Hardware Acceleration of Image Registration Algorithm on FPGA-based Systems on Chip

Ioannis Stratakos
National Technical University of Athens, Greece
istratak@microlab.ntua.gr

Dimitrios Gourounas
National Technical University of Athens, Greece
dimitrisgrn@hotmail.com

Vasileios Tsoutsouras
National Technical University of Athens, Greece
billtsou@microlab.ntua.gr

Theodore Economopoulos
National Technical University of Athens, Greece
teoeco@otenet.gr

George Matsopoulos
National Technical University of Athens, Greece
gmatso@esd.ece.ntua.gr

Dimitrios Soudris
National Technical University of Athens, Greece
dsoudris@microlab.ntua.gr

ABSTRACT

Image processing algorithms are dominating contemporary digital systems due to their importance and adoption by a large number of application domains. Despite their significance, their computational requirements often limit their usage, especially in deeply embedded designs. Heterogeneous computing systems offer a promising solution for this performance gap, leading to their ever increasing utilization by designers. This work targets the acceleration of an image registration pipeline on a System-on-Chip (SoC) including both general purpose and re-configurable computing elements. The evaluation of our proposed HW/SW co-designed image registration application on a state-of-the-art FPGA based SoC showcases its ability to outperform software designs leading to orders of performance speedup (up to 67x) against embedded CPUs.

CCS CONCEPTS

• **Computing methodologies** → **Image processing**; • **Hardware** → **Hardware accelerators**; **Hardware-software codesign**.

KEYWORDS

Image Registration, Downhill Simplex, Affine Transformation, Correlation Similarity Metric, Zynq

1 INTRODUCTION

Computing systems have invaded all aspects of human life in an effort to automatize and optimize important laborious tasks. Concepts and architectures such as Internet of Things and Edge computing have risen the bar in both computation and communication requirements of embedded systems [21]. In parallel, a flood of data is produced in unprecedented high rates leading to a saturation of the efficiency of general purpose computing systems and eventually paving the way for the integration of versatile and specialized computing accelerators [10].

A well established area of computing systems intervention is the medical domain, where electronic devices have revolutionized the way that patients are examined and treated [22]. While initially electronic devices were utilized mainly for medical data acquisition and visualization, there is a gradual wave of adoption of more sophisticated analysis mechanisms able to offload the critical tasks of a doctor. Towards this direction, the sampled patient data must be efficiently analyzed and correlated within short time frames, in order to assist diagnosis and reduce hospitalization. A significant amount of such algorithmic techniques stems from the image processing domain, since medical imaging is a well-established and essential part of accurate diagnosis [7]. A key requirement in medical applications is the accuracy and dependability of the derived diagnosis, ergo sophisticated algorithms and high resolution image acquisition techniques are utilized.

The inherent trade-off of this design choice is the severe toll on the computation requirements of the developed applications in order to abide by the high standard requirements. Nevertheless, adequately fast response latency and cost effective designs are required to allow for the successful adoption of new, high-accuracy image processing algorithms. Consequently, the incorporation of accelerating computing sub-systems in medical devices is an appealing and effective design choice.

One of the most common acceleration techniques of modern computing systems is the integration of Field Programmable Gate Arrays (FPGAs), which bare the responsibility of executing the heavy tasks of the application. FPGAs can produce order of magnitude of acceleration, while preserving a low power consumption profile of the target computing systems. These features have established them as an acceleration alternative in the medical domain as presented in the works of [4, 5, 11, 15].

This work targets image processing on FPGA based systems, with emphasis on medical applications, utilizing computing platforms which integrate general purpose computing elements (CPUs) and FPGAs on the same System-on-Chip. This combination provides the flexibility for efficient HW/SW co-design, which if applied properly can get the most out of the FPGA sub-system. Most importantly, the design principles of this system are similar to the FPGA systems for large scale deployments (e.g. cloud based systems), thus enhancing the re-applicability and importance of our proposed design.

On the algorithmic side, we focus on image registration algorithms which is a fundamental task frequently encountered in image processing applications. It is used in computer vision [14], medical imaging [13, 16], remote sensing [2], military automatic target recognition [12] and satellite imaging [19].

Our methodology involves benchmarking and analysis of the sub-components of the image registration algorithmic pipeline in order (i) to optimize the algorithmic structure for our target dataset and (ii) to locate and accelerate the computational hotspots of the algorithm. This approach enables us to develop an accurate and efficient accelerator, which in comparison to the embedded ARM processors of our target system achieves a speedup of up to 67x.

The rest of the paper is organized as follows. Section 2 briefly introduces related works regarding FPGA utilization in medical applications. Section 3 introduces the fundamentals of image registration pipelines and outlines their major building blocks. Section 4 details the utilized building methodology and design internals of our proposed HW/SW co-designed system. Section 5, describes the experimental evaluation analysis of our proposed accelerator, while Section 6 concludes the manuscript.

2 RELATED WORK

Regarding the acceleration of registration methods using an FPGA device, the work in [6] presents an FPGA-based architecture for accelerated implementation of mutual information (MI)-based deformable registration. Reduction on the execution time from hours to a few minutes is achieved, thus making it usable and efficient in a clinical environment.

In addition, [23] presents a high performance FPGA-based direct affine image registration core. The system runs at 100MHz and achieves a registration speed of 82fps for 640x480 images, while a floating point Matlab implementation on a 2.4GHz Intel Core 2 Quad required 5 seconds per frame, thus succeeding a 400x speedup. [18] presents an innovative method for representing and exploring the hardware design space when mapping image registration algorithms onto configurable hardware. A rigid image registration application under real-time performance constraints is designed with 74MHz maximum operating frequency.

The goal in [11] was to accelerate the demanding computational parts of the magnetic resonance imaging (MRI). Based on a previous work, a partial volume estimation (PVE) algorithm on an FPGA was implemented. Compared to an Itanium 2 CPU, speedups of 2.5x for the probability density estimation and 9.4x for the prior information estimation was measured.

Similarly, [5] explored the benefits of heterogeneous architectures that include FPGAs and GPUs, in order to meet the real-time execution requirements of an adaptive beamforming algorithm. They implemented the minimum-variance (MV) adaptive beamforming algorithm using the NVIDIA GTX 480 GPU and the ML605 FPGA board, reaching speedups of up to 38x and 39x, respectively.

3 IMAGE REGISTRATION

Image registration refers to the mapping between two images, both spatially and with respect to the intensity. It is used to match two or more pictures taken at a different time, or from different sensors/viewpoints [3]. One of the images is the fixed/source, while the others are the moving/targets. The reference frame in the fixed image is stationary, and the moving image is spatially transformed to match the target.

Image registration algorithms can be divided in two categories: intensity based and feature based [8]. Intensity-based methods compare intensity patterns within the images via correlation or error metrics. A pair of images, a similarity metric, an optimizer, and a geometrical transformation must be specified. Feature-based methods find correspondences between image features (such as points, lines and contours) and utilize them to estimate a global transformation which is finally applied to the moving image in order to align it.

In general, image registration can be described as an iterative trial-and-error procedure. A measure of similarity or distance is computed between the images at each iteration and used to determine if they are sufficiently aligned. This process is controlled by the optimizer starting from an initial state and determining subsequent steps to reach an optimal alignment. A brief presentation of each of the three main steps is presented below.

Measure of match: Image similarity metrics are methods used to calculate a quantitative evaluation of the similarity between two images. They act as a base in all registration applications, since they enable the optimizer to progress in the search of the optimum transformation. Commonly used measures include: *Sum of Absolute/Squared Differences, Correlation Coefficient, Matte's Mutual Information, Gradient Correlation* [20].

Two similarity metrics were examined in this study: the *Correlation Coefficient* and *Matte's Mutual Information*. The Correlation Coefficient provides a measure of the linear relationship between the two images. Matte's Mutual Information is a method that measures the amount of information that can be obtained for one of them by observing the other.

Transformer: The transformer maps points of the moving image to new locations in the transformed image. According to the registration problem the transformer can be either collinear or deformable. Collinear transformations are the *rigid*, the *affine* and the *projection*. The rigid transformation includes translation and rotation only, whereas the affine transformation also includes scaling and shearing [9]. This work uses an affine transformation around the center of the image.

Optimizer: The optimizer defines an efficient and often non-exhaustive strategy to search the allowed transformation space for the best match between the images. An optimizer can be categorized as gradient-based or gradient-free, global or local [20].

In this study the *Downhill Simplex* and the *Powell's Direction Method*, both of which are classified as gradient-free and local [20] are examined. Other methods include: *Gradient-Descent*, *Soblex*,

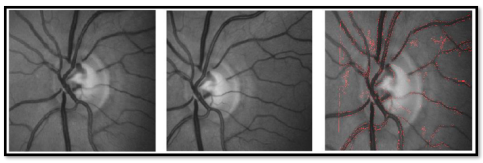


Figure 1: Registration example using the iris dataset

Simulated Annealing, etc. The Downhill Simplex utilizes the concept of the simplex, which consists of N+1 points in a N-dimensional space [17]. After initialization, the optimizer follows a series of steps that aim in moving the highest points of the simplex to lower ones. Convergence is usually achieved after multiple contractions of the simplex. The process terminates when the distance of the vector moved in two subsequent steps is below a threshold. Powell's is a direction method that produces N mutually conjugate directions. This method practically searches for a multidimensional function's minimum or maximum by moving along set directions.

4 PROPOSED HARDWARE ACCELERATOR

This section presents the HW/SW co-designed system that was developed. An exploration is necessary to determine the best combination of the optimizer and similarity measure. To achieve that, we utilized a dataset of gray-scale photographs of the iris of different eyes acquired by high resolution cameras (image size: 1000x1000). Registration of these eyes can have security or medical purposes [13]. Figure 1 illustrates an image registration example on the utilized dataset. The first two images correspond to different photographs of the same eye. The left one is the moving image, while the second the fixed one. The registration result is depicted in the third image, where the moving image has been optimally transformed to be aligned to the fixed. An edge detection and an image fusion have also been applied for a visual representation.

4.1 Evaluating the optimizer and similarity measure methods

As explained in Section 3, the *Downhill Simplex method* and *Powell's Direction method* are considered as the optimizing procedure, while for the similarity measure the *Correlation Coefficient* and *Matte's Mutual Information* are the methods under evaluation. Thus, four different combinations are derived and examined in order to choose the final configuration of the image registration application. The solution quality provided by these methods directly affects the registration's performance. For the dataset of the application, it was concluded that all four models had the same success rate.

The affine transformation parameters and their maximum allowed values also influence performance. We experimentally concluded that typical values of 30°for rotation and 10% for scaling are sufficient for our target dataset. Conversely, the maximum value of displacement is what affects the alignment process the most.

Table 1 presents how the similarity measure is affected by the maximum displacement value and the different configurations. All methods achieve a higher similarity (and consequently a better alignment), when the maximum displacement is increased to a value that can sufficiently express the required geometric transformation.

Table 1: Computed similarity measure for various maximum displacement. Scaling and rotation are fixed at 10% and 30° respectively.

Maximum Disp.	Downhill Simplex		Powell's Direction	
	Correlation	Mutual	Correlation	Mutual
50	0.157844	0.261021	0.227008	0.285423
100	0.231213	0.374724	0.487449	0.385665
200	0.783975	0.788725	0.784077	0.788861
300	0.784035	0.789102	0.784041	0.789113

Therefore, both optimizers and both similarity metrics noted similar performance, provided that they were equally initialized. In total, the chosen solver for the hardware acceleration makes use of the *Downhill Simplex Optimizer, the Affine Transformation Model and the Correlation Similarity Metric*. The aforementioned components were preferred mainly due to their popularity and conceptual simplicity.

Figure 2 illustrates a high-level representation of the processing flow for a general image registration solver. The dashed boxes are optional steps, which were not necessary in this work, while the boxes with solid contours are the ones that are part of our design. The flow of the application is as follows:

Step 1: Images are loaded either from non-volatile memory or the moving image is dynamically captured from an imaging device and the fixed image is extracted from a database.

Step 2: N+1 initial points are calculated. These points are required by the Downhill Simplex method and cover the whole range of the exploration space. For each of them an affine transformation is applied and the respective similarity measure between the fixed and the moving images is computed, thus creating the initial simplex in question.

Step 3: Starting from the initialization of Step 2, the optimizer begins the evaluation process of the termination criteria. If they are satisfied then a valid transformation has been found and the application moves on to the post-processing stages. If the termination criteria are not satisfied, then Step 4 is executed.

Step 4: The state of the transformation parameters are optimized and a new candidate transformation is determined. The corresponding similarity measure is computed and the optimizer again decides upon the completion or not of the registration procedure, by returning to Step 3.

4.2 Profiling of Image registration algorithm

The registration solver has been developed using the C programming language. A further profiling procedure was performed to

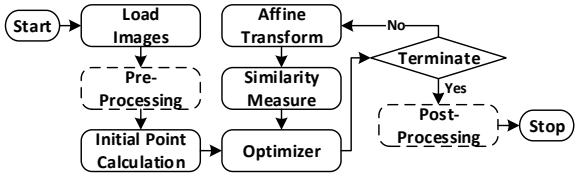


Figure 2: Processing flow of a general registration solver.

Table 2: Profiling of registration for two pairs of eye images.

	Total Execution Latency (sec)	Transform & Measure Execution Latency (sec)
Matching pair	39.923572	39.9216161
Differing pair	103.084555	103.080554

identify parts of the application that are computationally intensive, their memory requirements, as well as the type of operations involved. Besides its structure, the execution latency of the examined application is affected by the following parameters:

- *Image size*: As the image size becomes larger, the required latency to process it increases, becoming a serious bottleneck especially for image processing operations that are difficult to apply numerous pixels concurrently.
- *Tolerance threshold*: The threshold for determining a successful registration is a trade-off between accuracy and the required computations of the application.
- *Iteration number*: In case the optimizer is unable to converge to a solution (e.g. very low threshold), after a predefined number of iterations the registration process is stopped. In this case, the results may not be optimal and a post-processing step must be taken. This value was set to maximum 500 allowed iterations.
- *Maximum displacement/rotation/scaling*: The initial transformation parameters are directly affected by the maximum allowed values of these parameters. Certain transformations may be rejected if the computed transformed coordinates lay out of the image boundaries. The maximum rotation is fixed at 30°, scaling at 10% and displacement at 300 pixels (Section 4.1).

The initial profiling was performed on the 1000x1000 images. Moreover, to decouple the measured profile of the application from the properties of the examined pairs, we examine both cases of successful and unsuccessful registrations. Table 2 presents the measured latency of such an example, where there is a pair of matching input images (Matching pair) and a pair of non matching images (Differing pair) of the input dataset. The Table includes the total execution latency of the application, as well as the execution latency of its Transform & Measure function. The latency of this function requires almost 99% of the total latency in both cases. Consequently, this is the algorithmic part that will be the focus of our accelerated design on the FPGA fabric.

Based on the profiling, a high-level HW/SW partitioning of the registration solver is provided in Figure 3. The less computationally demanding parts, such as loading the images, tuning the transformation parameters and evaluating the termination criteria remain as software components, while the application of the candidate transformation to the pixel coordinates and the computation of the similarity measure are implemented on the FPGA.

4.3 Proposed HW accelerator architecture

The basic sub-modules of the implemented hardware accelerator are: (i) *the Transform*, which applies the affine transformation on every pixel. This unit takes as input the pixel coordinates and the

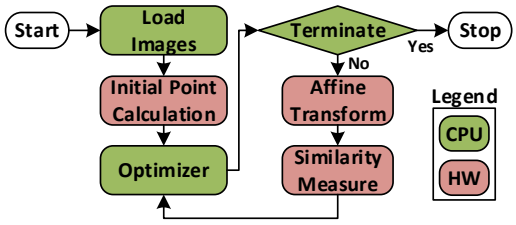


Figure 3: HW/SW partitioning of the image registration.

current transformation parameters and outputs the transformed coordinates, partitioned in their decimal and fractional parts, (ii) *the Interpolation Weights*, that generates the interpolation weights used for computing the final intensity value of the transformed pixel. It accepts the fractional parts of the transformed coordinates from the Transformation unit and produces the appropriate interpolation weights, (iii) *the Interpolation Calculation*, where the final intensity of the transformed pixel is computed. The inputs of this module are the weights and intensity values of the three pixels neighboring the transformed pixel, as produced on the Interpolation Weights unit, in addition to the weight and intensity value of the transformed pixel, (iv) *the Accumulations*, which comprise a number of accumulation units responsible for computing the sums needed for the final calculation of the correlation between the images. Its inputs are the intensity value of the interpolated pixel, coming from the Interpolation Calculation unit, and the corresponding intensity value of the pixel on the moving image.

Additionally, the hardware accelerator uses two memory modules for image storage purposes and a number of control units to synchronize its operation. These controllers are responsible for generating the pixel coordinates, managing memory accesses, delaying data and regulating the communication with the software application. The final architecture of the developed hardware accelerator is given in Figure 4.

4.3.1 HW/SW communication. In image processing applications one of the main design challenges is the efficient communication between the processing elements and the memory, so as to maintain a high throughput and low latency data transfers. To meet these requirements a Direct-Memory Access (DMA) based solution with streaming capabilities must be used. The streaming feature

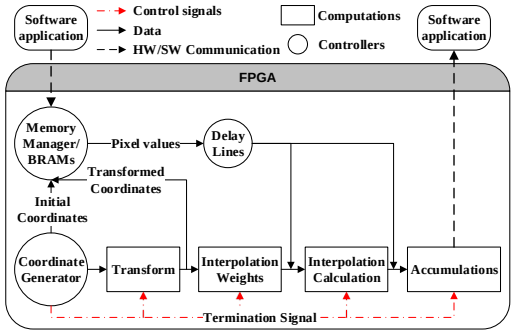


Figure 4: HW accelerator architecture.

will cover the throughput constraint and the DMA will provide direct access to the memory for the processing elements. The target platform offers these features through the use of the AXI communication infrastructure. Thus, the hardware accelerator adopts the AXI4-Stream for high-bandwidth image data transfers, while the AXI4-Lite is used for low-bandwidth communication.

4.3.2 Memory bottleneck. A common problem encountered in image processing applications deployed on FPGA devices is the limited BRAM resources to store image pixels for processing. To overcome this problem a common approach used is the sliding window technique [1]. The image registration application utilizes two images, the fixed one and the moving one. On the moving image, where the transformation is applied, the new pixel coordinates follow an irregular pattern, limiting the ability to apply this technique or find an efficient way to access the required pixel values. To bypass this problem, an offline preliminary step that downsizes the images from 1000x1000 to 256x256 was introduced. After ensuring that the quality of the results is not negatively affected by this strategy, it is now possible to fit both images in the BRAMS of the FPGA, allowing for a straightforward memory-mapped access.

5 EXPERIMENTAL EVALUATION

5.1 Experimental setup

The evaluation targets the Zybo development board, featuring the xc7z010 part of the Zynq-7000 SoC family devices. The Zynq platform consists of the Processing System (PS), hosting a dual-core ARM Cortex-A9 processor, and the Programmable Logic (PL) which refers to the FPGA fabric of the device. The HW/SW co-designed application is compared against software implementations executed on an Intel i5 4200U CPU and on the ARM CPU of the Zynq PS.

Each implementation is compared based on the quality of the results, which takes into consideration the calculated optimum transformation parameters and the similarity measure. Moreover, their execution latency is measured and the hardware cost in terms of FPGA resources is given. In order to have fair comparisons between the platforms all the measurements are performed after the dataset images are downscaled to 256x256 pixels in size in order to meet the memory constraint of the FPGA device (Subsection 4.3.2).

5.2 Evaluation of transformation parameters

Our first experiments aim at providing quantified evidence that the hardware accelerator provides accurate results with respect to the reference software application. The six affine transformation parameters (2 for rotation, 2 for scaling and 2 for displacement) and the final similarity measure are used as quality metrics, with the software implementation on Intel i5 serving as reference.

Table 3 gives the parameter values of the final transformation to be applied on the moving image, as well as the final similarity measure computed on the three different platforms. All the computations were performed with single precision floating-point arithmetic in order to have an initial estimate of the migration of some software functions to the FPGA. As can be seen, the values of the transformation parameters differ slightly, but the difference in the context of the specific test case is acceptable.

Table 3: Final affine transformation parameters and similarity measure on three different platforms.

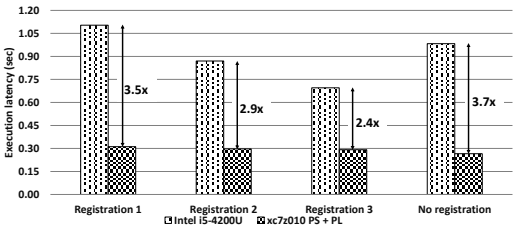
Transformation Parameters	Intel i5 4200U	xc7z010 PS	xc7z010 PS+PL
T1	1.002974	1.010045	0.990064
T2	-0.015803	-0.015795	-0.017116
T3	45.320175	45.801357	44.415871
T4	0.024869	0.027911	0.021519
T5	0.997309	1.003294	0.989229
T6	-16.065662	-15.859092	-15.909966
Measure of Match	0.792186	0.792874	0.780669

5.3 Co-designed system execution latency

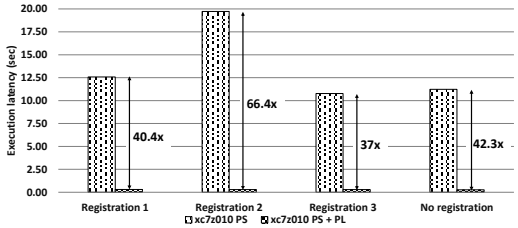
Having validated that the computed transformation parameters have an acceptable deviation from the reference and that the final similarity measure is acceptably close in all platforms, the execution latency of the proposed HW/SW co-designed system is compared against the software only implementations on Intel i5 4200U and on ARM Cortex-A9 (Zynq PS). The comparisons are performed for a single-threaded C code execution on all platforms. The operating frequencies of CPUs were preserved at their base supported frequency, i.e. 650 MHz for the ARM Cortex-A9 CPU and 1.6 GHz for the Intel i5 CPU. The FPGA accelerator operated at 100MHz as reported by the synthesis tool. Also, note that for the following experiments the downsized images were used.

Figure 5a shows the measured average execution latency over 100 runs of the co-designed application versus the software only application on Intel i5 for four different sets of eye images. The first three out of which lead to successful registration and the last one fails. The achieved execution speedup is also annotated on the figure. In all cases, there is a decrease in execution latency and the average achieved speedup over the four test cases is $\sim$ 3x. The variable execution latency stems from the fact that the termination criteria is directly related to (i) the maximum number of iterations allowed on the optimizer, (ii) the tolerance threshold of the optimizer and (iii) on the nature of the images used. Small deviations in the parameters generated at the end of an iteration may lead to a varying convergence trajectory, which is caused by a difference in the optimizing steps.

The experiment is repeated in a fully embedded execution environment and the obtained results can be seen in Figure 5b, which shows the execution latency of the presented design versus the software only implementation running on ARM Cortex-A9 (Zynq PS). It is clearly evident that there is a considerable decrease in execution latency in all four cases and the average speedup achieved is $\sim$ 46x. Again, observed fluctuation in the execution latency is due to the variable nature of the termination criteria of the optimizer. Nevertheless, the HW/SW co-designed application outperforms both software-only implementations, despite the use of single-precision floating-point implementation.



(a) Comparison against Intel i5-4200U @ 1.6 GHz



(b) Comparison against ARM A9 @ 650 MHz

Figure 5: Performance evaluation of the proposed design.

Table 4: Resource utilization on xc7z010 Zynq device.

xc7z010 PL Resources	Available	Used Resources	
		Accelerator	Communication
LUTs	17600	11396 (65%)	2768 (16%)
DFFs	35200	19965 (57%)	2656 (8%)
BRAMs	60	34 (57%)	1 (2%)
DSPs	80	76 (95%)	0 (0%)

5.4 FPGA resource utilization

Table 4 presents the resource utilization of the final HW architecture. Even though the co-designed application uses single-precision floating-point arithmetic, it can still fit on the FPGA device, despite the high DSP utilization (95%). However, the high DSP utilization presents a major bottleneck for the implemented application, as the parallelization of the application in the FPGA becomes prohibitive. Moreover, the majority of the FPGA resources have been allocated for computation and not communication. Regarding memory resources requirements, the utilization of BRAMs is around 60%, which allows for the prospect on-chip analysis of images of higher dimensions.

6 CONCLUSIONS

This paper presented the development of a HW/SW co-designed application and the deployment on a SoC-FPGA device for an image registration pipeline. The developed application was successfully mapped to one of the smallest Zynq devices, while incorporating single-precision floating-point arithmetic. Moreover, the quality of the proposed design was experimentally verified against pure software implementations showing high performance gains, while preserving algorithmic accuracy.

ACKNOWLEDGMENTS

The work in this paper has been partially funded by the European Union's Horizon 2020 research and innovation programme, under project SDK4ED, grant agreement No 780572.

REFERENCES

[1] Donald G. Bailey. 2011. *Design for Embedded Image Processing on FPGAs* (1st ed.). Wiley Publishing.
[2] Youcef Bentoutou, Nasreddine Taleb, Kidiyo Kpalma, and Joseph Ronsin. 2005. An automatic image registration for applications in remote sensing. *IEEE transactions on geoscience and remote sensing* 43, 9 (2005), 2127–2137.
[3] Lisa Gottesfeld Brown. 1992. A survey of image registration techniques. *ACM computing surveys (CSUR)* 24, 4 (1992), 325–376.
[4] Carlos R Castro-Pareja, Jogikal M Jagadeesh, and Raj Shekhar. 2003. FAIR: a hardware architecture for real-time 3-D image registration. *IEEE Transactions on Information Technology in Biomedicine* 7, 4 (2003), 426–434.
[5] Junying Chen, Alfred CH Yu, and Hayden K-H So. 2012. Design considerations of real-time adaptive beamformer for medical ultrasound research using FPGA and GPU. In *Field-Programmable Technology (FPT), 2012 International Conference on*. IEEE, 198–205.
[6] Omkar Dandekar and Raj Shekhar. 2007. FPGA-accelerated deformable image registration for improved target-delineation during CT-guided interventions. *IEEE Transactions on Biomedical Circuits and Systems* 1, 2 (2007), 116–127.
[7] Thomas Martin Deserno. 2011. *Biomedical image processing*. Springer Science & Business Media.
[8] Arthur Ardeshir Goshtasby. 2005. *2-D and 3-D image registration: for medical, remote sensing, and industrial applications*. John Wiley & Sons.
[9] John F Hughes, Andries Van Dam, James D Foley, Morgan McGuire, Steven K Feiner, David F Sklar, and Kurt Akeley. 2014. *Computer graphics: principles and practice*. Pearson Education.
[10] Christoforos Kachris and Dimitrios Soudris. 2016. A survey on reconfigurable accelerators for cloud computing. In *Field Programmable Logic and Applications (FPL), 2016 26th International Conference on*. IEEE, 1–10.
[11] Jahyun J Koo, Alan C Evans, and Warren J Gross. 2009. 3-D brain MRI tissue classification on FPGAs. *IEEE Transactions on Image Processing* 18, 12 (2009), 2735–2746.
[12] Igor V Maslov. 2004. Automatic image registration and target recognition with multiresolution hybrid evolutionary algorithm. In *Signal Processing, Sensor Fusion, and Target Recognition XIII*, Vol. 5429. International Society for Optics and Photonics, 180–188.
[13] George K Matsopoulos, Nicolaos A Mouravliansky, Konstantinos K Delibasis, and Konstantina S Nikita. 1999. Automatic retinal image registration scheme using global optimization techniques. *IEEE Transactions on Information Technology in Biomedicine* 3, 1 (1999), 47–60.
[14] Ntana Nkanza. 2005. *Image registration and its application to computer vision: mosaicing and independant motion detection*. Ph.D. Dissertation. University of Cape Town.
[15] Mohd Fauzi Bin Othman, Norarmalina Abdullah, and Nur Aizudin Bin Ahmad Rusli. 2010. An overview of MRI brain classification using FPGA implementation. In *Industrial Electronics & Applications (ISIEA), 2010 IEEE Symposium on*. IEEE, 623–628.
[16] Hanchuan Peng, Phuong Chung, Fuhui Long, Lei Qu, Arnim Jenett, Andrew M Seeds, Eugene W Myers, and Julie H Simpson. 2011. BrainAligner: 3D registration atlases of Drosophila brains. *Nature methods* 8, 6 (2011), 493.
[17] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. 2007. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.
[18] Mainak Sen, Yashwanth Hemaraj, William Plishker, Raj Shekhar, and Shuvra S Bhattacharyya. 2008. Model-based mapping of reconfigurable image registration on FPGA platforms. *Journal of Real-Time Image Processing* 3, 3 (2008), 149–162.
[19] Utsavkumar S Shah and Darshana Mistry. 2014. Survey of image registration techniques for satellite images. *International Journal for Scientific Research & Development* 1, 11 (2014), 2321–0613.
[20] Ramtin Shams, Parastoo Sadeghi, Rodney A Kennedy, and Richard I Hartley. 2010. A survey of medical image registration on multicore and the GPU. *IEEE signal processing magazine* 27, 2 (2010), 50–60.
[21] John A Stankovic. 2014. Research directions for the internet of things. *IEEE Internet of Things Journal* 1, 1 (2014), 3–9.
[22] Eric J Topol and Dick Hill. 2012. *The creative destruction of medicine: How the digital revolution will create better health care*. Basic Books New York.
[23] Brandyn A White. 2009. Using FPGAs to perform embedded image registration. *BSc. Major Thesis, Computer Engineering, University of Central Florida* (2009).