A Fast Approach for SAR Polarimetric Computations of Entropy/Alpha Angles

Amedeo Capozzoli, Claudio Curcio, Angelo Liseno, and Armando Marino

Abstract – We present an approach based on parallel computing on graphics processing units using the CUDA language for the evaluation of the Cloude and Pottier decomposition. The computation of eigenvalues and eigenvectors of the 3×3 covariance matrix is worked out by exploiting its Hermitian property by a "Cardano-like," direct analytical scheme. Numerical results show the benefits of the approach.

1. Introduction

Polarimetry [1, 2] is a widespread and assessed technique of synthetic aperture radar (SAR) due to its many applications, including agriculture monitoring [3], sea ice classification/characterization [4, 5], sea oil slick observation, [6] and target detection beneath foliage [7].

In 1996, Cloude and Pottier (C&P) [1] proposed to describe the polarimetric characteristics of natural scenes in terms of two parameters, namely, entropy and alpha angles, which account for the purity and type of the scattering mechanisms, respectively. Computationally, the decomposition consists of a pixelwise calculation of eigenvalues and eigenvectors of 3×3 covariance matrices. Although the calculation scheme appears to be simple if set up using library (e.g., Python, IDL) routines, the computation can be burdened for large SAR images, especially for the recently operating sensors, for which the final result can be obtained in long times. For example, in [8], approximately 250 TB of ScanSAR data were processed to produce forest maps with reliable information about the spatial and temporal behavior of tropical forests. Real-time SAR processing is a key technology in Earth observation, and improving the real-time SAR processing capability is a key issue [9]. Thus, achieving an adequate acceleration of polarimetric SAR methods appears to be timely and appealing.

Throughout the literature, a relevant effort has been devoted to acceleration of SAR processing in general [10, 11], many times reached by parallelization on graphics processing units (GPUs) [12–15]. However, the effort has been focused mainly on accelerating classical or advanced SAR image formation. Little has been done for classical SAR polarimetric schemes, as, even in recent contributions such as [16], the GPU has been adopted to speed up image formation and optimization.

Here, we introduce a computational approach based on parallel computing on GPUs using CUDA [17] and on a fast numerical computation of eigenvalues and eigenvectors of the covariance matrix for the evaluation of the C&P decomposition. Results on ALOS-PALSAR data show how speedups of more than 450 TB can be reached.

Manuscript received 30 December 2021.

A. Capozzoli, C. Curcio, and A. Liseno are with the Universitá di Napoli Federico II, DIETI, via Claudio 21, I 80125 Napoli, Italy; e-mail: a.capozzoli@unina.it.

A. Marino is with the Faculty of Natural Sciences, University of Stirling, Stirling FK9 4LA, United Kingdom; e-mail: armando.marino@stir.ac.uk.

2. Fast Processing for SAR Polarimetry

Today, GPUs are massively parallel machines in which thousands of cores can "simultaneously" execute millions of computational threads. The possibility of executing millions of threads "simultaneously" is related to the possibility of context switching, by which the latency of read/write memory operations is masked by computational tasks. Moreover, GPUs are now equipped with large memories to deal with large problems and have become an essential tool in many scientific applications. Thus, they enable the acceleration of highly burdened code sections for hybrid, CPU/GPU executions.

One area for which GPUs show the best performance is image processing, especially when the pixels require independent computations, leading to an "embarrassingly parallel" problem, as for the C&P decomposition.

The C&P decomposition relies on the computation of eigenvalues and eigenvectors of the covariance matrix for each image pixel. In the proposed parallel approach, each GPU computational thread is essentially appointed to compute the eigendecomposition of the covariance matrix for a different pixel.

Generally, in numerical computation, eigenvalues and eigenvectors are evaluated through library routines that typically implement iterative approaches. Thus, our problem requires a library capable of computing the eigendecomposition of a large number of small matrices, a tool that is not widespread. Nevertheless, for the problem under examination, it is possible to exploit the small size (3×3) of the covariance matrix as well as its Hermitian property to devise an alternative method using a direct analytical, "Cardano-like" scheme [18]. Being analytical, it is fast and simple since it does not require the setup of general purpose, parallel iterative numerical methods [19, 20] or the use of already developed CUDA parallel libraries [21]. It also accurate since it avoids the typical round-off errors of iterative methods, and the results do not depend on an initial guess.

3. The Cardano-Like Approach

To illustrate the proposed Cardano-like approach, we adopt the following rewriting of the coherency matrix $\underline{\underline{T}}$ [1, 22]:

$$\underline{\underline{T}} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12}^* & a_{22} & a_{23} \\ a_{13}^* & a_{23}^* & a_{33} \end{bmatrix} \quad (1)$$

The schemes to compute the eigenvalues and eigenvectors are summarized in the following two subsections.

3.1 Eigenvalue Calculation

To compute the eigenvalues, we resort to the characteristic equation of matrix (1), which is the third-order polynomial

$$\left|\underline{\underline{T}} - \lambda\underline{\underline{I}}\right| = P(\lambda) = \lambda^3 + c_2\lambda^2 + c_1\lambda + c_0 \quad (2)$$

where $\underline{\underline{I}}$ is the unit matrix and

$$\begin{cases} c_0 = & a_{11}|a_{23}|^2 + a_{22}|a_{13}|^2 + a_{33}|a_{12}|^2 - a_{11}a_{22}a_{33} \\ & -2\mathrm{Re}\left(a_{13}^* a_{12} a_{23}\right) \\ c_1 = & a_{11}a_{22} + a_{11}a_{33} + a_{22}a_{33} - |a_{12}|^2 \\ & -|a_{13}|^2 - |a_{23}|^2 \\ c_2 = & -a_{11} - a_{22} - a_{33} \end{cases} \quad (3)$$

Note that all the coefficients of polynomial $P(\lambda)$ are real.

Equation (2) can be solved by the method devised by Ferro, Tartaglia, and Cardano. In particular, the eigenvalues λ_i, $i = 1, 2, 3$, can be reliably computed by the numerically robust formulas

$$\lambda_i = \frac{\sqrt{p}}{3}x_i - \frac{c_2}{3}, \quad i = 1, 2, 3 \quad (4)$$

where $x_1 = 2\cos\varphi$, $x_2 = -\cos\varphi - \sqrt{3}\sin\varphi$, $x_3 = -\cos\varphi + \sqrt{3}\sin\varphi$,

$$\varphi = \frac{1}{3}\arctan\left\{\frac{\sqrt{27\left[\frac{1}{4}c_1^2(p - c_1) + c_0\left(q + \frac{27}{4}c_0\right)\right]}}{q}\right\} \quad (5)$$

$p = c_2^2 - 3c_1$, and $q = -(27/2)c_0 - c_2^3 + (9/2)c_1c_2$.

It can be stated that the absolute accuracy of the approach is $\mathcal{O}(\varepsilon\lambda_{\max})$, where $\lambda_{\max}$ is the largest eigenvalue and ε is machine precision. This may imply some loss of relative accuracy for very small eigenvalues. However, very small eigenvalues are symptomatic of a missing scattering mechanism; they are typically thresholded from a polarimetric point of view, and, accordingly, the increased relative error has no consequence.

3.1.1 *Eigenvectors Calculation:* Once the eigenvalues of $\underline{\underline{T}}$ have been computed, the calculation of the eigenvectors can be performed by resorting to the fundamental equation

$$\left(\underline{\underline{T}} - \lambda_i\underline{\underline{I}}\right) \cdot \underline{v}_i = 0, \quad i = 1, 2, 3 \quad (6)$$

By taking the Hermitian of (6), we have

$$\underline{v}_i^\dagger \cdot \left(\underline{\underline{T}} - \lambda_i\underline{\underline{I}}\right) = 0, \quad i = 1, 2, 3 \quad (7)$$

On multiplying (7) by the unit vector $\hat{e}_j$, having all the elements equal to 0 except for the jth, which is 1, we obtain

$$\underline{v}_i^\dagger \cdot \left(\underline{t}_j - \lambda_i\hat{e}_j\right) = 0, \quad i, j = 1, 2, 3 \quad (8)$$

where $\underline{t}_j$ is the jth column of $\underline{\underline{T}}$. Consequently, $\underline{v}_i^\dagger$ is simultaneously orthogonal to $(\underline{t}_1 - \lambda_i\hat{e}_1)$ and $(\underline{t}_2 - \lambda_i\hat{e}_2)$. Therefore, as long as such two vectors are linearly independent, $\underline{v}_i$ can be chosen as

$$\underline{v}_i = \left[(\underline{t}_1 - \lambda_i\hat{e}_1)\times(\underline{t}_2 - \lambda_i\hat{e}_2)\right]^* \quad (9)$$

If $(\underline{t}_1 - \lambda_i\hat{e}_1) = \mu(\underline{t}_2 - \lambda_i\hat{e}_2)$, then $\underline{v}_i$ can be chosen as

$$\underline{v}_i = \frac{1}{\sqrt{1 + |\mu|^2}}\begin{pmatrix} 1 \\ -\mu \\ 0 \end{pmatrix} \quad (10)$$

The above procedure fails in case of degenerate eigenvalues since, in this case, it determines only one of the two corresponding eigenvectors. If $\lambda_1 = \lambda_2$, then $\underline{v}_2$ is computed as the cross product of $\underline{v}_1$ and one of the columns of $\underline{\underline{T}} - \lambda_1\underline{\underline{I}}$. Indeed, from (8), $\underline{v}_2^\dagger$ is orthogonal to $(\underline{t}_j - \lambda_2\hat{e}_j)$, $j = 1, 2, 3$. Furthermore, it must be also orthogonal to $\underline{v}_1$. Accordingly, it can be chosen as

$$\underline{v}_2 = \left[\underline{v}_1 \times (\underline{t}_2 - \lambda_i\hat{e}_2)\right]^* \quad (11)$$

4. PyCUDA Implementation

The GPU implementation has been carried out by using the PyCUDA library [23], which enables accelerating the most time-consuming sections of a Python code through CUDA kernels. In this way, it has been possible to accelerate an already existing polarimetric decomposition code written in Python language. Such a code is a state-of-the-art, open-source SAR polarimetric code [24] that employs the Numpy library for eigenvalue and eigenvector calculation. For the problem at hand, such a library adopts the LAPACK routine _heevd, calculating the eigendecomposition of a complex Hermitian matrix using a divide-and-conquer algorithm.

The polarimetric decomposition has been fully accelerated in CUDA by appointing each computational thread to perform the calculations associated to a different pixel. In this way, each thread evaluates the eigendecomposition of a different matrix $\underline{\underline{T}}$ by exploiting the Cardano-like approach discussed previously. In other words, the original Python code runs on a

Table 1. Computational performance of the GPU vs. CPU approach

Size	GPU (s)	CPU (s)	Speedup
1248 × 2633	0.50	297	594
2496 × 5266	1.70	1198	705
3744 × 7899	3.76	2618	696

traditional CPU, iterates over the pixels sequentially in a loop, and exploits the LAPACK routine to compute eigenvalues and eigenvectors. On the other hand, the PyCUDA approach runs on a GPU, thus parallelizing over the pixels, and uses an analytical method for the same purpose.

The CUDA kernels implement the approach in [18], with the only difference being that double precision was considered, while here single precision has been dealt with.

5. Results

We now assess the performance of the proposed method by comparing it with that of the preexisting sequential Python code. The CPU and GPU codes have been run on the ml.p3.2xlarge instance of Amazon Web Services (Amazon SageMaker) comprising a Tesla V100-SXM2-16GB GPU, Compute Capability 7.0, and an Intel Xeon CPU E5-2686 v4 at 2.30 GHz.

The performance assessment has been worked out using polarimetric ALOS-PALSAR data of size 1248 × 2633.

A significant speedup has thus been achieved. To further assess the computational performance for larger data sets, the original image has been resampled with 2× and 3× factors and the computing times measured. Table 1 summarizes the achieved results. Again, the GPU implementation shows its significantly faster performance.

An accuracy analysis has been also performed. Figure 1 shows the map of the first computed eigenvalue (second and third eigenvalues are omitted for brevity), having ordered the eigenvalues in a decreasing order, while Figures 2–4 display the calculated corresponding α angles for the CPU and GPU cases. Likewise, Figures 5 and 6 can be used to compare the entropy and the average α angle for the sequential and parallel computations. As can be seen, the Cardano-like approach and the LAPACK-based approach return very similar results. This is confirmed by Table 2, which

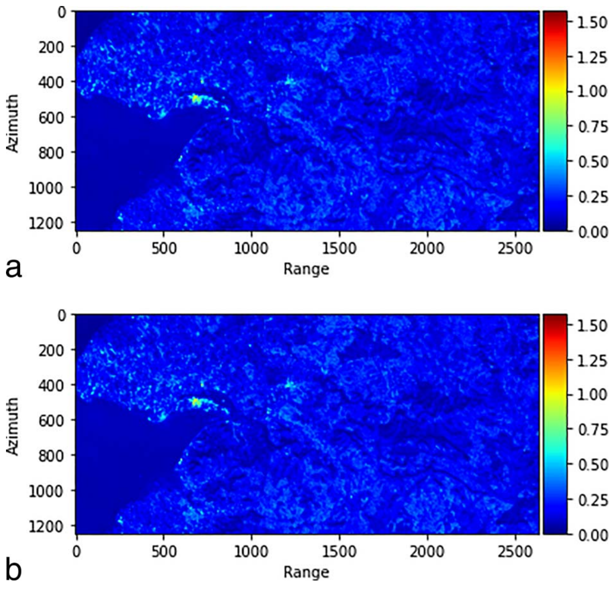

Figure 2. First alpha angle. Top: CPU. Bottom: GPU.

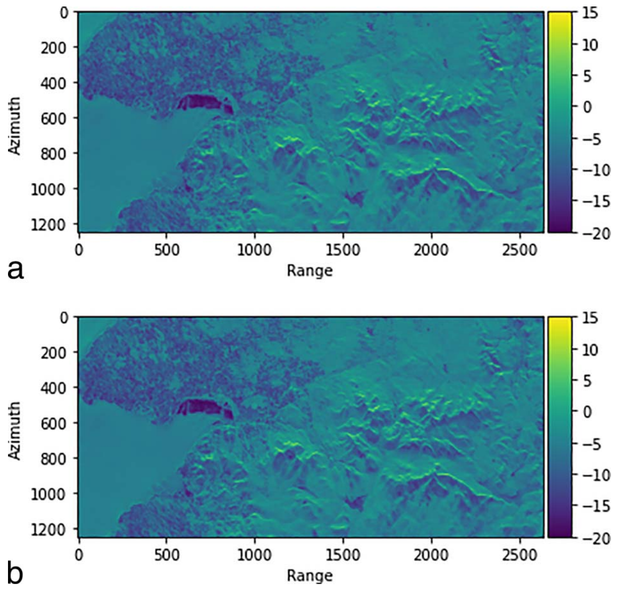

Figure 1. First eigenvalue. Top: CPU. Bottom: GPU.

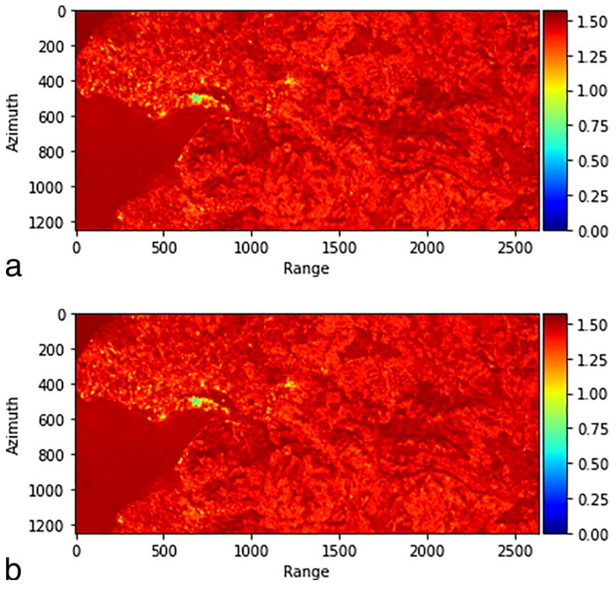

Figure 3. Second alpha angle. Top: CPU. Bottom: GPU.

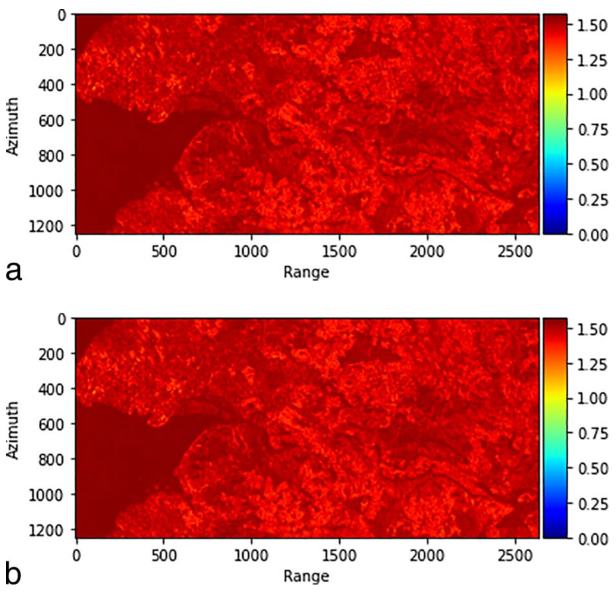

Figure 4. Third alpha angle. Top: CPU. Bottom: GPU.

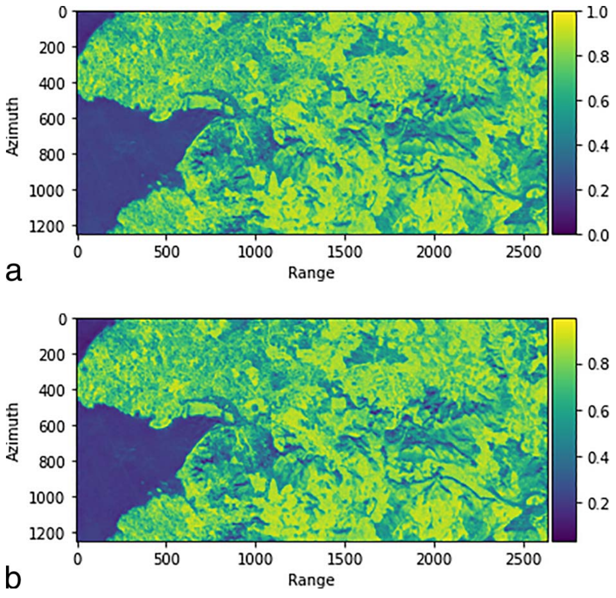

Figure 5. Entropy. Top: CPU. Bottom: GPU.

reports the root mean square errors achieved with GPU processing compared to CPU processing. The accuracies are consistent with the results in [18] if cast to single precision. Concerning the eigenvalues, the main discrepancies involve the smallest ones, as expected. The evaluation of the α angles exhibits an accuracy comparable to that obtained for the eigenvectors. The entropy computation has an accuracy close to that of the first eigenvalue since it provides the most relevant contribution. Opposite to that, the evaluation of the average α angle has an accuracy comparable to that relative to the eigenvector calculation.

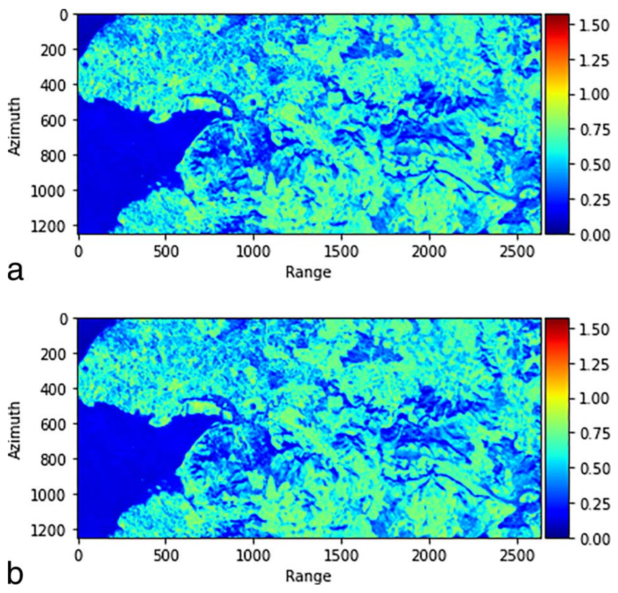

Figure 6. Average alpha angle. Top: CPU. Bottom: GPU.

Table 2. Percentage root mean square (RMS) error achieved with GPU processing

Parameter	Percentage RMS
λ_1	$4.02 \cdot 10^{-5}$
λ_2	$8.98 \cdot 10^{-4}$
λ_3	$1.09 \cdot 10^{-3}$
α_1	$9.77 \cdot 10^{-3}$
α_2	$5.30 \cdot 10^{-3}$
α_3	$3.24 \cdot 10^{-3}$
Entropy	$1.05 \cdot 10^{-5}$
Average alpha	$2.09 \cdot 10^{-3}$

6. Conclusions and Future Developments

We have accelerated the C&P polarimetric decomposition by an algorithmic/hardware approach. The numerical acceleration has been obtained by computing the eigenvalues and eigenvectors of the relevant coherency matrix by a Cardano-like approach. The hardware acceleration has been achieved by GPU processing. The approach can be easily extended to the computation of beta angles and generalized to cases when the relevant matrix to decompose is non-Hermitian [25].

7. Acknowledgment

This work was partially supported by MIUR through the "ICT for Health" project, Dipartimento di Eccellenza (2018–2022) "Ingegneria Elettrica e delle Tecnologie dell'Informazione."

8. References

1. S. R. Cloude and E. Pottier, "A Review of Target Decomposition Theorems in Radar Polarimetry," *IEEE*

Transactions on Geoscience and Remote Sensing, **34**, 2, March 1996, pp. 498-518.

2. A. Marino, S. R. Cloude, and I. H. Woodhouse, "A Polarimetric Target Detector Using the Huynen Fork," *IEEE Transactions on Geoscience and Remote Sensing*, **48**, 5, May 2010, pp. 2357-2366.
3. D. Mandala, V. Kumara, D. Rathaa, S. Deya, A. Bhattacharyaa, J. M. Lopez-Sanchez, H. McNairn, and Y. S. Rao, "Dual Polarimetric Radar Vegetation Index for Crop Growth Monitoring Using Sentinel-1 SAR Data," *Remote Sensing of Environment*, **247**, 2020, p. 111954.
4. M. Ghanbari, D. A. Clausi, L. Xu, and M. Jiang, "Contextual Classification of Sea-Ice Types Using Compact Polarimetric SAR Data," *IEEE Transactions on Geoscience and Remote Sensing*, **57**, 10, October 2019, pp. 7476-7491.
5. S. Singha, M. Johansson, N. Hughes, S. M. Hvidegaard, and H. Skourup, "Arctic Sea Ice Characterization Using Spaceborne Fully Polarimetric L-, C-, and X-Band SAR With Validation by Airborne Measurements," *IEEE Transactions on Geoscience and Remote Sensing*, **56**, 7, July 2018, pp. 3715-3734.
6. S. Angelliaume, O. Boisot, and C.-A. Guérin, "Dual-Polarized L-Band SAR Imagery for Temporal Monitoring of Marine Oil Slick Concentration," *Remote Sensing*, **10**, 7, p. 1012; doi: 10.3390/rs10071012.
7. H. Aghababaei, G. Ferraioli, L. Ferro-Famil, Y. Huang, M. Mariotti D'Alessandro, V. Pascazio, G. Schirinzi, and S. Tebaldini, "Forest SAR Tomography: Principles and Applications," *IEEE Geoscience and Remote Sensing Magazine*, **8**, 2, June 2020, pp. 30-45.
8. C. N. Koyama, M. Watanabe, M. Hayashi, T. Ogawa, and M. Shimada, "Mapping the Spatial-Temporal Variability of Tropical Forests by ALOS-2 L-Band SAR Big Data Analysis," *Remote Sensing of Environment*, **233**, September 2019, pp. 1-18, doi: 10.1016/j.rse.2019.111372.
9. G.-C. Sun, Y. Liu, M. Xing, S. Wang, L. Guo, and J. Yang, "A Real-Time Imaging Algorithm Based on Sub-Aperture CS-Dechirp for GF3-SAR Data," *Sensors*, **18**, 8, August 2018, pp. 1-15.
10. A. Capozzoli, C. Curcio, and A. Liseno, "Optimized NUFFTs for Interpolations in Back-Projection Algorithms," *URSI Radio Science Letters*, **2**, 2020, pp. 1-5.
11. A. Capozzoli, C. Curcio, A. Liseno, "NUFFT-Based Interpolation in Backprojection Algorithms," *IEEE Geoscience and Remote Sensing Letters*, **18**, 12, December 2021, pp. 2117-2121.
12. A. Capozzoli, C. Curcio, and A. Liseno, "GPU-Based ω-k Tomographic Processing by 1D Non-Uniform FFTS," *Progress in Electromagnetics Research M*, **23**, 2012, pp. 279-298.
13. O. Ponce, P. Prats-Iraola, M. Pinheiro, M. Rodriguez-Cassola, R. Scheiber, A. Reigber, and A. Moreira, "Fully Polarimetric High-Resolution 3-D Imaging With Circular SAR at L-Band," *IEEE Transactions on Geoscience and Remote Sensing*, **52**, 6, June 2014, pp. 3074-3090.
14. C.-Y. Chiang, K.-S. Chen, Y. Yang, Y. Zhang, T. Zhang, "SAR Image Simulation of Complex Target Including Multiple Scattering," *Remote Sensing*, **13**, 22, November 2021, pp. 1-22.
15. H. Quan, Z. Cui, R. Wang, and Z. Cao, "GPU Parallel Implementation and Optimisation of SAR Target Recognition Method," *Journal of Engineering*, **1**, 2019, pp. 1-5.
16. S. Porzycka-Strzelczyk, J. Strzelczyk, K. Szostek, M. Dwornik, A. Leśniak, J. Bała, and A. Franczyk, "Information Extraction From Satellite-Based Polarimetric SAR Data Using Simulated Annealing and SIRT Methods and GPU Processing," *Energies*, **15**, 1, December 2021, pp. 1-22.
17. A. Breglia, A. Capozzoli, C. Curcio, and A. Liseno, "CUDA Expression Templates for Electromagnetic Applications on GPUs," *IEEE Antennas and Propagation Magazine*, **55**, 5, October 2016, pp. 156-166.
18. J. Kopp, "Efficient Numerical Diagonalization of Hermitian 3×3 Matrices," *International Journal of Modern Physics C*, **19**, 3, 2008, pp. 523-548.
19. D. F. G. Coelho, R. J. Cintra, A. C. Frery, and V. S. Dimitrov, "Fast Matrix Inversion and Determinant Computation for Polarimetric Synthetic Aperture Radar," *Computers & Geosciences*, **119**, October 2018, pp. 109-114.
20. A. Cosnuau, "Computation on GPU of Eigenvalues and Eigenvectors of a Large Number of Small Hermitian Matrices," *Procedia Computer Science*, **29**, 2014, pp. 800-810.
21. C. Lessig, "Eigenvalue Computation With CUDA," *NVIDIA White Papers*, July 2012.
22. O. Lowenschuss, "Scattering Matrix Application," *Proceedings of the IEEE*, **53**, 8, August 1965, pp. 988-992.
23. Klöckner, N. Pinto, Y. Leed, B. Catanzaro, P. Ivanov, and A. Fasih, "PyCUDA and PyOpenCL: A Scripting-Based Approach to GPU Run-Time Code Generation," *Parallel Computing*, **38**, 3, March 2012, pp. 157-174.
24. A. Marino, "Pol-SAR Applications," 6th ESA Advanced Course on Radar Polarimetry, May 10–14, 2021.
25. V. I. Lebedev, "On Formulae for Roots of Cubic Equation," *Soviet Journal of Numerical Analysis and Mathematical Modelling*, **6**, 1991, pp. 315-324.