Parallel GLRT-Based SAR Tomographic Processing on Distributed Multi-GPU Platforms

Pasquale Imperatore, *Senior Member, IEEE*, Massimiliano Russo, Mehwish Nisar, Marco Lapegna, Diego Romano, and Antonio Pauciullo, *Member, IEEE*

***Abstract*—This article addresses the development of a multilevel parallel algorithm for detecting single scatterers in SAR tomography, targeted at multinode multi-GPU distributed computational architectures. Taking into account the computational structure of the canonical processing scheme based on the generalized likelihood ratio test considered in this article, an appropriate problem decomposition is adopted to devise and formulate an efficient parallel algorithm tailored for heterogeneous high-performance computing platforms, thereby enabling a hierarchical exploitation of different levels of parallelism, including both distributed and shared memory models. Experimental evaluations conducted on a multinode, GPU-enabled HPC cluster, using a high-resolution SAR dataset, exhibit substantial acceleration and scalability. These results quantitatively confirm the effectiveness of the proposed multilevel parallel framework and position the developed prototype as a promising, scalable solution for large-scale SAR tomographic processing applications. This is particularly relevant in light of upcoming SAR missions, which are expected to generate unprecedented volumes of data, thus demanding scalable and high-performance processing solutions.**

***Index Terms*—Electromagnetic scattering, graphical processing unit (GPU), high performance computing (HPC), parallel algorithms, synthetic aperture radar (SAR), tomography.**

I. Introduction

SPACEBORNE synthetic aperture radar (SAR) interferometry and tomography are critical technologies for advancing our understanding of Earth's surface dynamics, impacting the environment and building infrastructure. These techniques leverage radar signals from satellites to capture highly detailed 3-D information and subtle changes over large areas, making them essential for a wide range of scientific, environmental, and engineering applications. More specifically, repeat-pass differential SAR Interferometry (DInSAR) and its evolution into persistent scatterers interferometry (PSI), which allows the accurate localization of ground targets and the monitoring of possible ground displacements at the millimetric-per-year scale, has represented a breakthrough in the application of SAR to risk monitoring. Multipass and multiview SAR data are available today for most of the Earth's surface, thanks to repeated acquisitions by existing sensors along consistent orbital tracks. This wealth of data has driven the development of advanced techniques that, similarly to multipass DInSAR [1] and PSI [2], jointly and coherently process the available information to produce accurate physical measurements. Among them, SAR tomography—also known as 3-D SAR imaging—is based on the principle that acquisitions from different passes can be assimilated to signals received by elements of an antenna array aligned along the cross-radial (elevation) direction. Hence, as in the classic azimuth antenna synthesis process, a synthetic aperture can be generated to profile the scattering distribution also along the elevation direction as well, thereby enabling full 3-D imaging of the scene [3], [4]. Moreover, DInSAR and SAR tomography have been fused in differential SAR tomography processing, also known as 4-D SAR imaging (space-velocity) [5], [6], which overcomes the spatial resolution limits of classic PSI, allowing the separation of multiple interfering scatterers within the same pixel [6].

However, analysis of long-term multiview and multitemporal data requires the separation of useful signals from noise and clutter to identify, locate, and monitor the largest possible number of ground structures. At the same time, it is also necessary to minimize the risk of signal misinterpretation, even under relatively low signal-to-noise ratio (SNR) conditions. Standard PSI techniques detect single scatterers by comparing a decision variable—derived from matching observed phase values to a multibaseline/multitemporal model—against a predefined threshold. In contrast, 3-D/4-D tomography-based analysis detects single targets by assessing the similarity between the focused scattering distribution—along the elevation and elevation-velocity directions—and the ideal response of a point scatterer located at the specific position within the corresponding domain.

In [7], according to the tomographic approach, the problem was for the first time reformulated in the context of detection theory, by deriving a detection scheme based on the generalized likelihood ratio test (GLRT) for the identification of single

Received 30 July 2025; revised 15 September 2025, 29 December 2025, and 6 February 2026; accepted 2 March 2026. Date of publication 4 March 2026; date of current version 20 March 2026. This work was supported in part by the National Resilience and Recovery Plan through the National Center for HPC, Big Data, and Quantum Computing, European Union in the Next Generation EU plan framework, in part by European Union through Italian National Recovery and Resilience Plan of Next Generation EU, partnership on "Telecommunications of the Future" (program RESTART) under Grant PE0000001, and in part by the Italian Space Agency for using CSK SAR data. *(Corresponding author: Pasquale Imperatore.)*

Pasquale Imperatore, Mehwish Nisar, and Antonio Pauciullo are with the Institute for Electromagnetic Sensing of the Environment, National Research Council, 80131 Naples, Italy (e-mail: imperatore.p@irea.cnr.it; nisar.m@irea.cnr.it; pauciullo.a@irea.cnr.it).

Massimiliano Russo and Marco Lapegna are with the Department of Mathematics and Applications, University of Naples Federico II, 80138 Naples, Italy (e-mail: massimiliano.russo@unina.it; marco.lapegna@unina.it).

Diego Romano is with the Institute for High Performance Computing and Networking, National Research Council, 80131 Naples, Italy (e-mail: diego.romano@cnr.it).

Digital Object Identifier 10.1109/JSTARS.2026.3670717

and stationary scatterers in multidimensional SAR imaging. GLRT-based detection has subsequently been extended to handle multiple scatterers. In [8], a sequential two-stage scheme has been proposed for the identification of double scatterers. In [9], a different scheme is proposed for the detection of a number of scatterers—even more than two—and with differences in elevation and velocity below the classical Rayleigh resolution [6]. While the GLRT approach provides a powerful methodology for analyzing SAR data, its practical realization becomes extremely computationally demanding when applied to large multidimensional SAR data, making the analysis infeasible in a reasonable time-frame, particularly in high-resolution configurations. The challenge is further exacerbated by the growing volume and diversity of SAR data acquired by current and forthcoming satellite missions, operating across multiple frequency bands, polarimetric configurations, and acquisition modes [10].

The computational complexity inherent in SAR Tomography (TomoSAR) necessitates the development of highly parallel algorithms capable of efficiently handling large-scale, multibaseline datasets. This complexity is further amplified in large-scale Earth observation applications, where hundreds of acquisitions over wide geographic areas must be jointly processed for 3-D/4-D SAR imaging reconstructions. TomoSAR processing pipelines involve several intensive stages and the solution of large-scale inverse problems.

Leveraging modern heterogeneous high-performance computing (HPC) platforms—such as those listed in the TOP500 ranking [11]—offers a viable path to address these computational demands. These platforms integrate multicore CPUs, high-throughput GPUs, and domain-specific accelerators, enabling the deployment of hybrid parallel paradigms (e.g., using MPI + OpenMP/CUDA) that exploit both inter- and intranode parallelism [10], [12], [13], [14], [15], [16], [17]. To address these challenges, it is often necessary to develop dedicated parallel algorithms and implement advanced computing methodologies [10], [18], [19], [20], [21]. Specifically, efficient and high performance implementation methodologically requires careful parallel reformulation of inherent processing schemes, taking into account—for example, data locality, optimization of memory hierarchy, minimizing data transfers between host and device memory in GPU-accelerated kernels, optimizing problem decomposition strategies to reduce internode communication overhead in distributed systems and load unbalancing across heterogeneous resources. Moreover, algorithm scalability and performance portability are critical to ensure sustained throughput and processing efficiency as system size and data volumes scale.

In this article, we specifically refer to the processing scheme introduced in [7], which is considered a reference canonical problem for our investigation. To tackle the considered canonical problem, a dedicated multilevel parallel algorithm tailored for current heterogeneous high-performance computing platforms is developed following HPC methodologies. A specific experimental analysis is conducted using real high-resolution SAR data and a HPC distributed platform, to quantitatively demonstrate the benefits of the different degrees of parallelism incorporated into the implemented algorithm. The conducted experimental investigation aims to assess the scalable performance of the parallel strategy as both the problem size and computational resources increase.

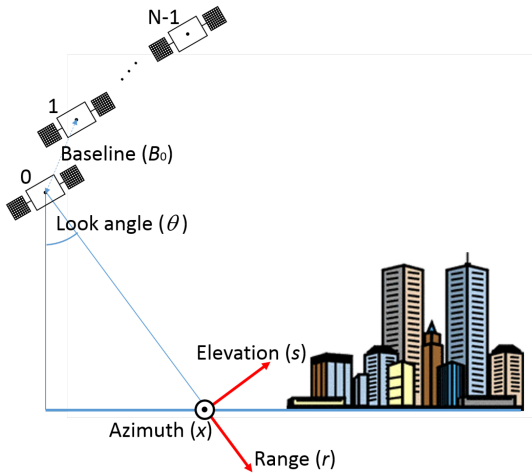

Fig. 1. Acquisition geometry of a multipass SAR system enables the synthesis of a long aperture along the elevation (s) direction.

In this context, scalable and high-performance TomoSAR processing solutions optimized for heterogeneous HPC architectures are key enablers for near-real-time 3-D/4-D SAR imaging of the Earth's surface, with impactful applications at continental or global scale ranging from urban infrastructure monitoring to environmental changes modeling and detection.

This article is organized as follows. The theoretical background of the considered canonical problem is presented in Section II. The multilevel parallel algorithm developed for the problem at hand is described in Section III. In Section IV, the experimental validation of the parallel prototype is discussed, with emphasis on the inherent performance analysis. Finally, Section V concludes the article.

II. Problem Formulation and Detector Design

Let us consider N SAR images acquired by a monostatic multipass system operating at wavelength λ and range distance r, as schematized in Fig. 1. We consider the problem of detecting the presence of a (single) target within each imaged azimuth-range pixel and estimating its geophysical parameters. The latter are usually given by the elevation (cross-radial) position s and the mean deformation velocity v, associated with possible movements of the ground. More specifically, we are interested in the detection of the so-called persistent scatterers (PSs), characterized by a very stable electromagnetic response between different SAR acquisitions [2].

PSI techniques address this problem by modeling the phase φ_i of the signal backscattered by a PS in a generic image pixel, received at the ith antenna and properly phase calibrated [1], [2], as

$$\varphi_i = 2\pi\xi_i s_0 + 2\pi\varsigma_i v_0 \qquad i = 0, \ldots, N-1 \tag{1}$$

where s_0 and v_0 are respectively the target elevation and mean velocity deformation, $\xi_i = 2b_i/\lambda r$ and $\varsigma_i = 2t_i/\lambda$ are the elevation and velocity frequency, corresponding to the orthogonal

baseline b_i and the temporal baseline t_i of the ith acquisition with respect to a reference master image. In this context, a target is identified as PS if the mismatch (i.e., the residual phase) between the model (1) and the observed phase in the complex data domain is above a suitable threshold, usually set by experimental evidence.

Unlike PSI, SAR tomography techniques, also known as multidimensional SAR imaging [5], [6], exploit both the amplitude and phase information of the received signal. This enables the reconstruction, for each azimuth-range image pixel, of the backscattering distribution in the elevation-velocity domain, i.e., the reflectivity profile γ of the pixel over the (s, v) domain. Fig. 1 shows the basic concept of 3-D SAR imaging, which synthesizes a long antenna in the cross range (elevation) direction.

Following the basis of the SAR tomography approach, a test for the detection of single PSs has been proposed in [7]. Here, the complex phase calibrated signal x_i acquired at the ith antenna and backsacttered, in a given pixel, by a PS located at point (s, v) of the 2-D elevation-velocity domain is modeled as

$$x_i = \gamma\, a_i(s, v) + w_i \tag{2}$$

where

$$a_i(s, v) = \exp(j2\pi\xi_i s + j2\pi\varsigma_i v) \tag{3}$$

is the complex phase term depending on the system and scatterer parameters; γ is the target complex reflectivity; w_i is the additive disturbance contribution affecting the received signal; finally, $j = \sqrt{-1}$. The model in (2) can be written in vector form as

$$\boldsymbol{x} = \gamma \boldsymbol{a}(s, v) + \boldsymbol{w} \tag{4}$$

where the N-length vectors $\boldsymbol{x}$ and $\boldsymbol{w}$ collect images and noise components, respectively; $\boldsymbol{a}(s, v)$ is the *steering vector*, which identifies the target direction in the complex N-dimensional vector space.

The problem of detecting the presence of a target with unknown backscattering coefficient γ and elevation-velocity parameters, i.e., unknown position (s, v) in the elevation-velocity plane, is cast in [7] in terms of the following binary hypotheses test:

$$\begin{aligned} &\mathcal{H}_0 : \boldsymbol{x} = \boldsymbol{w} \\ &\mathcal{H}_1 : \boldsymbol{x} = \gamma \boldsymbol{a}(s, v) + \boldsymbol{w}. \end{aligned} \tag{5}$$

According to the *Neyman–Pearson* criterion, the optimum solution to the hypotheses testing problem (5) is the likelihood ratio test (LRT) [22], i.e.,

$$L(\boldsymbol{x}) = \frac{p\left(\boldsymbol{x}; s, v, \gamma, \sigma^2, \mathcal{H}_1\right)}{p\left(\boldsymbol{x}; \sigma^2, \mathcal{H}_0\right)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \mathrm{T_s} \tag{6}$$

where $p(\boldsymbol{x}; s, v, \gamma, \sigma^2, \mathcal{H}_1)$ and $p(\boldsymbol{x}; \sigma^2, \mathcal{H}_0)$ are the probability density functions (pdf's) of $\boldsymbol{x}$ under hypotheses $\mathcal{H}_1$ and $\mathcal{H}_0$, respectively, $\mathrm{T_s}$ is the detection threshold, set according to the desired value of the false alarm probability P_{fa}. The latter is defined as the probability that the test decides for the presence of a target when it is actually absent (i.e., decides for $\mathcal{H}_1$ under $\mathcal{H}_0$).

Further developments require specifying the pdf's of $\boldsymbol{x}$ under $\mathcal{H}_1$ and $\mathcal{H}_0$. Assuming, as usual, that the noise vector $\boldsymbol{w}$ is distributed as a complex circular Gaussian random vector with zero mean and covariance matrix $E[\boldsymbol{w}\boldsymbol{w}^H] = \sigma^2 \boldsymbol{I}_N$, $E[\cdot]$ being the statistical expectation, $\boldsymbol{I}_N$ the $(N \times N)$ identity matrix, and $\sigma^2 > 0$, the two pdf's of $\boldsymbol{x}$ are given by

$$p\left(\boldsymbol{x}; s, v, \gamma, \sigma^2, \mathcal{H}_1\right) = \frac{1}{(\pi\sigma^2)^N} \exp\left(-\frac{\|\boldsymbol{x} - \gamma \boldsymbol{a}(s, v)\|^2}{\sigma^2}\right) \tag{7}$$

and

$$p\left(\boldsymbol{x}; \sigma^2, \mathcal{H}_0\right) = \frac{1}{(\pi\sigma^2)^N} \exp\left(-\frac{\|\boldsymbol{x}\|^2}{\sigma^2}\right) \tag{8}$$

which, substituted in (6), leads to the following expression for the LRT:

$$\mathrm{Re}\left\{\gamma^* \boldsymbol{a}^{\mathrm{H}}(s, v)\boldsymbol{x}\right\} \underset{\mathcal{H}_1}{\overset{\mathcal{H}_0}{\lessgtr}} \mathrm{T_s} \tag{9}$$

where $\mathrm{Re}\{\cdot\}$ denotes the real part of the argument, $(\cdot)^*$ is the conjugate operator, the superscript H denotes the Hermitian (conjugate transpose) operator, $\|\cdot\|$ denotes the Euclidean norm, and the same symbol $\mathrm{T_s}$ has been used to indicate the modified threshold. The above expression highlights that, for the case at hand, the optimum test cannot be implemented, since it requires the knowledge of the parameters s, v, and γ that, in practical situations, are unknown. A possible way to circumvent this drawback is to exploit suboptimum detection procedures such as the GLRT test [22].

The GLRT replaces the unknown parameters with their maximum likelihood (ML) estimates in the LRT (6). Specifically, it implements the following decision rule:

$$\frac{\max\limits_{s, v, \gamma, \sigma^2} p\left(\boldsymbol{x}; s, v, \gamma, \sigma^2, \mathcal{H}_1\right)}{\max\limits_{\sigma^2} p\left(\boldsymbol{x}; \sigma^2, \mathcal{H}_0\right)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \mathrm{T_s}\,. \tag{10}$$

Performing under each hypothesis the maximization over the respective unknown parameters, (10) can be recast as

$$\widehat{c} = \max_{s, v} \frac{|\boldsymbol{a}^{\mathrm{H}}(s, v)\boldsymbol{x}|}{\|\boldsymbol{a}(s, v)\|\,\|\boldsymbol{x}\|} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \mathrm{T_s} \tag{11}$$

where the optimization is carried out over the (s, v)-domain, $\widehat{c}$ explicitly denotes the test statistic, and $\mathrm{T_s}$ denotes the modified detection threshold. According to (11), the problem is formulated as a binary hypothesis test aimed at discriminating, on a pixel-by-pixel basis, between the presence (hypothesis $\mathcal{H}_1$) and the absence (hypothesis $\mathcal{H}_0$) of a target. The right-hand side of the test in (11), namely $\widehat{c} \in [0, 1]$, represents a measure of the match between the data and the assumed model—i.e., between the SAR multiacquisition and the steering vector associated with a specific point in the (s, v) domain.

Accordingly, whenever the presence of a target is declared, the ML estimators of residual elevation $\widehat{s}$ and the mean velocity $\widehat{v}$ of the target are obtained as

$$(\widehat{s}, \widehat{v}) = \arg\max_{s, v} \frac{|\boldsymbol{a}^{\mathrm{H}}(s, v)\boldsymbol{x}|}{\|\boldsymbol{a}(s, v)\|\,\|\boldsymbol{x}\|}. \tag{12}$$

In practice, optimizating in (11) is usually performed over a finite, uniformly spaced bidimensional grid in the (s, v)-plane, thus its discretized version becomes

$$\widehat{c} = \max_{s_n, v_n} \frac{|\boldsymbol{a}^{\mathrm{H}}(s_n, v_n)\boldsymbol{x}|}{\|\boldsymbol{a}(s_n, v_n)\| \, \|\boldsymbol{x}\|} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \mathrm{T_s} \tag{13}$$

where (s_n, v_n) is the nth point of the bidimensional discretized search grid $\mathcal{G}$ in the (s, v) domain, with $n = 0, \ldots, M_s M_v - 1$. We highlight that in brute-force nonlinear optimization over 2-D domains, grid-based search involves a fundamental tradeoff between accuracy and computational cost. Thus, finer grids yield asymptotically consistent estimates of the nonlinear optimum but at quadratic computational expense, whereas coarser grids preserve tractability at the price of systematic discretization bias that distorts the location of the optimal solution relative to the continuous problem.

The values of the test statistic $\widehat{c}$, along with the estimated parameters $\widehat{s}$, and $\widehat{v}$ at each image pixel constitute the three output maps of the algorithm. Evaluating these maps is computationally demanding. Consequently, as the number of acquisitions and image pixels to be processed increases, the overall computational burden may become prohibitive without the adoption of optimized, parallel processing strategies.

Although it is evident that the computation in (11)–(12), which can be carried out on a pixel basis, is amenable to parallelization, translating this natural potential into an efficient multilevel parallel algorithm—capable of fully exploiting the different levels of parallelism offered by current heterogeneous computing platforms—requires dedicated investigation, as discussed in the subsequent section.

III. Parallel Formulation of the Problem

This section is devoted to the parallel reformulation of the canonical processing scheme introduced in the previous section. The hierarchical multilevel parallelism pattern adopted in this article is derived from a suitable decomposition of the computational model, as discussed herein.

A. Target Heterogeneous Computing Architectures

In a typical heterogeneous multicore and multi-GPU node, the host features multiple identical cores and hierarchical memory, and it is connected to heterogeneous GPUs. The host and the devices have disjoint memory spaces, and explicit memory transfers are required for communication between them. Such distributed architectures are increasingly prevalent in the current HPC landscape and represent the primary target for the parallel solution we aim to develop [11].

Fig. 2 shows a representative scheme of a distributed heterogeneous multinode platform, where each node is composed of multiple CPUs and GPU devices, interconnected via a high-speed network.

B. Computational Cost

Preliminary, we concerns with the number of operations required by the problem considered. The computational costs

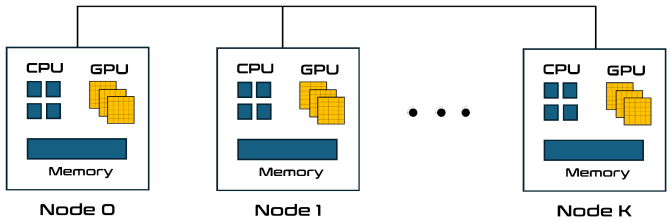

Fig. 2. Scheme of a distributed heterogeneous computing architecture.

can be evaluated in terms of the number of floating point operations needed to solve the considered canonical problem. It basically scales as follows:

$$C = N \times M \times N_{rg} \times N_{az} \tag{14}$$

where

1) $M = M_s M_v$ is the total number of prescribed discrete locations of a meshgrid in the search space (s, v);
2) N is the number of SAR images;
3) N_{rg} and N_{az} represent the SAR image size along the range and azimuth directions, respectively.

It should be noted that M is usually much larger than N, with M typically on the order of thousands (reaching up to 10 000 or more points for finer elevation grids), and N typically consisting of several tens of acquisitions (e.g., 20–200 acquisitions). On the other hand, the total number of pixels $N_{rg} \times N_{az}$ might be in the hundreds of millions.

C. Problem Decomposition

As a preliminary step, it is convenient to focus on the computationally dominant task pertinent to the evaluation of the product $\boldsymbol{a}^{\mathrm{H}}(s, v)\boldsymbol{x}$ underlying the optimization problem in (12). It should be noted that this term has to be evaluated over a prescribed meshgrid in the search domain (s, v), thus motivating the convenient introduction of a formal operator Ω, to be applied to a normalized version $(\boldsymbol{x}/\|\boldsymbol{x}\|)$ of the data vector $\boldsymbol{x} \in \mathbb{C}^N$ describing the set of SAR acquisitions for the pixel under test. The linear operator Ω can be represented by a complex matrix of size $M \times N$, whose rows are given by the vectors $\boldsymbol{a}^{\mathrm{H}}(s_m, v_m)$, with $m \in \{1, 2, \ldots, M\}$. Once $\Omega(\boldsymbol{x}/\|\boldsymbol{x}\|)$ has been evaluated, the results for the considered pixel in terms of estimated mean deformation velocity, residual topography, and coherence, can be straightforwardly obtained, according to (12) and (13). As will be discussed later, the computation of the steering vectors can be parametrically performed on a per-tile basis by logically partitioning the SAR images into Q tiles in a convenient manner.

In the context of parallel computing, the concept of homomorphism generally refers to structure-preserving transformations that can be used to divide a large problem into smaller subproblems that can be solved concurrently across multiple processing units. The decomposition strategy can then be systematically addressed using a formalism based on the concept of homomorphism, as discussed in [17], [18]. As a result, the computational scheme can be easily and formally structured as a parallel algorithm.

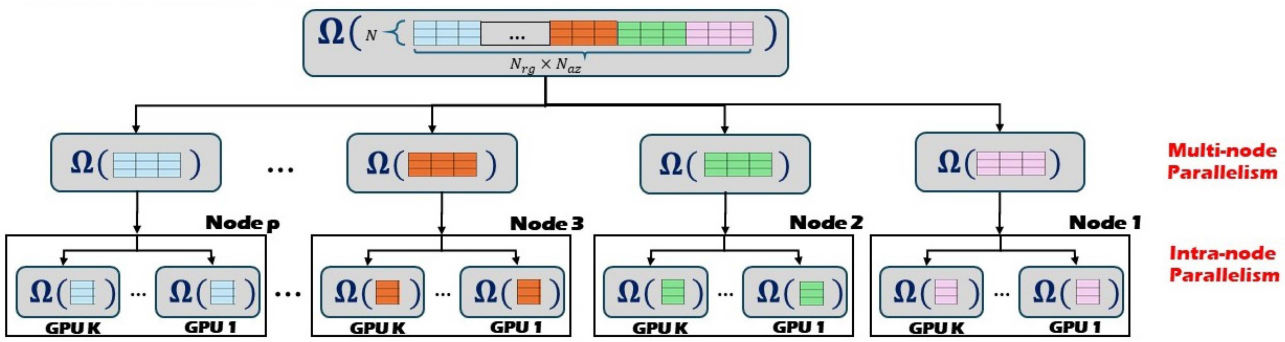

Fig. 3. Multilevel parallel pattern: $K \times P$ overall number of GPUs.

Consider objects of a structured type:

$$d = d_1 \oplus d_2 \oplus \cdots \oplus d_P \tag{15}$$

where $d_1, d_2, \ldots, d_P$, are objects of a structured type, and $\oplus$ is an operation that builds them into a larger object d, of the structured type. For the specific case considered in this article, d represents a reshaped version of the entire 3-D data stack (see also Fig. 3), which can be partitioned into a set of blocks, according to (15).

In its strict mathematical sense, a *homomorphism* is a structure-preserving map between two algebraic structures. Formally, for two algebraic structures $(D, \oplus)$ and $(E, \circ)$, a homomorphism $\Omega : D \to E$ satisfies the following condition [17], [18]:

$$\Omega(d_1 \oplus d_2 \oplus \cdots \oplus d_P) = \Omega(d_1) \circ \Omega(d_2) \cdots \circ \Omega(d_P) \tag{16}$$

for all $d_1, d_2, \ldots, d_P \in D$, where $\circ$ is the combine operation.

More specifically, the formalism in (16) exposes a first level of (distributed memory) parallelism available for the computation of $\Omega(d)$—that is, the fundamental processing operation to be applied to the entire 3-D data stack. This computation is equivalent to performing multiple independent tasks, each involving the same operation applied to different objects or data blocks (i.e., evaluation of $\Omega(d_p)$, with $p \in \{1, 2, \ldots, P\}$), which can therefore be carried out in parallel on different computational nodes. Moreover, each object d_p can be further partitioned as follows:

$$d_p = s_{p,1} \oplus s_{p,2} \oplus \cdots \oplus s_{p,K} \tag{17}$$

with $p \in 1, 2, \ldots, P$. Accordingly, a second level of (intra-node) parallelism concerns with distributing the computation $\Omega(d_p)$ assigned to the node p across multiple GPUs of the same node:

$$\Omega(s_{p,1} \oplus s_{p,2} \oplus \cdots \oplus s_{p,K}) = \Omega(s_{p,1}) \circ \Omega(s_{p,2}) \cdots \circ \Omega(s_{p,K}) \tag{18}$$

where K represents the number of GPUs per node.

In order to exploit the different levels of parallelism of the current available heterogeneus computational platforms, the considered canonical problem is broken into the processing of small chunks (of the reshaped 3-D data stack), according to (15) and (17) to be distributed across multiple computational units. Furthermore, the transformations in (16) and (18), which hierarchical decompose the original computational problem into smaller parts for parallel execution across two distinct levels of parallelism, preserve the computational structure of the problem while ensuring that operations on the subproblems remain consistent with the structure of the original problem. The resulting multilevel parallel pattern is depicted in Fig. 3.

TABLE I
COMPUTATIONAL BREAKDOWN OF (13)

Subtask	Total Cost per image pixel
Generate $\boldsymbol{a}(s, v)$	$O(NM)$
Compute $\boldsymbol{a}^{\mathrm{H}}(s, v)\boldsymbol{x}$	$O(NM)$
Normalization by $\|\boldsymbol{x}\|$	$O(N)$
Max over M values	$O(M)$

TABLE II
SOFTWARE ENVIRONMENT

Name	Language	Version
BLAS (FlexiBLAS)	C/C++	3.0.4
Lapacke	C/C++	3
Omp (OpenMP)	C/C++	GCC OpenMP
Cublas_v2 (CUDA)	CUDA-C	CUDA 11.0
Cuda_runtime	CUDA-C	CUDA 11.0
CuComplex (CUDA)	CUDA-C	CUDA 11.0
Mpi (OpenMPI)	C/C++	4.1.2

Finally, we address shared-memory parallelism at the thread level to take advantage of multicore CPUs [14]. It is important to note that steering vectors $\boldsymbol{a}(s_m, v_m)$, which in principle should be computed on a per-pixel basis, can be approximated with reasonable accuracy on a per-tile basis, with the tile size appropriately chosen. This implies a reduction of the inherent computational cost from $O(NMN_{rg}N_{az})$ to $O(NMQ)$, where Q denotes the number of image tiles (see also Table I). Hence, the computation of the vectors $\boldsymbol{a}(s_m, v_m)$, with $m \in \{1, 2, \ldots, M\}$, can be distributed across different CPU cores (multithreading). Subsequently, once the operator Ω has been computed on a per-tile basis efficiently using multicore CPUs, it can be applied to the data (so obtaining $\Omega(\boldsymbol{x}/\|\boldsymbol{x}\|)$) on a per-pixel basis using GPU architecture.

Within this framework, the problem is systematically decomposed into independent subproblems, achieving a suitable level

Algorithm 1: Distributed Multi-GPU GLRT-Based TomoSAR Processing (High-Level Workflow).

Require: SLC stack (or data cube), grid $\mathcal{G}$, tile partition into Q independent blocks, $\mathrm{T_s}$ is the detection threshold
Ensure: Detection map and estimated parameter(s) per pixel (e.g., peak index and GLRT value)
1: **Initialization: MPI_Init**(); determine rank r and number of ranks p
2: **Resource mapping:** bind each rank to one GPU device; create CUDA streams and GPU libraries handles
3: **Memory setup:** allocate device buffers and pinned host buffers (enabling asynchronous transfers)
4: **Data decomposition:** assign a set of tiles $\mathcal{Q}_r$ to rank r
5: ▷ *Each node uses all its GPUs; tiles are independent during the main compute phase*
6: **for** each tile $q \in \mathcal{Q}_r$ **do**
7: **Load tile data** into host pinned buffer
8: **Prepare auxiliary data on host** (e.g., baselines/time vectors; operator/steering components for $\mathcal{G}$)
9: **Offload to GPU(s)** using asynchronous transfers (Host → Device)
10: **Evaluate GLRT statistic on GPU(s)** over $\mathcal{G}$ (dominant compute phase)
11: **Reduce and decide** (compute $\widehat{c}$, apply detection threshold $\mathrm{T_s}$)
12: **Retrieve outputs** using asynchronous transfers (Device → Host)
13: **Store results** for tile q
14: **end for**
15: **Final aggregation:** collect results across ranks (parallel I/O)
16: **Finalize:** release resources; **MPI_Finalize**()

of computational granularity and making the overall solution highly scalable. The adopted decomposition enables the expression of parallelism at multiple levels of the parallel execution pattern, including both (multinode) distributed memory and (intranode) shared memory parallelism, the latter encompassing both multicore and many-core architectures.

The goal of this parallel reformulation is to achieve an efficient execution by minimizing both load imbalance and parallel overhead—two critical factors that strongly impact performance in high-performance computing environments. It should be noted according to the adopted pattern, the parallel algorithm does not require communication between processes during execution in a distributed memory environment.

TABLE III
LIBRARY MODULES USED FOR OUR PROTOTYPE

Blas	Cublas	CUDA Runtime
cblas_dcopy	cublasCreate	cudaMallocHost
cblas_scopy	cublasSetStream	cudaMalloc
cblas_dscal	cublasZgemm	cudaMemcpy
	cublasDestroy	cudaMemcpyAsync
		cudaGetDeviceCount
		cudaSetDevice
		cudaGetLastError
		cudaStreamCreateWithFlags
		cudaStreamSynchronize
		cudaStreamDestroy
		cudaFreeHost
		cudaFree

D. Parallel Implementation

The fundamental implementation mechanisms required for the realization of parallelism on heterogeneous architectures are briefly discussed here.

The proposed solution exploits both distributed-memory (multinode) and shared-memory (on-node) parallelism. According to Section III-C, the input SAR image stack is initially segmented into Q tiles, each representing a contiguous rectangular region in the range-azimuth domain. This tiling strategy enables efficient data partitioning and storage across hard drives. The data volume portion corresponding to each tile can be conveniently regarded as composed of P blocks, where P is the number of available computing nodes. At the top level, the generic object d_p is then constructed by collecting one block for each of the Q tiles, exploiting data access continuity and ensuring balanced workload distribution across nodes (load balancing). To ensure flexibility, the implemented data distribution scheme also accommodates any remaining blocks resulting from a nonexact division by P. Finally, the object d_p is further decomposed to ensure that the workload is evenly distributed across the GPUs of node p, according to (17).

Some considerations regarding the practical implementation of the adopted strategy are now in order. In the realm of high-performance computing, the development of libraries for parallel and distributed computing has predominantly focused on a subset of compiled languages, including Fortran, C, and C++. In contrast, interpreted languages (e.g., Python, IDL) tend to offer suboptimal performance [10], [15].

Specifically, for the prototype implementation, we use C and Cuda-C programming languages [23], as well as various libraries for solving numerical problems and implementing parallel patterns. For the implementation of parallelism with distributed memory, we use OpenMPI [24], while we use OpenMP for the (shared memory) implementation with multithreading parallelism [14], [25]. Moreover, the used libraries include OpenBLAS, CuBLAS, CUDA Runtime [26] (see Table II and Table III).

Considering the above-described implementation decomposition, each MPI process is tied to a single GPU device, forming a multinode, multi-GPU execution in which all GPUs work concurrently on different data blocks. Within each GPU, we exploit CUDA streams and device concurrency to maximize throughput. In particular, each GPU utilizes multiple CUDA streams (two in our prototype) to overlap data transfers between the host

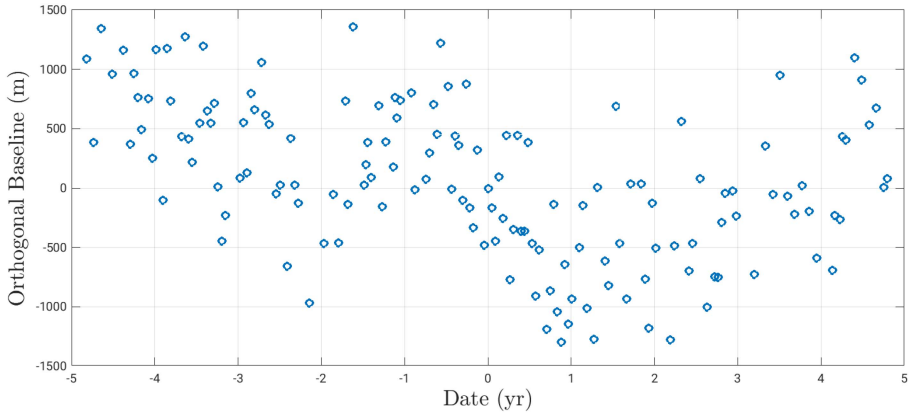

Fig. 4. Spatial-baseline/temporal-baseline distribution.

TABLE IV
SAR CSK DATASET

Parameter	Value
Orbit Direction	**Ascending**
Observation direction	**Right looking**
Polarization	**HH**
Number of images	**148**
Carrier Frequency (GHz)	**9.60**
Chirp Bandwidth (MHz)	**132.27**
Sampling Frequency (MHz)	**161.25**
Azimuth Bandwidht (kHz)	**2.63**
PRF (kHz)	**3.135**
Far Off-nadir Angle (degree)	**25.66**
Near Off-nadir Angle (degree)	**22.60**
Range Sample Spacing (m)	**0.93**
Azimuth Sample Spacing (m)	**2.21**
Range image size	**19662**
Azimuth image size	**19487**

and device with kernel executions. Algorithm 1 summarizes the overall workflow. Further implementative details are discussed in [27].

IV. Experimental Results

In this section, we first introduce a representative case study, including a description of the input dataset, the algorithm parameters used, and the resulting numerical outcomes (see Section IV-A). We then outline the computational environment (see Section IV-B) and describe the adopted parallel performance metrics (see Section IV-C). Finally, parallel performance analysis is addressed (see Section IV-D).

A. Case Study

For our experimental analysis, we employ a high-resolution X-band SAR image dataset composed of 148 SLC images acquired in Stripmap mode by the COSMO-SkyMed (CSK) constellation over the city of Salerno, Southern Italy, and its surrounding areas. The images were acquired during ascending passes of the satellites between October 2012 and May 2022, with a look angle ranging from 22.60° to 25.66°.

Each SAR image consists of approximately 383 megapixels (MP). The selected area covers about 45 km in the azimuth direction and 40 km in the range direction, including a portion of the sea and harbor areas. The main characteristic parameters of the SAR data stack are summarized in Table IV.

The spatial and temporal baseline distribution is shown in Fig. 4: the total spatial baseline span is approximately 2654 m, while the temporal baseline span is 9.6 years.

Test areas such as the selected one are particularly suitable for assessing the robustness of the detector against false alarms, as they may include nonstationary noise contributions from sea backscattering and impulsive noise generated by ships, which act as time-varying strong reflectors in certain acquisitions.

Preliminarily, the SAR dataset has been properly preprocessed, including geometric image coregistration and phase calibration to compensate for atmospheric disturbances. Moreover, a compensation of the phase terms related to the large-scale topography and the deformation contributions has been carried out according to the small-baseline subset approach operating at lower spatial resolution, see [28]. Different implementations of preprocessing operations are possible, and a detailed discussion on parallel solutions and parallel performance can be found in [10].

The GLRT-based parallel processing scheme considered in this article has been applied using the above-mentioned SAR data stack to retrieve the residual topography and deformation components of the detected PSs [7]. According to the selected baseline distribution, the resulting (theoretical) elevation and velocity resolutions are about 4.13 m (1.9 m in height) and 0.16 cm/year.

To facilitate the practical solution of the optimization problem presented in (12), the search domain was constrained by imposing bounds on the decision variables: the residual elevation span h was limited to the interval $[-a_s, a_s]$, with $a_s = 50$ m (i.e., a total range of 100 m), and the residual velocity was restricted to the interval $[-a_v, a_v]$, with $a_v = 2.0$ cm/year (i.e., a total range of 4 cm/year). We highlight that the selected elevation span h is lower than the unambiguous interval [29], which is evaluated roughly as 600 m in the considered case.

Moreover, the residual elevation and velocity bin sizes were set to 1.00 m (corresponding to a vertical binsize of 0.47 m) and 0.10 cm/year, respectively. Consequently, the number of discrete values considered is $M_s = 101$ for elevation and $M_v = 41$ for

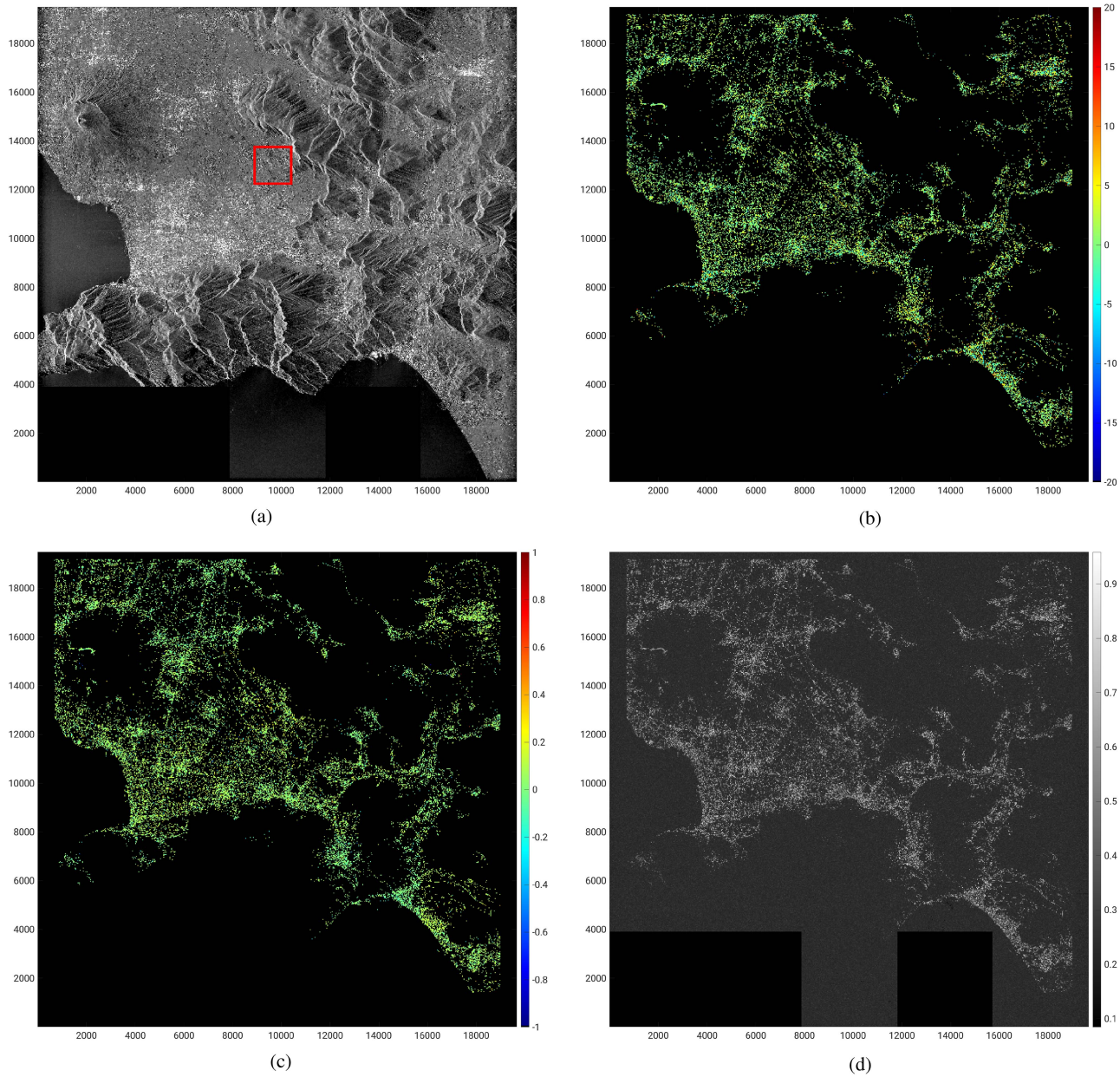

Fig. 5. Results obtained for the whole area of the considered test case. The range direction is from left to right; the azimuth direction is from bottom to top. (a) Multilooked SAR image intensity. (b) Residual topography $\widehat{s}$ (in meters) of the detected targets. (c) Residual deformation velocity $\widehat{v}$ (cm/year) of the detected targets. (d) Test statistic $\widehat{c}$.

velocity. It is worth noting that, according to the linear model introduced in Section II, the estimated residual velocity indeed represents a temporal mean value [28].

In Fig. 5, the range direction is from left to right; the azimuth direction is from bottom to top. In particular, the temporal multilook SAR intensity image of the whole area under investigation is shown in Fig. 5(a). This test site includes mountain and submontane reliefs, as well as urban and vegetated areas. From visual inspection, strong local radiometric distortions induced by reliefs can be recognized [30]. Black tiles are also noticeable at the bottom of the image, covering sea regions that have been equalized for further testing. It should be noted that urban areas and infrastructures typically provide higher coherence and are compatible with the PS model, whereas vegetated areas are more affected by temporal decorrelation. To demonstrate the practical effectiveness of the adopted canonical GLRT processing scheme in identifying PS, the resulting residual topography and residual deformation velocity of the detected targets are shown in Fig. 5(b) and (c), respectively. Finally, the test statistic $\widehat{c}$, which quantifies the degree of agreement between the adopted model and the processed data, is reported in Fig. 5(d). It should be clear that the pixels in the color-coded representation of Fig. 5(b) and (c) correspond exclusively to stable scatterers, i.e., those for which the test statistic $\widehat{c}$ exceeds the detection threshold $\mathrm{T_s}$.

The detection threshold $\mathrm{T_s}$ was set according to the desired probability of false alarm P_{fa}. The latter has been evaluated

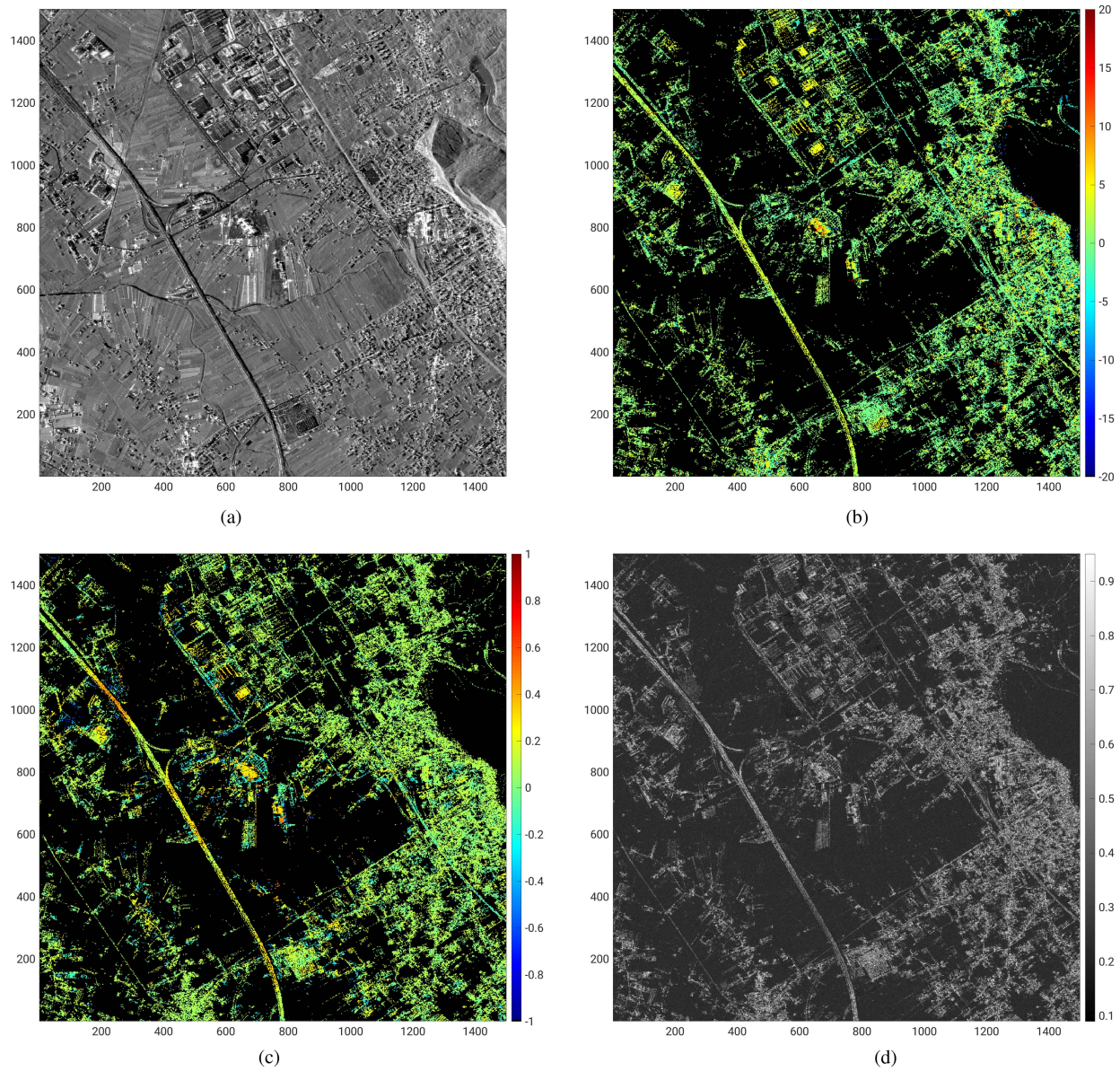

Fig. 6. Zoomed-in view of an area extracted from Fig. 5 (see the red box overlaid in Fig. 5(a)). (a) Multilooked SAR image intensity. (b) Residual topography $\widehat{s}$ (in meters) of the detected targets. (c) Residual deformation velocity $\widehat{v}$ (cm/year) of the detected targets. (d) Test statistic $\widehat{c}$.

resorting to a Monte Carlo approach, exploiting simulated data in the absence of PSs [7]. In particular, we set $P_{fa} = 10^{-5}$ corresponding to $\mathrm{T_s} = 0.36$.

To better analyze the dynamic of the measured processes and highlight the performance of the detection method, Fig. 6 provides a zoomed-in view of a small urbanized area, extracted from Fig. 5 and corresponding to the red box overlaid in Fig. 5(a). By visual inspection of Fig. 6, the correspondence between man-made structures (acting as PSs) and detected targets is also evident, confirming the high level of accuracy achieved by the GLRT method in the localization of single scatterers. In Fig. 6(c), localized areas subject to deformation can be clearly recognized, while the majority of pixels exhibit a zero residual deformation velocity (in green), as indicated by the color coding.

B. Computational Platform and Operating System

The experiments were carried out using a multinode heterogeneous HPC platform consisting of 23 nodes in an IBM cluster. Each node is equipped with a Power 9 processor that has 32 cores operating at 2.7 GHz (3.3 GHz in turbo mode) and supports four threads per core. Each node is also provisioned with 1024 GB of RAM, and 4 Nvidia V100 GPUs, each with 32 GB of memory (see Table V).

Nodes are connected via a high-speed Infiniband network, providing data transfer rates of up to 100 Gb/s. The scheme of the platform is pictorially depicted in Fig. 2. The cluster's software component is built on the Red Hat Enterprise Linux version 8.4 operating system, with workload management handled by IBM Spectrum LSF suite.

TABLE V
SPECIFICATION OF A NODE OF THE HETEROGENEUS COMPUTATIONAL PLATFORM

BEN Platform Node	CPUs	GPUs
Architecture	**2 × IBM POWER9™**	**4 × Nvidia V100**
Core Clock	**2.7 GHz**	**1530 MHz**
Number of Cores	**2 × 16**	**5120 CUDA**
Memory size	**1024 GB**	**32 GB**
Memory Bandwidth	**108 GB/s**	**897 GB/s**
Interconnection	**Mellanox connectx-5**	**PCIe 3.0 x16**

C. Performance Metrics

In order to rigorously quantify the performance characteristics of the parallel implementation under investigation, it is necessary to introduce appropriate performance metrics grounded in parallel computing theory [15], [16], [31]. Two canonical paradigms for such analysis are embodied in the notions of *strong scaling* and *weak scaling*, which constitute fundamental frameworks for the empirical and theoretical evaluation of parallel algorithms and architectures.

Strong scaling pertains to the scenario in which the total problem size W remains invariant, while the number of processing units p is increased. This effectively quantifies the ability of a parallel system to reduce the execution time of a fixed computational workload as parallel resources are augmented. The associated metric is the strong scaling speedup, denoted by S_p, and is formally defined as

$$S_p = \frac{T(W,1)}{T(W,p)} \tag{19}$$

where $T(W,p)$ represents the time required to solve a problem of size W using p parallel processing elements (PE). The numerator, $T(W,1)$, corresponds to the sequential execution time on a single processing unit, thus serving as the baseline reference.

Conversely, *weak scaling* investigates the system's capability to maintain constant execution time as both the problem size and the number of processors are increased proportionally, thereby preserving a fixed workload per processing unit. This performance indicator is particularly relevant for problems exhibiting extensive scalability or data parallelism. The weak scaling speedup $\tilde{S}_p$ is mathematically expressed as

$$\tilde{S}_p = \frac{p\,T(W,1)}{T(p\,W,p)} \tag{20}$$

where $T(pW,p)$ represents the execution time for a problem of size pW on p processing units, and $pT(W,1)$ represents the ideal scaled execution time assuming perfect linear scaling. In the ideal case of weak scaling, $\tilde{S}_p$ would also be equal to p, which means perfect scaling where the execution time remains constant as the problem size and the number of processing units scale together. Deviations from the ideal scaled speedup ($\tilde{S}_p < p$) typically arise from the parallel overheads, which can become more significant even when the workload per processing unit is kept constant as the scale increases. While the workload per processor remains constant, the total communication volume and the complexity of interprocessor dependencies can increase, leading to a decrease in efficiency at larger scales. Memory bandwidth limitations can also become a bottleneck in weak scaling scenarios.

It should be noted that the concept of weak scaling is closely related to *Gustafson's Law (GL)*, which suggests that as the number of processing units increases, the problem size can also be increased to maintain a constant execution time or even achieve a scaled speedup that grows linearly with the number of processing units, especially if the parallel portions of the workload scale effectively with the increased resources. GL emphasizes that the limitations imposed by the sequential fraction of the computation can be less restrictive if the problem size is allowed to grow with the available parallel resources [32], [33], [34].

In both scaling regimes, a further metric is provided by the concept of *parallel efficiency*, which measures the degree of resource utilization relative to the ideal linear scaling. For strong and weak scaling, the respective efficiencies ε_p and $\tilde{\varepsilon}_p$ are defined as

$$\varepsilon_p = \frac{S_p}{p}, \qquad \tilde{\varepsilon}_p = \frac{\tilde{S}_p}{p} \tag{21}$$

where ε_p quantifies the actual speedup achieved per processing unit in strong scaling, and $\tilde{\varepsilon}_p$ captures the same for weak scaling conditions. High values of efficiency (approaching 1) indicate minimal overhead and optimal scalability, whereas significant deviations reflect factors such as communication costs, synchronization overhead, and load imbalance.

This dual-scaling analysis provides a robust and comprehensive framework for evaluating the scalability and practical viability of parallel algorithms across a broad spectrum of computational architectures.

D. Scalable Performance Analysis

In this section, to assess the performance of the developed parallel algorithm, we present a thorough experimental analysis conducted on the heterogeneous computational platform introduced in Section IV-B.

We processed the entire SAR dataset described in Section IV-A, with a total memory usage of 371 gigabytes. The sequential processing time of the entire dataset requires approximately 33 h ($N_{rg} \times N_{az} = 383,153,394$, $N = 148$, $M = 4141$, $Q = 22$). As a preliminary step, we validate the implemented parallel prototype by comparing its numerical results with those produced by the sequential version of the algorithm, thereby verifying both functional correctness and numerical accuracy of the developed prototype.

A quantitative assessment of the acceleration and scalability performance of the implemented protopype is now presented. To highlight the speedup achievable by exploiting the various levels of parallelism available in the shared-memory architectures (intranode parallelism), for a prescribed chunk of the workload (33.3 h / 22 = 5443.6 s) assigned to a single computing node, we compare the corresponding execution times obtained using different incremental versions of the developed algorithm, as follows.

TABLE VI
COMPARISON OF DIFFERENT SHARED MEMORY (INTRANODE) IMPLEMENTATIONS

Implementation	Execution Time (sec)
Sequential (OpenBlas)	**5443.6**
Parallel Shared Memory (OpenMP, OpenBlas)	**294.8**
Single GPU (Cuda) with stream	**30.4**
4 GPUs (Cuda) with stream	**7.3**

First, we consider the scenario where only CPU-based parallelism is exploited across multicore processors, resulting in an execution time of 294.8 s. Next, we augment the degree of parallelism by leveraging one available GPU in addition to the multicore CPUs, which reduces the execution time to 30.4 s. Subsequently, by utilizing the full parallel capabilities of the node—including all four GPUs—the execution time further decreases to 7.3 s. For reference, the execution time of the sequential version of the algorithm is estimated at 5443.6 s.

The results are summarized in Table VI, showing the execution times for processing a prescribed workload assigned to a single computing node, for the different incremental implementations, thereby emphasizing the associated performance improvements. Overall, the full exploitation of intranode parallelism yields a speedup of approximately 743×, demonstrating the effectiveness of the proposed parallelization strategy.

This performance-oriented investigation also reflects the original and incremental methodology followed in the development of the parallel application, where each implementation increases the level of parallelism relative to its predecessor.

Table VII reports a stage-level timing breakdown for the single-GPU (1 stream) and multi-GPU (4 GPUs, streams) configurations. Specifically, we separate: i) *host-side data preparation* (baseline/time vector setup and host allocations), ii) *Host → Device memory copy* (input transfers), iii) *device compute* (cuBLAS routines and custom CUDA kernels), iv) *device reduction and decision* (reduction/selection of the GLRT statistic and final decision), and v) *Device → Host memory copy* (output transfers).

The results show that execution is strongly compute-dominated. In the single-GPU case, the mean total runtime is 30.4357 s, with device compute accounting for 29.3177 s (96.33%), while all noncompute stages contribute only 3.67% overall. In the multi-GPU case, the mean total runtime is 7.3263 s, with device compute representing 99.43% of the total time. Here, *device compute* is reported as the critical-path time, i.e., the maximum compute time across the 4 GPUs, as it determines the elapsed time of the multi-GPU execution.

To explicitly quantify load balancing across GPUs, we introduce the intranode imbalance index

$$\mathcal{I} = \frac{t_{\max} - t_{\text{mean}}}{t_{\text{mean}}} \tag{22}$$

where $t_{\max}$ and t_{mean} are, respectively, the maximum and mean per-GPU compute times within a run. Across ten multi-GPU runs, we measured $\mathcal{I}_{\text{avg}} = 3.51\%$ and $\mathcal{I}_{\max} = 8.85\%$, confirming that the adopted decomposition yields a well-balanced workload distribution across the four devices.

Subsequently, we analyze the scalable performance achieved using the multinode architecture. To quantitatively demonstrate the benefits of the implemented distributed parallelism, the metrics introduced in Section IV-C are employed. Performance evaluation was conducted over ten independent runs, with the average execution time used to evaluate representative metrics.

Specifically, the speedup obtained using the aforementioned heterogeneous HPC platform is evaluated under both strong and weak scaling scenarios (see Section IV-C), see Fig. 7. In all distributed experiments, each computing node is equipped with four GPUs (see Table V), and our implementation uses *all 4 GPUs available per node*. Therefore, if p denotes the number of computing nodes, the total number of GPUs used is $4p$. The scaling results reported in Fig. 7 are presented as a function of p (computing nodes), with a fixed baseline of four GPUs per node. It is worth noting that internode communication is limited to the initial distribution of independent data blocks and the final collection of results; the dominant compute phase executes without internode exchanges. We instrumented all MPI calls and observed that the aggregate MPI time remains negligible compared to the device-compute time, hence it does not materially affect the observed scaling behavior in Fig. 7.

Fig. 7(a) illustrates the experimentally measured speed-up S_p in a strong scaling scenario, where the entire data stack is processed while varying the number of engaged computational nodes. The ideal (linear) speedup, which represents a theoretical upper bound for paralell performance, is also shown (dashed line) as a reference. Similarly, the associated efficiency, ε_p, is depicted in Fig. 7(c).

We can observe substantial speed-up using the proposed implementation, with the experimental behavior being very close to the ideal one. The inherent discrepancy can be ascribed to communication overhead, load imbalance among computational nodes, and synchronization costs, which are typical sources of inefficiency in distributed parallelism.

In the weak scaling scenario, the obtained speedup $\tilde{S}_p$ and the corresponding efficiency $\tilde{\varepsilon}_p$ are presented in Fig. 7(b) and (d), respectively. As a result, the evaluated metrics exhibit an almost linear progression as the number of nodes increases, indicating that as both the problem size and computational resources increase, the parallel approach showed excellent overall speedup and scalability.

Therefore, a speedup of approximately 16× was achieved by exploiting the distributed (multinode) architecture, consisting of 23 nodes (92 GPUs), see Fig. 7(a). In conclusion, by combining the speedups from intranode and multinode parallelism, a total speedup of $743 \times 16 = 11\,888$ is achieved by fully exploting all the different levels of parallelism offered by the used heterogeneous computational platform, resulting in a reduction of the total execution time for processing the entire SAR dataset from 33.3 h to approximately 10 s.

E. Final Remarks

We presented the design, implementation, and scalable performance evaluation of a novel parallel GLRT-based tomographic processing scheme targeted to run on large distributed multinode, CPU-GPU heterogeneous architectures.

TABLE VII
STAGE-LEVEL TIMING BREAKDOWN FOR THE SINGLE-GPU AND MULTI-GPU CONFIGURATIONS

Stage	Single GPU [s]	Single GPU [%]	Multi GPU [s]	Multi GPU [%]
Host-side data preparation	0.2850	0.94	0.0000	0.00
Host→Device memory copy	0.0293	0.10	0.0280	0.38
Device reduction and decision	0.8006	2.63	0.0103	0.14
Device→Host memory copy	0.0031	0.01	0.0034	0.05
Device compute (cuBLAS + custom kernels)	29.3177	96.33	7.2846	99.43
Total	30.4357	100.00	7.3263	100.00

Percentages are relative to the corresponding total runtime.

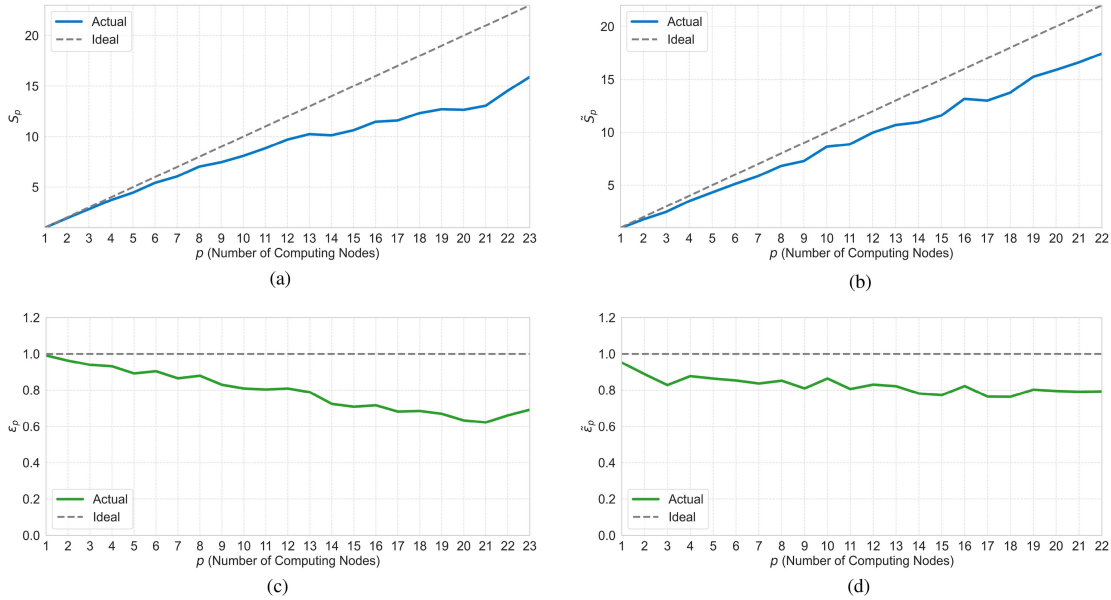

Fig. 7. Parallel performance results: (a) (Strong scaling) speedup, S_p, (b) (Weak scaling) speedup, $\tilde{S}_p$, (c) (Strong scaling) efficiency, ε_p, (d) (Weak scaling) efficiency, $\tilde{\varepsilon}_p$, as a function of the number of the engaged computing nodes p. Reference ideal speedup and ideal efficiency are also depicted (dashed lines). Each node is equipped with 4× V100 (Table V); p denotes number of nodes; each data point uses $4p$ GPUs. The reference configuration $p = 1$ corresponds to 1 node ≡ 4 GPUs. (a) Strong scaling—Speedup. (b) Weak scaling—speedup. (c) Strong scaling—Efficiency. (d) Weak scaling—Efficiency.

One of the key inherent challenges in designing a parallel scheme for the problem at hand on distributed platforms is the potential load imbalance among PEs, which can significantly degrade parallel efficiency. In multi-GPU environments, an additional challenge arises from the communication overhead between GPUs. The proposed multilevel algorithm addresses both issues by employing an effective data-parallel strategy that effectively distributes the workload, thereby achieving balanced load distribution across the available PEs.

Quantitative results from the experiments demonstrate that the proposed multilevel parallel scheme achieves excellent scalability and consistently high parallel efficiency across heterogeneous and distributed computing architectures. To the best of our knowledge, this represents the first demonstration of applying massively parallel computing to large-scale SAR tomographic processing on GPU-based distributed architectures. The proposed multilevel parallel framework fully exploits the available computational parallelism and thus offers a scalable and efficient solution for leveraging advanced heterogeneous supercomputing platforms.

V. CONCLUSION AND FUTURE DEVELOPMENTS

This article presents a parallel computing strategy for accelerating the detection of single scatterers in SAR tomography, specifically designed for multinode, multi-GPU distributed computational systems. The proposed multilevel parallel algorithm, which implements a canonical GLRT technique, is developed following HPC methodologies, thereby enabling hierarchical exploitation of the different levels of available parallelism. Special emphasis is placed on the methodological development involved in the parallel reformulation of the problem, including formal problem decomposition, expression of parallelism across multiple levels of the execution pattern, and the corresponding parallel implementation. Experiments were conducted using a large high-resolution SAR dataset and a powerful HPC platform hosted by CNR at Naples, Italy. The parallel performance analysis quantitatively confirmed the effectiveness of the proposed solution, demonstrating excellent speedup and scalability.

The developed parallel prototype thus represents a valuable tool for leveraging heterogeneous high-performance computing systems, enabling large-scale tomographic SAR analysis.

Future work will be focused on large-scale applications of the parallel prototype using large volumes of data from current and upcoming SAR missions, such as those from the NISAR platform [35]. Another promising direction consists in extending the proposed framework to support advanced GLRT formulations, thereby enhancing its applicability to more complex detection scenarios [8], [9]. Specifically, future research will investigate how the proposed parallel strategy can be extended

to cases involving double or multiple scatters detection. Finally, the exploitation of the scalable performance of the developed prototype by using advanced computational platforms—such as the Leonardo pre-exascale supercomputer Leonardo hosted by CINECA at the Bologna Technopole (Italy) that ranks ninth in the TOP500 global list [11]—represents an additional opportunity for future investigation.

ACKNOWLEDGMENT

The authors would like to thank Simona Verde for her help in setting up the SAR dataset.

REFERENCES

[1] P. Berardino, G. Fornaro, E. Sansosti, and R. Lanari, "A new algorithm for surface deformation monitoring based on small baseline differential SAR interferograms," *IEEE Trans. Geosci. Remote Sens.*, vol. 40, no. 11, pp. 2375–2383, Nov. 2002.

[2] A. Ferretti, C. Prati, and F. Rocca, "Nonlinear subsidence rate estimation using permanent scatterers in differential SAR interferometry," *IEEE Trans. Geosci. Remote Sens.*, vol. 38, no. 5, pp. 2202–2212, Sep. 2000.

[3] A. Reigber and A. Moreira, "First demonstration of airborne SAR tomography using multibaseline L-band data," *IEEE Trans. Geosci. Remote Sens.*, vol. 38, no. 5, pp. 2142–2152, Sep. 2000.

[4] G. Fornaro, F. Serafino, and F. Soldovieri, "Three dimensional focusing with multipass SAR data," *IEEE Trans. Geosci. Remote Sens.*, vol. 41, no. 3, pp. 507–517, Mar. 2003.

[5] F. Lombardini, "Differential tomography: A new framework for SAR interferometry," *IEEE Trans. Geosci. Remote Sens.*, vol. 43, no. 1, pp. 33–47, Jan. 2005.

[6] G. Fornaro, D. Reale, and F. Serafino, "Four-dimensional SAR imaging for height estimation and monitoring of single and double scatterers," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 1, pp. 224–237, Jan. 2009.

[7] A. De Maio, G. Fornaro, and A. Pauciullo, "Detection of single scatterers in multidimensional SAR imaging," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 7, pp. 2284–2297, Jul. 2009.

[8] A. Pauciullo, D. Reale, A. De Maio, and G. Fornaro, "Detection of double scatterers in SAR tomography," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 9, pp. 3567–3586, Sep. 2012.

[9] A. Budillon and G. Schirinzi, "GLRT based on support estimation for multiple scatterers detection in SAR tomography," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 9, no. 3, pp. 1086–1094, Mar. 2016.

[10] P. Imperatore, A. Pepe, and E. Sansosti, "High performance computing in satellite SAR interferometry: A critical perspective," *Remote Sens.*, vol. 13, no. 23, 2021, Art. no. 4756.

[11] "TOP500 global rankings," Accessed: May 15, 2025. [Online]. Available: https://top500.org/lists/top500/2024/11/

[12] Z. Wu, P. Ma, X. Zhang, and G. Ye, "Efficient management and processing of massive INSAR images using an HPC-based cloud platform," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 17, pp. 2866–2876, 2024.

[13] M. Russo, M. Nisar, M. Lapegna, D. Romano, A. Pauciullo, and P. Imperatore, "Detection of single scatterers in SAR tomography using multi-GPU platforms," in *Proc. Joint Urban Remote Sens. Event*, 2025, pp. 1–4.

[14] B. Chapman, G. Jost, and R. Van Der Pas, *Using OpenMP: Portable Shared Memory Parallel Programming*. Cambridge, MA, USA: MIT Press, 2007.

[15] T. Sterling, M. Anderson, and M. Brodowicz, *High Performance Computing: Modern Systems and Practices*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2017.

[16] R. Robey and Y. Zamora, *Parallel and High Performance Computing*. Shelter Island, NY, USA: Manning, 2021.

[17] D. B. Skillicorn, *Foundations of Parallel Programming*, vol. 6. Cambridge, U.K.: Cambridge Univ. Press, 1994.

[18] P. Imperatore, A. Pepe, and R. Lanari, "Spaceborne synthetic aperture radar data focusing on multicore-based architectures," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 8, pp. 4712–4731, Aug. 2016.

[19] P. Imperatore and E. Sansosti, "Multithreading based parallel processing for image geometric coregistration in SAR interferometry," *Remote Sens.*, vol. 13, no. 10, 2021, Art. no. 1963.

[20] P. Imperatore, A. Pepe, and R. Lanari, "Multichannel phase unwrapping: Problem topology and dual-level parallel computational model," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 10, pp. 5774–5793, Oct. 2015.

[21] T. Reza, A. Zimmer, J. M. D. Blasco, P. Ghuman, T. K. Aasawat, and M. Ripeanu, "Accelerating persistent scatterer pixel selection for INSAR processing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 1, pp. 16–30, Jan. 2018.

[22] S. M. Kay, *Fundamentals of Statistical Signal Processing: Detection Theory*, vol. 2. Upper Saddle River, NJ, USA: Prentice Hall, 1998.

[23] "Cuda documentation," 2026. [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[24] "Openmpi," 2026. [Online]. Available: https://www.open-mpi.org/

[25] "Openmp," 2026. [Online]. Available: https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-2.pdf

[26] "Cublas," 2026. [Online]. Available: https://docs.nvidia.com/cuda/cublas/index.html

[27] M. Russo, M. Nisar, A. Pauciullo, P. Imperatore, M. Lapegna, and D. Romano, "A multi-level parallel algorithm for detection of single scatterers in SAR tomography," in *Proc. 33rd Euromicro Int. Conf. Parallel, Distrib., Netw.-Based Process.*, 2025, pp. 544–551.

[28] G. Fornaro and V. Pascazio, "SAR interferometry and tomography: Theory and applications," in *Academic Press Library in Signal Processing*, vol. 2. Elsevier, 2014, pp. 1043–1117.

[29] G. Fornaro and A. Pauciullo, "LLMSE 3-D SAR focusing," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 1, pp. 214–223, Jan. 2009.

[30] P. Imperatore, "SAR imaging distortions induced by topography: A compact analytical formulation for radiometric calibration," *Remote Sens.*, vol. 13, no. 16, 2021, Art. no. 3318. [Online]. Available: https://www.mdpi.com/2072-4292/13/16/3318

[31] D. Eager, J. Zahorjan, and E. Lazowska, "Speedup versus efficiency in parallel systems," *IEEE Trans. Comput.*, vol. 38, no. 3, pp. 408–423, Mar. 1989.

[32] M. McCool, J. Reinders, and A. Robison, *Structured Parallel Programming: Patterns for Efficient Computation*. Amsterdam, The Netherlands: Elsevier, 2012.

[33] I. Foster, *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Reading, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[34] T. G. Mattson, B. Sanders, and B. Massingill, *Patterns for Parallel Programming*. London, U.K.: Pearson Education, 2004.

[35] B. Chapman and P. Rosen, "Overview of NASA's calibration and validation activities for the NISAR mission," in *Proc. 15th Eur. Conf. Synthetic Aperture Radar*. VDE, 2024, pp. 1101–1103.

Pasquale Imperatore (Senior Member, IEEE) received the Laurea degree (*summa cum laude*) in electronic engineering and the Ph.D. degree in electronic and telecommunication engineering, from the University of Naples Federico II, Naples, Italy, in 2001 and 2010, respectively.

He is currently a Senior Research Scientist at National Research Council, Institute for Electromagnetic Sensing of the Environment, Naples. Previously, he worked with Wise S.p.A., Naples, and later with CREATE-NET, Trento, Italy. Subsequently, he joined the Department of Biomedical, Electronic, and Telecommunication Engineering, University of Naples Federico II. His research interests include microwave remote sensing and electromagnetic propagation and scattering, with particular emphasis on theoretical scattering models, waves in random layered media, synthetic aperture radar data modeling, processing, and calibration, SAR interferometry and tomography, parallel algorithms, and high-performance computing. He has authored or coauthored numerous research articles in peer-reviewed journals, book chapters, and conference proceeding.

Dr. Imperatore is a member of the Topical Advisory Panel for Geomatics and the Earth and Planetary Science Scientific Advisory Board for IntechOpen. He was the lead editor for several journal special issues focused on electromagnetics and remote sensing and as a reviewer for numerous peer-reviewed journals. He is an Associate Editor for IEEE GEOSCIENCE AND REMOTE SENSING LETTERS and an Associate Editor for *Frontiers in Imaging*. Moreover, he has coedited two books in the field of remote sensing and geospatial technology.

Massimiliano Russo received the master's degree in computer science from the University of Naples Federico II, Naples, Italy, in 2024.

He developed a strong interest in high-performance computing and algorithm optimization. After graduation, he carried out research as a Research Fellow with the Department of Mathematics of the same university, continuing the project started with his thesis. He worked on developing high-performance versions of scientific software in C/C++ using CUDA, as well as OpenMP and OpenMPI libraries.

Mehwish Nisar received the Ph.D. degree in electronics engineering from the La Sapienza University of Rome, Rome, Italy, in 2022.

She is currently working as a Research Scientist with Institute for Electromagnetic Sensing of the Environment, National Research Council of Italy. Her primary research interests include electromagnetic modeling, SAR processing, analytical and numerical scattering models, and efficient parallel algorithm development for large-scale simulations.

Marco Lapegna received the Ph.D. degree in applied mathematics and computer science from the University of Naples Federico II, Naples, Italy, in 1991.

He is currently an Associate Professor of Computer Science. His primary research interests include algorithms, data structures, and computational environments for high-performance parallel, distributed, and cloud/edge computing, with a particular focus on scientific computing. He is actively involved in numerous national and international research projects. He has organized several international workshops in the context of major scientific conferences and an Associate Editor for leading academic journals.

Diego Romano received the Ph.D. degree computational and computer science from the University of Naples Federico II, Naples, Italy, in 2012.

He has been with the CNR since 2008. He is a Senior Research Scientist with the Italian National Research Council (CNR) in Naples. His research focuses on high-performance computing (HPC), GPU computing, scientific computing, and computer graphics. His work explores the application of parallel and distributed computing to solve complex problems, including developing parallel algorithms for edge computing, enhancing the performance of numerical algorithms on multicore CPUs and GPUs, and applying AI to hyperspectral image classification.

Antonio Pauciullo (Member, IEEE) received the Dr. Eng. (Hons.) and the Ph.D. degrees in information engineering from the University of Naples Federico II, Naples, Italy, in 1998 and 2003, respectively.

Since 2001, he has been with the Institute for Electromagnetic Sensing of the Environment, Italian National Research Council, Naples, where he currently holds a position of Senior Researcher. From 2003 to 2012, he was an Adjunct Professor of digital signal processing, University of Cassino, Italy. His current research interests are in the field of statistical signal processing with emphasis on synthetic-aperture-radar processing.