A Service-oriented FPGA-based 3D Model Acquisition System

Octavian Mihai MACHIDON[1], George OLARU[2]
[1]*Transilvania University of Brasov, 500036, Romania*
[2]*Tremend Software Consulting, Bucharest, 011253, Romania*
octavian.machidon@unitbv.ro, golaru@tremend.ro

***Abstract*—This paper proposes a non-contact, low cost 3D scanning solution using laser striping. The solution is composed of two main parts: the hardware setup - used for acquiring the object's 3D surface information, and the software part - that processes the information and obtains the 3D model representation of the object. We propose two major improvements over the traditional scanning solutions: the 3D information acquisition is based on a reconfigurable hardware platform – a Xilinx Spartan 6 FPGA – which adds flexibility and scalability to the scanning process, while the 3D model reconstruction is remotely available "as a Service", by the means of a web interface that abstracts away the complexity of the underlying processes and improves the performance, while granting easy sharing between users. By separating data capture process from the 3D model reconstruction tasks the system gains in portability - a feature that is absent for most existing solutions. The service-oriented approach brings on a performance gain, since the computational intensive tasks are handled by dedicated servers and ease of use of the system, because the user does not have to bother managing and using the software tools locally.**

***Index Terms*—Computer vision, Reconfigurable architectures, Virtual prototyping, Virtual reality, Web services.**

I. Introduction

A 3D scanner is a device that, by analyzing an object or a scene, collects 3D information about it. This is accomplished by creating a point cloud of geometric samples on the surface, which is further processed and used for reconstructing the shape of the particular object. The finality of the 3D scanning process is thus obtaining a 3D digital model to be used in a variety of applications [1]. There are a growing number of fields where 3D scanning is applied today: entertainment and consumer applications, historical preservation, medical imaging and surgical planning, robotics and even terrestrial scanners for acquiring 3D data of complex outdoor objects [2], etc.

Several technologies for building 3D scanners are available, each having its own particularities, strengths and weaknesses [2]. Such scanners have been the subject of significant prior work: in [4], the authors have developed a simple 2-camera system but with a manually held and moved laser pointer; [5] presents a cost-effective solution (using a camera, six lights and a projector) yet with no portability support. Another low-cost 3D scanner is presented in [6]; the system, being composed of a network of Raspberry Pi units, yields good performances however it has a higher complexity and no portability. Finally, the authors in [7] proposed the use of magnetic angular rate gravity (MARG) sensors for developing a moving 3D scanner.

This paper proposes a non-contact, low cost 3D scanning solution using laser striping. Laser striping is a principle used by most commercial scanners [8]. The solution proposed can be divided into two components: the hardware setup - used for acquiring the object's 3D surface information, and the software part - that processes the information and obtains the 3D model representation of the object.

The system has the following *modus operandi* (illustrated in Figure 1): the object to be scanned is placed on a turntable which is controlled by a stepper motor. A line laser projects a beam on the rotating object and a VGA camera captures images of this process. The laser, camera, and stepper motor are being controlled by the FPGA system, which is responsible also for capturing and storing the images taken and also transferring them via the Ethernet interface to a PC. The images containing the 3D information gathered are then processed online by a Web service via a web page and the 3D model reconstruction is afterwards available for download.

We propose two major improvements over the traditional scanning solutions, one for each of the two components: the 3D information acquisition is based on a *reconfigurable hardware platform* – a Xilinx Spartan 6 FPGA – which adds flexibility and scalability to the scanning process, while the 3D model reconstruction is remotely available *"as a Service"*, by the means of a web interface that abstracts away the complexity of the underlying processes and improves the performance, while granting easy sharing between users.

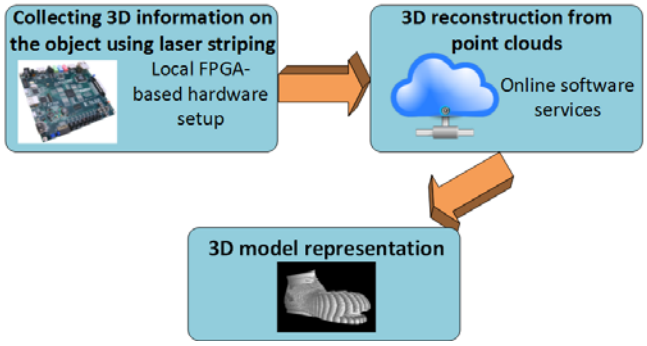

Figure 1. Service-oriented FPGA-based 3D model acquisition system overview

This paper is organized as follows: section II describes our solution's design and implementation. Section III presents the testing and validation of the entire system and 3D scanning results, and finally section IV concludes the paper and proposes future extensions on this subject.

Digital Object Identifier 10.4316/AECE.2015.04014

II. Design and Implementation

The core of the hardware implementation is the Digilent Atlys development board based on the Xilinx Spartan 6 LX45 FPGA chip. This board contains several high performance peripherals, Gigabit Ethernet and the on board 128 MB DDR2 memory being used for this particular implementation. The images are being captured using the MagnaChip HV7131GP CMOS image sensor [9], having a 640x480 VGA resolution. This sensor has been connected to the FPGA board using a dedicated PCB and connectors. Another component of the image acquisition system is a rotating turntable (on which the object is placed) which is controlled by the FPGA using a NMB stepper motor connected using a specialized driver circuit.

The system is using a laser striping principle. Thus, we have used a line laser (a red laser equipped with a cylindrical lens) that projects a beam on the object to be scanned. The laser and stepper motor are both connected to and controlled by the FPGA platform.

A. Hardware implementation

The MicroBlaze embedded software processing system (having a 32 bit RISC architecture) is controlling the functionality of the FPGA platform [10]. Being a soft processor core it is implemented during logic synthesis in the FPGA's programmable logic, being platform-independent. The instruction and program memories are implemented in the FPGA's BRAMs (block-RAMs).

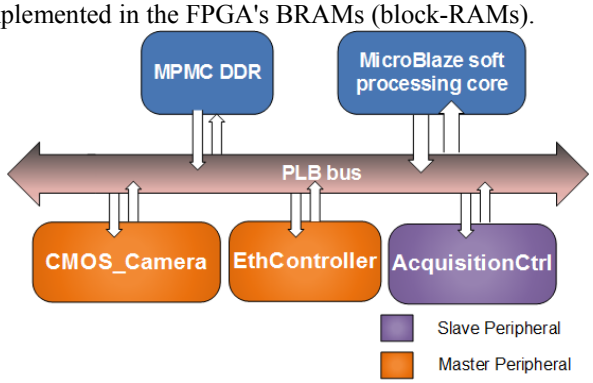

Figure 2. FPGA embedded system internal architecture

For interconnecting the processor and various IP cores and other integrated components, a PLB (Processor Local Bus) has been used.

The MPMC (Multi-Port Memory Controller) is an integrated controller generated by the Xilinx EDK software tool for an easy interfacing with the on-board memories [11] (in our case, the DDR memory used for storing captured frames). This peripheral is thus responsible for accessing the memory with read/write instructions via the PL.

We have implemented three IP Cores integrated into the embedded systems as peripherals (two of them master and one slave) responsible with controlling the camera, the stepper motor and for communicating via the network (sending captured images). The functionality of these peripherals is controlled by the embedded application running on the MicroBlaze processing core [12].

1) CMOS_Camera peripheral

The video camera, a CMOS image sensor, is interfaced using the *CMOS_Camera* master peripheral which is acting as controller for this device. It generates, using the *cmosCtrl* module, the signals needed for the initialization and correct operation of the image sensor (master pixel clock, reset and enable) and receives pixel data and synchronizing signals (Hsync and Vsync) from the camera.

Pixel data is received in YCbCr 4:2:2 format and then converted to RGB 8:8:8 using the *colorConversion* module.

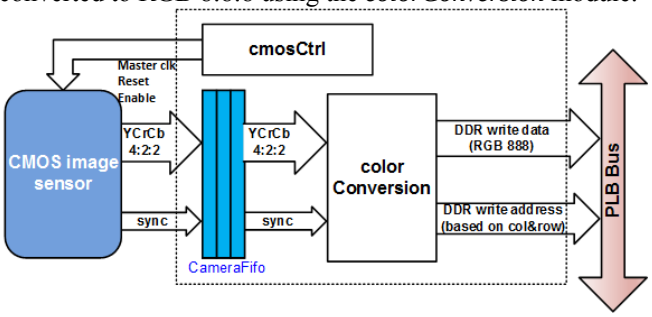

Figure 3. CMOS_Camera peripheral internal architecture.

During the conversion process, column and row information for each pixel is extracted from the two synchronization signals (*Hsync* and *Vsync*) [13].

This peripheral is controlled from MicroBlaze processor using software registers. The peripheral waits for a *captureImage* command from the MicroBlaze processing core in order for a new image to be acquired from the sensor. After receiving this instruction – written by MicroBlaze into a register -, it polls the synchronization signals in order to determine when a new frame can be received. Image data is being converted and written into the DDR memory via the PLB bus. The corresponding address for each write operation is computed based on the pixel row and column information.

After the image frame has been completely received and written into DDR, the peripheral enters the idle state and waits for a new capture command.

2) EthController peripheral

The network communication over the Ethernet interface for sending images is managed by the *EthController* master peripheral. This master peripheral is also DMA enabled so that it can read the frames from the DDR memory directly, keeping the processor free from unnecessary tasks. The internal architecture of this peripheral is illustrated in Figure 4. The two internal modules, *EthSend* and *EthReceive*, are responsible with sending/receiving data frames over the Ethernet to and from the PC.

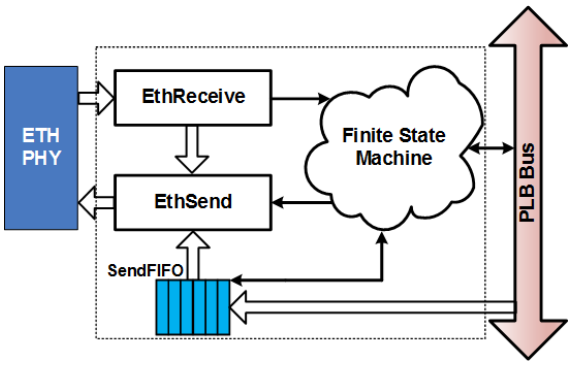

Figure 4. EthController peripheral internal architecture.

A proprietary protocol has been implemented, complying with the 802.3 standard [14], and a basic acknowledgement mechanism confirms the receival of each individual packet by the software application. The peripheral waits a certain

interval for the confirmation and at the end of that period, if the acknowledgement has still not been received, the packet is re-sent.

The protocol's packet structure is shown in Figure 5. This structure complies with the IEEE 802.3 protocol standard; it starts with the mandatory Preamble and SFD fields and the data field has been portioned according to the application's needs. The first byte indicates the type of packet (acknowledgement or containing image data), followed by either the confirmed packet's number, or the length and actual image data. Having to send quite large images, it is to expect that the majority of packets will have the maximum length of 1500 bytes. However, the module's implementation ensures that in the case of shorter lengths, an extra padding (byes having a zero value) is appended after the actual data in order to guarantee a minimum packet length of 64 bytes [15].

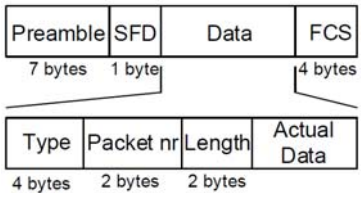

Figure 5. Diagram showing the structure of an Ethernet frame complying with the implemented proprietary protocol.

The transfer of a new image is initialized by the software application that sends a specific packet interpreted by the peripheral as a frame request. Upon receiving this signal, image data is being read from the DDR memory and encapsulated in Ethernet frames sent "on-the-fly". This is possible by using a synchronization FIFO in between the two clock domains – the Ethernet 25MHz transmit clock used for writing data on the interface, and the internal, 100MHz PLB bus clock used for accessing the DDR memory. This feature contributed to an important speed-up of the overall process.

3) AcquisitionCtrl peripheral

The mechanical rotation of the turntable is achieved by using a very fast and exact stepper motor. The *TurntableCntrl* slave peripheral is in charge of this functionality and also controls the triggering of the laser line. The C embedded application running on the MicroBlaze core sets up the software registers responsible with controlling the functionality of the stepper motor.

In order to synchronize the output signal driving the motor, a specific register - *configTimers* - was used for setting up the initial value of the internal timers.

The action register contains data for setting up both the motor and the laser's parameters (direction, number of steps, and the actual state of the laser - on/off). A status register plays the important role of signaling the fact that the rotation and the laser triggering events have finished successfully. At this point, the C application executes a delay before acquiring a new image to ensure that the new frame is captured without mechanical interference from the moving platform.

4) MicroBlaze C application

The embedded C program running on the MicroBlaze soft processing core "coordinates" the entire FPGA application. When executed, it first enters a initialization routine that sets up the software registers used by each peripheral (e.g. the DDR base address, the starting value for the timers inside the stepper controller module, etc.).

After initialization, a short demo of the entire system follows: an image is captured, the stepper motor rotates the platform a certain angle and the laser is activated. This is done to confirm that all devices are working properly and to make debug attempts easier. Following this demo the application enters its main loop that consists of six actions.

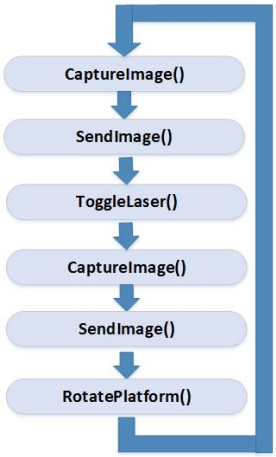

Figure 6. Schematic of the embedded C application's main loop

Two images are taken – one with just the object and one with the laser line projection – each of them being afterwards sent to the PC via the Ethernet. In between the two captures, the laser is toggled on/off and the platform is rotated a certain number of steps (according to the configuration registers). The difference between the two images taken produces the same result regardless of the order – which image is first – allowing for the loop to be configured as above; the order of the two images captured (with/without laser) alternating each time the loop is executed.

In order to improve the performance and speed up the image capture and transfer processes, we used a double-buffering approach, in other words while the first image is being sent, the second one is being captured, and so on. So as to enable such a procedure, each of the two images is being read in its own memory space, thus avoiding concurrent access to the same memory locations. This approach was being considered in the design stage of the system, and is one of the reasons for having two distinct IP cores (*CMOS_Camera* and *EthController*) implementing the two functionalities (image capture and transfer). Thus, the two peripherals operate simultaneously and are being arbitrated by the PLB Core Central Bus Arbiter.

B. Software implementation

1) C raw socket network application

This application implements the communication between the PC and the Atlys FPGA board over the Ethernet interface, using RAW sockets. This type of sockets is mandatory to be used since we have implemented our own proprietary protocol that requires communicating on the data-link layer (level 2 of the OSI stack) [16]. Thus, by working with raw sockets, the entire Ethernet frame (including headers) is being received by the application.

The application opens two RAW sockets, one for transmitting and one for receiving data. These sockets are

configured with the communication domain set to PF_PACKET (allowing working with RAW packets at the data-link layer), the sockets' type is SOCK_RAW and in order to enable packet traffic regardless of protocol, the ETH_P_ALL protocol is specified.

The application receives the packets sent by the FPGA system (containing the image data) and sends back to the board two types of packets: acknowledgement - for confirming each received packet, and request - sent after a complete image has been received in order to request a new one.

2) Image processing application

The image processing software has been developed in C using OpenCv 2.1 libraries [17]. In order for this application to provide the best results, two synchronization mechanisms have been implemented: calibration and angle adjustment.

Calibration is mandatory in the process of setting up the coordinates of the axis of rotation. The lack of proper calibration would lead to an erroneous perspective of the object (shifted towards the interior). At the beginning of each scanning session the first images are taken using a standard cube placed on the rotating platform. The laser projection lines, given the cube's particularities, are then used by the image processing software to calibrate the parameters [18].

Another important aspect is adjusting the angle between the camera and the laser. For best results it should be in the range 15-20 degrees. Again, not properly adjusting it would lead to a vertical distortion of the object's perspective.

After applying the two synchronization mechanisms described above, a correct translation from cylindrical to Cartesian coordinates can be performed.

The next process is extracting the laser line from the scanned images. This has been accomplished by implementing an algorithm that performs the difference between two image frames (the object in the same position, with and without the laser line projection). The resulting image contains only the extracted laser line. In order to reduce the unwanted “salt and pepper” noise from the image an extra median filter is applied. Further on the process of edge thresholding if performed that leads to a clearer outline of the projected laser line in the image [19].

After completing these operations, the midmost pixel located on the laser projection outline is selected from each line and, using its coordinates, the distance to the axis of rotation is computed. Thus the cylindrical coordinates of the specific point are obtained. The conversion to the Cartesian coordinates is done using the mathematical equation below (1):

$$x = \rho \cos \varphi$$
$$y = \rho \sin \varphi \quad (1)$$

Where:

- ρ is the Euclidean distance from the axis of rotation to the point P
- φ (the azimuth) is the angle between the reference position of the object and the current position at the specific moment during scanning

The image processing application uses these Cartesian coordinates to create two databases used for 3D model reconstruction [20]:

- One containing only the enumeration of the Cartesian coordinates - this is used for 3D modeling based on point clouds algorithms using MeshLab (an open source, portable, and extensible system for the processing and editing of unstructured 3D triangular meshes) [21].
- The second is a VRML (Virtual Reality Modeling Language) file.

3) Service-oriented integration

Since the software implementation deals with low-level protocols and complex operations involving several processes that need to be executed sequentially in order to obtain the 3D model of the scanned object, we have considered abstracting away the complexity of this flow by applying a service-oriented approach that offers the user a basic, easy-to-use web interface for controlling the entire scan process.

Service-oriented architectures are the ideal approach for developing flexible middleware solutions, solving the problem of inter-operability and thus being able to integrate "seamless" a variety of software and hardware technologies. These assets are enabled by the standards behind SOA (like XML, SOAP, WSDL, and UDDI) that allow a unified, service-based approach to different resources, thus abstracting away their specific functionality [22].

We have considered this approach not only useful, but mandatory in the sense that it can improve usability of the system and also enhance portability since the image acquisition and processing services are now remotely-available "in the cloud", and can be accessed from any location. This is important since it separates the hardware setup from the image processing software, which also gains in speed since it is performed on a dedicated powerful server.

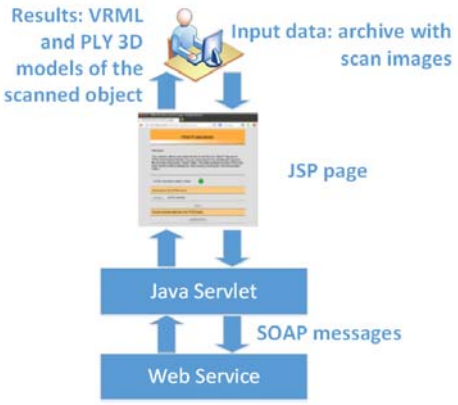

Figure 7. Web service-based work flow.

Our web-based implementation (described in Figure 7) provides a basic JSP (Java Server Pages) web interface that provides support for the I/O data flow between the user and the web service that runs the actual image processing tasks. The communication between the JSP page and the web service is intermediated by a Java servlet that translates the data flow to/from the JSP page into SOAP (Simple Object Access Protocol) - the key component of SOA [23].

The web service was implemented in Java and runs on a Glassfish 4.0 Server instance. It includes a method that creates and configures a local working environment on the server and implements the actual image processing tasks. The Java Servlet is responsible for encapsulating the

received image files as an archive into SOAP messages and forwarding them to the web service.

We have considered an optimization technique that, by processing the images "on-the-fly" (i.e. as they are being received), improves the image processing speed: after each pair of images is received, they are processed as shown above and the point coordinates are extracted into the database. After all the images are received, the method proceeds to execute the specific software tasks that result in the two files described above: a VRML 3D model of the scanned object, and a PLY file (Polygon File Format) - the 3D model obtained from point clouds. The Java servlet therefor acts like an extension to the server, improving its functionality.

III. Validation and Results

The system has been tested and validated both using specific simulation scenarios (regarding the HDL IP Cores – the integrated peripherals) and also by performing several scanning processes and analyzing the resulting 3D models.

In TABLE I**Error! Reference source not found.** a summary of the FPGA utilization is displayed, showing that the embedded system occupies around 20-30% of the resources available in terms of Slices (the Xilinx technology basic unit composed of LUTs and FFs) and 45% of the IOBs (Input/Output Buffers – the effective number of FPGA pins used). These results show that the design can be easily accommodated by the Spartan 6 FPGA, allowing future developments to be implemented in the programmable logic (like JPEG encoding and other image processing tasks) and also adding extra I/O connectivity.

TABLE I. SPARTAN 6 FPGA DEVICE UTILIZATION

Slice Logic	Utilization	
	Nr.	%
Number of Slice Registers	4312	7
Number of Slice LUTs	4777	17
Number of occupied Slices	6822	28
Number of bonded IOBs	99	45
Number of RAMB16BWERs	20	17
Number of DSP48A1s	13	22

Also, the data transfer between the FPGA board and the PC has been the subject of extensive testing, since it was implemented using our own proprietary protocol.

Fault insertion was implemented in order to test the communication and data transfer thoroughly. Thus, using on-board switches and buttons, several scenarios were implemented that allowed observing the behavior of the entire system in critical cases:

- turning off receiving of acknowledgement packets by the FPGA board for testing if the re-sending mechanism is working properly
- sending malformed packets that do not comply with the proprietary protocol to see how the software application deals with this issue

Furthermore, a benchmarking of the communication interface has been performed for both evaluating its performance and stressing the interface by emulating high traffic loads. This was implemented by sending a continuous data stream and monitoring the communication using *IPTraf* Linux utility. The results are shown in TABLE II below.

TABLE II. STATISTICS OF A DATA TRANSFER FROM PC TO FPGA

Metric	Value
Incoming packets	328655
Incoming bytes	248463 K
Outgoing packets	328655
Outgoing bytes	331256 K
Total packets	657310
Total bytes	579719 K
Incoming rates	2935.6 packets/s
	17754.5 Kbits/s
Outgoing rates	2935.4 packets/s
	23670.0 Kbits/s
Total rates	5871.0 packets/s
	41424.5 Kbits/s

The figures show that the transfer speed between PC and FPGA using the implemented proprietary protocol varied between 2.2 and 2.9 MB/s which is a good range for this application. This rate is however limited by the acknowledgement mechanism, which confirms individually each data packet, and can thus be improved. We are considering as a future development to implement a sliding-window type mechanism that would result in a potential speed-up of the data transfer.

In order to test the accuracy of the scanning process, different types of object were scanned, at different scanning distances in order to see the ability of the system to record surface details. Figure 8 below shows the scanning process and the resulting 3D model of an electrical connector block, showing that the scanner was able to correctly capture miniature details of the object (as small as a few millimeters).

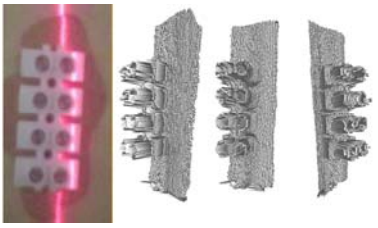

Figure 8. Scan results of an electrical connector block.

The detailed steps involved in the scanning process can be seen in Figure 9. The first image is the "clean" image taken of the object, while the second is a shot of the object in the same position, but with the laser line projection and containing the axis of rotation (with blue) reconstructed by the software application in order to compute the distance between it and the laser line. Finally, the bottommost image is the difference between the two above, showing the laser outline which will be further on used for computing the coordinates and distances.

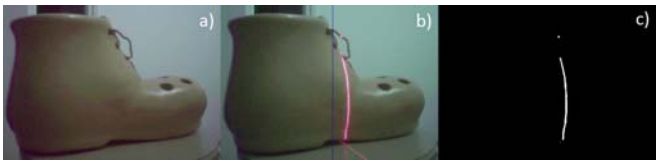

Figure 9. 3D modeling process of a clay figure. Simple image (a), laser projection and reference axis (b), image difference showing the laser outline (c).

As mentioned above, the scanning process generates two files, both containing a 3D models of the scanned object. In Figure 10 below it can be seen the VRML representation of the scanned object from Figure 9.

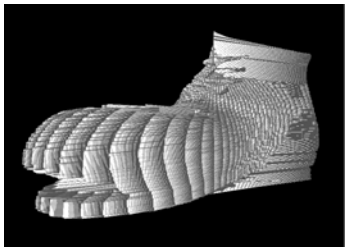

Figure 10. VRML 3D representation of a scanned object

IV. CONCLUSION

We proposed in this paper a novel approach based on a reconfigurable hardware platform and web services, thus gaining in *flexibility* and *scalability* over the traditional 3D scanning solutions. By separating data capture - based on the FPGA hardware setup - from the image processing and 3D model reconstruction tasks - available as services, *"in the cloud"* -, the system gains in *portability* - a feature that is absent for most existing solutions [24].

There is also a considerable *performance gain* since the intensive computational tasks - image processing and 3D model elaboration - are "outsourced" to powerful online computing servers.

Ease of use is another asset of our proposed implementation - the user does not have to bother managing and using a variety of complex software tools for obtaining the desired 3D representation of the scanned object, he benefits from the standard, friendly web interface that abstracts away the complexity of these tasks.

This solution has, compared to other commercial 3D scanners, a *low design and implementation cost*, especially since it is based on a reconfigurable hardware device (FPGA), which can be re-used for other designs, thus eliminating the costs of dedicated hardware. The *Software-as-a-Service* component also contributes to lowering overall costs since it supports multi-user access; therefor these users are exempted from having to manage software tools locally, on dedicated PCs.

Last but not least, the service oriented approach opens up new possibilities and applications since it can be used to obtain 3D models using data captured by other scanners that comply with the same principle; this makes it an important functionality that can be used both together and separately with the scanner setup.

We are considering as future developments transcending some of the image processing tasks from software to hardware, in order to benefit from the FPGA hardware acceleration given the fact that, as shown above, the utilization of the device allows such an approach. Also, we are focusing on improving the performance and overall speed of the system by improving the communication solution between FPGA and PC.

REFERENCES

[1] R. B. Catalan, E.I. Perez, B.Z. Perez. "Evaluation of 3D scanners to develop virtual reality applications." In Electronics, Robotics and Automotive Mechanics Conference, CERMA 2007, pp. 551-556, IEEE. Available: http://dx.doi.org/10.1109/CERMA.2007.4367744.

[2] X. Ning, Y. Wang, "Object Extraction from Architecture Scenes through 3D Local Scanned Data Analysis," Advances in Electrical and Computer Engineering, vol.12, no.3, pp.73-78, 2012, doi:10.4316/AECE.2012.03011

[3] Z. Lv, Z. Zhang. "Build 3D laser scanner based on binocular stereo vision." In 2011 Fourth International Conference on Intelligent Computation Technology and Automation, vol. 1, pp. 600-603. 2011. Available: http://dx.doi.org/10.1109/ICICTA.2011.158.

[4] N.A. Borghese, G. Ferrigno, G. Baroni, A. Pedotti, S. Ferrari, R. Savare. "Autoscan: A flexible and portable 3D scanner" IEEE Comput. Graph. Appl. No.18 (1998), pp. 38-41. Available: http://dx.doi.org/10.1109/38.674970.

[5] C. Rocchini, P. Cignoni, C. Montani, P. Pingi, R. Scopigno. "A low cost 3D scanner based on structured light." In Computer Graphics Forum, vol. 20, no. 3, pp. 299-308. Blackwell Publishers Ltd, 2001. Available: http://dx.doi.org/10.1111/1467-8659.00522

[6] J. Straub, S. Kerlin. "Development of a large, low-cost, instant 3D scanner". Technologies No. 2 (2014), pp. 75-95. Available: http://dx.doi.org/10.3390/technologies2020076

[7] D. Grivon, E. Vezzetii, M.G. Violante. "Development of an innovative low-cost MARG sensors alignment and distortion compensation methodology for 3D scanning application". Robot.Auton.Syst. No. 61 (2013), pp. 1710-1716. Available: http://dx.doi.org/10.1016/j.robot.2013.06.003

[8] O. Wulf, B. Wagner, "Fast 3D scanning methods for laser measurement systems," in International Conference on Control Systems and Computer Science (CSCS14), 2003.

[9] CMOS Image Sensor with Image Signal Processing - HV7131GP datasheet. Hynix, 2003. Available at: http://www.globaltec.com.hk/databook/hynix/Hyca3_V20.pdf

[10] MicroBlaze Processor Reference Guide Embedded Development Kit - EDK 14.1. Xilinx UG081, 2012. Available at: http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/mb_ref_guide.pdf

[11] LogiCORE IP Multi-Port Memory Controller (MPMC) (v6.03.a) Product Specification - Xilinx 2011. Available at: http://www.xilinx.com/support/documentation/ip_documentation/mpmc.pdf

[12] B. Muralikrishna, G.L. Madhumati, H. Khan, K.G. Deepika, "Reconfigurable System-on-Chip design using FPGA," 2nd International Conference on Devices, Circuits and Systems (ICDCS), 2014. Available: http://dx.doi.org/10.1109/ICDCSyst.2014.6926215.

[13] M.K. Birla, "FPGA Based Reconfigurable Platform for Complex Image Processing," IEEE International Conference on Electro/information Technology, pp.204,209, 2006. Available: http://dx.doi.org/10.1109/EIT.2006.252111

[14] IEEE 802.3™-2012 – IEEE Standard for Ethernet (accessed 10.08.2014), http://standards.ieee.org/about/get/802/802.3.html

[15] N. Alachiotis, S.A. Berger, A. Stamatakis. "Efficient PC-FPGA communication over Gigabit Ethernet." IEEE 10th International Conference on Computer and Information Technology, (CIT), pp. 1727-1734, 2010. Available: http://dx.doi.org/10.1109/CIT.2010.302.

[16] B.B. Hall. "Beej's guide to network programming: using Internet Sockets." (2012). Available online: http://beej.us/guide/bgnet/output/print/bgnet_A4.pdf

[17] G. Bradski, A. Kaehler. Learning OpenCV: Computer vision with the OpenCV library. " O'Reilly Media, Inc.", 2008.

[18] F.A. Van Den Heuvel, "Object reconstruction from a single architectural image taken with an uncalibrated camera." Photogrammetrie Fernerkundung Geoinformation (2001): 247-260.

[19] Z. Wang, D. Zhang. "Progressive switching median filter for the removal of impulse noise from highly corrupted images." IEEE Transactions on Circuits and Systems II, 46, no. 1 (1999): 78-80. Available: http://dx.doi.org/10.1109/82.749102.

[20] R.T. Whitaker. "A level-set approach to 3D reconstruction from range data." International Journal of Computer Vision 29.3 (1998): 203-231. Available: http://dx.doi.org/10.1023/A:1008036829907

[21] T.P. Kersten, M. Lindstaedt. "Image-based low-cost systems for automatic 3D recording and modelling of archaeological finds and objects." In Progress in cultural heritage preservation, pp. 1-10. Springer Berlin Heidelberg, 2012. Available: http://dx.doi.org/10.1007/978-3-642-34234-9_1

[22] M.D. Hansen. SOA Using Java Web Services. Pearson Education, 2007.

[23] B. Perry. Java Servlet & JSP Cookbook." O'Reilly Media, Inc.", 2004.

[24] W. Böhler, A. Marbs. "3D scanning instruments." In Proceedings of the CIPA WG 6 International Workshop on Scanning for Cultural Heritage Recording, Ziti, Thessaloniki, pp. 9-18. 2002.