FPGA Implementation of 2D Cross-Correlation for Real-Time 3D Tracking of Deformable Surfaces

Amir HajiRassouliha[1*], Thiranja P. Babarenda Gamage[1], Matthew D. Parker[1], Martyn P. Nash[1,2], Andrew J. Taberner[1,2], Poul M. F. Nielsen[1,2]

[1] Auckland Bioengineering Institute, [2] Department of Engineering Science
The University of Auckland, Auckland, New Zealand, [*]ahaj975@aucklanduni.ac.nz

***Abstract*— 3D surface measurements have many industrial and medical applications. We have previously used 3D surface deformation and tracking to identify mechanical properties of the skin. To be able to detect dynamic changes in the surface of the skin we need to have a real-time 3D measurement system. A significant portion of the computation time for tracking the changes is spent during 2D cross-correlation of surface images. This study focuses on improving cross-correlation speed by taking advantage of parallel computation in field programmable gate arrays (FPGAs).**

We have implemented variable size 2D cross-correlation computations using the Xilinx System Generator tool in the Virtex-6 LX240T FPGA. We have also proposed a hierarchical approach for finding the cross-correlation peak in order to efficiently use our method for different image sizes. Furthermore, the use of RAM blocks instead of shift registers in our design has lowered the resource requirements compared with other FPGA implementations.

Preliminary results for our special design indicate better than 200 times speed up compared with a PC with an Intel Xeon E5620 CPU (2.4 GHz clock speed, 4 cores and 8 threads) and 12 GB DDR3 RAM and also 190 times speed up in comparison to an NVidia GForce GT 525M as the graphics processing unit (GPU).

***Index Terms*-- FPGA implementation, 2D Cross-Correlation, Real-time 3D measurement, Xilinx System Generator, GPU surface deformation measurement, template matching.**

I. INTRODUCTION

3D geometric measurement and tracking of a deformable object is desirable in various fields. 3D measurement has been applied in several different industrial [1] and medical applications [2].

Tracking 3D geometric changes of the surface of skin under deformation can improve our understanding of its mechanical properties. Biomechanical properties of the skin have been determined by various methods [3],[4] and used to inform computational models. Computational modeling of the skin can assist in studying wrinkling [4], and is also beneficial for planning plastic surgery [5]. In our application, we use 3 cameras and a surface indenter (microrobot) to measure the strain fields of the skin based on surface deformations [6]. The cameras are placed around the tissue obtain sufficient images for 3D reconstruction while overcoming occlusions caused by the surface indenter (Fig. 1). The procedure for measuring 3D deformation in our system is shown in Fig. 2. The first step in this procedure is to find camera parameters that will be used for the triangulation process. Then some corresponding feature points should be found in images before and after the deformation. The displacement between these points identifies the surface deformation and stereoscopic triangulation gives this deformation in 3D.

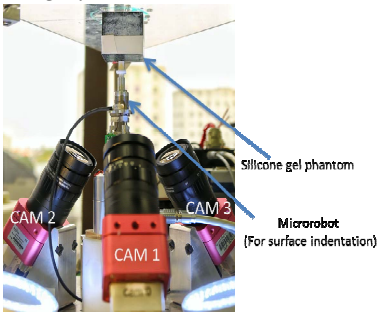

Figure 1. The camera setup for 3D measurements. There are three cameras (CAM 1, 2, 3) placed around the tissue, a microrobot for surface indentation and a silicon gel phantom as the soft tissue.

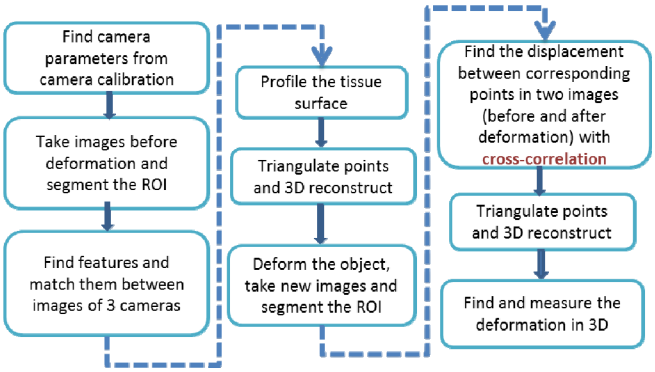

Figure 2. The procedure for measuring the 3D deformation in our application.

Cross-correlation (CC) is a widely recognized approach to determine the displacements between two images [7], and is also used in our application to find feature point displacement

978-1-4799-0883-7/13/$31.00 ©2013 IEEE

between deformed and undeformed images. However, as can be seen in the 2D CC equation for real values(1), the large number of multiplications and accumulations means that CC is computationally expensive.

$$\sum_{x=-\infty}^{\infty}\sum_{y=-\infty}^{\infty} I_1(x,y) \times I_2(x-i, y-j) \qquad (1)$$

In many applications there is a need for real-time systems to track the deformations dynamically. FPGAs and graphics processing units (GPUs) are good candidates for real-time systems, as they offer high speed parallel processing capabilities. GPUs are comprised of a large number of cores which make them suitable for image processing tasks with high level of data paralelism [8]. FPGAs have also been used for some real-time video and image processing tasks [9],[10] and their use is increasing as a result of the recent technological and manufacturing developments. FPGA architecture is suitable for implementing the cross-correlation (CC) algorithm [11]. The performance evaluation of FPGA and GPU is highly dependent on the hardware choice and the application. Fowers *et al* [12] showed that the FPGA implementation was the fastest for a floating-point 1D convolution. In addition to the performance, FPGAs have the advantage of architectural felexibility in desiging algorithms for a specific application. This felexibility, which has made it possible to have efficient designs, is the main reason for choosing the FPGA for our implementation.

In this paper, we describe the implementation of a variable size 2D CC in an FPGA with a novel hierarchical approach for finding the cross-correlation peak in an effort to bring the performance of our 3D surface deformation measurement system towards real-time.

The FPGA implementation is described in section II. In section III, a detailed analysis of the design is provided and a novel hierarchical approach for finding the cross-correlation peak is suggested. Furthermore, in this section speed performance is compared with a PC with an Intel Xeon E5620 CPU and also with a NVidia GForce GT 525M GPU [13]. Finally, in section IV are the discussion and the conclusion.

II. METHODOLOGY

We used the "Xilinx System Generator" tool (SysGen)[14] in our project. This tool has been designed for the bit- and cycle-accurate implementation of signal and image processing algorithms in Xilinx FPGAs and has reduced the development time considerably. In addition, this tool enables the possibility of hardware co-simulation [14].

The normalised cross-correlation (NCC) technique is a more powerful method in image registration and tracking compared with non-normalised cross-correlation (NNCC) but it is more computationally expensive [15]. Fast NCC has been suggested as a means to reduce the complexity by normalising the results of NNCC by pre-computing integrals of the image and the square of the image in the search window [15]. Since the aim of this implementation is for real-time tracking of surface deformations, this approach of fast NCC has been chosen for the FPGA implementation design to reduce the complexity and increase the speed.

Implementation of the 2D CC in the spatial domain was chosen for our design because its implementation has many advantages over the spectral domain implementation and is more efficient in resource allocation and maximum speed, especially for small image sizes. For more details see Lindoso [11].

Huber *et al* [16] implemented one dimensional auto-correlation, with the counter-based correlation, by using delay lines to compute correlation. To employ CC in image processing, we have implemented 2D CC. Unlike these implementations [16]-[18], instead of using a large number of shift registers, we have used block RAMs for storing the data. The use of block RAMs eliminates the need for exploiting shift registers and considerably reduces the area occupied by the design. A novel hierarchical approach has also been suggested at the end to efficiently use our method for different image sizes and take advantage of parallel processing of the FPGA.

The implementation of this design is simulated based on Xilinx ML605 boards which contains the Virtex-6 LX240T FPGA and the PCIe interface [19]. This board has been chosen since Virtex-6 and PCIe are suitable for image processing applications and can represent a realistic cost-efficient choice for our application. The default width of two images is chosen to be 32px because in our 3D tracking system the size of the searching window for finding the displacements is 32px × 32px [6]. However, this size can be varied from 1px to 32px. The height of the images can also be varied between 1px to 512px. The value for the height of the images has been chosen in a way to make it easier to find the place of the subset image in a bigger neighbourhood. Because memory size in the Virtex-6 LX240T is 36bits × 512 [20] we have the most optimum resource allocation for the size of 512 for the length of RAMs. It is possible to use a different number of modules in parallel for bigger images.

Our algorithm is designed to be block-based in order to make the code implementation and debugging easier and faster with Xilinx SysGen, rather than with direct VHDL coding. The SysGen tool has previously been used for implementing real-time algorithms and has demonstrated its advantages [21],[22].

The purpose of our FPGA design is to match a subset image of the deformed surface to the initial undeformed image, in order to measure the displacement and track material points. Therefore, for each camera two images (before and after the deformation) are used to track the points in the procedure that is shown in Fig.2. These images are grayscale as it is more suitable for tracking surface deformations [7]. The CC thus starts from a position where the subset image has complete overlap with the initial image and ends when it reaches the end of any overlap. As a result, the first position is where top of two images are aligned and the last position is where they are aligned at the bottom. The last position depends on the size of the initial image, which can be variable (Fig. 3.)

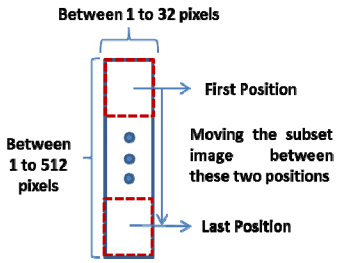

Figure 3. The cross-correlation algorithm for two images. Dashed lines show the subset image and solid lines show the initial image. The first position is where top of two images are aligned and last position is where they are aligned at the bottom.

The multiplication part of the CC algorithm is applied to all the pixel values in the width of images in each row in parallel. The output of the multiplication is stored in an accumulator for all the subset image's rows (corresponding to the image height). This accumulator has the same role as the ∑ operator in (1) and its value quantifies the resemblance between the subset image and the initial image.

After calculating the value of CC for one row, the subset image will move one pixel down and the value of the accumulator will reset to calculate the value of the CC for the new position. Fig.4 shows this concept:

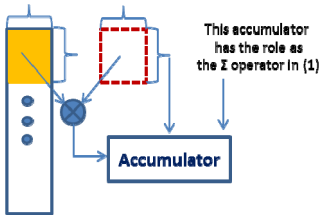

Figure 4. Calculating the CC value for each relative position of the subset and the initial images

The design implemented in SysGen can be divided into three main stages.

- Load and initialisation: In this stage, data of two images is loaded and stored in multiple block-RAMs. The initial inputs are also set in this stage.
- Synchronisation: In this stage, the outputs of each set of block RAM are synchronized in a way that we can have pixel correspondences between two images for each CC value.
- Calculating the CC value: In this stage, the intensities of corresponding pixels from each image are multiplied, and the sum is computed.

These three stages are described in more detail:

A. *Load and initialisation*

For each image, four block RAMs of size 64bit × 512 are selected. Each pixel is 8 bits deep, therefore 64bits can store the intensities of 8pixels. The width of each block RAM is chosen to be 64bits because it is the standard width for PCIe and DDR RAM which are usually available on the FPGA boards suitable for real-time applications (such as ML605 that has been selected for the simulation of our algorithm). The intensity values of 8 pixels are then concatenated to form 64bits, as shown in fig. 5.

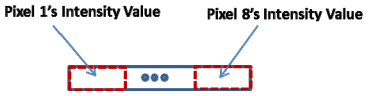

Figure 5. Concatenating the intensity values for 8 pixels to form a 64bit digit.

The heights of the initial and subset images are also set during this stage.

B. *Synchronisation*

To have pixel correspondences between pixel locations in two images with variable sizes, the corresponding pixels have to be read at the same time. In order to have this synchronisation in the FPGA, the flow diagram shown in Fig. 6 is implemented.

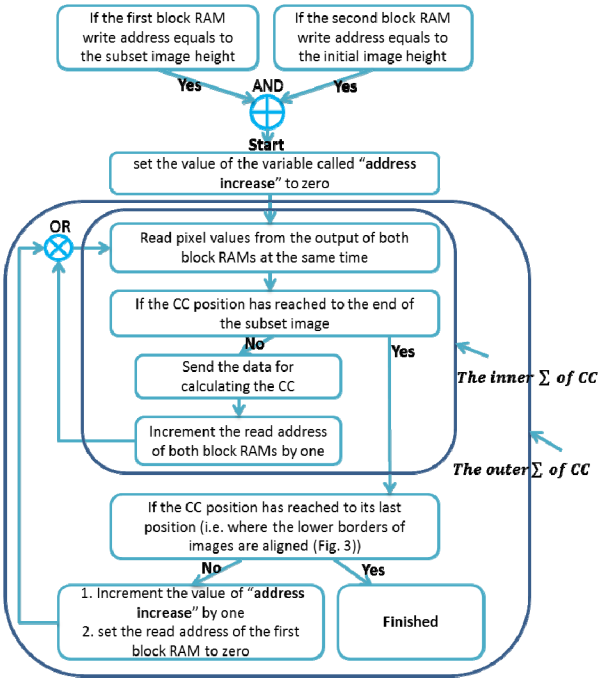

Figure 6. The flow diagram of the procedure for synchronisation between two images to have a complete pixel correspondence.

The whole procedure is comprised of two main loops, one for the inner Σ and the other for the outer Σ in (1). The variable "address increase" is used to control the limits of the outer Σ, therefore it is set to zero at the beginning of the process and is automatically incremented by one when one set of CC is finished. This ensures that the next CC value will start with a

new position for the subset image on the initial image. To have variable size for the CC, the criteria for both loops are checked based on the size of two images, which is set in the initialisation stage. Several flags are used in the process of checking and controlling these conditions and to loop through all the lines of data and synchronise the computations in the parallel FPGA environment.

C. Calculating the CC value

For calculating the CC value it is essential to multiply the corresponding pixels of two images. Before multiplying the pixels, a register with an enable port is placed to make it possible to send the appropriate data to the multiplication unit.

The use of this register allows us to achieve the maximum possible speed, as the multiplication can be done at the main clock frequency. Three main clock delays are chosen for the multiplication unit for the pipelined operation. Adjacent to the multiplication unit is an accumulator to store the data for all the pixels in the width of images. The accumulator is chosen to be DSP48E1 [23] to increase the speed and reduce the usage of logic gates. Multiplication and accumulator units are shown in Fig. 7. The reset signal is sent to this accumulator to reset it at each new placement of two images. As the accumulator resets at the end position of the subset image, the height of the subset image can be varied between 1px and 512px (Fig. 3).

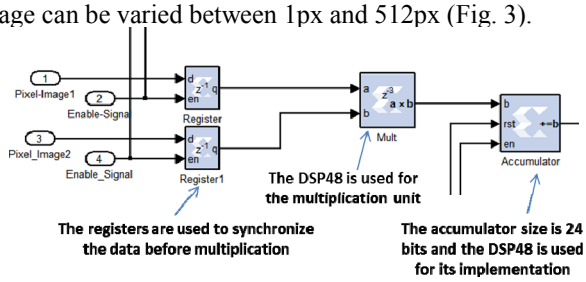

Figure 7. Multiplication unit and accumulator for calculating CC value

To preserve the correspondence between "data" and "data enable" which shows the data validity, the delays in any operation on the data should also be considered in the enable signal. For this reason, the total number of delays for multiplication and accumulation are added to the "enable signal" at the end of this stage. In order to perform parallel multiplication of every pixel, 32 multiplication and 32 accumulation units are required.

At the last step of this stage, the outputs of the CC for each position of two images are added together and the final value for CC is found. The total number of CC values at each position is equal to the height of the subset image (Fig. 4). For adding these CC values, pipelines are used and all the addition units work at the main clock frequency. The data valid signal is added by the help of the "register" at the end (Fig. 8).

The CC value can thus be sent to any other module, or can be written in external or internal RAM with the help of data enable (data valid).

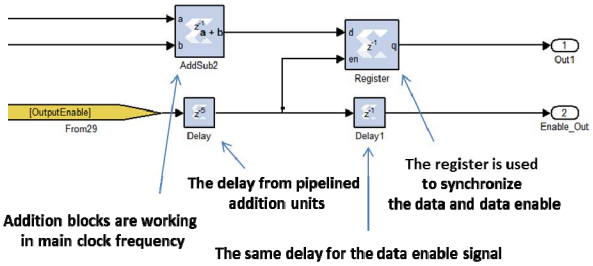

Figure 8. Using the register to synchronize the output data and the data enable signal

III. RESULTS

The cross-correlation results have been tested and validated with inputs generated from a Matlab program. To test the algorithm, counters with values between 0 to 512 are applied in all the inputs of 8 RAMs. The height for two images is set to be 32 for the subset image and 512 for the initial image. A part of the final result is shown in Fig. 9, where output demonstrates there is a complete synchronisation between the data and data enable signals. The start value of the cross correlation for the next row of data has shifted up because the input data values were counters, and thus the stored data increases in each new row.

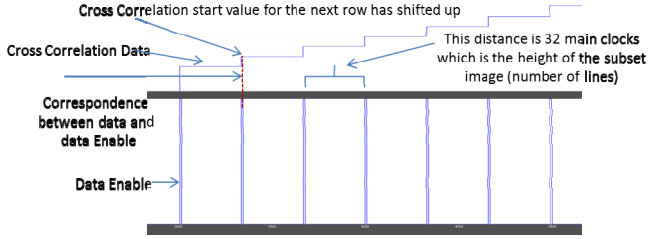

Figure 9. A Part of the final result. There is complete synchronisation between data and data enable and the CC start value has shifted up for the next line.

Xilinx ISE software [24] can provide more details of the design. The SysGen output was imported into it to build a project. Device utilisation summary of the ISE project showed that the total number of occupied slices for this design is 543 slices (approximately 1 percent of the 37,680 available slices in the Virtex-6 LX240T.) The logic gates inside the slices are different across other FPGAs, so more accurate details about this can be found in the utilisation report of Xilinx PlanAhead [24]. This report is shown in Fig. 10.

The DSP48E1 blocks are used for calculating the CC value. For each pair of pixels (from two images) one DSP48E1 is used for multiplication and one is used for accumulation. Therefore, 64 DSP48E1 blocks are required for 32 pixels

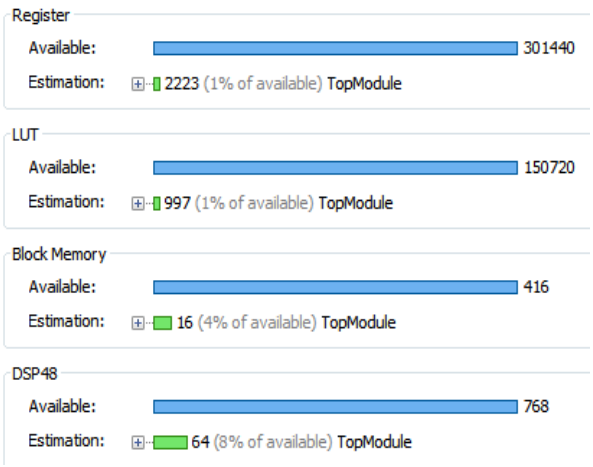

Figure 10. Utilisation report of the proposed algorithm in the Virtex-6 (LX240T)

(width of images) as can be seen in the utilisation report (Fig. 10).

There were 8 block memory units in this design with the size of 64bit × 512 and each of them has occupied 2 Block RAMs, which is the optimum number for the width of 64bits. Accordingly, the total number of block RAMs for this design was 16 (Fig. 10).

The maximum achievable frequency of this design was reported to be 252.876 MHz (Minimum period: 3.955ns) by the synthesis report of the ISE project. The occupied resources for this design were less than other implementations in the FPGA [18]-[20] for the same size of the CC. As the FPGA of the design will affect the maximum possible frequency, reliable comparison with other implementations is not practicable. Nevertheless, the maximum frequency that we could reach is 252 MHz, or around half the speed of DSP48E1 blocks [23]. In addition, in our design we can have variable size for two images.

The total number of delays in this implementation are 45 main clock cycles so, for the subset image of size 32 and initial image of size 256, it will take (45 + (256-32) × 32) = 7213 main clock cycles to make all the CC values ready. The main clock frequency can be chosen to be 252 MHz according to the ISE synthesis report, so the total time to calculate the CC value for these two images will be 7213/252 M = 28.6µs.

This module covers 32 bits of the initial image's width. However, feature matching and displacement computation usually involves finding the peak CC value over the entire initial image. To achieve the best performance for our module in finding the CC's peak, a hierarchical approach with a number of CC modules in parallel should be used. This hierarchical approach is described here for the width size of 256 pixels for the initial image as an example:

- At the first step, the initial image is divided to 16 equal slices with half size overlap and the CC value is found for all the slices at the same time. (Fig. 11).
- In the next step, two neighbor slices with maximum values of CC are chosen (for our example this will be 32 + 16 = 48 pixels) and the procedure in the first step is repeated for these slices.
  Steps one and two are repeated until the pixel number less than (32 + number of slices) is reached. For this example, this number is (32 + 16 = 48). At this step, the maximum CC value can be found at once in parallel modules of CC (because the shift between modules will be one pixel and the maximum value (peak) for the CC can be found with one pixel resolution).

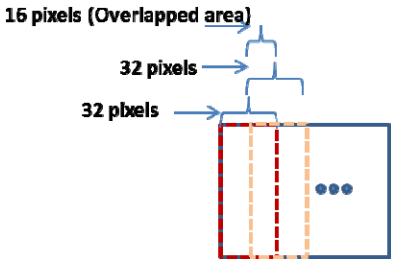

Figure 11. Two neighbour slices in hierarchical approach

This execution time is not exact because we should add the time required for finding the maximum value of the CC and loading the data for the new set of CC. To have a comparison with a Matlab program, the time needed for the cross correlation in Matlab is measured by using this command "conv2 (rot90(Subset image, 2), Initial image)" [26] which is the fastest way to find the CC values in Matlab. The time needed for this operation for the same size of images was 19.3ms on a PC with an Intel Xeon E5620 CPU (2.4 GHz clock speed, 4 cores and 8 threads) and 12 GB DDR3 RAM, a rate that is around 235 times slower than our implementation in the FPGA. For bigger image sizes the ratio of execution time in the FPGA over the CPU will decrease as we can use more parallel modules in the FPGA. Another advantage of using the FPGA over the CPU is that we can divide the whole image to the smaller region of interests (ROIs) and do the CC in parallel for all the ROIs and subsets at the same time.

Hence, the total time spent for an initial image with width of 256 pixels will be approximately 3 × 28.6µs = 85.8µs.

Both the FPGAs and the GPUs are parallel hardware. The Matlab's parallel computing toolbox [27] has also been used to implement the same function in an NVidia GForce GT 525M [13] as our GPU. The execution time in the GPU was 16.3ms which is faster than the CPU but still 190 times slower than our FPGA implementation.

IV. Discussion and Conclusion

In this paper, we have proposed a new algorithm for implementation of 2D CC of variable size images. In this method the CC is calculated in the FPGA and a solution is suggested to perform the normalization on the PC based on

[15]. The utilisation report shows our approach is well-designed for resource allocation and has occupied fewer resources in comparison to similar FPGA designs [16]-[18]. The utilisation report (Fig. 10) shows that we have used less than 1% of the available registers and LUTs of the Virtex-6 LX240T. The large number of unoccupied slices will allow us to implement many CC modules at the same time and will make it possible to add other parts of the algorithm to this FPGA implementation.

Use of SysGen for our design will help to easily change the code for other configurations, import data to our algorithm in Matlab for debugging, and perform hardware and software co-simulation. Another advantage of our implementation is that it can also be used for matrix multiplications for different matrix sizes.

It is important to consider that in both of our execution time comparisons, there are several reasons for the estimated difference between the FPGA, GPU, and CPU implementations. First, our hierarchical approach has reduced the number of calculations considerably (as discussed in section I the flexibility of the FPGA has allowed us to implement our customized design which was not easy to do that in the GPU). Second, we have not included the time needed for transferring the data to the FPGA board. However, this transfer time to the FPGA can be minimized by using the DDR RAMs in the FPGA board.

By adding several modules to the FPGA, the maximum frequency will decrease because it will be more difficult to place and route all the logic gates. Nevertheless, this reduction of speed is not expected to amount more than 10%.

The speed up comparison with the CPU would be more realistic if we compare the execution time of the FPGA to a compiled high level language code. Furthermore, in computation of the CC, the CPU only uses one of its cores because it is a complete sequential task and it can't divide that between different cores. Therefore, the CPU has not taken advantage of multi-cores and multi-threads.

The comparison against the GPU will also be more meaningful if we compare that with a more powerful GPU with more cores.

It is the authors' intention to implement a hierarchical CC in the FPGA, and the normalisation part in the PC, for improving the robustness of our 3D deformation tracking system. In addition, results from some images of surface deformation in our system will be assessed, and further comparisons will be made against a C++ and MKL [28] implementation of the CC to provide a better indication of the increase in speed. To gain more accuracy, sub-pixel interpolation also will be added to the FPGA implementation. These steps will, hopefully, lead to an accurate real-time 3D surface deformation measurement and tracking system.

REFERENCE

[1] M. Malesa, K. Malowany, U. Tomczak, B. Siwek, M. Kujawińska, and A. Siemińska-Lewandowska, "Application of 3D digital image correlation in maintenance and process control in industry", Computers in Industry, Volume 64, Issue 9, pp. 1301–1315, December 2013.

[2] S. Amin Yavari, J. van der Stok, H. Weinans, A. Zadpoor, "Full-field strain measurement and fracture analysis of rat femora in compression test", Journal of Biomechanics, V. 46, Issue 7, 26, pp 1282-1292, 2013.

[3] C. Flynn, and B .A.O McCormack, "Simulating the wrinkling and aging of skin with a multi-layer finite element model", Journal of Biomechanics, Volume 43, Issue 3, 10, pp: 442-448, 2010.

[4] C. Flynn, A.Taberner, and P. Nielsen, "Modeling the mechanical response of in vivo human skin under a rich set of deformations" Annals of biomedical engineering, 39(7), pp: 1935-1946. 2011.

[5] R.J.Lapeer, P.D. Gasson, and V. Karri, "Simulating plastic surgery: From human skin tensile tests, through hyperelastic finite element models to real-time haptics", Progress in Biophysics and Molecular Biology, Volume 103, Issues 2–3, pp:208-216, 2010.

[6] M.D Parker, M. Azhar, T.P. Babarenda Gamage, D. Alvares, A.J. Taberner, P.M.F. Nielsen,, "Surface deformation tracking of a silicone gel skin phantom in response to normal indentation," (EMBC), pp.527,530, Sept. 2012.

[7] B. Pan, K. Qian, H. Xie, and A. Asundi,;"Two-dimensional digital image correlation for in-plane displacement and strain measurement: a review", Measurement science and technology, 20(6), 062001. 2009.

[8] D. Castaño-Díez, D. Moser, A. Schoenegger, S. Pruggnaller,& A. S. Frangakis, "Performance evaluation of image processing algorithms on the GPU.", Journal of structural biology, 164(1), pp: 153-160, (2008).

[9] D. Chaikalis, N. P Sgouros, D. Maroulis, "A real-time FPGA architecture for 3D reconstruction from integral images", J. of Visual Communication and Image Representation, 21(1), pp: 9-16. 2010.

[10] K. Ambrosch, and W. Kubinger, "Accurate hardware-based stereo vision", Computer Vision and Image Understanding, 114(11), pp: 1303-1316, 2010.

[11] A. Lindoso, L. Entrena, "High performance FPGA-based image correlation", Journal of Real-Time Image Processing, 2(4), pp: 223-233. 2007.

[12] J. Fowers, G. Brown, J. Wernsing, and G. Stitt, "A performance and energy comparison of convolution on GPUs, FPGAs, and multicore processors." ACM Transactions on Architecture and Code Optimization (TACO), 9(4), 25, (2013).

[13] http://www.geforce.com/hardware/notebook-gpus/geforce-gt-525m

[14] http://www.xilinx.com/tools/sysgen.htm

[15] J. P. Lewis, "Fast Template Matching", Vision Interface, p. 120-123, 1995

[16] N. Huber, M. S. Hromalik-Pouchet, T. D. Carozzi, M.P. Gough, and A. M. Buckley, "Parallel processing speed increase of the one-bit auto-correlation function in hardware", Microprocessors and Microsystems, 35(3), pp: 297-307. (2011).

[17] S. Hasan, S. Boussakta, A. Yakovlev, "Improved parameterized efficient FPGA implementations of parallel 1-D filtering algorithms using Xilinx System Generator," (ISSPIT), pp.382,387, Dec. 2010

[18] J. Ding, X. Du, X. Wang, J. Liu, "Improved real-time correlation-based FPGA stereo vision system". ICMA, pp: 104-108. IEEE, 2010.

[19] http://www.xilinx.com/products/boards-and-kits/EK-V6-ML605-G.htm

[20] http://www.xilinx.com/support/documentation/ip_documentation/blk_mem_gen/v7_2/pg058-blk-mem-gen.pdf

[21] A.Thangavelu, M. V. Varghese, M. V. Vaidyan, "Novel FPGA based controller design platform for DC-DC buck converter using HDL Co-simulator and Xilinx System Generator," (ISIEA), pp.270,274, 2012

[22] V.R. Pamula, W. Strauss, J. Bernhard, "Model Based Development of the Digital Part of a RFID Transponder with Xilinx System Generator for a FPGA Platform," RFID Technology (EURASIP RFID), pp.124,127, 2012.

[23] http://www.xilinx.com/products/silicon-devices/fpga/virtex-6/modals/600mhz-dsp48e1-slices.html

[24] http://www.xilinx.com/products/design-tools/ise-design-suite/index.htm

[25] http://www.xilinx.com/tools/planahead.htm

[26] http://www.mathworks.com.au/help/matlab/ref/conv2.html

[27] https://www.mathworks.com.au/products/parallel-computing/index.html

[28] http://software.intel.com/en-us/intel-mkl