Near Real-Time 3D Reconstruction from InIm Video Stream

D. Chaikalis, G. Passalis, N. Sgouros, D. Maroulis, and T. Theoharis

Department of Informatics and Telecommunications, University of Athens, Ilisia 15784, Athens, Greece
{dhaik,passalis,nsg,dmaroulis,theotheo,rtsimage}@di.uoa.gr

Abstract. An efficient hardware architecture for the acceleration of an integrated 3D reconstruction method is presented, targeting demanding dynamic Integral Imaging applications. It exploits parallel processing and features minimized memory operations by implementing an extended-access memory scheme. Its reduced data throughput, thanks to optimized data utilization, makes it suitable for single FPGA device implementation. Results reveal that the hardware system outperforms the software method by an order of magnitude. Moreover, its processing rate surpasses the typical rate of a dynamic Integral Imaging acquisition system, thus making a significant step towards real-time 3D video reconstruction.

Keywords: Integral Imaging, object reconstruction, 3D video, architecture, hardware.

1 Introduction

Three-dimensional (3D) object extraction in real world scenes is known as one of the major challenges in the stereoscopic vision field. Typical applications include video tracking and machine vision systems as well as virtual reality environments. [1,2]. A number of systems have been developed that perform these tasks, most of them based on classic two-view stereoscopic cameras or multiple cameras arranged in an array, or systems that use a combination of range finding sensors and laser beams along with high resolution cameras to capture the texture of the objects [3]. However these systems are fairly complex and require accurate calibration procedures. Moreover most of them are bulky and have high initial costs which make them prohibitive for uses other than large television or cinema productions [3,4].

An alternative method that is characterized as the near ideal multiview technique [5] functions on the principle of Integral imaging (InIm) . InIm is based on Integral Photography which was initially proposed by Lipmann back in 1908 [6]. The operational principle of an InIm capturing setup is shown in Fig. 1. This technique uses a lens array (LA) over a high resolution Charge Coupled Detector (CCD). Each lens in the lens array produces a part of the InIm, which is called Elemental Image (EI). The resulting image can be used for stereoscopic viewing using an appropriate Liquid Crystal Display equipped with a LA. The capturing devices can be made compact enough, without moving parts and hence ensuring portability with no need for calibration. Moreover as

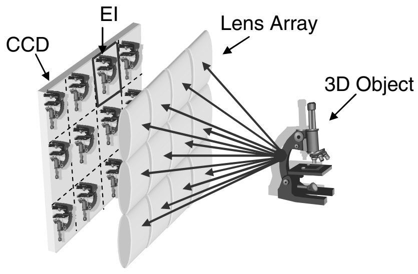

Fig. 1. A typical InIm capturing setup

sensor resolution increases and new materials can be used for optical components construction like LAs the technique can be made affordable for low-end applications.

1.1 Motivation

InIm has unique characteristics that can be used for several medical, educational and cutting edge applications, as it can provide a virtual environment with an enhanced perception of reality and allow real-time manipulation. High quality 3D object reconstruction from dynamic InIm can further benefit these applications as 3D information can be efficiently represented, stored and retrieved. However, in order to target demanding real-time 3D video applications, such an approach must be combined with a robust acceleration method, notably hardware-oriented optimized solutions. This is necessary as the volume of information that is produced by a practical dynamic InIm system cannot be processed in real-time by the current generation of CPUs.

1.2 Related Work

The problem of 3D object reconstruction using Integral Images (InIms) has been addressed by several researchers [7-10]. Most of the proposed InIm reconstruction algorithms are mainly applied to InIms of small objects that do not span many EIs because of their size. This limitation is enforced in order to avoid stitching problems during the reconstruction stage caused by abrupt discontinuities due to depth estimation errors [7]. Moreover EI modification techniques are proposed in an effort to increase depth accuracy [8]. Note that in several works (such as Shin [11]) the term *3D object reconstruction* is used to describe the generation of 2D images from multiple views and focus depths from a single Integral Image.

A method that is focused on the reconstruction of a fully 3D surface model is proposed in [12], where 3D shape and texture of real-life objects are reconstructed using the InIm technique. The method extends the classic stereo correspondence problem 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 elemental lenses 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 [13]. Other efforts turn to clustering for boosting performance [14]. They all target typical stereoscopic systems, and no attempt has been presented to accelerate a full 3D surface model reconstruction method.

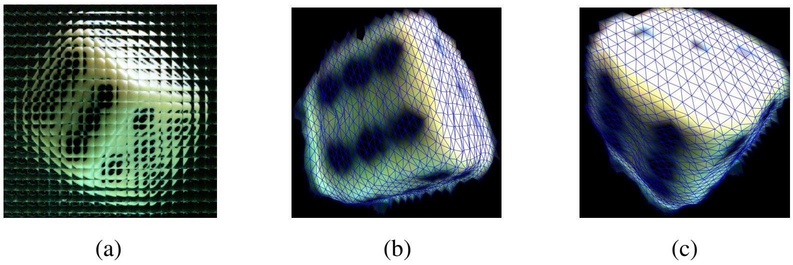

Fig. 2. Reconstruction of a dice: (a) Integral Image with f=3.3mm; (b, c) Reconstructed 3D object rendered with triangulation superimposed

1.3 Overview

In this paper, an architecture is proposed for efficiently enhancing the performance of a 3D reconstruction method, such as the one presented in [12]. The proposed hardware illustrates a significant speed improvement over the software method, by parallelizing time-consuming and repetitive processing tasks that form the inner loop of the reconstruction algorithm and favor hardware implementation. Moreover, it demonstrates optimized data utilization by applying specific memory data arrangements. Implementation results in an FPGA device reveal that the hardware system can process 3D image data in a rate greater than 1 *fps*, outperforming the software application by at least one order of magnitude.

The rest of the paper is divided into 4 sections. In section 2, the 3D reconstruction method [12] is outlined. In section 3, we describe the hardware system and present its main design considerations. The timing and implementation results are given in section 4, along with comparison against the software system. Finally, the results are discussed and future work is presented in section 5.

2 3D Reconstruction from InIm Algorithm

The method proposed in [12], estimates the 3D shape and texture of an object from a single Integral Image. To this end a two step process is applied: first, 3D points (vertices) on the surface of the object are computed and second, these points are connected in a polygonal (e.g. triangular) mesh. The reconstruction algorithm can be summarized as follows:

- *Vertex Grid Computation:* Vertices are computed using the central pixel of each lens, forming a rough regularly sampled vertex grid.
- *Grid Refinement and Triangulation*: The grid is subdivided, new vertices are computed and the refined grid is triangulated.
- *Post-Processing*: The final grid is filtered in order to improve reconstruction quality (e.g. noise reduction).

Given an Integral Image (produced by a lens array with known focal length f), we first compute 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. Note that all EIs have the same pixel resolution which is determined by the acquisition device.

We define the distance $D(p_1, p_2)$ between two pixels ($p_1 = [u_1\ v_1]^T$ and $p_2 = [u_2\ v_2]^T$) from different EIs using a simple but effective L_1 metric:

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

where E_1 and E_2 are the two EIs, and W defines the size of the comparison window.

We subsequently extend the above distance metric to more than two EIs. In practice, we use $2N+1$ neighboring EIs per direction, thus forming a symmetrical neighborhood area of radius N around each EI (see Fig. 5). The best correspondence has the minimum sum of the distances over all neighbors:

- For the central pixel $p_{k,l}$ of each EI $E_{k,l}$
- Find the 3D vertex P that minimizes the expression:

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

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

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. 3 we can project the central pixels from neighboring EIs (computed in the previous step) onto a specific EI. Additional vertices that we will use for refinement will be derived only from pixels between the central pixels of the lens and the correspondences of the central pixels of the neighboring lenses. The algorithm that subdivides the grid of a lens can be summarized as follows:

```
• Let the current lens be denoted by $L_{00}$, and its up, right and diagonal neighbors by
  $L_{01}$, $L_{10}$ and $L_{11}$ respectively. Let the 3D vertices computed using the central pixel
  of the above lenses (in the previous step) be denoted by $V_{00}$, $V_{10}$, $V_{01}$ and $V_{11}$.
    • Project $V_{00}$, $V_{10}$, $V_{01}$ and $V_{11}$ in $L_{00}$ as $p_{00}$, $p_{10}$, $p_{01}$ and $p_{11}$ respectively.
        • For $j$ from 1 to $S-1$ do
              For $i$ from 1 to $S-1$ do
                  $p' = (S-i)(S-j)/S^2 p_{00} + i(S-j)/S^2 p_{10} + (S-i)j/S^2 p_{01} + i \cdot j/S^2 p_{11}$
                  Compute the reconstructed vertex V' using the vertex grid computation algorithm and add it to the grid.
```

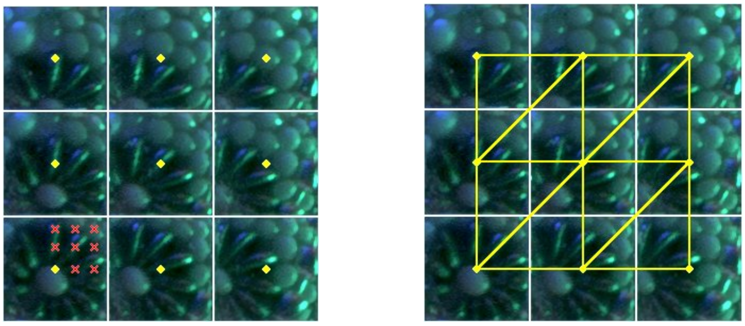

Fig. 3. EIs from a real-life object (f=3.3mm): (a) A 3x3 neighborhood with central pixels marked yellow. The correspondences of the central pixels in the bottom-left EI are marked red. (b) Triangulation of the same neighborhood using only central pixels, superimposed over the image.

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 detailed in the previous section, the pixel distance metric $D(p_1, p_2)$ constitutes the core of the algorithm. The calculation of the metric in software is time-consuming due to the complex nature of the absolute value calculation procedure and the subsequent multitude of additions. Moreover, the repetitive nature of the metric computations favors hardware implementation in order to improve performance. Furthermore, an optimized architecture can sufficiently eliminate the redundant memory accesses of the algorithm, imposed by the traversal of the comparison window area.

In the proposed hardware system that targets real-time InIm reconstruction, the distance metric is implemented using the Sum of Absolute Differences (SAD) metric. A detailed description of the SAD architecture can be found in [15, 16]. 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 EIs is achieved by using application-specific look-up tables and an optimized memory organization architecture.

3.1 Implementation Considerations

The implementation of a completely parallel SAD Unit for simultaneous pixel comparisons of the entire window area can offer the most significant speed increase. Such an implementation though poses the problem of high memory bandwidth demands, and multiple FPGAs should be used [17]. In the proposed implementation, a $(2W+1)$ SAD Unit is designed as the core processing element in order to target a single FPGA device. This unit can perform pixel comparisons and additions using 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 $(2W+1)x(2W+1)$ block comparison is available after $2W+1$ clock cycles. The outline of this SAD Unit is depicted in Fig. 4.

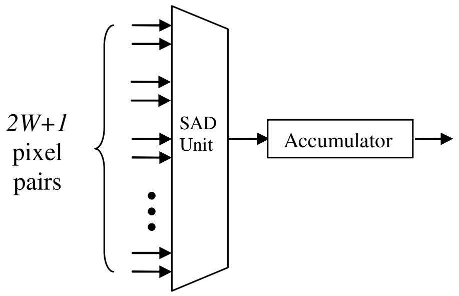

Fig. 4. The implemented SAD Unit outline

The structure of the SAD Unit can be exploited in an array of M units, 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 pixels is read from the search area. 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 array also removes the need for a parallel comparison unit, 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 can be compared in order to determine the smallest value using a sequential comparator. 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 $(2W+1)+M+1$.

The design of the FPGA memory takes into account the need for immediate access to arbitrary blocks in an EI, which is useful for the grid refinement stage. The 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, $2W+1$ memory modules are used for every EI. Each of these memory modules stores pixel lines of the EI, in intervals of $2W+1$ lines. For example, in the first memory module, lines $0,\ 2W+1,\ 2\cdot(2W+1),\ 3\cdot(2W+1)\ \ldots$ etc are stored. This way, $2W+1$ pixels of each line can be accessed in every clock cycle regardless of the pixel's coordinates (and hence block position) in the EI.

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 and the block size W.

3.2 Practical Dynamic Acquisition System

A practical dynamic InIm acquisition system was considered as a case study. Using the typical InIm configuration, the lens arrays is positioned between the camera's objective lens and the CCD. Assuming a contemporary CCD, a resolution of 16 megapixels is possible at approximately 1 *fps*. For a square CCD, this resolution corresponds to 4096x4096 pixels. This system is considered practical as it can provide dynamic EI with sufficient resolution using contemporary hardware. 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 3.3mm) the best setup would correspond to 64x64 pixels per lens, resulting in 64x64 lenses (and corresponding EIs). For 64x64pixels per EI, the optimal window is 11x11 ($W=5$). For this configuration, typical values for the remaining parameters are $N=3$ and $S=2$.

The algorithm is divided into 2 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 3 additional pixels for each EI (for $S=2$). For the utilized configuration, there are 64x64 EIs, but we use only the inner 58x58 (so that each central EI has 3 neighbors per direction). Thus, the maximum size of the search area is $M=30$ (this is correlated with the EI resolution), which also defines the size of the SAD Unit array. An outline of the search areas for neighboring EIs is depicted in Fig. 5.

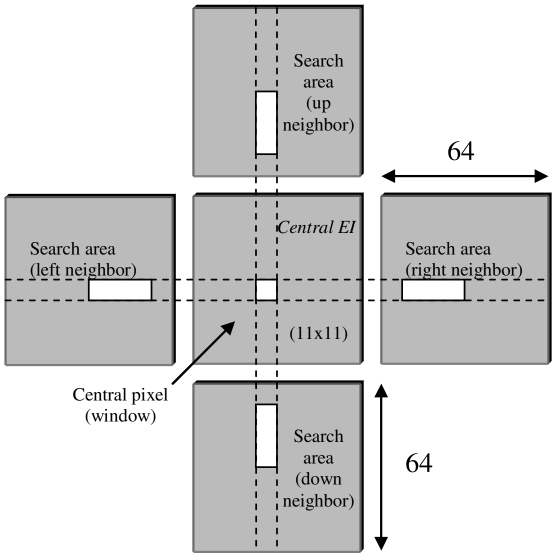
Search area (up neighbor)
64
Central EI
Search area (left neighbor)
Search area (right neighbor)
(11x11)
Central pixel (window)
Search area (down neighbor)
64

Fig. 5. Elemental Image (EI) size, block size and search area outline for the practical acquisition system. For depiction clarity, N=1.

The proposed architecture which is depicted in Fig. 6 implements a SAD Array with 30 11x1 SAD Units, each one having the ability to perform calculations on a 11x11 block of 8-bit pixel values in 11 clock cycles. Each unit's output is connected to an accumulator where the final summation takes place. Each unit begins its calculations one clock cycle after the previous one, so that the pixels of the central EI can be correctly delayed by propagation, until the last unit. In this fashion, the 30 SAD Units produce 30 values that correspond to the same number of comparisons of one central EI block with one of the four neighboring EIs. The calculated values are produced in a sequential manner, which justifies the single output data bus of the SAD array.

These values are stored as temporary results in a 30-cell memory, in order to be added to the values of the next three SAD calculations of the remaining neighbors. In every calculation cycle, the previously stored values are added to the new ones, and the result is stored. After the calculations for the last block, there is no need to store the outcome of the final additions. Instead, they are compared in order to determine the minimum value. It is this value that corresponds to the best match of the central EI block to its neighbors ones. Once the smallest value is determined, the positions of the pixels in the neighboring EIs are defined. These pixels are the best matches to the pixel of the central EI.

According to the memory scheme explained in subsection 3.1, eleven memory modules are designed for every EI. Each of these memory modules stores pixel lines (rows or columns, depending on the position) of the EI, in intervals of eleven lines. For example, in the first memory module, lines 0, 11, 22, 33 etc are stored.

Moreover, the horizontal neighbors are stored row-wise and the vertical neighbors column-wise. The central EI is stored in two different ways, both row- and column-wise. A schematic representation of the position of pixel rows or columns in each memory module is illustrated in Fig. 7.

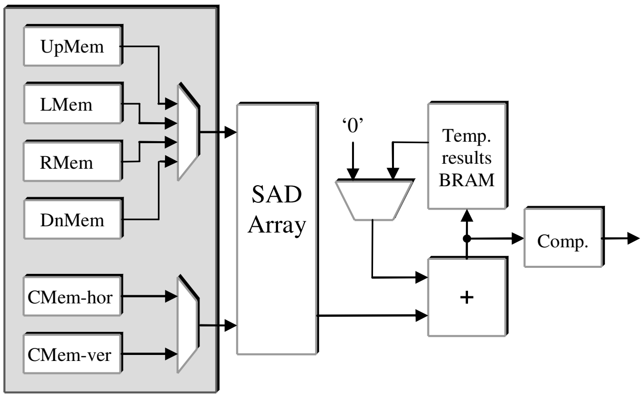

Fig. 6. Architecture of the hardware system

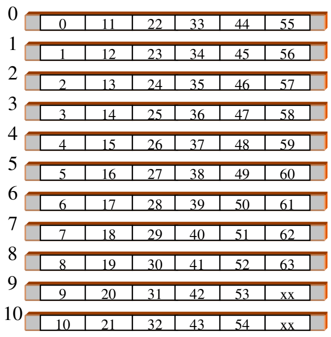

Fig. 7. EI line positioning in each FPGA memory module

In total, 66 memory modules of this type are needed, namely 22 for the two central EI instances and 11 for each of the four neighboring EI. The number of memory cells for each memory module is proportional to the EI size – for a 64x64 EI, 378 memory cells are needed. For the sake of clarity, the 11 memory modules for each EI are grouped in a single memory block in Fig. 6.

4 Results

The initial grid computation requires each EI and its neighbors up to N to be loaded to the input FPGA memory modules. The calculation for the 3D position of the central pixel of each EI takes place in three stages, one for each quadruplet of neighbors, and costs 175 clock cycles (cc) for N=1. The memory transfer cost is 256 cc for each 4 EIs. For N=1, the central EI and 4 neighbors are transferred, so the cost for this transfer is 512cc. For successive neighborhood radii, only the 4 neighbors are swapped, and the cost is 256 cc for each of these stages. Respectively, for the grid refinement, three more pixels must be determined for each EI. For the utilized configuration, there are 64x64 EIs, but we use only the inner $(64-2N)\times(64-2N)$ (so that each central EI has N neighbors per direction). The cost for processing the whole 4096x4096 image is dependent on the neighborhood radius N .

Implementation of the proposed hardware system on a Celoxica RC1000 development board fitted with a Xilinx Virtex-E 2000 device [18] determine that the maximum operating frequency is 43Mhz. The hardware performance is compared to the software process of vertex grid computation and grid refinement, when executed on a PC with a Pentium 2.4GHz CPU and 512MB of RAM. The specification of the PC and FPGA are of the same technological era.

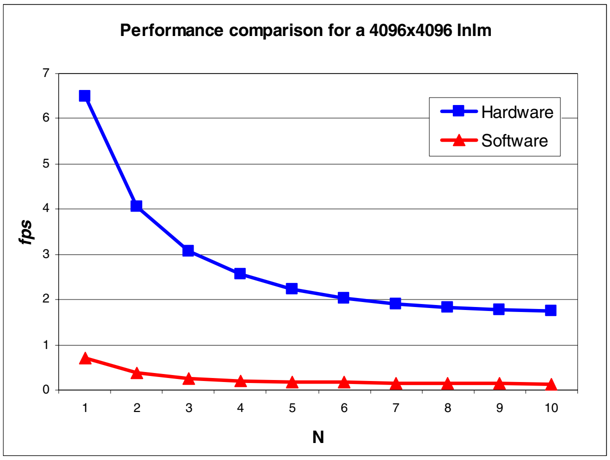

Fig. 8. Hardware/software performance comparison for an 4096x4096 InIm for varying N

As revealed in Fig. 8, the performance of the hardware is approximately one order of magnitude better than the software algorithm. For example, for N=3, more than 3 4096x4096 images can be processed by the FPGA per second, while in software no more than 27% of an image of this size is processed. Moreover, the hardware system operates at a rate greater than 1 *fps* even for large 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.

5 Conclusions

This paper presents an efficient hardware architecture for an integrated 3D reconstruction system based on InIm. The architecture features extensive pipelining and minimizes data reutilization by incorporating a specific data memory arrangement. Moreover its reduced data throughput leads to the successful implementation of the digital system in a single FPGA device. The results reveal that the implemented hardware system can successfully process InIm data of significant resolution at a rate of 6 *fps,* outperforming the processing rate of a typical InIm acquisition system. The acceleration compared to the corresponding software implementation is more than one order of magnitude. The performance increase offered by the architecture contributes significantly towards real-time 3D video reconstruction.

Future work involves the development of other time-consuming tasks of the 3D reconstruction process in hardware. The migration of the hardware system to a larger FPGA device will be considered, which will allow us to implement more processing elements and explore architecture enhancements that can offer even greater performance gain.

Acknowledgements

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, and 75% by the European Social Fund.

References

1. Park, J.-I., Inoue, S.: Acquisition of Sharp Depth Map from Multiple Cameras. Signal Processing: Image Communication 14, 7–19 (1998)
2. Ko, J.-H., Hwang, D.-C., Jung, Y.-W., Kim, E.-S.: Intelligent Mobile Robot System for Path Planning Using Stereo Camera-Based Geometry Information. In: Proceedings of SPIE, vol. 6006, pp. 232–243 (2005)
3. Javidi, B., Okano, F. (eds.): Three-Dimensional Television. Video, and Display Technologies. Springer, Berlin (2002)
4. Kawakita, M., Iizuka, K., Aida, T., Kikuchi, H., Fujikake, H., Yonai, J., Takizawa, K.: Axi-Vision Camera (Real-Time Distance-Mapping Camera). Appl. Opt. 39, 3931–3939 (2000)
5. Son, J.-Y., Javidi, B.: Three-Dimensional Imaging Methods Based on Multiview Images. J. Display Technol. 1, 125–140 (2005)
6. Lippman, G.: La Photographie Integrale. C. R. Acad. Sci. 146, 446–451 (1908)
7. Park, J., Kim, Y., Kim, J., Min, S., Lee, B.: Three-Dimensional Display Scheme Based on Integral Imaging with Three-Dimensional Information Processing. Optics Express 12, 6020–6032 (2004)
8. Park, J., Jung, S., Choi, H., Kim, Y., Lee, B.: Depth Extraction by Use of a Rectangular Lens Array and One-Dimensional Elemental Image Modification. OSA Applied Optics 43, 4882–4895 (2004)
9. Kishk, S., Javidi, B.: Improved Resolution 3D Object Sensing and Recognition using Time Multiplexed Computational Integral Imaging. Opt. Express 11, 3528–3541 (2003)
10. Frauel, Y., Javidi, B.: Digital Three-Dimensional Image Correlation by Use of Computer-Reconstructed Integral Imaging. Appl. Opt. 41, 5488–5496 (2002)
11. Shin, D., Kim, E., Lee, B.: Computational Reconstruction of Three-Dimensional Objects in Integral Imaging Using Lenslet Array. Japanese Journal of Applied Physics 44(11), 8016–8018 (2005)
12. Passalis, G., Sgouros, N., Athineos, S., Theoharis, T.: Enhanced reconstruction of 3D shape and texture from integral photography images. OSA Applied Optics 46, 5311–5320 (2007)

13. Kolar, A., Graba, T., Pinna, A., Romain, O., Granado, B., Ea, T.: An Integrated Digital Architecture for the Real-Time Reconstruction in a VsiP Sensor. In: 13th IEEE International Conference on Electronics, Circuits and Systems, pp. 144–147 (2006)
14. Falcou, J., Sérot, J., Chateau, T., Jurie, F.: A Parallel Implementation of a 3D Reconstruction Algorithm for Real-Time Vision. In: PARCO 2005, Parallel Computing, 13 - 16 September, Malaga (2005)
15. Chaikalis, D., Sgouros, N., Maroulis, D., Papageorgas, P.: Hardware Implementation of a Disparity Estimation Scheme for Real-Time Compression in 3D Imaging Applications. Journal of Visual Communication and Image Representation 19(1), 1–11 (2008)
16. Maroulis, D., Sgouros, N., Chaikalis, D.: FPGA-based Architecture for Real-Time IP Video and Image Compression. In: IEEE International Symposium on Circuits and Systems, Island of Kos, Greece, May 21-24, pp. 5579–5582 (2006)
17. Wong, S., Stougie, B., Cotofana, S.: Alternatives in FPGA-based SAD Implementations. In: IEEE I.C. on Field Programmable Technology (FPT 2002), Hong Kong (2002)
18. Celoxica RC1000-PP development board: Hardware Reference, `http://www.celoxica.com`