takes into account the need for direct access to arbitrary blocks in an EI, which is useful for the grid refinement stage. The compared block positions are pre-determined and stored in a look-up table. For each pixel comparison, the appropriate block positions are fetched from the look-up table and are translated into address words. The memory modules are designed with the ability to uninterruptedly feed the SAD array with image data at the needed rate, regardless of the block position of the central EI. On this account, z memory modules are used for every EI. Each of these memory modules stores k pixel lines of the EI, in intervals of z lines, where k is dependent on EI size. For example, in the first memory module, lines $0, z, 2z, 3z, \ldots, kz$ are stored. In this way, z pixels of each line can be accessed in every clock cycle regardless of the pixel's coordinates (and hence block position) in the EI. The memory arrangement is depicted in Fig. 9.

Moreover, the horizontally adjacent EIs (i.e. left and right neighbors) must be stored row-wise and the vertically adjacent EIs (i.e. up and down neighbors) column-wise. This arrangement favors fast calculations of the sums regardless of the direction of the search area. Due to the implemented search method, the central EI must be stored in two different ways, both row- and column-wise. The total memory size is dependent on the EI size L and the block size z.

3.2. Implementation details

A practical dynamic InIm acquisition system was considered as a case study. Using the typical InIm configuration, the lens array is positioned between the camera's objective lens and the CCD. Using a square contemporary CCD, an image resolution of 2048×2048 pixels can be used for acquisition so that it can provide sufficient dynamic InIm using contemporary devices. Given the tight space inside a camera, only a lens array with a low focal length can be used. For such a lens array (with focal length 3.3 mm) the best set-up would correspond to 64×64 pixels per lens, resulting in 32×32 lenses (and corresponding EIs). For 64×64 pixels per EI, the optimal window is 11×11 ($W = 5$), since it provides relatively stable results with respect to different texture detail. The neighboring area is restricted to sizes of $N = 1$—4, since for larger sizes the reconstruction accuracy gain is relatively small compared to the computational cost, especially for medium and large depths. For the same reason, the subdivision parameter is set to $S = 2$ according to practical system considerations. Further analysis on the selection of the parameter values can be found in [10]. The parameter values are summarized in Table 2.

For the evaluation of the architecture performance, the system was implemented using the Xilinx ISE tools on a PLDA PCI-Express development board using the Xilinx Virtex-5 LX110T FPGA device [18]. The implemented algorithm is divided into two steps, the initial grid computation and grid refinement. For the first step, we only compute the 3D position of the central pixel of each EI. For the second step, we compute the 3D position of three additional pixels for each EI (for $S = 2$). For the utilized configuration, there are 32×32 EIs, but we use only the inner $(32 - 2N) \times (32 - 2N)$ (so that each central EI has N neighbors per direction). The maximum size of the search area is correlated with the EI resolution and it also defines the size of the SAD Unit array. An outline of the search areas for neighboring EIs is depicted in Fig. 10.

In the current framework, we implement $M = 5011 \times 1$ SAD Units for parallel operation on two neighbors. This scheme is based on the observation that, when considering opposite neighbors together, the total number of calculations remains constant. This is due to the definition of the search area which depends on the position of the block in the central EI. As this block's position moves away from the center, the size of the search area in one neighbor

Table 2
The parameters of the practical system and their values.

Parameter name	Symbol	Implementation value(s)
Neighboring area radius	N	1–4
Subdivision	S	2
Comparison window radius	W	5
Comparison window size	z	11
PE number	M	50
EI dimension	L	64

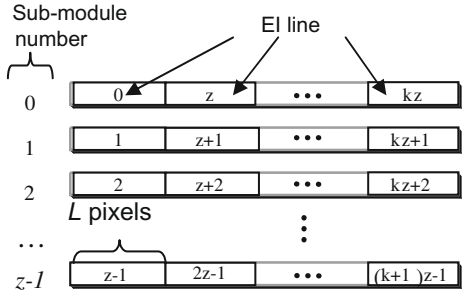

Fig. 9. The number of memory modules and EI data arrangement.

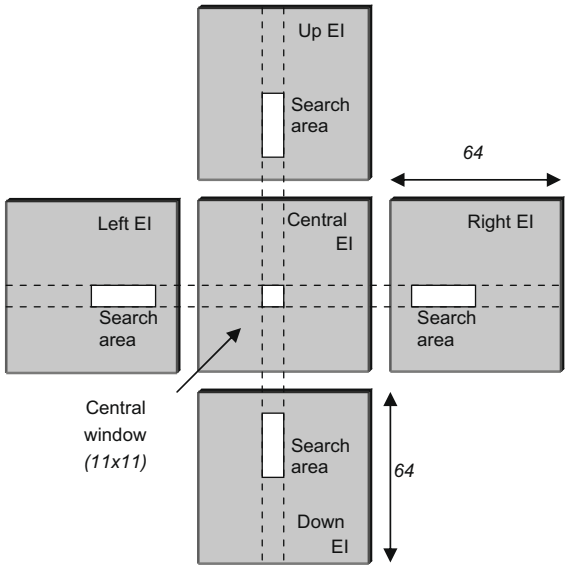

Fig. 10. Elemental Image (EI) size, block size and search area outline for the practical acquisition system. For depiction clarity, $N = 1$.

increases as in the opposite neighbor is reduced in the same amount. The sum of opposite search areas, which also defines the size M of the SAD Unit array, can be calculated based on Table 1 to $M = L - 2W - 4$. For the specific values of L and W in the practical system, it is derived that $M = 64 - 10 - 4 = 50$.

Compared with the single-neighbor approach [17], this scheme requires extra circuitry that allows the array of 50 SAD Units to operate as two sub-modules of variable size each. The central EI's block arrives at the input of each sub-module, and the neighboring blocks are also distributed to the correct SAD Units. Additionally, the output of this scheme is properly wired in order to handle the variability of the comparisons it must carry. Each unit's output is directed to the proper selector unit of the two available, according to the comparison block coordinates of the central EI. Each selector is connected to one input of the 2-to-1 adder, which adds up the SAD values of each proper pair of blocks to one intermediate value, which is stored in the temporary results block RAM.

The outline of the proposed metric calculation architecture is depicted in Fig. 11. The three main assets of this architecture compared to the baseline approach presented in [17] are:

- The doubling of speed that it achieves: the system simultaneously processes two neighbors of each EI. The speed increase of this architecture is almost 100% compared to single neighbor processing, since only one latency clock cycle is added to the end of the datapath for the extra addition that is required.
- The flexibility that it offers for the position of the block in the central EI: moving the block away from the center of the central EI leads to search area size variations in the neighboring EIs. The system can cope with these variations thanks to the ability of the PEs to operate on either of two search areas, alleviating the need of either restricting the central block position to specific coordinates or using more PEs.
- The more efficient use of the PEs: summing up the sizes of the two search areas that are simultaneously processed, the total size is the same to the number of implemented PEs. The PEs are always active during the processing regardless of the search area sizes. On the contrary, if operating on a single search areas that can vary in size, the maximum number of PEs should be implemented. Some of them would remain idle when the search area is smaller than the maximum given size.

The memories are organized in a way that a central EI and two neighboring ones can be simultaneously accessed by the system. In our architecture, each EI storage memory is comprised of 11 memory sub-modules, each storing six lines of 64 data bytes. An example for the left neighbor memory module (LMem) contents is shown in Table 3.

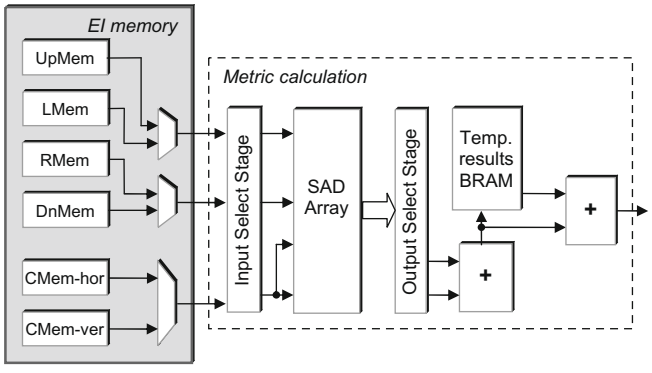

Fig. 11. The outline of the metric calculation architecture.

Table 3
The EI line arrangement in the memory modules for the left neighbor.

Memory sub-module number (left)	EI line number
0	0, 11, 22, 33, 44, 55
1	1, 12, 23, 34, 45, 56
2	2, 13, 24, 35, 46, 57
3	3, 14, 25, 36, 47, 58
4	4, 15, 26, 37, 48, 59
5	5, 16, 27, 38, 49, 60
6	6, 17, 28, 39, 50, 61
7	7, 18, 29, 40, 51, 62
8	8, 19, 30, 41, 52, 63
9	9, 20, 31, 42, 53
10	10, 21, 32, 43, 54

The central EI memory occupies 22 memory sub-modules, one more is used for the intermediate and one for the final results. The remaining FPGA memory is used for the neighboring EIs. In total, 24 neighboring EIs are stored in the FPGA memory, according to the LX110T device specifications [18]. As a result, the maximum neighborhood level that can be processed without reloading image data to the FPGA memory is $N = 6$. When addressing neighborhoods of seven or more adjacent EIs, the remaining neighbors must be loaded in the already read memory modules of the previous neighbors, without stalling the process.

The image data from the neighbors are forwarded to the proper PEs, which are determined by the coordinates of the central EI. This way, more PEs can be assigned to operate on the bigger search area and less on the smaller one, increasing system efficiency. The central EI window serves as an input on two PEs, namely the first ones to operate on each search area. The data are propagated to one next PE on each clock cycle, to create the appropriate window comparison pair on every PE. The output of the PEs need similar organization in order to separate the outputs according to the neighbor each output is related to. A properly designed selector directs the results of the first comparison to the first input of the successive adder, and the results of the second comparison to its second output. This way, the SAD results of each window pair in each search area are added and stored in the temporary results' block RAM. Using the same procedure, the next SAD results are computed and added in the following phase from the remaining two neighbors of the same neighborhood level. Adding these results with the temporarily stored ones in a final addition, the required SAD values are derived. These values represent the comparison results between a pixel in the central EI and determined quadruples of pixels in the search areas of the four neighboring EIs.

The final processing stage requires the results to be compared in order to determine the minimum value. A comparator is implemented in order to perform this calculation, which takes place in a sequential manner. This is possible because each SAD result is produced one clock cycle after the previous one, thus a comparator needs to operate only on two values on each clock cycle. The implementation of a sequential comparator leads to reduced area coverage and higher operation speed of the specific circuit, posing no bottleneck on the entire digital system. The complete FPGA architecture is depicted in Fig. 12.

4. Results

The total clock cycle delay imposed by the system for the calculation of the minimum SAD value for four EI neighbors adds up to 113 clock cycles. This number breaks down to 100 clock cycles for propagating the data two times from the first to the last of the 50 PEs and the rest clock cycles for propagating the data through the remaining datapath (adder trees in the PEs, intermediate and final adder).

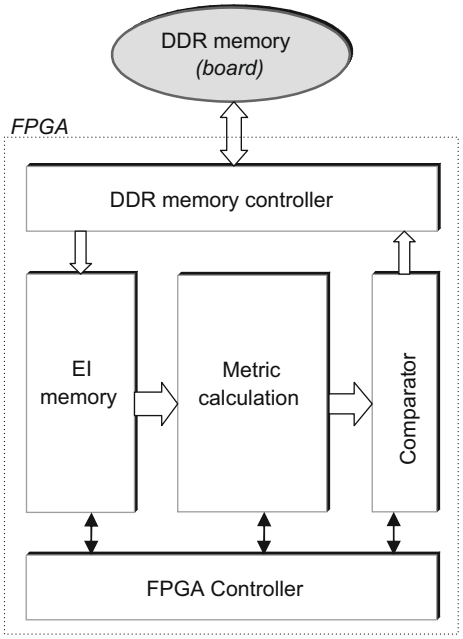

Fig. 12. The proposed FPGA architecture.

Hardware implementation results on the Virtex-5 LX110T FPGA reveal that the system can be clocked with a frequency of 170 MHz. A variety of different neighborhood area radii was tested in order to evaluate the performance of the proposed architecture.

Memory operations dominate over the total processing time in the specific platform implementation. Using 64-bit transfers from the DDRAM modules to the FPGA, 512 cc are needed for each EI to be moved to the hardware system memory. Processing time thus depends on the size of the neighborhood area N, which determines the number of EI transfers to the hardware system memory. For each N, a quadruplet of neighboring EIs must be transferred twice to the system memory, once for the central pixel calculation and one more time for the refinement calculation stage.

The integrated hardware performance is compared to the software process of vertex grid computation and grid refinement, when executed on a PC with an Intel Core2Duo 2 GHz CPU and 2048 MB of RAM, which is of the same technological era. The results are depicted in the diagram of Fig. 13.

As revealed in Fig. 13, the performance of the hardware is approximately seven times better than the Core2Duo software performance for every neighbourhood radius tested. As expected, the neighborhood radius greatly affects the processing performance, because more calculations are required for each EI. This aggravation diminishes as N increases, since as explained in Section 3.2 the number of central EIs decreases according to the expression $(32-2N)\times(32-2N)$. The hardware system achieves real-time processing rates for lower neighbourhood area radii, while it operates at a rate close to 15 fps even for larger values of N. This rate surpasses the estimated acquisition rate of a dynamic InIm acquisition system, and therefore poses no bottleneck to a robust integrated system.

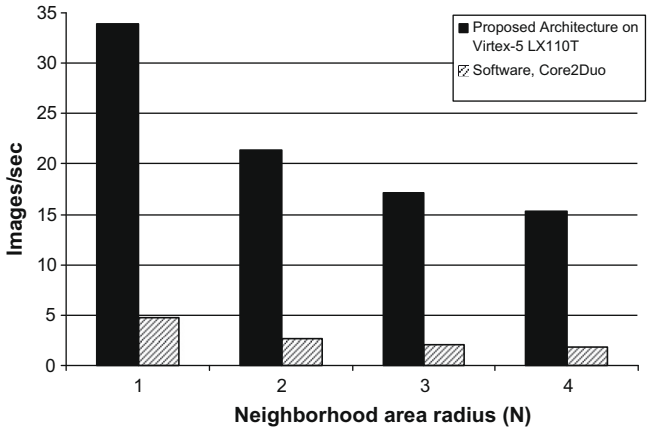

Fig. 13. Performance comparison of the proposed architecture and the software approach.

5. Conclusions

Fully 3D shape and texture reconstruction is a highly intensive processing task that needs to be efficiently addressed in order to move towards real-time 3D applications. In this paper, a hardware implementation for the acceleration of a software approach to 3D surface model reconstruction is presented. The proposed parallel digital system features maximized processing throughput and minimized memory read by data reutilization. The implemented PEs operate on two search areas simultaneously, thus reducing idle time and efficiently exploiting the available area on the reconfigurable device. Operating results reveal that the hardware architecture is able to process 3D data in a rate that surpasses a contemporary InIm system's throughput capability. Although developed for the specific 3D reconstruction algorithm proposed by Passalis et al. [10], the hardware system can address the acceleration of any 3D reconstruction technique which resolves the correspondence problem using distance metric calculations on neighboring images, such as multi-view image and video sequences. The proposed architecture can support future 3D applications that will require real-time representation of 3D objects using high-resolution InIms that have enhanced depth and detail characteristics.

References

[1] O. Schreer, P. Kauff, T. Sikora, 3D Video Communication: Algorithms, Concepts and Real-Time Systems in Human Centred Communication, Wiley, New York, 2005.
[2] J.-Y. Son, B. Javidi, Three-dimensional imaging methods based on multiview images, J. Display Technol. 1 (2005) 125–140.
[3] Y.-S. Ho, S-Y. Kim, E.-K. Lee, Three-dimensional video generation for realistic broadcasting services, in: Proceedings of the IEICE, 23rd International Technical Conference on Circuits/System Computation and Communications, 2008, TR1–TR4.
[4] G. Lippmann, La Photographie integrale, C.R. Acad. Sci. 146 (1908) 446–455.
[5] J. Park, Y. Kim, J. Kim, S. Min, B. Lee, Three-dimensional display scheme based on integral imaging with three-dimensional information processing, Opt. Exp. 12 (2004) 6020–6032.
[6] J. Park, S. Jung, H. Choi, Y. Kim, B. Lee, Depth extraction by use of a rectangular lens array and one-dimensional elemental image modification, OSA Appl. Opt. 43 (2004) 4882–4895.
[7] S. Kishk, B. Javidi, Improved resolution 3D object sensing and recognition using time multiplexed computational integral imaging, Opt. Exp. 11 (2003) 3528–3541.
[8] Y. Frauel, B. Javidi, Digital three-dimensional image correlation by use of computer-reconstructed integral imaging, Appl. Opt. 41 (2002) 5488–5496.
[9] D. Shin, E. Kim, B. Lee, Computational reconstruction of three-dimensional objects in integral imaging using lenslet array, Jpn. J. Appl. Phys. 44/11 (2005) 8016–8018.
[10] G. Passalis, N. Sgouros, S. Athineos, T. Theoharis, Enhanced reconstruction of 3D shape and texture from integral photography images, OSA Appl. Opt. 46 (2007) 5311–5320.
[11] A. Kolar, T. Graba, A. Pinna, O. Romain, B. Granado, T. Ea, An integrated digital architecture for the real-time reconstruction in a VsiP sensor, in: Proceedings of the 13th IEEE International Conference on Electronics, Circuits and Systems, 2006, pp. 144–147.
[12] J. Falcou, J. Serot, T. Chateau and F. Jurie, A parallel implementation of a 3D reconstruction algorithm for real-time vision, in: PARCO 2005, Parallel Computing, 13–16 September, Malaga, 2005 pp. 663–670.

[13] D. Chaikalis, N. Sgouros, D. Maroulis, P. Papageorgas, Hardware implementation of a disparity estimation scheme for real-time compression in 3D imaging applications, J. Vis. Commun. Image Rep. 19 (1) (2008) 1–11.
[14] D. Maroulis, N. Sgouros, D. Chaikalis, FPGA-based architecture for real-time IP video and image compression, in: IEEE International Symposium on Circuits and Systems, Island of Kos, Greece, 2006, pp. 5579–5582.
[15] S. Wong, B. Stougie, S. Cotofana, Alternatives in FPGA-based SAD implementations, in: IEEE I.C. on Field Programmable Technology (FPT'02), Hong Kong, 2002, pp. 449–452.
[16] Torres-Huitzil, C., Arias-Estrada, M., Real-time image processing with a compact FPGA-based systolic architecture, Real-Time Imaging 10 (2004) 177-187.
[17] D. Chaikalis, G. Passalis, N. Sgouros, D. Maroulis, T. Theoharis, Near real-time 3D reconstruction from InIm video stream, in: Springer Lecture Notes in Computer Science, vol. 5112, Image Analysis and Recognition, 2008, pp. 336–347.
[18] Xilinx Virtex-5 Datasheet. Available from: <http://www.xilinx.com>.