Towards an AI Engine-based library for similarity metrics computation

Davide Ettori, Federico Mansutti

June, 30 2024

POLITECNICO
MILANO 1863

POLITECNICO MILANO 1863
NECST laboratory

Abstract

Similarity metric computation is one of the most compute-intensive steps in several procedures related to Machine Learning, image analysis, Computer Vision, and many others. Among these applications, image registration stands as pivotal. Employed in medical imaging and robotics, it uses similarity metrics to guide the heuristic registration algorithm. In this context, Cross-Correlation (CC), Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), Squared Cross-Correlation (SCC) and Root Mean Squared Error (RMSE) are promising, opening to high-performant solutions without significant accuracy drops. This project focuses on tailoring CC, MSE, PSNR, SCC and RMSE procedures to unleash the benefits of cutting-edge Versal technologies. While those metrics have been implemented with varying degree of success on FPGA architectures, this is the first implementation, to the best of our knowledge, of MSE, CC, PSNR, SCC and RMSE acceleration on a Versal machine with AI Engine. We present an AI Engine-based library for Similarity Metric Computation (AIESMC) for both 2D and 3D images. AIESMC is able to generate various accelerators, letting the user choose the metric to use and the number of kernels on the AI Engine. By applying vector operations, map-reduce, axi burst and multiple kernels, a speedup over 5x was achieved compared to the SW version. The stability is also improved, by orders of magnitude. Anyway, it's not only limited to this application and it can be useful whenever we need a fast and reliable way of computing similarity metrics

1 Introduction

In the field of imaging, the core concept behind numerous image enhancement and data integration techniques involves capturing multiple images and merging their contents to derive more detailed information. A notable example of this is a surgical navigation system, which assists surgeons by providing image-based guidance during procedures. This system necessitates the integration of preoperative images with those taken during the surgery. However, without properly aligning these images to the same reference system, the resulting fusion can yield poor or inaccurate results. Consequently, many methods utilize Image Registration (IR), a well-established paradigm that aligns one or more images captured under different conditions to a common reference. Traditional IR consists of three primary components: a geometric transformation, an optimizer that seeks the optimal transformation parameters, and a similarity metric that measures the accuracy of the transformation. The main focus of this work is towards the **similarity metrics**, since it's one of the bottlenecks of the entire IR pipeline [3]. In particular we present an AI Engine-based library for Similarity Metric Computations (AIESMC). While our initial goal was just to provide an accelerator for MSE and CC, the scope of the project has expanded to prove that a wide range of similarity metrics can benefit from being implemented on a VLIW architecture. The result is an efficient and usable library that accelerates MSE, CC, PSNR, SCC and RMSE similarity metrics. The main advantages of AIESMC are versatility and customizability, in fact, it is designed to work with both 2D and 3D images, having square or rectangular shape. It's equipped with a tool that generates and configures the project files required to build and run on Versal and is customizable on the metric type, the size of the images (width, height and depth) and in the number of AIE kernels. It's thought to be usable even by people without a very specific knowledge of FPGA, since the provided interface is intuitive, and fully automated.

1.1 Context

AIESMC is designed to be inserted into an IR pipeline [3] or [6]. The key idea is to use AIESMC as a component which takes in input 2 images (2D or 3D) an returns, as output, the floating point value of the chosen similarity metric. The size of the images should be passed as a parameters, before the value of the pixels, but the 2 images must have the same size. This means we are supposing that the rotations and translations necessary for IR are performed by the other components of the pipeline, as in the FABER toolchain [3]

The literature regarding the acceleration of similarity metrics using FPGA-based system is not expansive, and even more so when targeting VLIW architecture and Versal-

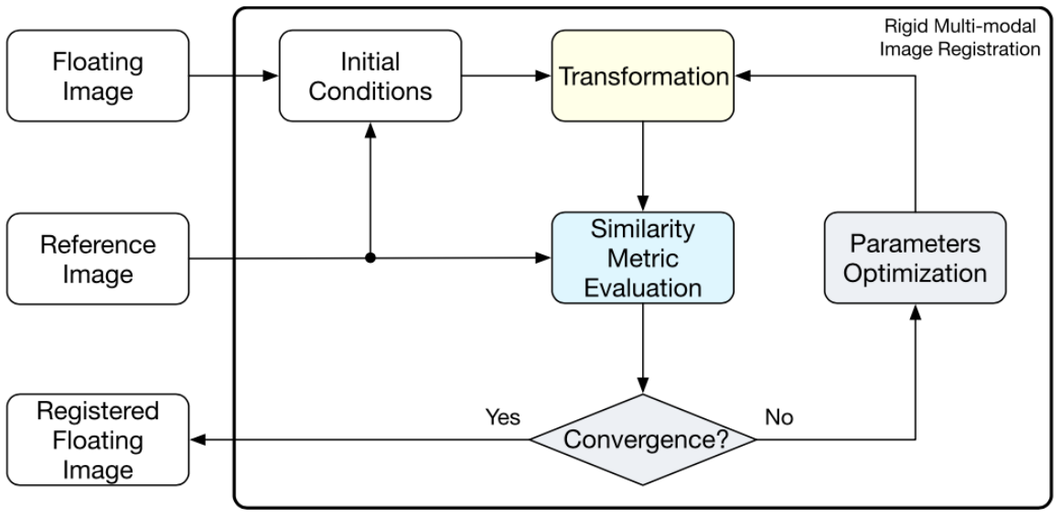

Figure 1: General IR pipeline [3]

based systems. Our work is meant as a proof of concept, useful to expand and explore this promising possibility, analyzing 5 important metrics.

1.2 Architectural Choice

AIESMC is implemented on a Versal machine, equipped with AI Engine. This choice is driven by the fact that, being a VLIW architecture it can execute multiple basic operations (addition, subtraction, multiplication, ...) in parallel. This is done by fetching a Very Long Instruction, composed by smaller operations scheduled by the compiler. This allows for out-of-order execution while keeping the in-order issue of each single operation. Obviously, since this belongs to the static optimization family it relies on the compiler to find potential parallelism and maximize ILP (Instruction Level Parallelism) by code motion, loop unrolling and software pipelining. Advantages are: a relatively simple Hardware, easy to increase number of Function Units and good loop level parallelism. Disadvantages are: a huge number of registers, large data transport and increased code size. Since there are plenty of parallelizable, repetitive and independent operations in similarity metrics computation, Versal architecture is very promising and thus examined in this work.

1.3 Related Work

1.3.1 Mean Squared Error and Root Mean Squared Error

The metric is defined as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Ref_i - Float_i)^2 \tag{1}$$

$$RMSE = \sqrt{MSE} \tag{2}$$

where:

- n is the number of data points,
- Ref_i is the value of the pixel in reference image,
- $Float_i$ is the value of the pixel in floating image

The Mean Squared Error is one of the most explored and studied metrics in literature due to its simplicity and intuitive meaning. It's characterized by fast computation and large capture radius with the main downsides being translation/rotation/scaling variant and the inability to assign different weights to different pixels. The RMSE is often also considered as it has the same units as the target. For example, the MSE is in pixels squared while the RMSE is in pixels. It also penalizes differences in pixels less severly than its counterpart.

While the MSE is often studied alone, we are not aware of any significant work on MSE acceleration on FPGA and image registration, due to its inability to respond accurately to image transformation. One approach found in literature is GPU based [2] which uses SIMD (Single Instruction Multiple Data) approach to accelerate the computation by dividing the image in smaller chunks and combining the results afterwards. This is the same approach as the map-reduce implemented in AIESMC.

1.3.2 Peak Signal-to-Noise Ratio

The metric is defined as follows:

$$\text{PSNR} = 10 \log_{10} \left(\frac{\max^2}{\text{mse}} \right) \tag{3}$$

where:

- max is the maximum pixel value of the images (255 for gray-scale),
- mse is the mean squared error of the images

PSNR correlates better with human visual perception than MSE because it accounts for the logarithmic sensitivity of human vision to changes in brightness. PSNR also normalizes the MSE by the maximum possible pixel value, allowing for easier comparison across different image types and scales.
The PSNR is shown in the work [5] and here, it is explained that PSNR is commonly used to evaluate image reconstruction quality: a value between 30 and 50 denote high fidelity, while a value over 50 says the results are nearly identical, on the other hand values below 10 indicate a huge difference in the images. To the best of our knowledge, there has been no implementation in literature on both FPGA systems and Versal.

1.3.3 Cross-Correlation and Squared Cross-Correlation

The metric is defined as follows:

$$CC = \frac{\sum_{i=1}^{n} Ref_i * Float_i}{\sqrt{\sum_{i=1}^{n} Ref_i^2 \sum_{i=1}^{n} Float_i^2}} \tag{4}$$

$$SCC = CC^2 \tag{5}$$

where:

- Ref_i and $Float_i$ are the values of the pixels of the images,
- n is the number of data points.

This metric has a limited capture radius [-1, 1] and it's invariant to linear scaling and shifting. The latter is a very desirable property in Image Registration as it allows image comparison between not only transformed images but also different modalities. AIESMC also implements the squared version (SCC, sometimes called Coefficient of Determination) of CC, since in IR it is often analyzed whether there exists a correlation, independently of its sign. In this case the radius is [0, 1].
This metric is the most discussed in literature and it has been often accelerated on FPGA systems. Most of the state of the art tends to focus on the specific case where the floating image is smaller than the reference image and optimizing by batch, sliding the floating image on top of the reference image. This optimization is done by saving sums and squared sums of pixels between different computations of CC [8]. Other implementations include employing convolutional structures [4] and the Fourier transforms [9]. While these approaches are noteworthy, AIESMC is a library meant to be used in an image registration pipeline and output a single similarity, hence we employed a different approach.

2 Implementation of AIESMC

2.1 Data types

Since pixels range from 0 to 255, the smallest data type is the unsigned 8 bit integer. Its implementation significantly speeds up data transfers to main memory and the input streams to the kernel. It's implementation, however, has the main disadvantage of requiring conversion to a 32 bit integer for add, subtract and multiply operations which can be costly in terms of performance.

The architectural choice for AIESMC is the unsigned 8 bit integer as this disadvantage can be mostly offset by increasing kernel count to allow for further parallelization. The unsigned 8 bit integer was grouped into chunks of 16 forming arbitrary precision unsigned integers of 128 bits. This grouping allows better synchronization and transfer time between the processing units.

2.2 Vector Operations

The AI Engine of Xilinx's Versal platform excels in accelerating complex vector operations crucial for high-performance computing tasks. These engines employ a highly parallelized architecture capable of performing numerous simultaneous calculations, significantly enhancing the processing speed of vector and matrix computations. By leveraging specialized vector processors, the AI Engine can efficiently execute operations such as additions, multiplications and dot products completely in parallel; as explained in [10]. We used those vector operation extensively in our AI Egnine kernel, since all the metrics considered are defined as a series of repetitive operations on distinct elements. This configuration is a typical example of a computation that can be accelerated by vectorized operations, hence AIESMC calculations are parallelized on vectors of 32 elements.

2.3 Compiler Optimizations

The AI Engine code is enhanced by *pragmas*, which are used to guide the compiler in optimizing the code. The first one is *chess loop range*, useful to avoid checking the loop conditions for the first N iterations (this is particularly convenient when the loop size is exactly known, wasting zero time to evaluate the branch condition). Finally, we made each computation equal and independent from the previous ones, so that we could apply software pipelining with *chess prepare for pipelining*. This instruct the compiler to overlap the execution of the loops, achieving an high level of parallelism, specifically on the VLIW machine.

2.4 Burst Transfer

Medical images can reach up to 5 megapixels (MP) for computed radiography (CR) and 10 MP for digital mammography [7]. To handle large amounts of pixels, the PL logic implements AXI BURST tranfers to aggregate memory accesses to the DDR to maximize the throughput bandwidth and minimize the latency. This improves the throughput by reading large chunks of data into a single request and, the larger the size of the data, the higher the throughput, which makes AIESMC especially suitable to handle large images. Specifically, AIESMC reads from memory in chunks of 1024 uint8, which corresponds to the max read burst length - 256 int.

Algorithm 1 burst PL logic for one kernel

```
globaldata:
    burst_count = number of burst requests
    burst_length = burst size

    buffer1 [burst_length]
    buffer2 [burst_length]

procedure:
    from i = 0 to burst_count:
        read request on the first input (image 1)
        read request on the second input (image 2)
```

```
12
13          from j = 0 to burst_length:
14              read from input 1 and store on buffer1
15              read from input 2 and store on buffer2
16
17          from j = 0 to burst_length:
18              write to output from buffer1
19              write to output from buffer2
20
21  end procedure
```

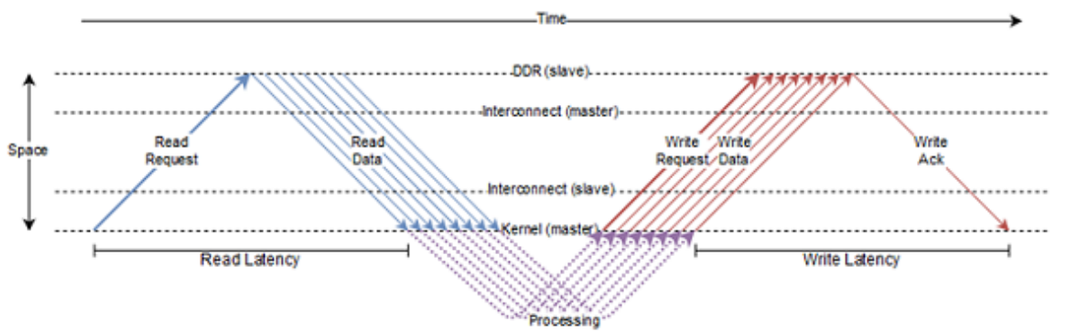

Figure 2: Burst transfers [1].

2.5 Map reduce

Each HW implementation of the five metrics exploits the map-reduce pattern which improves performance at the cost of resource usage. Currently, each metric supports execution on either 1 or 2 AI engines. Theoretically, our architecture can be pushed up to 32 kernels bringing increased performance. Each MSE and PSNR kernel calculates the MSE of its portion of the image and forwards it to the adder. In the case of the PSNR, a scalar logarithm is applied before the output.

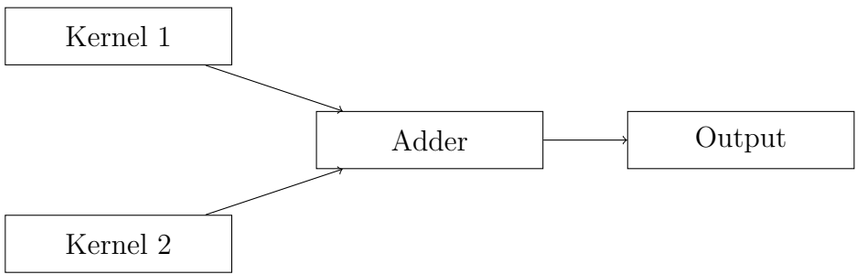

Figure 3: Map reduce for two kernels

The implementation for the CC follows the same paradigm of map-reduce shown above with some notable changes: each kernel passes the product of reference and floating pixels (numerator) and the sum of the reference and floating pixels squared (denominator) eq. (4). This was a necessary architectural choice derived from the difference in denominators calculated in each kernel.

3 Experimental Setup

We built AIESMC using the Vitis environment, employing diverse levels of parallelization on various AI Engines. First of all, we simulated our kernels on a couple of architectures (x86 and VLIW) thanks to the Vitis software and finally we compiled the templates and tested our results on the AMD VCK5000 Versal made available by the HACC cluster at ETH Zurich.

3.1 Architecture

AI Engines are architected as 2D arrays consisting of multiple AI Engine tiles and allow for a very scalable solution across the Versal portfolio, ranging from 10s to 100s of AI Engines in a single device. The benefits includes: C programmable, compile

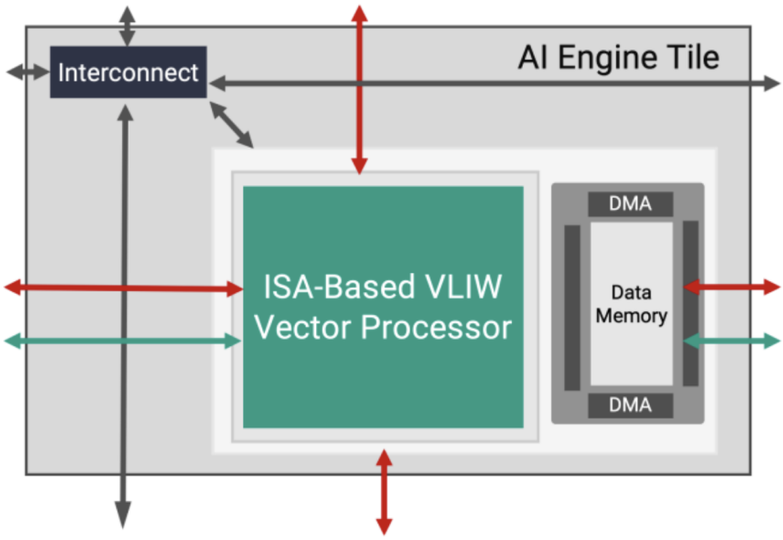

Figure 4: Single AI Engine unit [11]

in minutes, dedicated instruction and data memories and dedicated connectivity for scheduled data movement. The AI engine architecture is designed for efficency and it delivers up to 8X better silicon area compute density, reducing power consumption by nominally 40%. Each AI Engine tile consists of a VLIW, (Very Long Instruction Word), SIMD (Single Instruction Multiple Data) vector processor optimized for signal processing applications. The AI Engine processor can run up to 1.3GHz enabling very efficient, high throughput and low latency functions.

3.2 Board

The AMD VCK5000 Versal™ development card is built on the AMD 7nm Versal adaptive SoC architecture and is designed to optimize signal processing applications. Fully supported and compilable by Vitis™ and Vitis AI, the VCK5000 domain-specific architecture brings the performance of 400 AI Engines with energy efficiency of PL and high connectivity of Network on Chip, while keeping ease-of-use in mind with C/C++ software programmability.

3.3 Infrastructure

All our tests were run on the Heterogeneous Accelerated Compute Clusters (HACC) of ETH Zurich. HACCs are equipped with the latest AMD Xilinx hardware and software technologies for adaptive compute acceleration research. Each cluster is specially configured to enable academic teams to conduct state-of-the-art HPC research. In the following paragraph, AIESMC on AMD VCK5000 Versal is compared with an Intel Xeon Processor Cascadelake 8 core 3.3Ghz running a SW implementation of all the metrics.

4 Results

4.1 Method

We run batteries of 100 test, with 3D images, of size 512 x 512 x 10 (2.621.440), composed by random pixel in the interval [0, 255], inside the HACC by ETH Zurich. We measured the execution time of both the Hardware (Versal) and the Software (CPU), and compared them by Speedup (ratio of mean execution time) and F-ratio (ratio of the standard deviation of execution time). The Software data is always at the numerator of those ratio, while the Hardware data is at the denominator, this means: The higher, the better in both cases.

All the optimizations mentioned before were applied for the final version of each metric accelerator, this lead to an increasing of both compilation time and speedup. On the other hand, the F-ratio remains quite similar, indicating that the HW acceleration has been is very stable throughout all the optimizations.

4.2 Performance

The results obtained show a significant improvements in performance, from which the whole Image Registration pipeline can benefit. In the Table 1 we show the speedups and F-ratio considering 2 different type of HW time, to better understand the performance of the accelerator. In the first case, the total HW time to write input streams, compute the metric and read the output streams was considered. The second analysis also includes the time taken to load data into the board main memory and to load the the final output back (transfer time) allowing for a fairer comparison towards total SW time. This is important to distinguish, since the transfer time is not optimizable from the accelerator point of view, it represents a fixed cost that all accelerator must pay. **Following we present the results for the fully optimized AIESMC with images are of size 512 x 512 x 10.**

Obviously, when not considering transfer time the accelerator shows much better performance and that's because we mainly focus on big images, which are the most challenging when applying Image Registration. Hence the transfer time from/to the board is not trascurable, since we have a limited bandwidth which we might saturate. We are also providing some absolute data, which can be useful to immediately understand the performance of AIESMC when computing each metrics. Anyway, those cannot be directly compared between each other, since different metrics require different computations which can be very significant (CC requires much more operation than MSE, for instance). We can also observe that the presence of the logarithm in the PSNR is slightly influencing the performance, in a negative manner.

Metric	**Speedup**	**F-ratio**	**Time (ms)**	**STD (ms)**
MSE — RMSE				
Only Computation	4.5x	205	3.08	0.02
With Transfer	2.9x	12	5.12	0.8
CC — SCC				
Only Computation	3.9x	144	4.03	0.07
With Transfer	2.25x	15.2	6.13	0.67
PSNR				
Only Computation	4.05x	213	3.39	0.03
With Transfer	2.4x	12.4	5.37	0.56

Table 1: Performance Metrics Comparison on Optimal Range

Then, we also measured those performance with different sizes of images and it turns out that the optimal size is around 512 x 512 x 10, as we said before. This is a very relevant range since, for example, medical images for computed radiography and digital mammography are exactly in this range. RMSE and SCC performance are analogue to their counterpart, MSE and CC respecively.

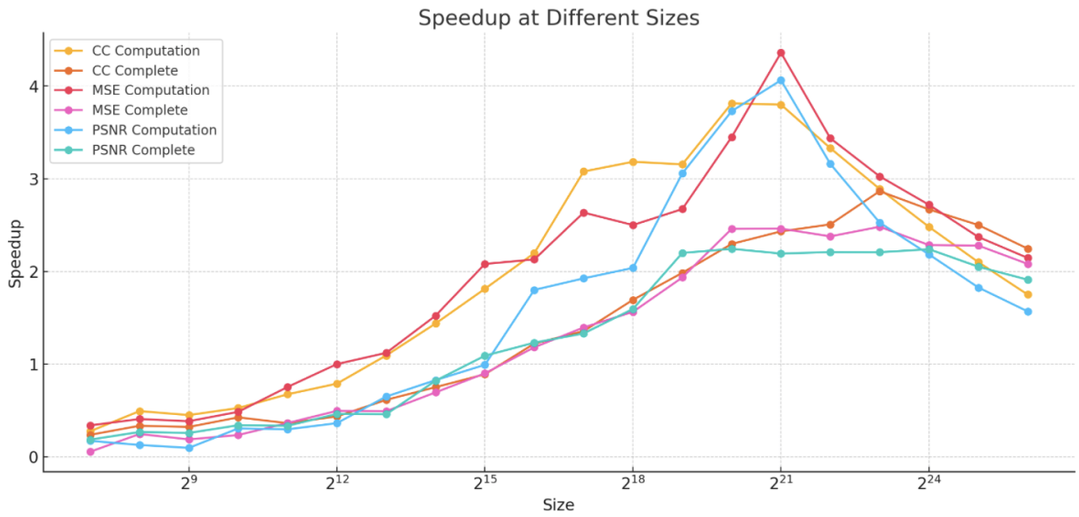

Figure 5: Speedup with different sizes
log scale on X axis

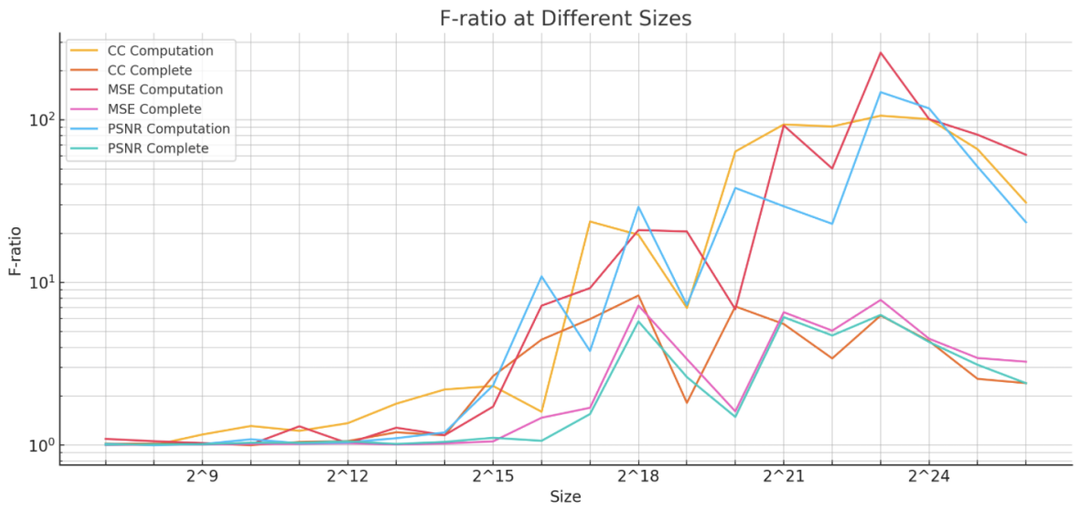

Figure 6: F-ratio with different sizes
log scale on both axis

4.3 Evaluation

fig. 5 shows that the speedup is quite smooth, providing the best performance in the range of medical images [7]. fig. 6, instead, depicts that the F-ratio is generally more noisy but is much greater in magnitude. This clearly indicates that AIESMC implementation is way more stable than the software counterpart, avoiding significant oscillation in execution times.

table 1 shows, the best performance is achieved by the MSE, which requires the least amount of computations. The PSNR has a similar behaviour, but the introduction of base-10 logarithm slows the execution, which cannot be avoided by parallelization. Moreover, even though cross-correlation is inherently the most demanding one, it's still faster than the SW counterpart computed with the CPU on the HACC.
Finally, we the standard deviation in execution time in versal is up to 200x lower compared to SW, as whosn by the F-ratio, implying that the HW version provided by AIESMC is much more stable, a very desirable behaviour especially for very intensive Image Registration tasks.

Concerning the size of the images, we see a peak of performance around 2^{21} (512 x 512 x 8) for the speedup and around 2^{23} (512 x 512 x 32) for the F-ratio; this is the range of most medical scans. Obviously, as images get extremely large the performance starts to decay as the bandwidth of the machine start to saturate from too many pixels.

5 Conclusion

We presented AIESMC, an AI engine-based library for accelerated computing of various similarity metrics. AIESMC is very flexible since it can automatically generate the templates for any supported combinations of parameters, while adapting the map-reduce pattern in the AIE part and the axi-burst code in the PL part. Currently, it supports 5 metrics (MSE, CC, PSNR, RMSE and SCC), different image dimensionality (1D, 2D or 3D) and multiple kernels (1 or 2). It shows significant speedup and improved stability in all those metrics. AIESMC is meant to be used in Image Registration pipeline to improve the similarity metric computation (up to 4.5x), which is often times one of the bottlenecks and using AIESMC makes the whole process faster and much more stable. Anyway, it's not only limited to this application and it can be useful whenever we need a fast and reliable way of computing similarity metrics. AIESMC shows that VLIW-based architecture such as the Versal AI Engine can be applied successfully to similarity metrics computation and acceleration, even though data transfer time remains one of the bottlenecks, a typical issue of HW accelerators.

References

[1] AMD. *Vitis High-Level Synthesis User Guide: AXI Burst Transfers*. 2022. URL: https://docs.amd.com/r/2022.2-English/ug1399-vitis-hls/AXI-Burst-Transfers.

[2] Peter Bui and Jay Brockman. "Performance Analysis of Accelerated Image Registration Using GPGPU". In: (Mar. 2009), pp. 38–45. DOI: 10.1145/1513895.1513900. URL: https://doi.org/10.1145/1513895.1513900.

[3] Eleonora D'Arnese et al. "Faber: A Hardware/SoftWare Toolchain for Image Registration". In: *IEEE Transactions on Parallel and Distributed Systems* 34.1 (Jan. 1, 2023). Published: 02 November 2022, pp. 291–303. ISSN: 1045-9219. DOI: 10.1109/TPDS.2022.3218898.

[4] Yufei Ma et al. "Optimizing the Convolution Operation to Accelerate Deep Neural Networks on FPGA". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26.7 (July 2018). Date of Publication: 03 April 2018, pp. 1354–1367. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2018.2815603.

[5] A. Obukhov and A. Kharlamov. *Dct8x8*. NVIDIA Software Development Kit (SDK). 2008.

[6] Mainak Sen et al. "RECONFIGURABLE IMAGE REGISTRATION ON FPGA PLATFORMS". In: (Nov. 2006). Date of Conference: 29 November 2006 - 01 December 2006, Date Added to IEEE Xplore: 15 August 2008. ISSN: 2163-4025. DOI: 10.1109/BIOCAS.2006.4600331.

[7] Shawn Wang. "Preparing effective medical illustrations for publication (Part 1): pixel-based image acquisition". In: *Biomedical Imaging and Intervention Journal* 4.1 (Jan. 2008). Epub 2008 Jan 1, e11. DOI: 10.2349/biij.4.1.e11.

[8] Xiaotao Wang and Xingbo Wang. "FPGA Based Parallel Architectures for Normalized Cross-Correlation". In: (Dec. 2009). Date Added to IEEE Xplore: 26 April 2010, pp. 26–28. DOI: 10.1109/ICISE.2009.603.

[9] Jason W. H. Wong, Caterina Durante, and Hugh M. Cartwright. "Application of fast Fourier transform cross-correlation for the alignment of large chromatographic and spectral datasets". In: *Analytical Chemistry* 77.17 (Sept. 2005), pp. 5655–5661. DOI: 10.1021/ac050619p.

[10] Xilinx Inc. *Xilinx AI Engine*. https://docs.amd.com/r/en-US/ug1079-ai-engine-kernel-coding/AI-Engine-API-Overview. Accessed: 2024-06-26.

[11] Xilinx Inc. *Xilinx AI Engine*. https://www.xilinx.com/products/technology/ai-engine.html. Accessed: 2024-06-26.