Design and Implementation of Image Analysis and Processing Algorithms in Field-Programmable Gate Arrays for Real-Time Operation

Dionisios P. Chaikalis*

National and Kapodistrian University of Athens
Department of Informatics and Telecommunications
dhaik@di.uoa.gr

Abstract. The need for increased realism in several medical, communication and entertainment applications continuously promotes the importance of three-dimensional display systems. However, the volume of information that a three-dimensional display system must cope with prohibits real-time performance for many applications like three-dimensional video. In this dissertation, the acceleration of time-consuming algorithms that are used for the compression, reconstruction and display of Integral Photography images is addressed. The acceleration is realized by proposing novel digital architectures that confront the problem of processing large volumes of data. A number of optimization techniques like data reutilization and implementation of a large number of processing elements used for parallel operation are adopted in this context. The performance increase that is obtained compared to the software approaches renders them capable or real-time performance. A study for the acceleration of a rendering method for three-dimensional data based on Ray Tracing concludes this dissertation.

Keywords: Three-dimensional display, Autostereoscopy, Integral Imaging, Field-Programmable Gate Arrays, Architecture.

1 Introduction

The rapid increase in processing power and graphic card acceleration, combined with improvements in high fidelity optical systems, over the past few years, revived the interest for three-dimensional (3D) applications. Many promising technologies evolved, ranging from the classic stereoscopic ones, like polarizing and eye shuttering glasses [1], to most sophisticated techniques like autostereoscopic displays [2]. Autostereoscopic display devices provide 3D stereoscopic view without the need of additional glasses, as all optical components are integrated in the display, reducing eye fatigue. A method that is characterized as the near ideal multiview technique [3] functions on the principle of Integral imaging (InIm). InIm is based on Integral Photography which was initially proposed by Lipmann back in 1908 [4]. A simple InIm capturing setup is built using a CCD sensor and a lens array as shown in Fig 1. The object is projected through the lens array on the CCD surface forming a number

* Dissertation advisor: Dimitris Maroulis, Associate Professor

of different projections equal to the number of the lenses in the lens array. These projections are usually called Elemental Images (EIs). An InIm display setup uses a high resolution LCD display in conjunction with an appropriate lens array to produce high quality full parallax stereoscopic images.

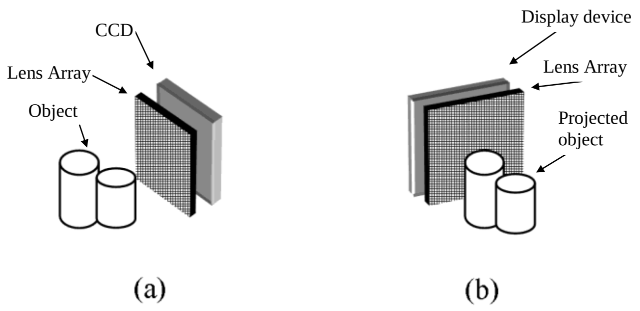


Fig. 1. A typical InIm (a) capturing and (b) display setup.

InIm can provide a virtual environment with an enhanced degree of reality and 3D perception which can benefit several medical, educational, entertainment and cutting edge applications [5,6]. However, such applications must cope with high resolution InIms which lead to high bandwidth requirements for the storage and reproduction of 3D objects and scenes. 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, the computationally intensive tasks must be addressed with a robust acceleration method, notably hardware-oriented optimized solutions.

The main aim of the current work is to propose hardware architectures for the acceleration of time-consuming tasks in 3D imaging, such as compression and reconstruction. These architectures can contribute to the development of an integrated system for autostereoscopic data manipulation, such as the one presented in Fig. 2 ranging from acquisition to representation, targeted to real-time applications in the 3D imaging field. The algorithms selected for acceleration achieve high quality results in coding and reconstruction of 3D data, rendering the increase of their performance vital for demanding real-time systems. The acceleration methods address the most time-consuming tasks of the algorithmic approaches, and achieve high processing rates and efficient data use thanks to specific design choices.

The remaining paper is organized in four sections. In section two, the proposed InIm compression architecture is outlined and its main performance results are presented. Section three discusses the proposed InIm reconstruction architecture and summarizes its performance compared to a software approach. A combined architecture merging compression and reconstruction of 3D data is presented in section four, along with its performance according to system requirements. Section five outlines the main considerations in virtual InIm systems when they are combined with Ray Tracing rendering machines. Finally, the conclusions and future work issues and enhancements are presented in the last section.

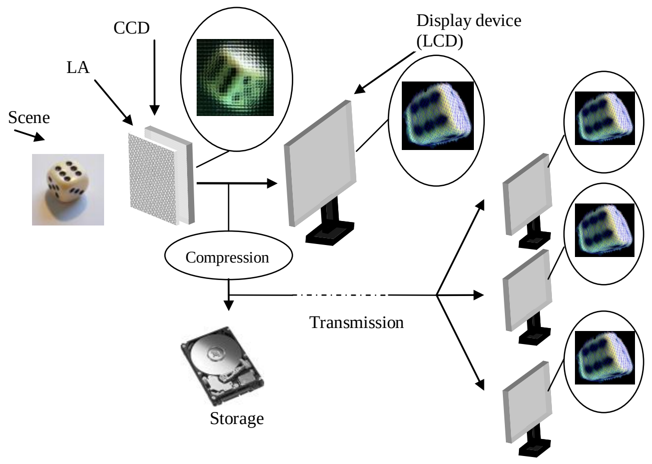


Fig. 2. Schematic representation of an integrated InIm system which combines data compression and reconstruction.

2 InIm Compression Architecture

The inherent properties of an InIm image gave rise to new techniques for the compression of highly correlated data, in order to provide efficient case-oriented algorithms that achieve high compression ratios over traditional methodologies. In the current dissertation, a novel hardware implementation for the acceleration of an efficient InIm compression algorithm is presented, targeted to real-time 3D capturing and display applications. The algorithm selected for acceleration [7] adopts the original MPEG coding scheme in a certain extent, with major changes in the motion vector estimation unit. Two of the most significant features of this alternative that derive from the properties of the InIm are the a priori knowledge of the search window size and the motion vector directionality. These properties further simplify the process, favoring real-time applications, where high compression ratios must be combined with real-time performance of the whole process.

The digital architecture exploits the main algorithmic features in order to increase processing performance. By adopting extensive pipelining schemes and using a large number of processing elements, high level of parallelism is achieved. Moreover, memory access is minimized by properly arranging the image data to the memory modules and organizing the processing elements to a specific topology.

Fig. 3 presents a block diagram of the implemented encoder. The Disparity Estimation Unit (DEU) uses the image data from the Block RAM to generate the disparity vectors. The Difference Generation Unit (DGU) calculates the differences between the blocks in P-type EIs and the estimated blocks in the I-type EIs, using the results from the DEU. The Compression Unit (CU) generates the final compressed blocks that will be used at the reconstruction stage. It is also responsible of compressing and decompressing the I-type EIs, used in the disparity estimation procedure. The CU consists of three two-dimensional Discrete Cosine Transform

(2D-DCT) modules, one 2D Inverse-DCT (2D-IDCT) module and the appropriate Quantization and Inverse Quantization modules. The Address Generation Unit (AGU) and the control unit are necessary for synchronizing the data transfers; the control unit exchanges control signals with every other unit in the FPGA and also with the board components and the host PC, while the AGU provides the address words to the FPGA and board memory modules.

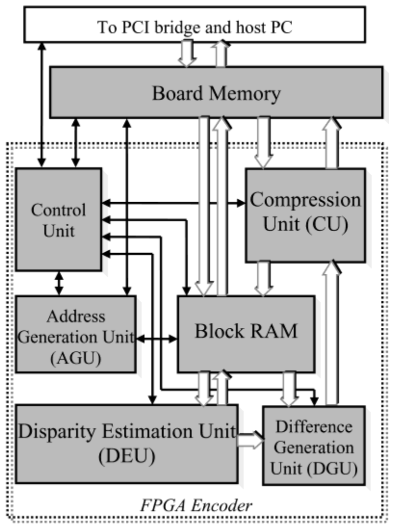

To PCI bridge and host PC
Board Memory
Control Unit
Compression Unit (CU)
Address Generation Unit (AGU)
Block RAM
Disparity Estimation Unit (DEU)
Difference Generation Unit (DGU)
FPGA Encoder

Fig. 3. The main units that comprise the FPGA encoder.

A distance metric is implemented in order to calculate the disparity vectors. The Sum of Absolute Differences (SAD) metric is selected since it is shown to be efficiently implemented in hardware [8], and its accuracy is sufficient for the unidirectional exhaustive search of the technique. The SAD calculation adds up the absolute differences between corresponding elements in the predetermined macroblocks. For an 8×8 pixel block, the SAD value is calculated by the formula:

$$SAD(U,V) = \sum_{i=0}^{m} \sum_{j=0}^{n} \left| U(i,j) - V(i,j) \right| \tag{1}$$

where i, j are spatial coordinates in the pixel domain and U,V represent $m \times n$ pixel blocks in adjacent image blocks. The actual coordinates of these blocks in the corresponding macroblocks are determined by the search algorithm.

As shown in Fig. 4, the encoder achieves real-time performance in all standard cases while real–time performance is also maintained for high resolution InIms. The high throughput of the encoder for low resolution InIms makes it suitable for processing a large number of Integral video streams in real-time, while rates above the threshold of 30 InIm images per second of operation for high resolution InIms show that it is a robust solution in cases where quality can not be compromised to achieve real-time performance.

For further evaluation of the performance of the encoder, a comparison with the homologous software approach is also presented. The results reveal that software processing requires several seconds, even for images at the low end of the tested dimensions. Also, compared to the software approach, the implemented encoder has demonstrated an acceleration factor lying in the range of 300 to 370, as reveled in . Fig. 5.

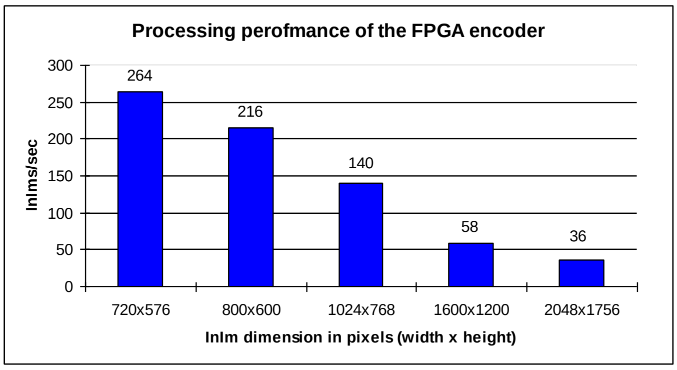


Fig. 4. The processing performance (in InIm images/sec) of the encoder measured as a function of the image resolution in pixels.

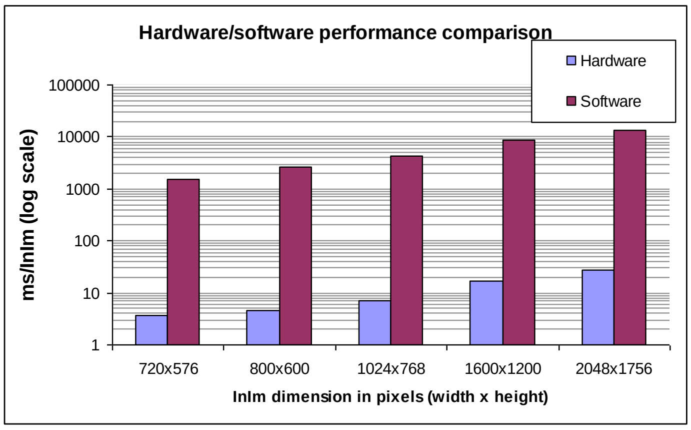


Fig. 5. FPGA encoder versus software performance comparison measured in processing time per InIm for several image dimensions.

3 InIm Reconstruction Arcthitecture

Three-dimensional object reconstruction or tracking algorithms use estimations of projections of the object in a large 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 to accelerate 3D reconstruction applications utilize dedicated platforms, most notably FPGA devices for the implementation of the digital architecture [9] while other researchers use clustering for boosting performance [10]. 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 dissertation, a robust, parallel digital architecture for 3D object reconstruction acceleration is presented. The architecture is based on an algorithm that is focused on the reconstruction of a fully 3D surface model [11], where 3D shape and texture of real-life objects are reconstructed using the InIm technique. The output of the process is the 3D polygonal representation of the object's shape and texture, as shown in Fig. 6. By efficiently exploiting the properties of the reconstruction algorithm, the implemented architecture demonstrates extensive processing capability. The Processing Elements (PE) operate simultaneously on multiple image areas and memory reads are minimized.

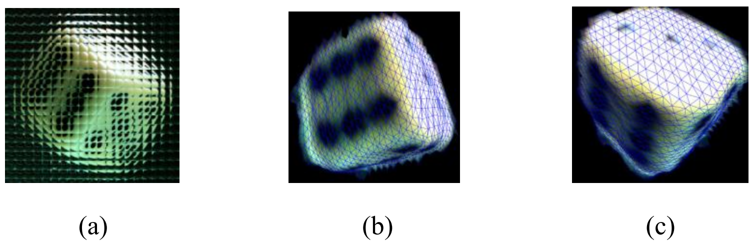


Fig. 6: Reconstruction of a dice: (a) Integral Image with f=3.3mm; (b, c) Reconstructed 3D object rendered with triangulation superimposed.

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 in order to extract depth information for a given pixel and place it on the appropriate position in the grid. 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:

$$D(p_1, p_2) = \sum_{j=-W}^{W} \sum_{i=-W}^{W} \left| E_1(u_1 + i, v_1 + j) - E_2(u_2 + i, v_2 + j) \right| \qquad (2)$$

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 $z = 2W + 1$. This metric is extended to more than two EIs. In practice, $2N + 1$ neighboring EIs are used per direction, thus forming a symmetrical neighborhood area of radius N around each EI. The best correspondence has the minimum sum of the distances over all neighbors.

The implemented 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, and in the next step more positions are computed, based on the refinement requirements

The outline of the proposed metric calculation architecture is depicted in Fig. 7. The three main assets of this architecture compared to the baseline approach presented in [12] are the doubling of speed that it achieves, the flexibility that it offers for the position of the block in the central EI and the more efficient use of the PEs.

As revealed in Fig. 8, the performance of the hardware is approximately 7 times better when compared to the software performance measured on a Core2Duo-based desktop PC, for every neighbourhood radius tested. 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.

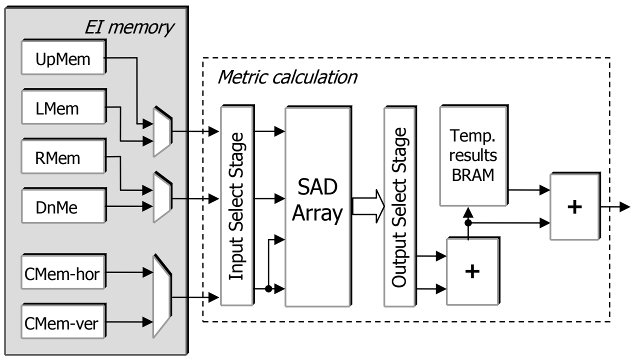


Fig. 7: The outline of the metric calculation architecture.

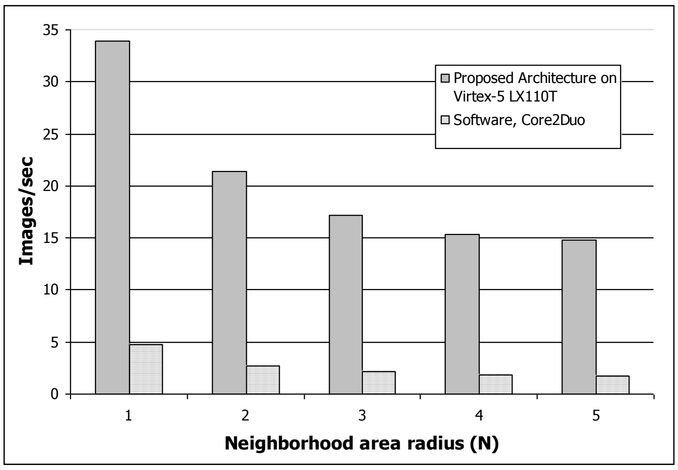


Fig. 8: Performance comparison of the proposed architecture and the software approach.

4 Common InIm processing datapath

As mentioned in the previous sections, the pixel distance metric constitutes the computational core of both the compression and reconstruction algorithms. In this section, a combined hardware processor that implements the distance metric is proposed in a way that either process can be executed according to the momentary need of the 3D system.

Based on the requirements of the two specific processes, the combination of the reconstruction and compression architectures to a unified digital system is presented in Fig. 9. The system is comprised of 57 SAD Units in order to carry out all the needed calculations for both processes. The first 50 Units are also used by the reconstruction process. The input and output selection stages apply only to 3D reconstruction, where the Units must operate on two search areas simultaneously, and the results to be properly forwarded to the adder inputs. The reconstruction process also uses a dedicated reconstruction module, where the intermediate and final additions take place, and the intermediate results are temporarily stored. The comparison stage is common to both processes, and contains 7 separate sequential comparators, in order to accommodate all the required comparisons of the compression process. One of these comparators is also used for the reconstruction results, in order to determine the best match for the central EI pixel window.

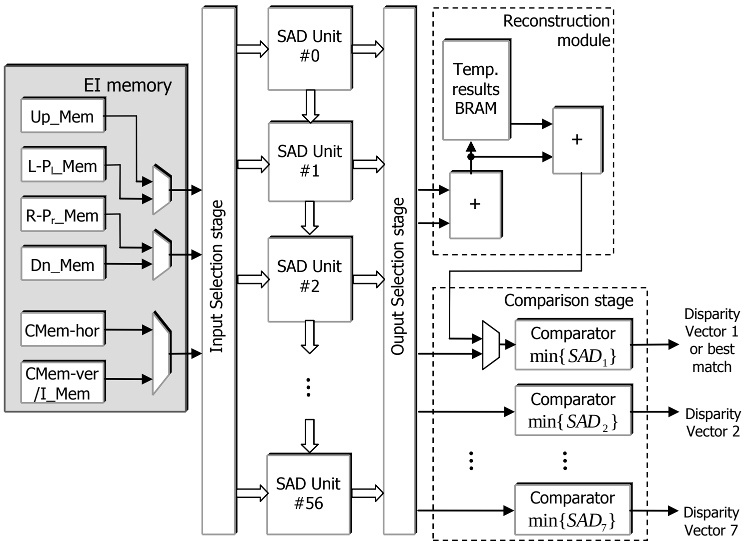


Fig. 9. The main modules of the proposed architecture.

The implemented SAD Units can process an 11×11 pixel window which is needed for the reconstruction calculations. Since the compression process operates on 8×8 block size, the block is zero-stuffed with three additional rows and columns, in order to reach the 11×11 size needed for the SAD Unit operations. The zero stuffing alleviates the need for different processing elements and doesn't impose any delay on the processing time.

The memory design takes into account the need for immediate access to arbitrary blocks in an EI, which is useful for the reconstruction calculations. 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. The data arrangement favors fast calculations of the sums regardless of the direction of the search area.

As depicted in Fig. 10, more than 30 InIms can be processed every second by the proposed architecture for either process. In the reconstruction process, the total InIm processing rate reaches up to over 65 InIms/sec for the smallest value of N, and remains over 30 even for larger N values that signify the increase in reconstruction quality. The compression achieves rates above 80 InIms/sec even for the smallest group of EIs ($N = 1$). It should be noted that the system achieves real-time performance even when both processes are applied to the InIm data sequentially. This is extremely useful since each InIm can be compressed and stored right after acquisition once, and then be available for 3D reconstruction or transmission as many times as needed.

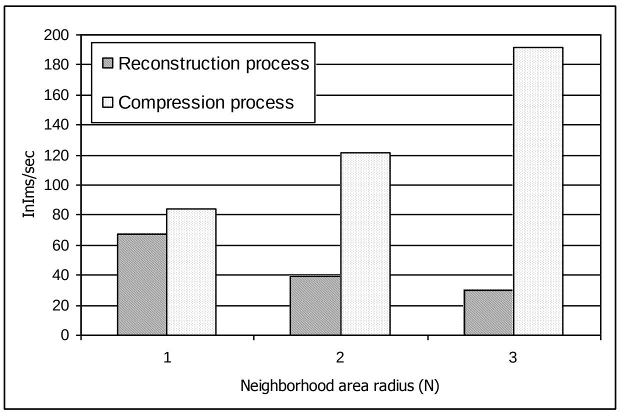


Fig. 10. Processing performance of the digital system for the reconstruction and compression processes.

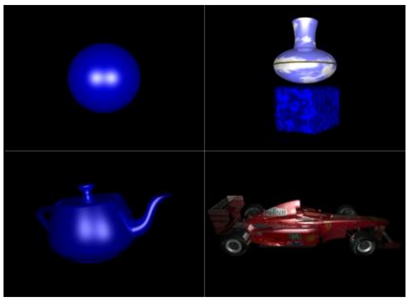
5 Performance considerations for InIm data rendering in a Ray Tracing environment

The use of Ray-Tracing (RT) engines [8] for computer generated InIms [13-15] is a highly promising technique, as RT engines provide scene renderings characterized by increased realism. All capturing optics can be modeled within the ray tracer as ordinary scene objects further simplifying the architecture of the virtual InIm capturing setup [15,16]. However, the high complexity introduced by modeling the acquisition optics in the RT prohibits their use in real-time applications. The lens array introduces a specific type of complexity in an RT scene due to the repeatability of the lens objects. This fact is not taken into account by generic hardware acceleration methods for RT tasks [17,18].

In the dissertation a detailed evaluation procedure for the additional overhead that is introduced by including a lens array containing a large number of lens objects in the scene is described. As the number of lenses that contribute to the InIm generation greatly affects performance, and there is a different behaviour if the lens is hit by light rays emanating from scene objects or not, a sliding window cross correlation technique is introduced to provide an accurate estimation of the active lenses of the lens array. As the number of lenses hit by light rays can be calculated derive a relation between the number of lenses and rendering time is finally derived. Using a number of different scene types and respective complexities, the rendering performance is calculated.

A virtual InIm camera is assembled by constructing a lens array from individual lenses using Constructive Solid Geometry (CSG) principles. The simulation of the capturing optics has been realized by modeling the lens array as an ordinary object of the Three-Dimensional (3D) scene [14] using the ray tracer's Scene Description Language (SDL). This approach takes advantage of the optimized algorithms implemented in POV-Ray in order to produce high quality photorealistic InIms.

Two types of POV-Ray scenes are constructed, containing objects described using CSG or triangle primitives, in order to evaluate the effect of the lens array in different scene types. Figure 11(a) presents four different objects and Fig. 11(b) illustrates the corresponding objects when the lens array is inserted in the scene. More images are created by scaling the object behind the lens array, thus varying the number of EIs that depict part of an object in the scene. In order to evaluate which of the lenses project part of the object or correspond to noise, metioned as Number of Active Lenses (NAL), the cross correlation is calculated between rectangular image parts of adjacent elemental images. The degree of correlation between an elemental image and its immediate neighbors determine if its corresponding lens is active or not.


(a)

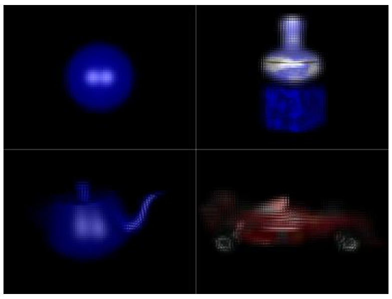

(b)

Fig. 11. The four reference images rendered using POV-Ray. From top left to bottom right: the sphere, vase, teapot and car objects (a) without lens array and (b) with lens array.

In order to evaluate the complexity introduced by the existence of a lens array in a scene, we measure t_l and t_{nl} which are the rendering times with and without the lens array respectively. Next we calculate the rendering time ratio $a = t_l / t_{nl}$ for each of the generated scenes. The results versus the NAL values are plotted in Fig. 12. As shown in the figure, the scenes that are solely assembled of CSG objects are affected in a greater degree by the number of active lenses ($t_l > 5 \cdot t_{nl}$) in regard to the scenes generated using triangle primitives ($t_l > 2.8 \cdot t_{nl}$). In addition a general increase in the rendering time ratio occurs in both scene types as the NAL value increases, since more lenses are utilized in order to generate an InIm.

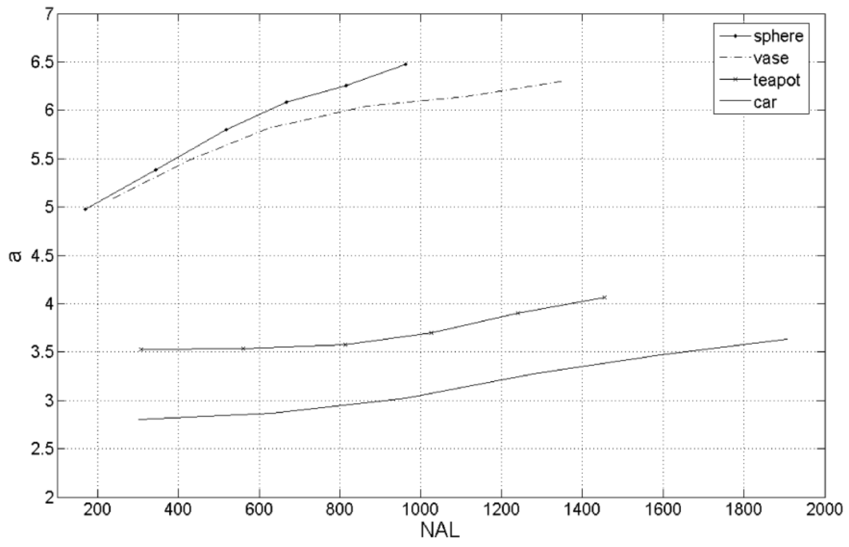


Fig. 12. The rendering time ratio for the 4 POV-Ray InIm scenes.

6 Conclusions and Future Work

In this paper a series of methods and hardware architectures are proposed, which contribute to the most crucial issue towards real-time 3D display systems – the time-consuming processing due to the volume of information that these systems manipulate. The proposed approaches exploit the benefits of hardware implementations in creating extensively parallel and high throughput architectures. In the same time they take advantage of the inherent redundancy in InIms in order to optimize data use and PE design. The significant results presented in the current work denote that 3D display methods together with optimized hardware solutions is a robust combination that can help real-time 3D imaging in its way of offering a practical, realistic solution in a number of educational, medical, entertainment and communication applications. Future work includes implementation of supplementary modules for 3D image processing and enhancing the performance of rendering machines through specific optimization based on benchmarking results during 3D data rendering.

References

1. Developers Handbook, Stereographics, 1997, www.stereographics.com.
2. M. Halle, "Autostereoscopic displays and computer graphics", Computer Graphics, 31(2), ACM SIGGRAPH, pp. 58-62, 1997.
3. J. -Y. Son B. Javidi, "Three-Dimensional Imaging Methods Based on Multiview Images," J. Display Technol. 1, pp. 125-140, 2005.
4. G. Lippman, "La Photographie Integrale," C. R. Acad. Sci., 146, pp. 446-451, 1908.
5. H. Liao, S. Nakajima et. al., "Intra-operative Real-Time 3-D Information Display System Based on Integral Videography", MICCAI'01, LNCS 2208, pp. 392-400, 2001.
6. P. Harman, "Home based 3D entertainment-an overview", Proc. ICIP(1), pp. 1-4, 2000.
7. Sgouros N., Andreou A., Sangriotis M., Papageorgas P., Maroulis D., Theofanous N.: Compression of IP Images for Autostereoscopic 3D Imaging Applications. In: 3rd International Symposium on Image and Signal Processing and Analysis (ISPA), Rome, Italy, September 18-20, 2003
8. 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 (ISCAS), May 21-24, Island of Kos, Greece, 2006
9. 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. 13th IEEE International Conference on Electronics, Circuits and Systems, pp 144-147, 2006
10. J. Falcou, J. Sérot, T. Chateau and F. Jurie: A Parallel Implementation of a 3D Reconstruction Algorithm for Real-Time Vision. PARCO 2005, Parallel Computing, 13 - 16 September Malaga, 2005
11. G. Passalis, N. Sgouros, S. Athineos, and T. Theoharis: Enhanced reconstruction of 3D shape and texture from integral photography images. OSA Applied Optics, 46:5311–5320, 2007
12. 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, pp. 336-347, 2008
13. POV-Ray: http://www,povray.org
14. S. Athineos, N. Sgouros, P. Papageorgas, D. Maroulis, M. Sangriotis, N. Theofanous: Photorealistic Integral Photography Using a Ray Traced Model of the Capturing Optics. Journal of Electronic Imaging, 15(4), pp. 43007-43014, 2006.
15. G. Milnthorpe, M. McCormick, N. Davies: Computer Modeling of Lens Arrays for Integral Image Rendering. IEEE Computer Society, Proc. of EGUK'02, pp 136-141, 2002
16. R. Olsson, Y. Xu: An Interactive Ray-Tracing Based Simulation Environment for Generating Integral Imaging Video Sequences. Proc. SPIE, vol. 6016, pp150-157, 2005
17. IngoWald: Realtime Ray Tracing and Interactive Global Illumination, PhD thesis, Computer Graphics Group, Saarland University, 2004
18. N. A. Carr, J. Hoberock, K. Crane, J. C. Hart: Fast GPU Ray Tracing of Dynamic Meshes using Geometry Images, Proceedings of the 2006 conference on Graphics Interface, pp. 203-209, 2006.