UNIVERSITÀ DEGLI STUDI DI BERGAMO

Dipartimento di Ingegneria e Scienze applicate

Dottorato di ricerca in Ingegneria e Scienze applicate

3D SURVEYING, DIGITAL REPRESENTATION AND NUMERICAL ANALYSIS FOR THE DOCUMENTATION AND MONITORING OF BUILT HERITAGE AND CIVIL ENGINEERING

Relatore:

Chiar.mo Prof. Paolo Riva

Correlatore:

Chiar.mo Prof. Alessio Cardaci

Tesi di Dottorato

Pietro AZZOLA

Matricola n. 1003327

CICLO XXXVIII

Acknowledgments

I wish to thank the University of Bergamo for offering me the opportunity to participate in the doctoral program in Intercultural Humanistic Studies, coordinated by Prof. Alessandra Marini. I am deeply grateful to all the professors of the Department of Engineering and Applied Sciences of the University of Bergamo, who have supported and guided my academic journey up to this point. I am especially indebted to the professors who directly supported me in the development of this dissertation: the supervisor Prof. Paolo Riva and the co-supervisor Prof. Alessio Cardaci. Finally, I would also like to express my gratitude to all my colleagues, whose presence and collaboration have enriched my entire course of studies.

To all those who, through the dignity of their labor and the integrity of their service, work for the advancement of peace among peoples and nations.

Contents

Introduction **1**

Chapter 1. Theoretical framework and state of the art **5**

1.1 Surveying techniques for built heritage 5

1.2 Data acquisition in Structural Health Monitoring 6

1.2.1 Active sensing technologies: laser scanning and LiDAR 6

1.2.2 Vision-based passive sensing techniques 7

1.2.3 Hybrid approaches and multi-scale fusion 7

1.3 The point cloud as an analytical dataset 8

1.3.1 Geometric interpretation and dimensional analysis 8

1.3.2 Deformation and structural interpretation 8

1.3.3 Local and global referencing elements 8

1.3.4 Global geospatial frameworks and GNSS integration 9

1.3.5 Local coordinate systems and structural logic 9

1.4 Workflow for point cloud processing 10

1.4.1 Point cloud segmentation for structural analysis 10

1.5 Segmentation strategies 13

1.5.1 Manual and assisted partitioning via CloudCompare 13

1.5.2 Deterministic ROI extraction and geometric analysis 14

1.5.3 Deep learning paradigms: complexity vs. supervision 14

1.6 Software environments for 3D point cloud visualization and processing 15

1.6.1 Data processing and editing platforms 15

1.6.2 Image-based reconstruction environments 16

1.6.3 Mesh processing and visualization tools 16

1.6.4 Specialized platforms for survey data management 17

1.6.5 Comparative considerations 17

1.7 Automation in surveying processes: existing solutions and gaps 18

1.8 Data post-processing and dimensional analysis in built heritage studies 19

1.9 Geomorphometry and curvature analysis 21

1.9.1 Theoretical foundations and evolution of the discipline 22

1.9.2 Differential descriptors: slope, aspect, and curvature 23
1.9.3 Scale dependencies and computational constraints 24
1.10 Research gaps in 3D surveying and representation 24

Chapter 2. Integrated surveying in built heritage: case studies 29
2.1 Framework comparison 29
2.2 Case study 1: Military village of Filon dei Mot (Stelvio Pass, Italy) 30
2.2.1 Site conditions 30
2.2.2 Description of the integrated survey methodology adopted 31
2.2.3 Discussion of the main outcomes and contributions to heritage knowledge . . 32
2.3 Case study 2: Reinforced-concrete arch bridge over the Carso river (Nembro, Italy) . 37
2.3.1 Site conditions 37
2.3.2 Integrated survey methodology 38
2.3.3 UAS acquisition design and photogrammetric parameterisation 39
2.3.4 Deliverables and Level of Detail (LoD) 39
2.3.5 Data interrogation and interpretation protocol 39
2.3.6 Experimental note: RAW vs. RGB photogrammetry 40
2.3.7 Accessibility constraints, occlusions, and data fusion 40
2.3.8 Geometric morphology and measured layout 40
2.4 Case study 3: Gleno Dam (Vilminore di Scalve, Italy) 43
2.4.1 Site conditions 43
2.4.2 Description of the integrated survey methodology adopted 43
2.4.3 UAS acquisition design and photogrammetric parameterisation 46
2.4.4 UAS multispectral survey of the basin: vegetation indices and bio-colonisation cues 50
2.4.5 Discussion of the main outcomes and contributions to heritage knowledge . . 50
2.4.6 Experimental evaluation of RAW vs. RGB photogrammetry 51
2.5 Case study 4: Geotechnical centrifugal test surveying 52
2.5.1 Aim of the survey workflow 53
2.5.2 Image acquisition protocol (numerical settings) 54
2.5.3 Processing pipeline and quality control thresholds 55
2.5.4 Derived products of the survey 55
2.5.5 Test scale and geometry (context for survey settings) 56

2.5.6 Practical considerations for replication 57
2.5.7 Limitations 57
2.6 Comparative discussion and lessons learned 57
2.7 Comparative synthesis of the case studies 59

Chapter 3. Methodological framework 63
3.1 Curvature-based geomorphometric workflow 63
3.1.1 Protocol design and data homogenization 63
3.1.2 Numerical estimation via covariance analysis 63
3.1.3 The dual-level analytical approach 63
3.1.4 Interpretation and interdisciplinary application 64
3.2 Slope and curvature analysis from XYZ data 64
3.2.1 Mathematical definition and physical meaning 65
3.2.2 Outcomes 66
3.2.3 Neighborhood-based normal estimation and surface definition 67
3.2.4 Orientation descriptors in CloudCompare: dip and dip direction 67
3.2.5 Motivation for planar sectioning alongside 3D visualization 67
3.2.6 An assisted, GUI-based workflow for section generation 67
3.2.7 Software integration and standardized management of point clouds 68
3.2.8 Implementation strategy: Python-driven MATLAB output for advanced plotting 68
3.2.9 Directional estimation of slope and curvature on user-defined sections 68
3.2.10 Outlook 68
3.3 A MATLAB script for analytical analysis of point cloud sections 69
3.3.1 Purpose 69
3.3.2 Inputs and key parameters 69
3.3.3 Core routine: `process_section` 74
3.3.4 Applying the routine to both epochs 76
3.3.5 Reference polyline and piecewise slopes 76
3.3.6 Visualization: three diagnostic panels 76
3.3.7 Spline sampling and `.poly` export 76
3.3.8 Excel export: sheets and variables 77
3.3.9 Interpretation of the derivatives 77
3.3.10 Assumptions and practical notes 78

3.3.11 Representation of transducers measurements along section AA' 78
3.4 Integration with Python scripting: automating data acquisition strategy 79
3.4.1 Installing PyAutoGUI and Python environment 79
3.4.2 Installing PyAutoGUI 80
3.5 Python scripting for dimensional analysis with MATLAB 80
3.6 Step-by-step explanation of the Python wizard (PyAutoGUI + Tkinter) 80
3.6.1 Goal and scope 80
3.6.2 Dependencies, imports, and environment 81
3.6.3 GUI utility functions (file and folder pickers, numeric input, yes/no) 81
3.6.4 Wizard start and input files (Initial/Final) 82
3.6.5 Primary numeric parameters (validated prompts) 82
3.6.6 Reference polyline (default vs. manual) 84
3.6.7 Output names and folder (robust path handling) 85
3.6.8 Formatting numeric vectors for MATLAB injection 85
3.6.9 User-experience and robustness notes 86
3.6.10 What follows downstream (context) 86
3.6.11 Dimensional verification against design/reference models 86

Chapter 4. Experimental applications and case studies 87
4.1 Description of selected case studies 87
4.2 Point cloud processing and dimensional analysis workflows 87
4.3 Case study of a geotechnical application 89
4.4 Arch bridge over the Carso River 106
4.4.1 Bridge arch survey: application of the algorithm 106
4.5 Parametric workflow for automated analysis of arch dam sections from raster profiles 112
4.5.1 Aim and scope 112
4.5.2 Data preparation and scale control 112
4.5.3 High-level workflow 113
4.5.4 Geometric conventions 113
4.5.5 Circle fitting: strategy and justification 114
4.5.6 Radial thickness computation 115
4.5.7 Pixel-to-Circle adherence: residuals by angle 115
4.5.8 User interaction and ROI strategy 115

4.5.9 Multi-arch batch and outputs . . . 116
4.5.10 Parameter choices and sensitivity . . . 116
4.5.11 Uncertainty and quality control . . . 117
4.5.12 Reading the results . . . 117
4.5.13 Limitations and extensions . . . 125
4.5.14 Reproducibility and data management . . . 126
4.5.15 User protocol . . . 126
4.5.16 Data analysis summary . . . 127

Chapter 5. Methodological recommendations and operational guidelines . . . 129
5.1 Methodological strengths and research contributions . . . 129
5.2 Dimensional analysis protocols using MATLAB . . . 130
5.3 Integration with open-source data formats and tools (interoperability) . . . 133

Conclusions . . . 135
5.4 Summary of main findings . . . 135
5.5 Implications for the surveying and conservation community . . . 136
5.6 Identified limitations and challenges . . . 137
5.7 Proposed future research developments . . . 137

Appendices . . . 139
Appendix A. Code listing . . . 141

List of Figures . . . 225

Bibliography . . . 229

Introduction

In recent decades, the documentation and management of built heritage have undergone a profound transformation due to the rapid development of digital surveying technologies. Architectural and structural environments, once documented through manual measurement and traditional representation, are now increasingly analyzed using three-dimensional (3D) data acquisition and processing techniques. Situated at the intersection of geomatics, architectural drawing, and numerical analysis, this research investigates how advanced 3D surveying techniques—such as terrestrial laser scanning (TLS) and digital photogrammetry—can support both geometric reconstruction and diagnostic analysis of built structures in heritage conservation and civil engineering.

The field of architectural and engineering surveying has evolved significantly from its origins in analogue photogrammetry and manual drafting. Today, integrated survey methods based on active optical sensors—such as TLS systems or airborne LiDAR—and passive techniques—such as multi-view photogrammetry from terrestrial or aerial platforms—allow for the rapid collection of vast amounts of spatial information. These data, generally encoded as 3D point clouds, have not only revolutionized how we record material culture but have also introduced new potential for quantitative evaluation and decision-making. The point cloud model, conceived as a multidimensional and multisource dataset, is no longer limited to visualization purposes; instead, it provides the basis for integrated analyses in which geometry, semantics, and performance evaluation support informed interventions on the built environment.

The increasing availability and scale of point cloud datasets have, however, generated new challenges. While acquisition technologies have become more accessible and automated, the post-acquisition processing and interpretation of geometric data remain complex and labor-intensive tasks. In particular, the transformation of raw 3D points into meaningful numerical representations—profiles, sections, deviation maps, and surface descriptors—often relies on fragmented workflows that lack standardization and are poorly integrated with diagnostic or design processes. This fragmentation becomes especially problematic in contexts where metric accuracy, procedural repeatability, and interoperability between different disciplines (e.g., surveying, structural analysis, or materials science) are essential.

In this light, the central objective of the present research is to address the methodological and technological gap between the creation of 3D point clouds and their analytical exploitation in built heritage and civil engineering contexts. The research aims to enable new connections between surveying and structural disciplines, promoting an approach where the geometry derived from digital

models can directly support maintenance, conservation, and monitoring. By fostering an integrated workflow—from survey acquisition to numerical analysis—the present work contributes to both theoretical and applied advances in the representation and interpretation of 3D data.

To this end, the thesis is structured around three core axes of development: (i) the analysis and synthesis of current survey methodologies based on TLS and photogrammetry; (ii) the establishment of a standardized workflow for extracting and analyzing metric information from 3D datasets; and (iii) the development of specific tools and scripts—primarily in Python and MATLAB—to support the processing, segmentation, and geometric comparison of point clouds.

The first axis addresses the state of the art in 3D surveying within the built environment, focusing on the technical characteristics, acquisition principles, and integration potential of active and passive optical sensor systems. Particular attention is given to their ability to generate high-resolution, dense geometric datasets suitable for detailed architectural and structural analysis. The review also highlights critical gaps in current representation methodologies, especially regarding the lack of unified visual conventions and analytical standards for point-based datasets.

The second axis proposes and tests a workflow that facilitates both graphical and numerical representation of point clouds. Central to this contribution is the application of surface geomorphometry—typically used in natural terrain modelling—to built geometry for the purpose of automated diagnostic analysis. Metrics such as slope, curvature, and topographic indices are adapted to architectural forms and reinterpreted in terms of their potential for detecting deformation, material loss, geometric deviation, and other morphological anomalies.

Finally, the third axis focuses on the development of an operational pipeline that enables the differential analysis of point clouds against reference or design geometries. Through custom scripts and graphical user interfaces designed within the CloudCompare, Python, and MATLAB environments, the research formalizes a repeatable process for extracting profiles and cross-sections, fitting analytical curves using spline-based methods, and computing variance between “as-built” and “as-designed” conditions. This contribution is validated through an array of case studies, including the analysis of a reinforced concrete bridge in Nembro and the post-load assessment of a scaled laboratory model in a geotechnical centrifuge test, each reflecting a different phase of the survey-analysis integration continuum.

The findings suggest that differential and derivative-based analysis of point clouds offers a promising pathway toward the numerical expression of geometric configurations. By quantifying curvature, gradients, and profile deviation, the workflow demonstrated in this thesis enables a shared analytical language that can link structural behaviour, material diagnostics, and graphic representation. Slope and

curvature, when understood as geometric invariants, become universal measures of shape—bridging modelling, geotechnical assessment, and structural inquiry.

Beyond technical contributions, the research also advocates for a mindset shift in how point cloud models are perceived and utilized. Rather than being viewed as static outputs of a survey campaign, point clouds are understood here as dynamic repositories of knowledge, capable of informing analytic inquiry and fostering interdisciplinary collaboration. The implications of this paradigm extend from heritage conservation policies to the design of engineering management systems, emphasizing the value of digitization not only as documentation but as the substrate for predictive and performance-based analysis.

The conclusion of the thesis reflects on the limitations of current computational routines in relation to professional practice and identifies future directions for more robust integration of artificial intelligence, automated feature extraction, and open-source pipelines. The proposed methodology is scalable and adaptable, opening a pathway for future developments in surveying education, diagnostic automation, and data-driven conservation strategies.

This doctoral research contributes to a growing field of inquiry at the crossroads of 3D surveying, technical drawing, and numerical modelling, confirming the relevance of integrated workflows for the contemporary management of built heritage and engineered structures. By rethinking the role of metric representation within the survey process and equipping it with computational tools for direct analysis, the present work affirms the essential unity between measurement, representation, and interpretation in the digital age.

Chapter 1. Theoretical framework and state of the art

1.1 Surveying techniques for built heritage

Surveying techniques applied to built heritage have progressively evolved in response to increasing demands for geometric reliability, completeness of documentation, repeatability of measurements, and multi-scale representation. Early direct methods—such as manual measurement and tachymetric surveys—provided the first systematic metric descriptions of architectural artefacts, establishing foundational procedures for geometric control and dimensional verification. In contemporary practice, however, indirect digital approaches based on image- and range-based sensing technologies have become predominant, fundamentally transforming the scale, density, and analytical potential of acquired data [1].

Image-based techniques, particularly close-range photogrammetry—including UAS-assisted acquisition—enable high-resolution surface reconstruction and radiometric documentation, facilitating detailed material and morphological analysis. Range-based methods, most notably terrestrial laser scanning (TLS), generate dense and metrically robust point clouds capable of accurately capturing complex geometries and irregular architectural configurations. The complementary characteristics of these techniques have encouraged the widespread adoption of multi-sensor acquisition strategies, integrating photogrammetric and laser-scanning datasets to enhance completeness, redundancy, and geometric reliability [2].

Recent contributions in the literature indicate that the primary challenges in heritage surveying are no longer limited to data acquisition itself, but rather concern the rigorous management of geometric accuracy, registration consistency, uncertainty quantification, and methodological reproducibility. Systematic reviews emphasize the necessity of structured control network design, explicit error modelling, and transparent reporting of accuracy indicators, particularly in applications involving deformation assessment and long-term structural monitoring [3, 4]. Such considerations reinforce the transition from descriptive documentation toward metrically validated measurement frameworks.

Moreover, ongoing research trends focus on increasing the level of automation within processing workflows, including advanced point-cloud segmentation, feature recognition, and machine-learning-assisted classification. These developments aim to reduce operator subjectivity, enhance processing efficiency, and improve the consistency of extracted geometric information while preserving metric robustness [3]. Within this evolving context, surveying is increasingly conceptualized not merely as

geometric recording, but as a structured, traceable, and repeatable measurement process supporting conservation diagnostics, structural interpretation, decay analysis, and multi-temporal comparison. Accordingly, state-of-the-art heritage surveying can be characterized by four interrelated principles: (i) multi-scale acquisition strategies; (ii) sensor integration and methodological redundancy; (iii) explicit accuracy validation and uncertainty management; and (iv) reproducible and traceable digital workflows. Collectively, these principles define the contemporary methodological framework within which built-heritage documentation and analysis are situated.

1.2 Data acquisition in Structural Health Monitoring

The digitization of the built environment has become a fundamental pillar in contemporary Structural Health Monitoring (SHM) and architectural documentation. The transition from manual measurements and point-based sensor networks to full-field monitoring is driven by the emergence of three-dimensional (3D) point cloud technology [5]. A point cloud is defined as an unordered collection of discrete 3D points within a coordinate system, where each point is characterized by spatial coordinates (X, Y, Z) and may include additional attributes such as intensity, R, G, B colour values, or thermal data [6, 7]. This chapter discusses the taxonomy of surveying techniques available to reconstruct models of reality across diverse spatial scales, ranging from localized architectural elements to expansive landscapes.

1.2.1 Active sensing technologies: laser scanning and LiDAR

Light Detection and Ranging (LiDAR), commonly referred to as 3D laser scanning, represents a predominant active sensing methodology. It operates by emitting high-frequency laser pulses and measuring the time or phase shift of the reflected signal to calculate distances with millimeter-level precision [8, 6]. LiDAR systems are categorized based on their deployment platforms, which dictate the scale of the reconstructed reality.

Terrestrial Laser Scanning (TLS)

TLS is a stationary method typically mounted on a tripod. It provides the highest accuracy and measurement rate among LiDAR systems, making it the preferred tool for high-fidelity documentation of architectural elements and buildings [8]. In construction management and facility diagnostics, TLS is utilized to capture as-built geometries, allowing for quality inspection and the detection of surface defects such as concrete spalling [6, 5].

Mobile and Airborne Laser Scanning (MLS and ALS)

For larger spatial scales, such as infrastructures and urban landscapes, mobile and airborne systems provide the necessary mobility. MLS involves sensors mounted on vehicles or robots to map road networks and tunnels [8]. Conversely, ALS—mounted on aircraft or Unmanned Aerial Vehicles (UAVs)—facilitates the acquisition of city-scale terrain models [6]. While ALS may offer lower accuracy compared to TLS, its ability to capture hard-to-reach areas, such as roofs and bridge decks, is vital for holistic structural assessments [8, 5].

1.2.2 Vision-based passive sensing techniques

Passive methodologies rely on capturing ambient light to infer 3D geometry. These techniques are highly scalable and have gained popularity due to the widespread availability of high-resolution digital cameras and advanced processing algorithms [9].

Close-range and UAS photogrammetry

Photogrammetry involves the reconstruction of 3D point clouds from a series of overlapping 2D images captured from different perspectives. The core algorithms, Structure from Motion (SfM) and Multi-View Stereo (MVS), estimate camera poses and triangulate keypoints to generate dense datasets [9, 6]. UAS photogrammetry has revolutionized the inspection of large-scale infrastructures like bridges and dams, providing a cost-effective alternative to LiDAR while offering high-resolution texture information [5, 7].

Stereo vision and RGB-D Sensing

Real-time 3D reconstruction is increasingly supported by stereo cameras and RGB-D sensors (e.g., Microsoft Kinect). These devices utilize binocular parallax or integrated depth sensors to generate coloured point clouds [6]. While their range is limited, they are instrumental in robotic navigation and the detailed dimensional analysis of indoor environments [9].

1.2.3 Hybrid approaches and multi-scale fusion

The complexity of modern infrastructure often requires a hybrid strategy to bridge the gap between macroscopic structures and microscopic defects [10]. Multi-sensor fusion acquisition integrates data from LiDAR, photogrammetry, and even thermal imagers to create a comprehensive digital twin [5].

A notable development in this field is the integration of infrared (IR) imaging with SfM. By using thermograms as input for SfM algorithms, researchers can generate 3D thermal renderings that identify subsurface defects—such as moisture infiltration or poor insulation—which are invisible in the traditional spectrum [7]. Furthermore, multi-scale robotic approaches now combine low-resolution LiDAR for global environment mapping with high-resolution laser line scanners for the sub-millimeter measurement of hairline cracks (< 0.01 mm) [10]. These hybrid workflows ensure that high-resolution data is prioritized for regions of interest (ROI), optimizing computational resources [10].

1.3 The point cloud as an analytical dataset

Regardless of the acquisition methodology, the final output shares consistent characteristics: a dense 3D point cloud. This consistency allows point clouds to serve as a common dataset for advanced quantitative analysis central to structural engineering [5, 6].

1.3.1 Geometric interpretation and dimensional analysis

Point clouds provide the raw data required for geometric morphology analysis. This involves direct measurement of structural dimensions, checking construction quality against as-designed BIM models (scan-to-BIM comparison), and performing accessibility diagnoses [6, 5].

1.3.2 Deformation and structural interpretation

For SHM, multi-temporal differential analysis is used to monitor structural changes over time. By aligning point clouds from different epochs (Registration), analysts can quantify displacements, settlements, and tilt angles [5]. The point cloud acts as a substrate for Finite Element Method (FEM) updating, enabling the simulation of mechanical performance based on as-is geometry [5, 7]. In conclusion, the homogeneity of the point cloud output enables a standardized processing pipeline—including cleaning, registration, and segmentation—that supports the complex structural interpretations required for modern asset management.

1.3.3 Local and global referencing elements

The positioning of three-dimensional data within a coherent spatial framework is a fundamental requirement for the utility of point clouds in structural analysis and reality modelling [5]. Surveying sensors generate data within varied coordinate systems depending on their internal capabilities and the presence of positioning modules. As established in recent literature, the choice between local and

global referencing is dictated by the specific analytical objectives of the survey, ranging from broad territorial mapping to high-precision structural diagnostics [8, 6].

1.3.4 Global geospatial frameworks and GNSS integration

Global referencing relies on the integration of Global Navigation Satellite Systems (GNSS) to place surveyed models within a geospatial reference framework. This is particularly prevalent in mobile and airborne platforms, such as Airborne Laser Scanning (ALS) and Mobile Laser Scanning (MLS), which incorporate GPS receivers and Inertial Measurement Units (IMU) to record absolute coordinates and system orientation (roll, pitch, and yaw) [8, 5].

The adoption of georeferenced systems offers significant advantages when datasets must be incorporated into broader territorial analyses or compared with external geospatial information. Furthermore, global referencing is essential for multi-temporal differential analysis, where point clouds acquired across different epochs must be precisely aligned to monitor structural changes, such as settlements or volumetric variations, over time [7, 5]. In scenarios where sensors lack native GNSS capabilities, global referencing can be achieved through the integration of Ground Control Points (GCPs)—distinct markers with known global coordinates that allow the transformation of the model via translation and rotation matrices [7, 6].

1.3.5 Local coordinate systems and structural logic

Despite the benefits of georeferencing, geometric and structural interpretations of the built environment often necessitate the use of local coordinate systems. Civil infrastructures and architectural elements typically exhibit high degrees of regularity, symmetry, and orthogonality, reflecting construction logics based on rectilinear components [11]. Measurements conducted within a global system can introduce unnecessary complexity when the primary goal is to evaluate the relative geometric relationships of a specific asset.

Local treatment allows for the alignment of the point cloud with the structural logic of the building. For instance, converting data from a sensor-centric coordinate system to a structural coordinate system facilitates the interrogation of dimensional characteristics along principal axes [11, 5]. This approach is highly efficient for detecting out-of-plane displacements, tilt angles, and the deformation of predominantly orthogonal elements such as beams, walls, and columns [11, 10].

1.4 Workflow for point cloud processing

The workflow proposed in this research primarily relies on the local processing of point clouds to optimize the measurement and interrogation of surveyed data. By adopting a local reference system, the analytical procedures focus on evaluating geometric characteristics along principal directions aligned with the building's layout.

This strategy supports a standardized processing pipeline—including registration, segmentation, and semantic classification—where elements are identified based on their orientation and position relative to the structural framework [6, 10]. Specifically, the use of local systems enhances the accuracy of sub-millimeter measurements, such as crack widths and surface irregularities, by ensuring that the analysis remains focused on the intrinsic geometry of the structural member rather than its global geospatial position [10, 11]. In conclusion, while global frameworks provide essential context for territorial integration, the local treatment of point clouds provides the precision and efficiency required for the advanced structural analyses central to this thesis.

1.4.1 Point cloud segmentation for structural analysis

Point cloud datasets acquired through terrestrial laser scanning (TLS), mobile mapping systems, or photogrammetric reconstruction provide dense geometric representations of built environments. These datasets are widely used in structural health monitoring (SHM), deformation analysis, and heritage documentation because they capture spatial information with high geometric accuracy and full-field coverage. Recent developments in three-dimensional sensing technologies have significantly expanded the role of point cloud data in engineering analysis workflows, enabling non-contact inspection and digital documentation of structural elements [11, 12]. In this context, segmentation represents a fundamental processing stage that enables the extraction of meaningful subsets of data from large and complex point cloud scenes.

The general processing pipeline for point-cloud-based structural monitoring typically includes data acquisition, registration, filtering, segmentation, and subsequent geometric analysis or modelling [5]. Within this pipeline, segmentation acts as an intermediate step that separates the point cloud into smaller and more manageable components based on geometric, radiometric, or semantic characteristics. As highlighted in recent reviews of point-cloud-based structural monitoring methodologies, segmentation is essential for transforming raw spatial datasets into analyzable geometric entities [5].

Role of segmentation in structural monitoring

In many structural monitoring studies, segmentation is used to identify features such as cracks, material deterioration, or geometric anomalies through automated classification algorithms. These approaches often rely on surface descriptors, colour attributes, or intensity values to classify points belonging to specific structural components or damage patterns [12]. However, the objective of segmentation within the scope of the present doctoral research differs from typical feature-detection tasks.

Rather than identifying damage patterns directly within the entire dataset, segmentation is primarily employed to isolate specific portions of the point cloud that will subsequently undergo detailed quantitative analysis. In other words, the segmentation process is intended to extract geometrically coherent subsets of data corresponding to architectural or structural elements. These subsets can then be analyzed independently using dimensional or geometric evaluation techniques. This approach allows the reduction of dataset complexity and enables more targeted analyses of structural components without relying on automated feature classification.

Such a strategy is particularly useful when dealing with dense point clouds generated from laser scanning or photogrammetric surveys. These datasets often contain millions or billions of points representing multiple structural and environmental elements. Without segmentation, performing precise geometric analysis on selected structural components becomes computationally inefficient and methodologically impractical. Consequently, segmentation can be regarded as a necessary preprocessing step that facilitates subsequent quantitative investigations.

Traditional segmentation approaches

Early segmentation approaches relied on geometric heuristics or region-based methods. Algorithms such as region growing, clustering, or model fitting are frequently employed to separate surfaces based on local geometric continuity. These methods typically exploit spatial proximity, surface normals, curvature values, or planarity indicators to group neighboring points that belong to the same structural surface.

Region growing techniques represent one of the most widely used approaches in engineering applications. Starting from seed points, neighboring points are progressively aggregated if their geometric properties satisfy predefined similarity criteria. These criteria may include thresholds on normal orientation, curvature, or point-to-plane distance. Such techniques have been applied in various studies focusing on infrastructure monitoring and geometric assessment of structural elements [13].

Another commonly used strategy involves clustering algorithms that group points based on similarity in spatial or radiometric attributes. These approaches are particularly useful when dealing with heterogeneous datasets containing multiple structural components. Clustering-based segmentation has been successfully applied in several point cloud damage detection workflows, where geometric descriptors or intensity attributes are used to distinguish damaged areas from intact surfaces [12]. While these methods can achieve reliable results, they often require careful parameter tuning and may struggle in highly complex scenes.

Deep learning-based segmentation methods

Recent advances in machine learning and artificial intelligence have introduced a new class of segmentation methods based on deep neural networks. Architectures such as PointNet and PointNet++ process raw point clouds directly and learn hierarchical geometric features for classification or segmentation tasks. These approaches have demonstrated significant potential in automatically identifying structural components and damage patterns from large datasets [14].

Deep learning-based segmentation models are capable of producing results more rapidly once trained, especially when applied to large-scale datasets. By learning geometric patterns from annotated datasets, neural networks can automatically classify points according to structural categories or damage indicators. This capability has stimulated extensive research in automated infrastructure inspection and disaster assessment [14].

Nevertheless, despite their efficiency and growing adoption, these models still present several practical limitations. Training deep neural networks requires large labeled datasets that are often difficult to obtain in engineering contexts. Moreover, the reliability of the results depends strongly on the quality and representativeness of the training data. As a consequence, automated segmentation results frequently require expert supervision, verification, and validation before they can be used in engineering analysis. For this reason, current workflows in structural monitoring often combine automated segmentation with manual verification steps to ensure the reliability of the extracted features.

Segmentation strategy adopted in this research

Considering the advantages and limitations of existing methods, this research adopts a hybrid segmentation strategy combining manual operations and region-of-interest (ROI) extraction techniques. Manual segmentation allows the precise identification of structural elements based on expert interpretation of the dataset and reference graphical information derived from architectural drawings or

survey documentation. Although this process requires a higher level of user interaction, it provides a high degree of control over the selection of structural components.

Complementary to manual segmentation, ROI-based extraction techniques are employed to isolate specific subsets of the point cloud based on geometric reference features. In this approach, graphical elements such as reference points, polylines, or surfaces are used to define spatial boundaries that delimit the portion of the point cloud to be extracted. This method enables the rapid selection of structural elements corresponding to predefined geometric references while preserving the spatial accuracy of the dataset.

The integration of these two approaches supports a workflow that is both flexible and reliable. Manual segmentation ensures accurate interpretation of complex architectural geometries, while ROI-based extraction facilitates efficient data handling when working with large point cloud datasets. Ultimately, the segmented point clouds obtained through this process serve as the basis for the subsequent geometric and dimensional analyses presented in the following sections of this study.

1.5 Segmentation strategies

In the context of SHM and architectural heritage preservation, the processing of three-dimensional (3D) datasets begins with the fundamental task of segmentation. As systematically reviewed by [5], point cloud segmentation is the process of partitioning a global dataset into distinct regions characterized by specific geometric or semantic properties. Within this doctoral research, segmentation is not merely viewed as a classification or feature detection exercise, but rather as a critical preparatory phase designed to isolate specific portions of the point cloud. These isolated sub-datasets, or Regions of Interest (ROI), are subsequently subjected to high-fidelity analytical routines to quantify structural pathologies or geometric anomalies.

1.5.1 Manual and assisted partitioning via CloudCompare

While automated workflows are increasingly prevalent, manual and assisted segmentation remain indispensable for complex datasets, particularly in the field of cultural heritage where unique architectural elements defy standardized algorithmic rules [15]. CloudCompare serves as a primary open-source platform for these tasks, providing a robust suite of tools for the interactive manipulation of 3D data.

Researchers frequently utilize CloudCompare's polyline selection and segmentation tools to manually delineate elements, a process that is essential for creating high-quality ground truth labels for subsequent machine learning evaluation [14]. Beyond purely manual intervention, assisted methods—such

as scalar field thresholding and spatial cropping—allow for the efficient extraction of structural components based on intensity, colour, or geometric descriptors. Despite being more time-consuming than fully automated approaches, manual segmentation ensures a level of precision and quality control that is currently unmatchable by unsupervised algorithms in unstructured environments [5].

1.5.2 Deterministic ROI extraction and geometric analysis

For many SHM applications, segmentation can be achieved through deterministic rule-based methods that leverage the inherent geometric properties of the point cloud, such as surface normals, curvature, and local point density. Recent advancements in automation for construction components have demonstrated that normal vector-based filtering can effectively isolate target surfaces for flatness assessment and deformation monitoring [13].

This research emphasizes ROI-based methods that extract elements by identifying reference graphic features. For instance, statistical outlier removal (SOR) and voxel-grid downsampling are utilized to regularize datasets, followed by curvature analysis to locate potential surface anomalies irrespective of the underlying geometry [12]. Such approaches are particularly effective for identifying spalling or material loss in masonry and reinforced concrete structures, where the deviation from a reference plane or fitted primitive serves as a proxy for damage [16, 13].

1.5.3 Deep learning paradigms: complexity vs. supervision

The emergence of Deep Learning (DL) has shifted the paradigm of segmentation toward end-to-end models capable of processing raw coordinates directly. Architectures such as PointNet++ and GNN-based models enable the rapid semantic segmentation of large-scale datasets, significantly reducing the time required for data processing. However, the implementation of these models introduces considerable complexity.

As noted by [14], while DL models like CrackEmbed can achieve high precision in segmenting cracked regions from disaster sites, they require extensive annotated datasets and significant computational resources. Furthermore, a critical limitation remains: the outputs of DL models often require expert supervision and validation to mitigate false positives caused by environmental noise or incomplete scanning [5]. In high-stakes SHM scenarios, the lack of transparency in "black-box" models necessitates a hybrid approach, where automated results are cross-verified against manual or geometry-based segmentations to ensure data integrity.

The segmentation strategy adopted in this thesis combines the efficiency of automated ROI extraction with manual refinement procedures. By integrating rule-based geometric filtering with assisted tools

like CloudCompare, the research enables a scalable yet accurate workflow for isolating structural elements, laying the foundation for advanced displacement and damage quantification.

1.6 Software environments for 3D point cloud visualization and processing

The rapid diffusion of terrestrial laser scanning (TLS), image-based photogrammetry, and mobile mapping systems has resulted in an exponential increase in the availability of three-dimensional (3D) spatial datasets. These point cloud datasets are widely used for applications such as structural monitoring, heritage documentation, infrastructure inspection, and environmental modelling. However, the efficient interpretation of such datasets requires specialized software tools capable of handling large volumes of spatial information and supporting different stages of the processing workflow [17].

Rather than being used independently, these tools typically form part of a broader computational ecosystem in which multiple software platforms interact within the same processing pipeline. Modern workflows commonly include stages such as data acquisition, preprocessing, registration, segmentation, model generation, and visualization. Consequently, different software environments have evolved to address specific phases of this workflow.

1.6.1 Data processing and editing platforms

A first group of tools focuses on direct manipulation and inspection of raw point cloud data. These environments provide functionalities such as filtering, registration, coordinate transformations, and statistical analysis of geometric attributes. Among them, CloudCompare represents one of the most widely adopted open-source platforms in geomatics and engineering research. The software supports a wide range of formats and provides interactive tools for editing and analyzing point clouds or triangular meshes [18]. Because of its flexibility and lightweight architecture, CloudCompare is frequently used for exploratory analysis and preprocessing operations in survey-based research workflows.

Although similar functionalities are available in several platforms, software packages differ significantly in their ability to process extremely large datasets. Tools such as ParaView, for example, are specifically designed for high-performance scientific visualization. Unlike conventional point cloud editing software, ParaView relies on parallel computing architectures and can process datasets containing billions of points. This makes it particularly suitable for multidisciplinary simulations or large-scale environmental models, although it may require greater technical expertise to operate effectively.

1.6.2 Image-based reconstruction environments

A second category of software focuses on the generation of point clouds from image datasets. These systems typically rely on Structure-from-Motion (SfM) and Multi-View Stereo (MVS) algorithms to reconstruct 3D geometry from overlapping photographs. The open-source software COLMAP is a widely used example of such pipelines. COLMAP implements a complete SfM reconstruction workflow that estimates camera poses and produces sparse and dense point clouds from unordered image collections [19]. Incremental SfM pipelines such as those implemented in COLMAP remain among the most accurate and robust approaches for image-based 3D reconstruction, although they may require substantial computational resources when applied to very large image sets.

Commercial photogrammetric platforms offer similar functionalities but typically provide more integrated workflows and optimized performance. For instance, Agisoft Metashape (formerly PhotoScan) and RealityCapture combine camera alignment, dense reconstruction, mesh generation, and texture mapping in unified software environments. Their graphical interfaces simplify parameter configuration and enable faster processing pipelines, particularly when GPU acceleration is available. Because of these characteristics, such tools are frequently adopted in professional applications including drone-based mapping, architectural documentation, and film production.

Despite their different implementation strategies, both open-source and commercial photogrammetry tools share common algorithmic foundations. They rely on feature detection, image matching, and bundle adjustment to estimate camera geometry and reconstruct the three-dimensional structure of the scene. These techniques remain central to modern image-based modelling workflows and continue to evolve through improvements in feature extraction and multi-view optimization algorithms.

1.6.3 Mesh processing and visualization tools

Beyond point cloud generation and editing, many workflows require the creation of polygonal surfaces and realistic visual representations. Mesh-oriented software environments therefore play a complementary role in the visualization and interpretation of survey results. MeshLab, for example, provides advanced tools for mesh simplification, topology correction, and texture mapping. Such capabilities are particularly important in cultural heritage documentation, where high-quality surface models are needed to represent complex architectural geometries.

Other visualization platforms emphasize rendering capabilities rather than geometric processing. Blender represents a notable example in this category. Although originally developed for computer graphics and animation, Blender has increasingly been used in scientific visualization and digital heritage reconstruction. Its physically based rendering engines allow the production of photorealistic

images and animations derived from survey datasets, supporting both communication and dissemination of 3D documentation results.

A different visualization paradigm is introduced by real-time interactive platforms such as Unity3D. In contrast to traditional modelling environments, these systems enable immersive exploration of spatial datasets through virtual and augmented reality technologies. Such approaches have recently gained attention in fields such as engineering training, infrastructure inspection, and cultural heritage communication, where interactive visualization can enhance user engagement and understanding of complex spatial environments.

1.6.4 Specialized platforms for survey data management

In addition to general-purpose tools, some software platforms are designed to manage datasets produced by specific scanning hardware. For example, Faro SCENE is optimized for processing terrestrial laser scanning data acquired with FARO instruments. Its functionalities include scan registration, multi-station alignment, and quality assessment of point cloud datasets. Similarly, Autodesk ReCap facilitates the integration of point cloud data within Building Information Modelling (BIM) environments, supporting workflows such as scan-to-BIM and digital twin creation.

These specialized environments often provide highly optimized algorithms for specific sensors or file formats. However, they may also introduce interoperability limitations due to proprietary data structures. For this reason, many workflows combine specialized acquisition software with open processing platforms such as CloudCompare or MeshLab to enable broader data exchange.

1.6.5 Comparative considerations

Although the point cloud tools serve similar overarching purposes, their design philosophies differ considerably. Open-source platforms tend to prioritize transparency and flexibility, enabling researchers to experiment with new algorithms and processing strategies. In contrast, commercial platforms typically focus on performance optimization and user-friendly interfaces, making them attractive for industrial applications where efficiency and reliability are critical.

Another important distinction concerns scalability. Some environments are optimized for editing individual point clouds or mesh models, while others are designed to process massive datasets or integrate heterogeneous information sources. As point cloud datasets continue to grow in size and complexity, future software developments will likely emphasize high-performance computing, automated feature extraction, and machine-learning-based analysis methods.

Overall, the diversity of available software platforms reflects the multidisciplinary nature of point cloud research. Rather than competing directly, many of these tools complement one another within integrated workflows that combine acquisition, processing, modelling, and visualization capabilities.

1.7 Automation in surveying processes: existing solutions and gaps

Automation has increasingly permeated surveying workflows, aiming to reduce operator dependency and accelerate data processing. Automated feature extraction, scan registration, and image orientation tools have significantly improved efficiency in digital surveying pipelines. Recent developments in photogrammetry and laser scanning software have introduced automated procedures for camera alignment, point cloud classification, and surface reconstruction, allowing large datasets to be processed with minimal manual intervention [20, 21]. In particular, modern photogrammetric pipelines based on Structure-from-Motion (SfM) and Multi-View Stereo (MVS) algorithms enable the automatic generation of dense point clouds and textured 3D models from large image collections.

Furthermore, software packages increasingly incorporate machine learning and artificial intelligence techniques to support advanced processing tasks such as semantic segmentation, automatic mesh generation, and object detection in point cloud datasets. Deep learning approaches have demonstrated promising results in automatically identifying structural elements, architectural features, and damage patterns within complex spatial datasets [22]. These approaches can significantly reduce the time required for manual classification and annotation, particularly when applied to large-scale surveying campaigns.

Despite these advances, several limitations persist, particularly in heritage contexts characterized by architectural complexity, ornamental richness, and material heterogeneity. Cultural heritage sites often present irregular geometries, high levels of decorative detail, and non-standard construction materials that challenge conventional automation algorithms. As highlighted in recent studies on 3D digitisation workflows, automated classification models frequently struggle to correctly interpret intricate geometrical patterns or to separate overlapping architectural elements [21]. Occlusions, lighting variability, and incomplete datasets may further reduce the reliability of automated reconstruction pipelines.

Another challenge concerns the quality and consistency of the datasets themselves. Heritage surveys frequently contain noise, missing regions, or inconsistencies resulting from acquisition constraints such as limited accessibility or unfavorable environmental conditions. Such issues may significantly affect the performance of automated algorithms, particularly when dealing with high-resolution point clouds derived from terrestrial laser scanning or UAS photogrammetry [5]. In these cases, expert intervention remains necessary to validate results and refine processing parameters.

Additional gaps remain within the broader surveying workflow, which is typically distributed across multiple software environments specialized in different processing stages. While some tools excel at image-based reconstruction, others are optimized for point cloud processing, visualization, or mesh editing. As a result, interoperability issues and fragmented workflows often arise, requiring manual data exchange between software platforms [23].

Post-processing operations also remain only partially automated. Tasks such as mesh cleaning, topology optimization, texture refinement, and metadata attribution still require significant user interaction, particularly when high geometric fidelity is required for documentation or conservation purposes. Consequently, current research efforts increasingly focus on the development of hybrid approaches combining automation with expert supervision. These approaches include context-aware algorithms, interactive machine learning systems, and semi-automated workflows that allow operators to guide the processing pipeline while maintaining high computational efficiency [22].

Overall, while automation has significantly improved the efficiency of modern surveying processes, the complexity of heritage environments continues to require a careful balance between automated procedures and expert-driven interpretation. Future research is therefore expected to focus on the development of adaptable AI models capable of integrating geometric, semantic, and contextual information to improve the robustness and reliability of automated survey workflows.

1.8 Data post-processing and dimensional analysis in built heritage studies

Post-processing of surveying data represents a fundamental stage in the digital documentation workflow of built heritage. Following data acquisition through terrestrial laser scanning (TLS), photogrammetry, or hybrid survey techniques, raw datasets must undergo a sequence of processing operations aimed at improving data quality and enabling meaningful geometric interpretation. Typical post-processing procedures include noise filtering, outlier removal, point cloud registration, meshing, and the derivation of geometric measurements from spatial datasets [21]. These operations transform raw spatial observations into structured datasets suitable for geometric analysis, conservation assessment, and architectural interpretation.

Dimensional analysis techniques play a central role in extracting quantitative information from three-dimensional survey models. Orthogonal projections, cross-sectional profiles, and deviation mapping are commonly employed to evaluate geometric properties and structural conditions of architectural elements. Such analyses allow researchers to derive accurate measurements of structural components, document morphological irregularities, and identify deviations from theoretical geometries. In her-

itage contexts, dimensional analysis supports not only metric documentation but also the interpretation of historical construction practices and structural transformations over time [24].

The integration of geometric analysis within digital survey workflows has significantly enhanced the ability to detect deformation phenomena and structural anomalies. Advanced software environments enable the comparison between survey datasets and reference geometries through best-fit models or parametric surfaces. Deviation analysis based on such models is particularly effective for identifying patterns of structural displacement, including wall bulging, column misalignment, or foundation subsidence in historical structures [22]. These techniques allow the quantitative evaluation of geometric discrepancies between measured surfaces and idealized reference geometries, thereby providing objective indicators of structural deformation.

Visualization tools further support the interpretation of these analyses by generating colour-coded deviation maps that highlight geometric discrepancies across architectural surfaces. Such graphical representations allow conservation professionals and engineers to quickly identify areas affected by deformation, material loss, or structural instability. When combined with statistical descriptors such as mean deviation, standard deviation, and spatial distribution metrics, deviation maps provide a robust framework for assessing structural conditions and communicating results to multidisciplinary stakeholders involved in conservation projects [25]. Consequently, these visualization techniques play an important role in bridging the gap between technical analysis and decision-making processes in heritage management.

Recent methodological developments have expanded the capabilities of post-processing techniques through the integration of advanced computational approaches. Cloud-to-cloud comparison algorithms, for instance, allow the direct comparison of two point clouds without the need for intermediate surface reconstruction. These approaches have proven particularly useful for detecting small geometric changes and monitoring deformation patterns in large datasets acquired during successive survey campaigns [5]. By comparing spatial datasets collected at different times, researchers can quantify structural evolution and identify progressive deterioration processes affecting heritage structures.

Another important methodological trend concerns the integration of multi-temporal datasets for long-term monitoring. Multi-temporal data fusion techniques combine information from repeated surveys in order to detect structural changes or environmental impacts over time. These approaches have been successfully applied to the monitoring of historical buildings, archaeological sites, and cultural landscapes, providing valuable insights into the dynamic behaviour of heritage structures under environmental and anthropogenic pressures [26]. The integration of temporal datasets further enhances the potential of digital survey methods for preventive conservation and risk assessment.

Parametric modelling environments also play an increasingly important role in post-processing workflows. Parametric and procedural modelling techniques enable complex geometries captured through survey operations to be translated into simplified analytical representations. These models facilitate the abstraction of architectural elements into geometrically meaningful components that can be analyzed within Building Information Modelling (BIM) or Heritage-BIM (HBIM) frameworks [27]. Through this process, point cloud data can be transformed into semantically structured models integrating geometric, historical, and material information, thus supporting advanced heritage documentation and conservation planning.

Despite these methodological advancements, the reliability of post-processing results remains strongly dependent on the quality of the initial survey data. Factors such as sensor accuracy, acquisition geometry, spatial resolution, and environmental conditions may significantly influence the precision of derived measurements. Moreover, operator decisions related to filtering strategies, registration methods, and modelling parameters may affect the reproducibility and reliability of the resulting analyses. Consequently, careful survey planning and methodological transparency remain essential components of any digital heritage documentation workflow.

Overall, post-processing and dimensional analysis constitute critical phases in the interpretation of digital survey data. By combining advanced computational tools with rigorous methodological frameworks, researchers can derive meaningful quantitative insights into the geometry, structural behaviour, and historical evolution of built heritage. Continued research in this area is expected to further enhance the integration of automated analysis methods, multi-temporal monitoring strategies, and semantically enriched modelling environments in support of heritage conservation and management.

1.9 Geomorphometry and curvature analysis

The numerical examination of point clouds represents a fundamental stage in digital surveying and three-dimensional representation within architectural and structural studies. Contemporary 3D datasets are characterised by a high density of spatial information, improved metric reliability, and an increasing level of internal coherence [28]. Spatial data no longer function solely as a geometric depiction of reality. Instead, they constitute a structured source of information that supports the interpretation of morphological features and spatial relationships. A point cloud should therefore not be regarded simply as a collection of discrete coordinates. Instead, it can be understood as an analytically exploitable dataset that enables analytical procedures extending beyond purely descriptive or figurative purposes, facilitating a deeper investigation of the built environment. The *point* represents the discrete sampling of a surface and, through abstraction, can be interpreted as an algebraic entity

suitable for conceptual and mathematical analysis. Consequently, the reconstructed *surface* is not merely a geometric output but becomes itself an analytical object through which spatial behaviour can be examined. First-order derivatives describe local variations within a neighbourhood and are typically associated with parameters such as slope and aspect. Second-order derivatives, in contrast, provide insight into intrinsic geometric behaviour through curvature, allowing the identification of planar areas, concave and convex regions, as well as transitional zones and geometric discontinuities.

This analytical framework originates from the tradition of differential geometry, whose foundations were laid by Carl Friedrich Gauss. In his work, Gauss demonstrated that the understanding of a surface does not primarily depend on its global configuration but rather on the way it varies locally [29]. From this perspective, a surface becomes geometrically meaningful only when its infinitesimal variations are numerically described and interpreted. Derivative-based parameters should therefore be considered not merely as additional attributes but as conceptual instruments that enable a rigorous interpretation of geometric structures. Building upon this line of thought, Bernhard Riemann emphasised that the geometric properties of space arise from local relationships between neighbouring elements, meaning that metric characteristics are determined by infinitesimal relations within the continuum [30]. Felix Klein later reinforced this conceptual shift by proposing that geometry should be interpreted as the study of invariant properties under transformation rather than the description of rigid figures [31]. Although these ideas were originally developed in a theoretical framework, they have clear implications for digital surveying. In this context, analytical tools derived from differential calculus provide the means to interpret spatial data: slope expresses the magnitude and direction of local variation, while curvature reveals intrinsic geometric behaviour and structural stability. Numerical analysis, therefore, should not be viewed merely as a computational procedure but as a methodological approach that makes explicit and measurable the infinitesimal variations which, according to the tradition of differential geometry, contain essential information about the structure of the built environment.

1.9.1 Theoretical foundations and evolution of the discipline

Geomorphometry serves as the quantitative interface between landform description and mathematical modelling. Historically rooted in the transition from qualitative geomorphology to numerical terrain analysis during the late 20th century [32], it has evolved from 2D cartographic analysis into the complex processing of high-density 3D datasets. Central to this field is the conceptualization of a surface not as a static geometric entity, but as a queryable structure whose local behaviours reveal intrinsic morphological properties [28].

This perspective is anchored in classical differential geometry, drawing heavily from the Gaussian paradigm that a surface's identity is defined by infinitesimal local changes rather than its global arrangement. In contemporary surveying, this allows point clouds to transcend their role as simple coordinate sets, becoming informative resources for diagnosing structural health and architectural transformations.

1.9.2 Differential descriptors: slope, aspect, and curvature

The characterization of a digital surface is primarily achieved through the computation of first and second derivatives.

First-order derivatives: gradient and direction

First-order descriptors include slope and aspect, which collectively represent the local gradient vector. Geometrically, the slope angle (β) defines the intensity of elevation change relative to the tangent plane. In geomorphometric workflows, the slope magnitude can be expressed as:

$$S = \sqrt{\left(\frac{\partial \zeta}{\partial x}\right)^2 + \left(\frac{\partial \zeta}{\partial y}\right)^2} \tag{1.1}$$

where ζ represents the elevation function. In architectural contexts, these descriptors are vital for identifying intentional discontinuities, such as moldings, or incidental irregularities like settlement-induced deformations [33].

Second-order derivatives: the curvature

Second derivatives provide measures of curvature, which capture the rate of change of the tangent plane and distinguish between concave, convex, and planar regions [34]. Curvature is not a single value but a set of directional metrics:

- **Profile Curvature:** Measures the rate of change of slope along the direction of the gradient, highlighting ridges and depressions.
- **Plan Curvature:** Quantifies the divergence or convergence of flow across the surface.
- **Gaussian (K) and Mean (H) Curvatures:** Derived from principal curvatures (κ_1, κ_2), these provide intrinsic shape descriptors. Gaussian curvature is defined as:

$$K = \kappa_1 \cdot \kappa_2 \tag{1.2}$$

The mean curvature is:

$$H = \frac{\kappa_1 + \kappa_2}{2} \tag{1.3}$$

These metrics are essential for detecting material loss, localized stress concentrations, and anisotropic deformation patterns in built structures [28].

1.9.3 Scale dependencies and computational constraints

A defining challenge in curvature analysis is the influence of scale and resolution. Because differential quantities depend on the configuration of neighboring points rather than isolated coordinates, the choice of neighborhood size (k or radius r) significantly impacts the stability of the estimate [35, 33].

- **Noise and Smoothing:** Second-order derivatives are highly sensitive to sensor noise. Inadequate smoothing can lead to instability, while excessive smoothing may obscure critical structural details.

- **Sampling Density:** Non-uniform point distribution, common in photogrammetric or laser scanning datasets, requires robust regularization to ensure comparable results across different survey epochs.

1.10 Research gaps in 3D surveying and representation

These technologies allow the generation of dense point clouds and high-resolution models capable of supporting a wide range of applications, from conservation planning to structural assessment and virtual dissemination [36, 37]. The increasing accessibility of these methods, together with the development of automated photogrammetric pipelines and improved scanning hardware, has progressively transformed 3D surveying into a fundamental component of contemporary heritage documentation workflows.

However, the technological maturity of these methods should not be interpreted as a state of methodological closure. On the contrary, the consolidation of TLS and photogrammetry has revealed several unresolved challenges that extend beyond the acquisition phase and concern the entire survey process, including data integration, information modelling, analysis, and long-term management of digital heritage datasets. These challenges highlight the necessity for more integrated, systematic, and standardized approaches capable of bridging the gap between data acquisition and knowledge extraction, in particular in the representation of data analysis.

A central issue concerns the integration of heterogeneous survey outputs into unified and interoperable digital frameworks at different scales and disciplines. The inherent complexity of built heritage—often characterized by irregular geometries, stratified construction phases, and heterogeneous materials—frequently requires the combined use of multiple surveying techniques, such as terrestrial laser scanning, close-range photogrammetry, and UAS-based photogrammetry. Each technique provides complementary information but also introduces specific limitations related to scale, resolution, or acquisition conditions [38, 39]. The effective fusion of these heterogeneous datasets into coherent and semantically structured three-dimensional models therefore remains a major research challenge. Addressing this issue would facilitate the development of more comprehensive digital representations capable not only of capturing geometric accuracy but also of embedding historical, material, and interpretative information within a unified digital environment.

Equally significant is the need to transform 3D survey models from static repositories of geometric data into dynamic, information-rich platforms. Contemporary research increasingly emphasizes the potential of digital models to integrate multiple layers of knowledge, including conservation records, material diagnostics, historical documentation, and monitoring data acquired through sensors or diagnostic investigations [40]. In this perspective, the 3D model evolves from a purely geometric representation into a knowledge infrastructure capable of supporting interdisciplinary analysis. Nevertheless, the integration of such heterogeneous information raises several technical and methodological challenges, including software interoperability, data structuring, metadata management, and the definition of standardized protocols for updating and validating survey datasets over time.

Another critical issue concerns the limited standardization of survey workflows. In many cases, survey procedures remain strongly project-specific, with acquisition strategies, processing methods, and data interpretation practices tailored to individual case studies. While this flexibility can be advantageous in dealing with complex heritage contexts, it also limits the reproducibility and comparability of results across different projects. Several studies have highlighted the need for transparent and replicable workflows that encompass the entire survey pipeline—from acquisition planning and data registration to modelling, representation, and analytical interpretation [41]. Establishing generalized protocols would significantly enhance the scientific robustness of survey methodologies and facilitate the development of shared digital repositories and collaborative research infrastructures.

Within this framework, the role of architectural representation remains central. Despite the increasing diffusion of three-dimensional models, two-dimensional drawings derived from metric data continue to play a fundamental role in the interpretation and communication of survey results. Architectural drawings, when rigorously generated from point clouds or photogrammetric models, represent not

only graphical outputs but also analytical tools capable of synthesizing complex spatial information and supporting critical interpretation of architectural form and construction techniques. In this sense, drawing acts as an interpretative bridge between digital measurement and historical knowledge, enabling the transformation of raw geometric data into structured architectural information.

Particular attention should also be devoted to the development of analytical methodologies derived from three-dimensional survey data. Metric analyses extracted from point clouds and digital models provide essential insights into construction techniques, geometric regularities, deformation patterns, and dimensional relationships within historic structures. Such analyses are increasingly employed in structural assessments, conservation diagnostics, and the reconstruction of building phases. Nevertheless, systematic procedures for performing dimensional and geometric analyses from survey datasets are still relatively underdeveloped, and the interpretation of point cloud data often depends on manual or semi-manual procedures that limit reproducibility.

In this context, the development of computational tools capable of automating specific stages of the analytical workflow represents a promising research direction. The integration of scripting environments and numerical analysis tools with digital survey data offers new opportunities for extracting quantitative information from point clouds and derived models. Automated or semi-automated procedures may support the identification of geometric patterns, the extraction of architectural elements, and the analysis of dimensional relationships within complex architectural systems. Such approaches can significantly enhance the interpretative potential of survey data, transforming digital models into analytical instruments for architectural and engineering research.

Finally, emerging developments in digital heritage research suggest the need to further explore the integration of 3D survey data with advanced digital environments such as HBIM platforms, digital twins, and immersive visualization systems. These approaches aim to connect geometric documentation with information modelling and real-time monitoring, enabling more effective management and interpretation of heritage assets. However, the methodological integration of accurate survey data within such environments remains an open challenge, particularly with regard to data structuring, semantic enrichment, and the preservation of metric reliability.

In parallel, there is an increasingly evident need for a standardized framework for the analysis of survey outcomes that goes beyond purely visual and graphical inspection of 2D drawings or 3D renderings. While visualization remains indispensable for interpretation and communication, it is often insufficient to reveal subtle geometric features, construction defects, and irregularities embedded in the measured data. These aspects frequently cannot be inferred from the sole observation of 2D imagery and instead require a more rigorous, analytic reading of the point cloud itself, capable of quantifying deviations,

discontinuities, and geometric anomalies through reproducible metrics and dedicated processing strategies.

Such a shift also highlights the importance of multidisciplinary collaboration, since the interpretation of complex geometric evidence is inherently linked to structural behaviour, material technologies, conservation knowledge, and historical construction practices. Consequently, future research should aim to develop a methodological approach that is both multidisciplinary and multiscale, enabling the interrogation of point cloud datasets at different levels of detail and supporting the extraction of new information through innovative analytical tools.

Overall, addressing these research gaps will be essential in order to fully exploit the potential of consolidated surveying technologies and to extend their role beyond documentation. The future development of the field will increasingly depend on the ability to integrate acquisition technologies, analytical methodologies, and digital representation systems into coherent workflows capable of supporting both scientific investigation and heritage conservation practices.

Chapter 2. Integrated surveying in built heritage: case studies

2.1 Framework comparison

This chapter builds upon the theoretical framework and state-of-the-art discussion developed in Chapter 1, where contemporary surveying paradigms were framed in terms of accuracy control, multi-sensor integration, interoperability, and the disciplined transition from three-dimensional acquisition to drawing-based communication. The present chapter operationalises those principles through a structured set of case studies collected during the author's doctoral programme. Rather than isolated demonstrations, the examples constitute a curated record of survey campaigns designed to generate metrically reliable *point clouds* and *mesh models* that will form the core datasets for the numerical analyses presented in the subsequent chapters of the thesis.

The chapter is organised around four applied studies that span complementary domains of the built and physical environment: (i) a high-altitude First World War military settlement at Filon dei Mot (Stelvio Pass, Bormio, Italy), where extreme logistics, environmental hazards, and rapid landscape change demand robust and repeatable documentation workflows; (ii) an early twentieth-century reinforced-concrete arch bridge in Nembro, Italy, surveyed under severe occlusion conditions within a narrow gorge; (iii) the remains of the Gleno Dam in Vilminore di Scalve, Italy, a complex infrastructure fragment embedded in an unstable alpine valley, documented through corridor-oriented UAS photogrammetry and comparative processing strategies; and (iv) a controlled laboratory scenario in which close-range photogrammetry is used to reconstruct and compare pre-/post-test geometries in geotechnical centrifuge experiments. Together, these studies are intended to illustrate how state-of-the-art methods can be adapted to heterogeneous constraints—from inaccessible alpine ridges to confined infrastructure sites and reproducible experimental environments—while maintaining a coherent standard of metric control.

Across the case studies, particular emphasis is placed on integrated acquisition and referencing strategies that combine range-based and image-based sensing. Terrestrial laser scanning (TLS) is adopted where dense, stable metric baselines are required for architectural detail and dimensional verification, while terrestrial and UAS photogrammetry are employed to extend coverage, reduce occlusions, and enhance textural legibility. Where appropriate, GNSS and total-station networks are used to establish a shared geodetic frame (GCPs and independent check points), ensuring that multi-

source datasets are co-registered, comparable, and reusable. The chapter also documents processing decisions that directly influence the quality and analytical readiness of the outputs, including the selection of reconstruction parameters, the export of interoperable formats, and the management of resolution and scale consistency for drawing-grade deliverables.

In line with the representational focus established in Chapter 1, the chapter presents a set of graphical outcomes that demonstrate the practical results obtainable from these workflows. Beyond immersive 3D inspection, the models are deliberately translated into two-dimensional products that remain central to architectural and engineering communication: orthophotos, elevations, and sections, together with axonometric and perspective visualisations. In the Filon dei Mot study, for instance, co-registered TLS–UAS point clouds and textured meshes support the production of high-resolution plates and "tomographic" Monge-derived views that enhance the reading of out-of-plane deviations and wall deformations. In the Nembro bridge and Gleno Dam studies, disciplined 3D→2D pipelines yield drawing-scale outputs (e.g., 1:50) suitable for metric annotation, comparative readings against historical documentation, and the mapping of discontinuities. In the centrifuge case, the generation of strictly co-registered pre-/post-test meshes enables quantitative differencing and the extraction of displacement fields, slope/gradient indicators, and section-based measurements.

The ultimate function of this chapter is therefore twofold. First, it establishes a transparent and replicable methodological foundation for producing coherent, metrically controlled 3D datasets from heterogeneous survey contexts, providing concrete evidence of achievable outputs, limitations, and best practices. Second, it prepares the analytical transition that motivates the remainder of the thesis: the point clouds and mesh surfaces produced in the present chapter are not treated solely as representational artefacts, but as *analysis-ready geometries*. They will serve as the input for the differential and curvature-based investigations developed in the following chapters, where local surface behaviour—expressed through derivatives, slope, and curvature—will be quantitatively examined to support morphological interpretation, diagnostic assessment, and comparative evaluation of the built environment.

2.2 Case study 1: Military village of Filon dei Mot (Stelvio Pass, Italy)

2.2.1 Site conditions

The Filon dei Mot military village occupies a precipitous ridge above the Stelvio Pass within the Ortles-Cevedale massif, a theatre of the so-called "White War" (1915-1918) where extreme altitude and climate conditioned military logistics, settlement patterns, and constructional choices [42]. Italian

defences in this sector privileged forward positions and lightweight works rather than massive forts, resulting in a network of small "military villages" linked by trenches and firing posts; two nuclei were established, one on the Filon dei Mot crest and another on the Buse plain below, with layouts adapted to topography (irregular, ridge-conforming plan at Filon; circular scheme at Buse) (Figure 1) [42]. The ridge location—though spatially constrained—was selected for offensive/defensive advantages: continuous visual control of enemy movement and favourable artillery geometry from elevation [42]. *In situ* remains document accommodations and service structures (dormitories, armoury, infirmary, canteen, kitchen with oven, cistern), largely in dry-stone masonry with internal timber linings and insulating fillings, protected on the eastern side by a substantial wall with loopholes; trench walkways connected the village to artillery emplacements, the cableway to Buse, and a circular lookout post [42]. The site's historical visibility has increased as glacier retreat progressively exposes artefacts, underscoring the urgency of documentation and conservation planning [42].

Figure 1: Stelvio Pass: overview of high-altitude military works. Composite view with a satellite basemap highlighting the principal First World War installations in the Ortles–Cevedale sector, including *Filon dei Mot*, *Le Buse*, *Rese Alte Scorluzzo*, *Rese Basse Scorluzzo*, *Riparo Scorluzzino*, *Fort Goldsee*, *Goldsee Stellung*, and *Lempruch Lager*. The lateral photographic strips provide on-site views and documentary frames that contextualise the remains and illustrate their current state of preservation. The figure establishes the spatial framework adopted in the chapter and locates the Filon dei Mot survey within its territorial setting and historical landscape. *Image credit:* [42].

2.2.2 Description of the integrated survey methodology adopted

Fieldwork was scheduled within the narrow seasonal window between late-summer thaw and early-winter snowfall (late summer–early autumn 2022), balancing access, safety, and data quality [42]. Given high-altitude hazards (steep, unstable slopes, permafrost, strong wind, high radiance) and lo-

gistical constraints (instrument transport, thermal stress), the methodology combined range-based and image-based sensing with geodetic control: terrestrial laser scanning (TLS) for detailed architecture and surfaces; UAS-based and terrestrial photogrammetry for global coverage and occlusion reduction; and a GNSS/total-station control network for robust georeferencing [42, 43, 44]. Technical measures mitigated environmental effects (e.g., managing glare and snow reflectance; stabilising instruments in low temperatures), while GNSS data were transformed into a local Cartesian frame to align all datasets and handle discrepancies between WGS84 and local datums [42]. Data fusion produced co-registered point clouds, textured meshes, and a Digital Elevation Model (DEM); a dedicated "tomographic" workflow generated Monge double-projection views from the 3D model which, when superimposed on textured orthophotos, enhanced detection of wall deformations and out-of-plane displacements relevant to conservation diagnostics [42, 45, 46, 47, 48].

2.2.3 Discussion of the main outcomes and contributions to heritage knowledge

The integrated survey yielded a high-resolution, metrically controlled 3D model of the Filon dei Mot complex that exceeds the completeness attainable by single-technique campaigns (Figure 2) [42]. Derived deliverables included plans, elevations, and cross-sections suitable for 1:50 representation, with reported thickness accuracy on the order of ±1.5 cm and ground sampling distances of approximately 0.5 mm in detailed outputs [49]. Such resolution supported systematic mapping of construction techniques (dry-stone walling with insulated cavities, timber-lined interiors, lightweight roofing assemblies), functional zoning, and circulation across multi-level stair-linked terraces [42]. The combined TLS–UAS–DEM products enabled targeted assessments of deformation patterns, local instabilities, and weathering/decay processes, informing prioritisation for conservation and risk mitigation [42, 43, 44, 46]. From an interpretative standpoint, tomographic/Monge visualisations improved the analytical reading of geometry, facilitating identification of out-of-plane anomalies and stratigraphic relationships otherwise obscured in conventional shaded models [42, 47, 48]. Beyond documentation, the model functioned as a multidisciplinary knowledge platform by integrating historical-cartographic evidence and enabling the communicative functions of drawing—communication, documentation, and valorisation—in an environment that is physically remote and operationally hazardous [42]. Finally, the work situates Filon dei Mot within broader environmental dynamics: the exposure of remains due to glacial retreat provides research opportunities while posing preservation challenges, reinforcing the value of repeatable, standardised workflows for long-term monitoring and public engagement [42].
Data processing for both terrestrial laser scanning (TLS) and photogrammetry was performed using proprietary software (FARO SCENE and Agisoft Metashape), with computational parameters cali-

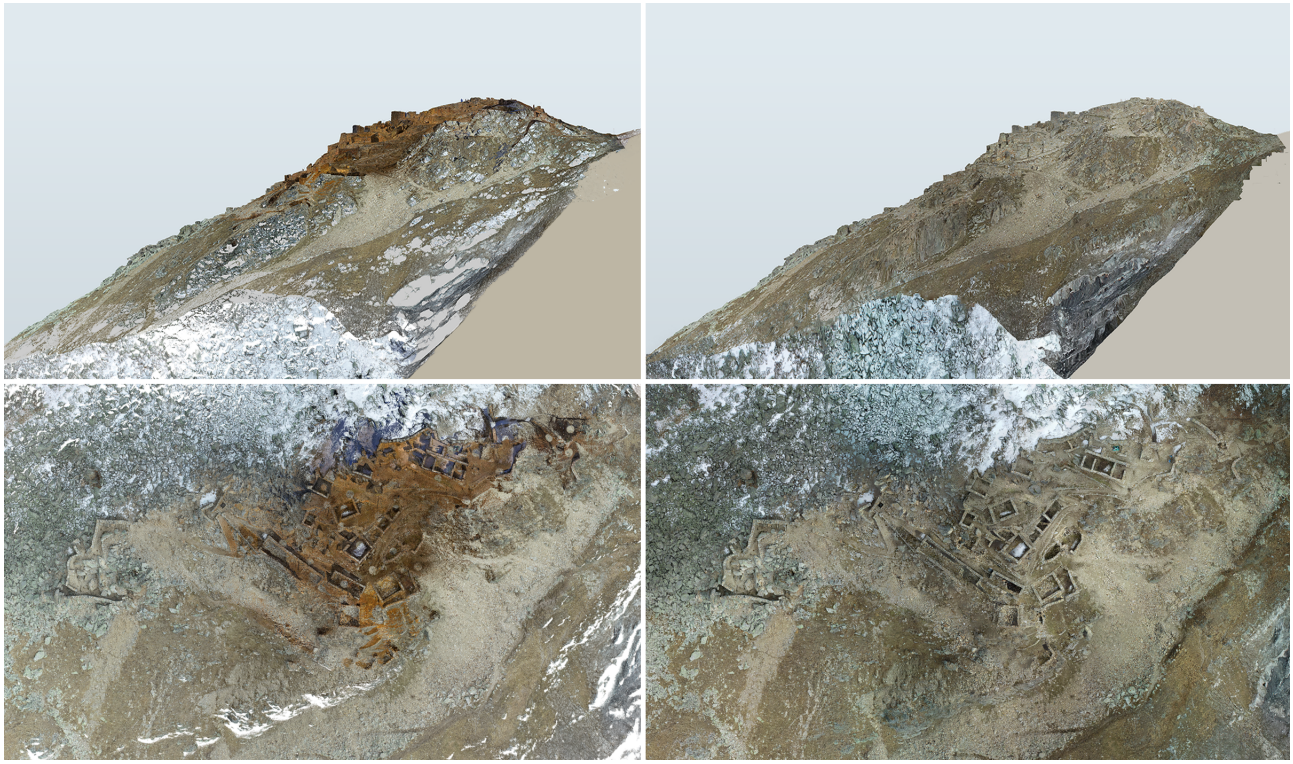

Figure 2: Filon dei Mot: oblique and planimetric views from the integrated survey. Composite plate showing four co-registered visualizations exported as high-resolution images from the integrated TLS–UAS workflow. Clockwise from top left: (1) oblique composite rendering with layer-based overlays to enhance the legibility of ridge-top structures; (2) oblique natural-colour photogrammetric texture emphasising morphology and settlement distribution; (3) planimetric orthophoto highlighting the layout of buildings, terraces, and trench connections; (4) planimetric composite with controlled semi-transparency and occasional false-colour mapping to distinguish sensor provenance and to support simultaneous reading of geometric and textural information. The montage underpins the 3D-to-2D translation adopted in this chapter and provides a consistent basis for metric annotations and comparative analysis. *Image credit:* [42].

brated to the required level of detail and, correspondingly, to the intended scale of representation. This tuning ensured that the resulting datasets were not only visually coherent but also metrically consistent with the project's deliverables.

The three-dimensional models support immersive visual inspection; however, they yield their most robust measurable outcomes when translated into traditional two-dimensional drawing outputs (orthographic plans, elevations, and sections). In this regard, Figures 3 and 4 illustrates the analytical readings conducted on the elevations and masonry of the military village, demonstrating how the 3D-derived representations underpin precise metric assessments.

GNSS-supported photogrammetry produced a high-fidelity, photorealistic model characterized by a high level of geometric detail and excellent chromatic rendition, the latter facilitated by favourable visibility and lighting conditions during the survey campaign. The analysis of elevations was conducted by generating orthographic views and superimposing the photogrammetric outputs onto the TLS

dataset, with the dual objective of enhancing interpretability and enabling a direct metric comparison between the two results. Within this integrated framework, the TLS acquisition serves as the primary metric reference for the mountaintop structures, while the UAS-based photogrammetric survey enabled rapid coverage of a broader area, thereby providing the necessary territorial context for interpretation and multi-scalar analysis.

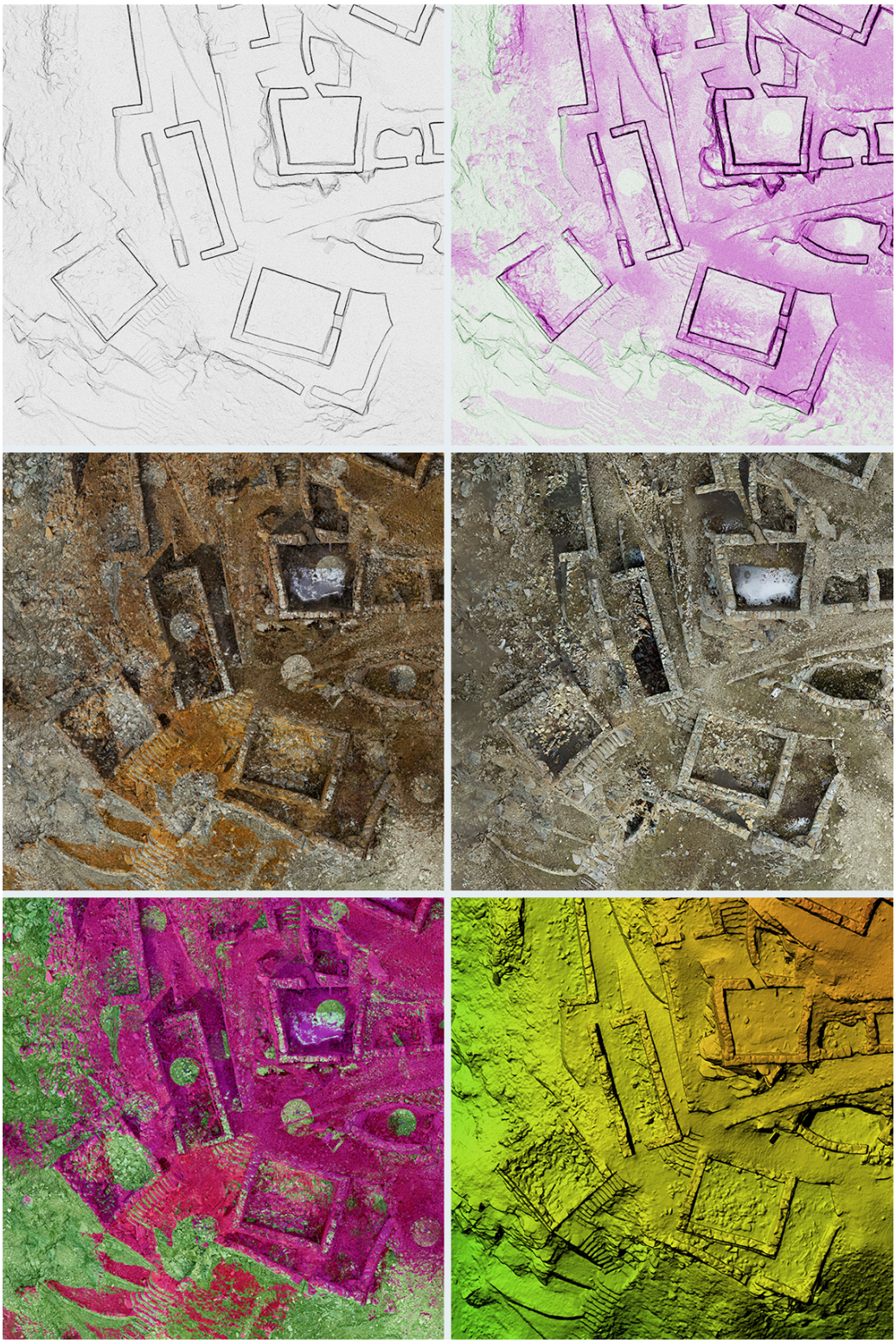

Figure 3: Filon dei Mot: orthographic elevation and masonry reading. Orthographic elevation derived from the integrated TLS–UAS dataset, exported as a high-resolution TIFF and composed through a controlled layer-based workflow. The plate synthesises photogrammetric texture and TLS-derived shading/tomographic renderings to enhance the legibility of joints, openings, discontinuities, and out-of-plane deviations along the ridge-side façades of the military village. The image provides the graphical basis for subsequent metric annotations and comparative readings discussed in the text. ***Image credit:*** **[42].**

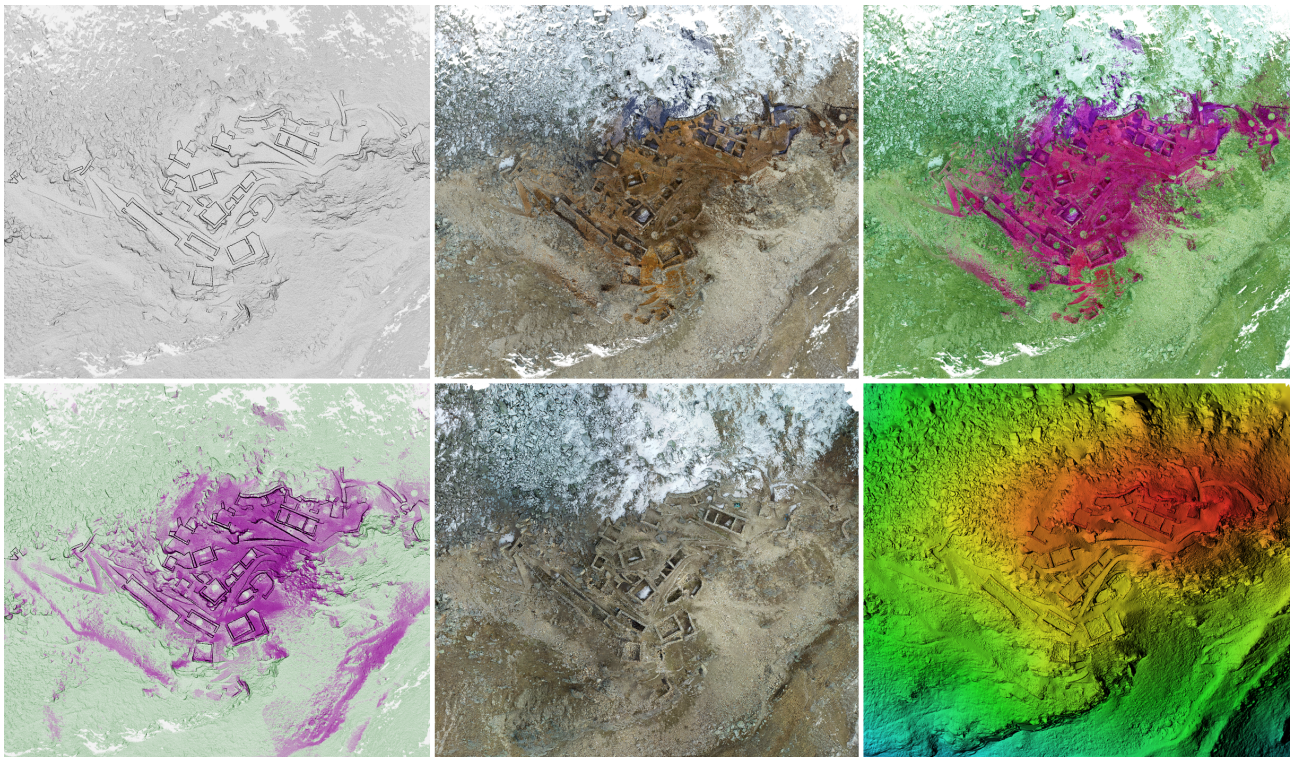

Figure 4: Filon dei Mot: planimetric orthophoto and comparative metric analysis. Top-down orthographic mosaic co-registered in a common reference frame, combining photogrammetric texture with TLS intensity information to support cross-checking of geometry and the analytical reading of terraces, circulation paths, and construction sequences. The figure exemplifies the 3D-to-2D translation adopted for publication-quality outputs and serves as a reference for the stratigraphic and deformation assessments reported in the chapter. ***Image credit:*** **[42].**

The derived models were exported as point clouds in interoperable formats (LAS, ASCII TXT, E57, PLY, among others) to ensure long-term accessibility and software independence. These datasets can be explored, queried, and measured within open-source viewers such as *CloudCompare*—which is free of charge—and, where required, within proprietary platforms that offer advanced inspection and reporting functionalities. In a Building Information Modelling (BIM) context, the point clouds may be imported to enable direct redrawing and vector tracing (e.g., polylines, section lines, reference grids), thus providing a robust basis for segmentation and semantic enrichment of architectural elements.

For publication and comparative analysis, a two-dimensional representation strategy was adopted. Orthographic views co-registered in a common reference frame were exported as high-resolution TIFF images from both the laser-scanning and the photogrammetric workflows, allowing precise overlay, visual cross-checking, and metric interrogation. The final plates were composed using a professional image-editing environment with layer management (*Adobe Photoshop*), which enabled consistent harmonisation across sources by adjusting tonal curves, applying controlled false-colour mappings, and balancing lights and shadows.

Within a rigorous, layer-centric workflow, complex image generation and enhancement were achieved by combining adjustment layers (Curves, Levels, Exposure, Hue/Saturation, Colour Balance, Selective Colour, Channel Mixer, Black & White, Gradient Map, Colour Lookup with LUTs), blending modes (Multiply, Screen, Overlay, Soft Light, Colour Dodge and Burn, Difference, Luminosity), and masking strategies (layer masks, vector masks, clipping masks, as well as luminosity masks refined via *Apply Image* and *Calculations*). Smart Objects were employed to preserve editability and to host Smart Filters (e.g., Gaussian Blur, Unsharp Mask, High Pass for detail enhancement, Camera Raw Filter for global tonal calibration, Lens and Perspective corrections), while layer styles (e.g., Stroke, Inner/Outer Glow) and Fill Layers (Solid Colour, Gradient, Pattern) facilitated non-destructive annotation and cartographic symbology.

Rather than resorting to advanced image-synthesis procedures, the compositing strategy centred on the superposition of co-registered layers exported as high-resolution TIFFs from the TLS and photogrammetric workflows. Each layer was edited individually—adjusting tonal response, local contrast, and colour balance—and blended through controlled semi-transparency (alpha) to preserve the geometric correspondence while enhancing visual discrimination among sources. This approach enabled semi-transparent composite views in which information from different sensors can be read simultaneously, with false-colour representations employed, where appropriate, to clarify provenance (e.g., TLS intensity or shaded relief) against natural-colour photogrammetric textures. The result is a set of analytically legible plates that maintain metric integrity while supporting cross-comparison and interpretative reading of the site [42, 47, 48].

2.3 Case study 2: Reinforced-concrete arch bridge over the Carso river (Nembro, Italy)

2.3.1 Site conditions

The 1912 deck-stiffened reinforced-concrete arch bridge over the Carso river in Nembro (Val Seriana) is an early Italian example of Maillart-type design, built for the Bergamo–Albino tramway. Its setting—a narrow gorge with steep flanks and dense vegetation—produces self-occlusions and mixed illumination that demand an integrated multi-sensor survey, with rigorous geodetic control and a disciplined 3D→2D translation approach for drawing-grade documentation [50, 51].

2.3.2 Integrated survey methodology

A range+image workflow combined terrestrial laser scanning (TLS) and UAS photogrammetry within a shared geodetic frame. A GNSS/total-station network materialised Ground Control Points (GCPs) and independent Quality Control Points (QCPs). TLS stations on both gorge banks captured the intrados, abutments, and under-arch shaded areas; UAS flights acquired deck, spandrels, and the environment, using mixed nadir and convergent oblique images to stabilize block alignment and rich textural modelling, especially in occluded areas [52]. Data registration followed a target-assisted workflow, with final seven-parameter rigid alignment; residuals were assessed via QCPs and planar patches. Images were processed with a "RAW-first" approach to preserve radiometric fidelity under high dynamic range; parallel RGB-derived reconstructions were conducted for controlled comparisons. Dense image-derived models and TLS clouds were fused at the point-cloud level to leverage completeness and surface detail, following integrated Cultural Heritage (CH) documentation best practices [53, 54]. Outputs were exported in interoperable formats (E57, LAZ, PLY), accompanied by metadata for reuse (Figure 5).

Figure 5: Nembro bridge: 3D model. Photogrammetric and TLS-based reconstruction of the bridge geometry. ***Image credit: Cardaci, Versaci, Azzola (2023) [50].***

2.3.3 UAS acquisition design and photogrammetric parameterisation

The target drawing scale (1:50) defined the Ground Sampling Distance (GSD) via

$$\mathrm{GSD} = \frac{p\,H}{f},$$

with flight altitude H and exposure time t_e calibrated so that ground motion blur satisfies $s_{\mathrm{px}} = \frac{v\,t_e}{\mathrm{GSD}} \ll 1$. Longitudinal and lateral overlaps were set at 80%. Convergent oblique captures (30–40°) improved angle diversity and intrados visibility [52]. Camera calibration and interior orientation fidelity were cross-checked using comparative software testing in complex environments [55]. Flight planning adhered to national and European UAS regulations for confined operations in Open A2 scenario.

2.3.4 Deliverables and Level of Detail (LoD)

Architectural outputs (at 1:50) included:

- **True orthophotos** of deck, intrados, and abutments—suitable for mapping cracks, joints, leaching/efflorescence, and bio-colonisation cues.
- **Orthogonal elevations** and longitudinal/cross-sections with vector overlays (cracks, seams) on raster backdrops to retain metric fidelity.
- **Intrados/extrados profiles** sampled at regular stationing; thickness t measured orthogonally, reported as $(t_{\min}, \bar{t}, t_{\max})$ per station.
- **Axis conformity plates**: centreline derived via spline through intrados keypoints; deviation field $\delta(x) = z_{\mathrm{surveyed}}(x) - z_{\mathrm{ideal}}(x)$ enables as-planned vs. as-built comparison [50].
- **Scalar field renderings** (e.g., Directional Illumination (DIP) / Eye-Dome Lighting (EDL) shading or curvature) on point-clouds to enhance discontinuities and support segmentation/contour extraction [54].

LoD choices for plate production (e.g., intrados façades) follow recent CH documentation guidance on drawing-scale fidelity [56].

2.3.5 Data interrogation and interpretation protocol

(i) Geometry. Plots of station-wise thickness, axis elevation, and curvature trends (haunch vs mid-span) are cross-checked with sections. Uncertainty is bounded by $\pm\varepsilon$, where $\varepsilon = \max\{\mathrm{RMSE}_{\mathrm{reg}}, 2 \cdot \mathrm{GSD}\}$, paralleling the Gleno case workflow.

(ii) Surface condition. High-frequency shading/curvature maps and true orthos are used to trace cracks, map efflorescence, and detect casting seams. When available, small multispectral overlays (plate-scale co-registration) qualitatively signal biofilm presence for targeted inspection. The synergy of range+image data enhances detectability of fine discontinuities [53, 54].

(iii) Communication. Plates are delivered as plane representations, at fixed ppi with scale bars; vector layers annotate centrelines, stationing, thickness bars, and measurement values. This disciplined 3D→2D transition ensures reproducible documentation and archivable comparison to archival design documentation [51].

2.3.6 Experimental note: RAW vs. RGB photogrammetry

A paired subset adhered to the same RAW vs RGB protocol as in Case Study 2.4. Metrics included: (a) alignment robustness (aligned images count, tie-points/image, reprojection error distribution), (b) QCP RMSE and planar patch deviations, (c) completeness/density in fixed ROIs, (d) edge acuity in orthophotos. RAW ingestion improved keypoint stability in mixed lighting and minimized occlusion voids in the intrados—producing crisper 1:50 orthos with minimal radiometric post-processing [52, 55, 54].

2.3.7 Accessibility constraints, occlusions, and data fusion

The bridge is only *partially accessible* for terrestrial and aerial acquisitions: vegetation, adjacent constructions, and gorge morphology generate persistent *self-occlusions* that prevent full visibility and stable line-of-sight, with the most critical gaps in the inter-pillar zones above the arches on the western side. Nevertheless, the *fusion* of UAS photogrammetry and TLS within a common geodetic frame provided an *as-built* geometric nucleus suitable for drawing-scale interpretation. In particular, the overlap of the photogrammetric model with the TLS cloud enabled a high-precision planimetric localisation of the spandrel pillars above the arches and a robust reconstruction of the intrados/extrados geometry of the arches, which could be directly compared to the constructive scheme documented by *Cesare Pesenti* [57] and discussed for Nembro in [50, 51].

2.3.8 Geometric morphology and measured layout

Arches. The principal bearing system is a pair of parallel reinforced-concrete arches stiffened by a deck supported through vertical spandrel pillars. TLS+UAS sections confirm a shallow parabolic-like intrados with a low rise-to-span ratio, in agreement with the graphical–analytical statics illustrated by

Pesenti [57]. Residual analysis on polynomial fits highlighted slight curvature irregularities and local offsets at the springings (Figure 6).

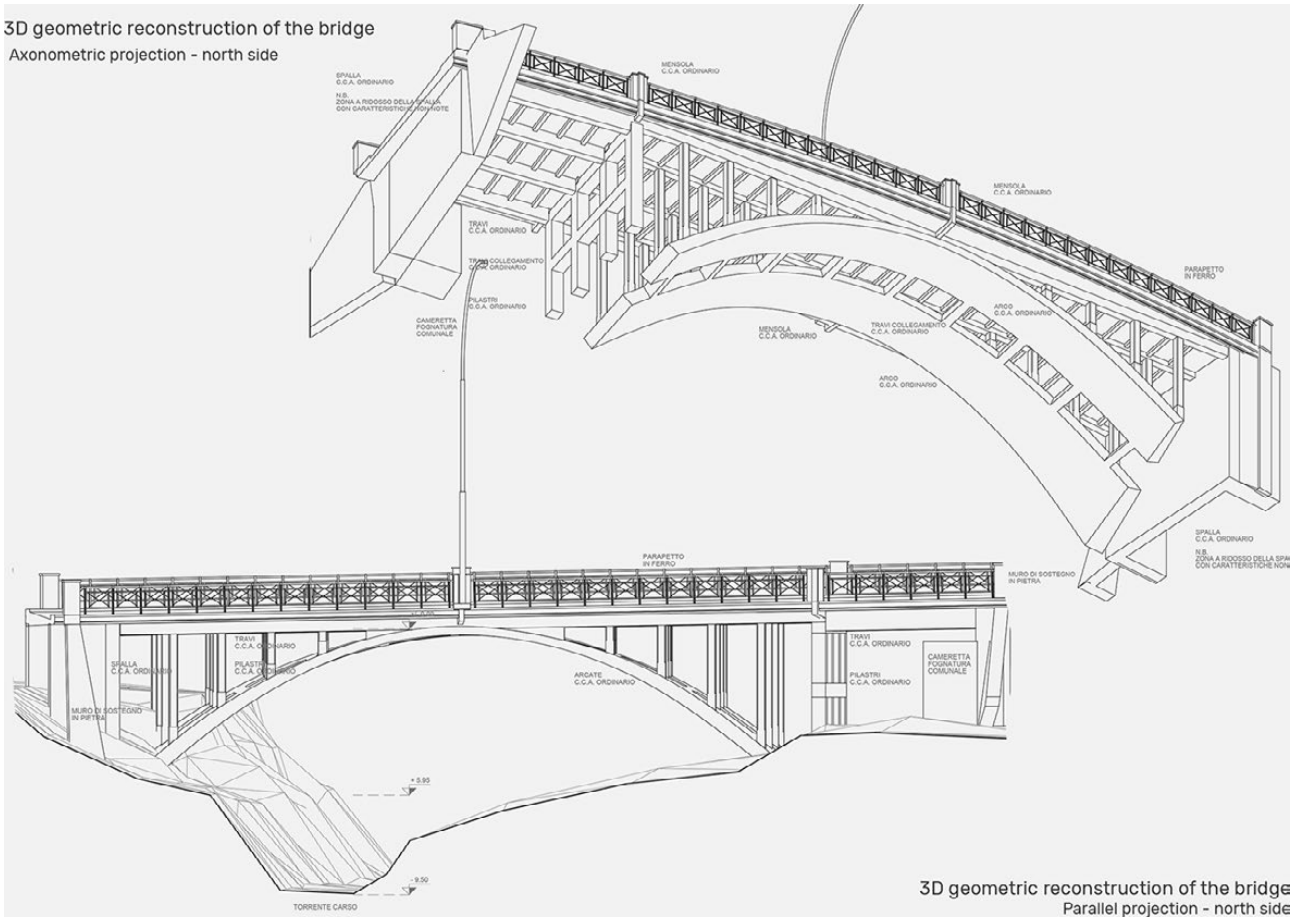

Figure 6: Nembro bridge: the as-built state. Graphical representation of the modelled bridge, as a result of 3D surveying. ***Image credit: Cardaci, Versaci, Azzola (2023) [50].***

Spandrel pillars. The deck rests on a double row of spandrel pillars orthogonal to the arch extrados. Their *planimetric spacing*, reconstructed by intersecting TLS profiles with photogrammetric meshes, matches the constructive grid described by Pesenti. Occluded pillars on the west side were partially interpolated from visible segments. Orthogonality and verticality checks reveal deviations within tolerance, consistent with construction techniques of the time (Figure 7).

Deck and cornices. The deck is formed by longitudinal beams, secondary cross-members and a slab, with cantilevered sidewalks on both sides. The point cloud clearly records the *plan curvature* (radius $\sim 110\,\mathrm{m}$) and a mild crossfall in transverse sections. Cornice ridges and parapet plinths are visible as edge features in the TLS model.

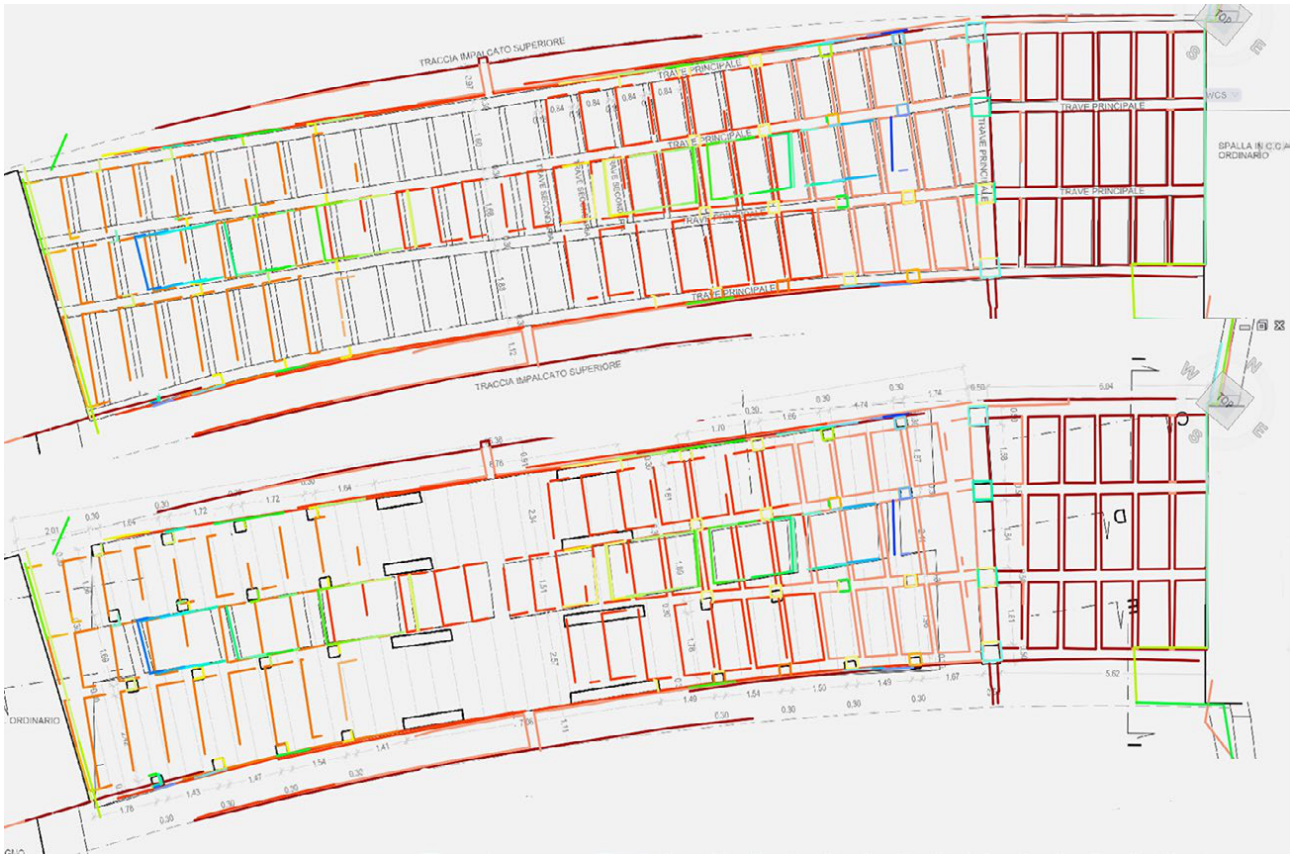

Figure 7: Nembro bridge: comparison of the aimplified structural scheme (grey-scale), with the real as-built state (colour-scale). Reconstruction of the bridge from historical design documents. ***Image credit: Cardaci, Versaci, Azzola (2023) [50].***

Abutments and joints. Abutments are reconstructed as near-vertical planes, with planar residuals serving as indicators of erosion and deformation. TLS shading (DIP/EDL) enhances the visibility of construction joints and pour lines, providing reliable annotation for 2D drawings.

As-planned vs. as-built comparison. The surveyed intrados axis $z_{\text{surveyed}}(x)$ was compared with the design ideal $z_{\text{ideal}}(x)$ derived from Pesenti's scheme. The deviation field

$$\delta(x) = z_{\text{surveyed}}(x) - z_{\text{ideal}}(x)$$

was mapped across stations, with crown and haunch envelopes annotated in orthophotos. Pillar spacing and orthogonality were checked against the design grid, showing consistency within a conservative error bound $\pm\varepsilon$ with $\varepsilon = \max\{\text{RMSE}_{\text{reg}},\ 2 \cdot \text{GSD}\}$. (Figures 8,6)

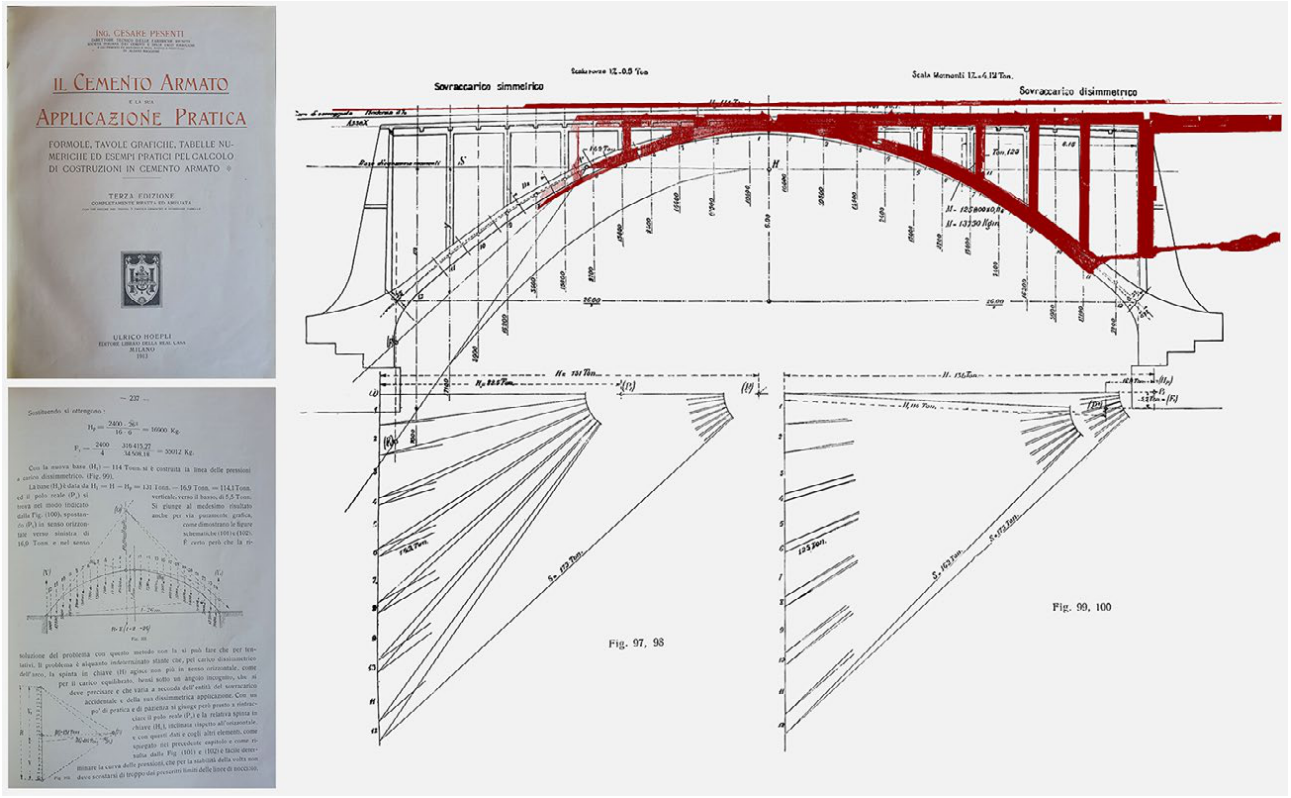

Figure 8: Nembro bridge: as-built vs Pesenti design scheme. Comparison between Pesenti's calculation (1913) and TLS survey geometry. ***Image credit: Cardaci, Versaci, Azzola (2023) [50].***

2.4 Case study 3: Gleno Dam (Vilminore di Scalve, Italy)

2.4.1 Site conditions

The remains of the Gleno Dam occupy a high-altitude gorge in Val di Scalve (Central Italian Alps), where a multiple-arch concrete dam —completed in 1923— catastrophically failed on 1 December 1923 during first impoundment, releasing a flood wave that propagated through the Scalve valley toward Lake Iseo. The event, reconstructed from archival sources and hydraulic back-analyses, is a canonical case in dam-break literature and a touchstone for contemporary risk modelling in alpine settings [58]. Today the site presents a complex palimpsest—breached structural fragments embedded in steep, unstable slopes; heterogeneous lithology; dense vegetation; and microclimatic variability—which imposes strict constraints on access, positional control, and sensor deployment, and motivates multi-scale documentation coupling architectural detail with geomorphic context.

2.4.2 Description of the integrated survey methodology adopted

A geodetically controlled, multi-sensor workflow integrated image-based surveying with rigorous quality control. A GNSS/total-station network established planimetric and altimetric consistency, materialising Ground Control Points (GCPs) and independent Quality Control Points (QCPs). Un-

crewed Aerial System (UAS) acquisition was planned for a corridor-like morphology, tuning baselines and overlap to the intended Ground Sampling Distance (GSD) and line-of-sight constraints typical of linear infrastructures [59] (Figure 9). Where feasible, direct georeferencing via Real-Time Kinematic (RTK) equipped UAVs was tested to reduce GCP logistics while preserving centimetric accuracy—a configuration increasingly adopted in hazardous or access-limited environments [60].

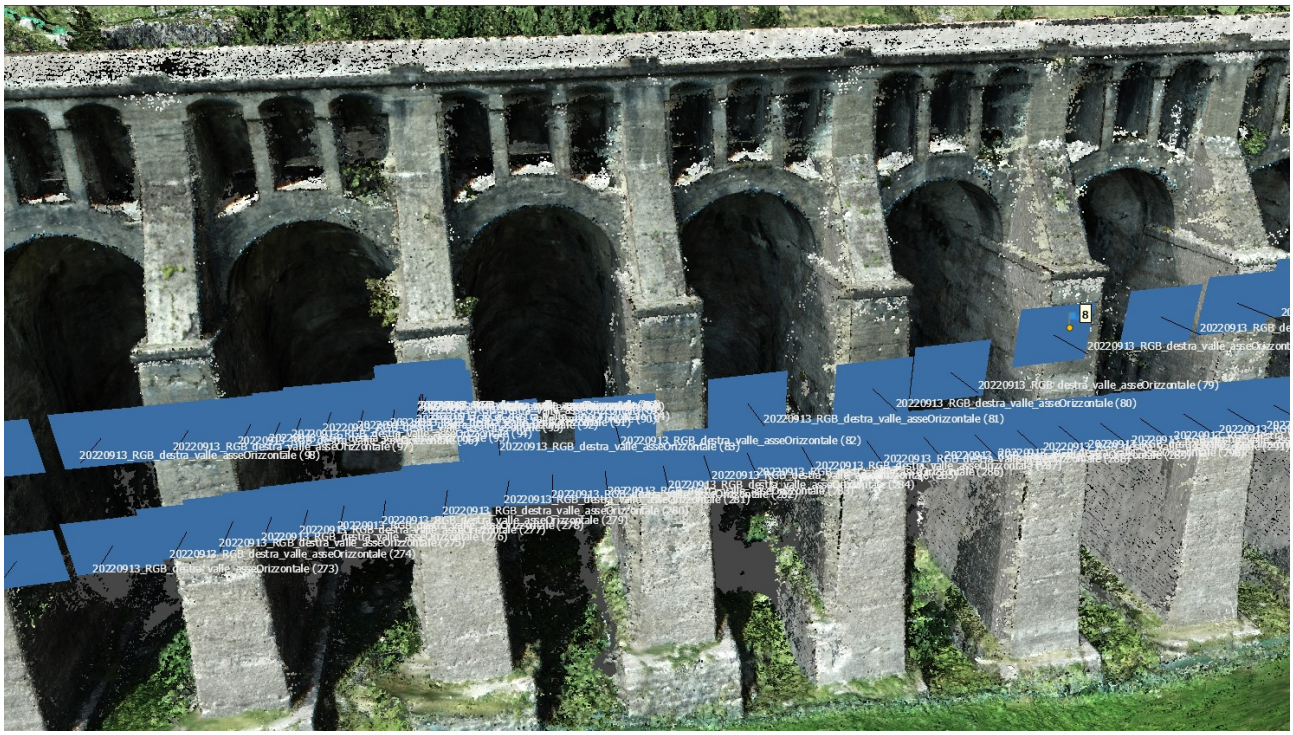

Figure 9: Gleno Dam: UAS photogrammetric 3D model with photo capture points. View of the reconstructed model, showing the aligned camera stations/poses (blue image footprints) superimposed on the dense model. The display documents the acquisition geometry adopted for image alignment and dense reconstruction, highlighting the corridor flight along the arch sequence. *Image credit: author*.

Photogrammetric processing followed a RAW-first strategy to preserve sensor-level radiometry and improve alignment robustness under high-contrast lighting, with parallel RGB-based reconstructions produced for controlled comparison. Differences in keypoint stability, reprojection error distributions, and completeness were quantified against the same QCP set [61]. Dense point clouds and textured meshes were exported in interoperable formats (E57, LAS/LAZ, PLY) for long-term archiving and downstream analysis (Figures 10 and 11).

For representation and interpretation, outputs were translated into two-dimensional orthophotos, elevations, and sections; in a layer-based CAD/GIS environment, co-registered raster plates and vector annotations (e.g., crack traces, joints, construction seams) supported comparative readings without altering the underlying metric integrity.

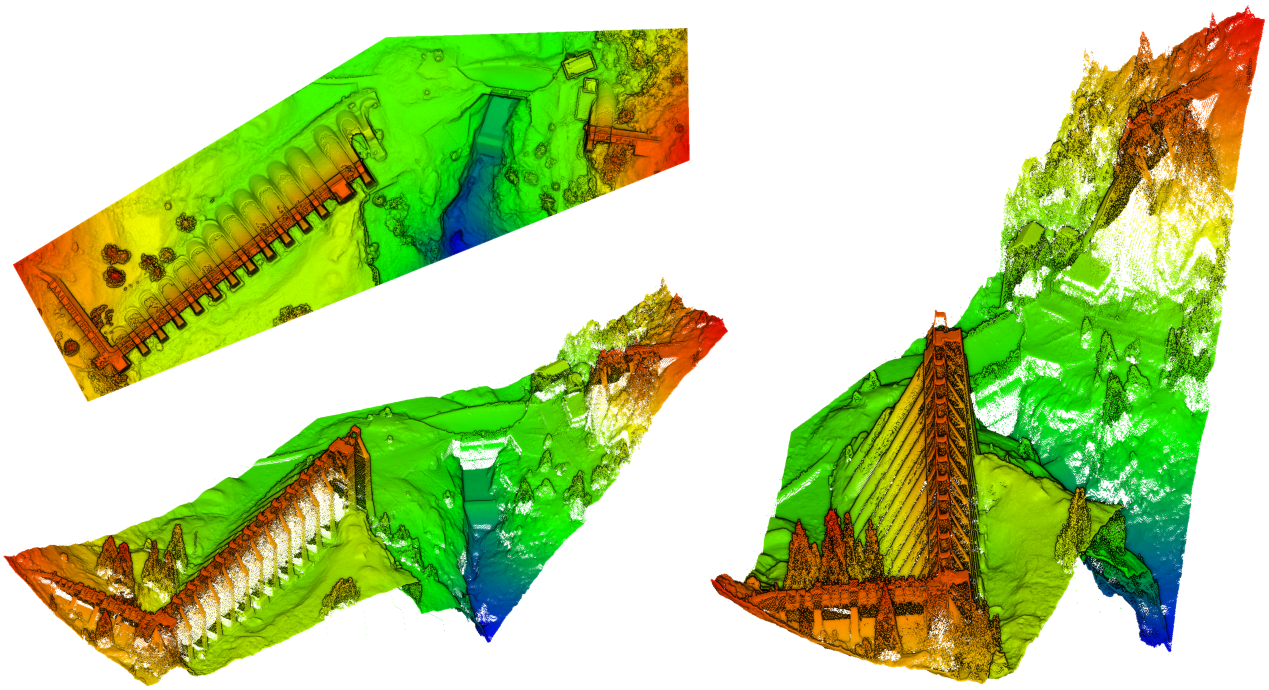

Figure 10: Gleno Dam: elevation-coded views of the photogrammetric 3D model. Hypsometric rendering of the UAS-derived 3D model, shown in plan (upper left) and two oblique perspectives (lower left and right). The colour ramp (red–yellow to green–blue) encodes relative elevation, enhancing the legibility of the dam crest, arch sequence, abutments, and surrounding slope morphology. These views support contour mapping and the metric readings discussed in Section 2.4.5. *Image credit: author*.

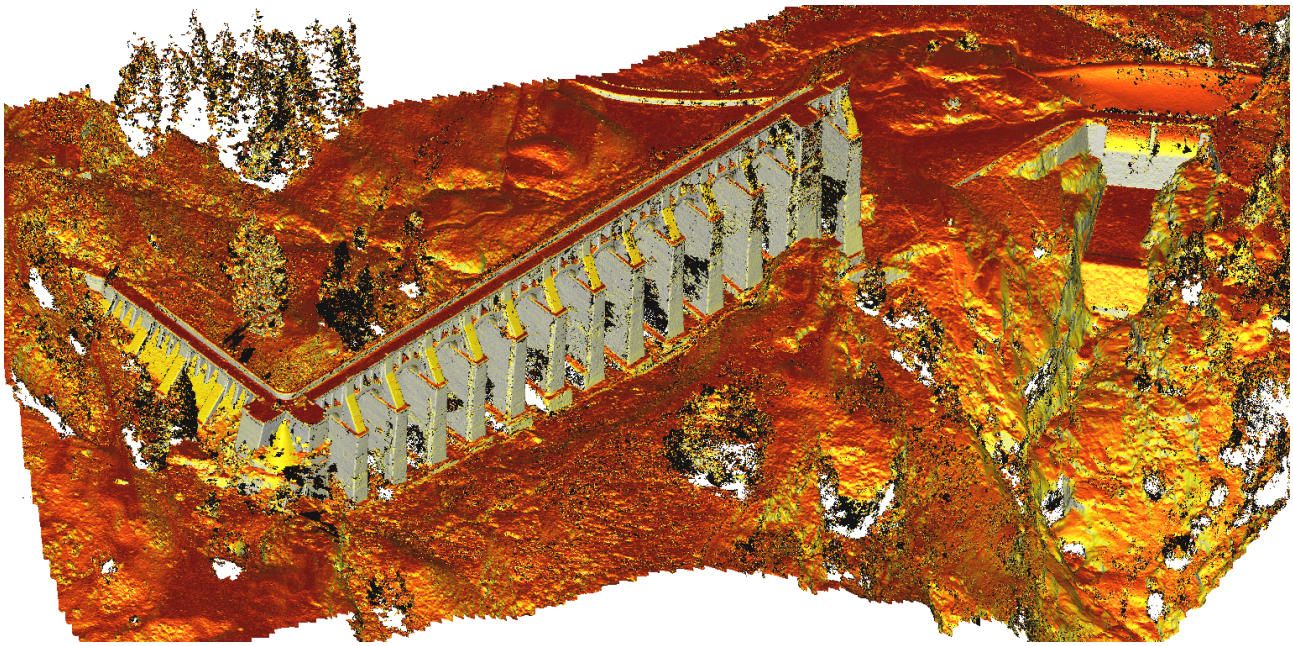

Figure 11: Gleno Dam: CloudCompare DIP rendering. Directional/illumination (DIP) scalar–field visualisation of the co-registered TLS–UAS point cloud. The warm–to–cool colour ramp enhances local relief and discontinuities, making the dam crest, arch sequence and adjacent slope breaks immediately legible. The rendering supports qualitative checks on morphology and registration and serves as a guide for subsequent segmentation and contour extraction. *Image credit: author*.

2.4.3 UAS acquisition design and photogrammetric parameterisation

The UAS survey design was defined in compliance with the applicable local, national and European aviation regulations for remotely piloted aircraft, and parameterised to achieve a Ground Sample Distance (GSD) compatible with the intended drawing scale of 1:50. The target GSD governed the choice of flying height through the standard relation

$$\mathrm{GSD} = \frac{p\,H}{f},$$

where p is the physical pixel size of the sensor, H the mean height above ground, and f the focal length.

Temporal sampling was set by the exposure time t_e and the capture interval Δt. The permissible image smear ("drag") on the ground was bounded by

$$s = v\,t_e \quad \text{(m)} \qquad \text{and} \qquad s_{\mathrm{px}} = \frac{v\,t_e}{\mathrm{GSD}} \quad \text{(px)},$$

and kept well below one pixel ($s_{\mathrm{px}} \ll 1$) to preserve edge definition. Given the chosen along-track overlap O_L and the ground footprint length L_g, the cruise speed followed from

$$v = \frac{L_g\,(1 - O_L)}{\Delta t},$$

while the lateral strip spacing w was set by the across-track overlap O_S and footprint width W_g,

$$w = W_g\,(1 - O_S).$$

These relations ensured that the prescribed longitudinal and transverse overlaps were met and that the plan of flight lines (course-aligned) was consistent with the corridor morphology of the site.

Photographic parameters were determined according to a conventional photogrammetric planning workflow: the target GSD fixed the nominal object distance; depth of field and hyperfocal distance were verified to guarantee sharpness across relief; and a colour calibration card (colour checker) was used in the field to derive a white-balance compensation and a consistent colour temperature for subsequent radiometric normalisation.

The resulting settings (flight speed, height, capture interval, image overlaps and ancillary options) are summarised in Table 1, which consolidates the values read from the mission planners and used in the acquisition.

The mutual alignment of terrestrial laser scanning (TLS) and photogrammetry was achieved within a single geodetic reference frame by means of a GNSS-supported topographic control network (total station, Ground Control Points and independent check points). The resulting co-registered dataset—dense

Parameter	Value	Units / Notes
Waypoints qty	33 pts	Points
Flight length	1709.2	m
Course count	16	Lines
Cover area	13.11	ha
Flight time (est.)	25 min 59 s	Estimated
Photos (est.)	409	Estimated
Batteries (est.)	2 sets approx.	
Capture interval	4.10	Distance between shots [m]
Camera model / settings	P4 Multispectral Camera	
Shooting angle	Course aligned	
Capture mode	Capture at equal time interval	
Flight course mode	Inside Mode	
Speed	5.6	m/s
Shutter interval	3.0	s
Height (AGL)	12.0 and 120.0	m
Ground resolution	6.4 and 64.0	mm/px
Front overlap	80%	
Side overlap	80%	
Course angle	35° 90°	
Margin	0.0	m
Motion blur (est.)	6.12	mm
End-mission action	Return to Home	

Table 1: UAS flight–planning parameters for the DJI Phantom 4 Multispectral mission.

point clouds and textured meshes—yields a three-dimensional model that is both metrically rigorous and chromatically faithful. This "as-built" representation enabled (i) verification of the constructed state against the historical design documentation (planarity, alignments, geometric offsets; qualitative assessment of finishes and material palettes based on calibrated textures); and (ii) the derivation of an analysis-ready geometric nucleus for numerical modelling.

For structural assessment, the surveyed geometry was simplified into a finite-element (FE) domain through surface extraction and topology cleaning, with boundary conditions inferred from the observed support configurations and construction joints. Material parameters were assigned from diagnostic evidence and literature-consistent ranges, and linear elastic modal analysis was performed to estimate natural frequencies and mode shapes under operational conditions. The numerical predictions were then confronted with in-situ dynamic identification obtained from a short ambient-vibration test: an array of accelerometers installed at the crest and other accessible points recorded the response to environmental excitation. Experimental modal parameters, extracted via frequency-domain peak-picking, were compared with the numerical counterparts using standard correlation measures (e.g.,

Modal Assurance Criterion),

$$\mathrm{MAC}(\boldsymbol{\phi}_i, \boldsymbol{\psi}_j) = \frac{\left|\boldsymbol{\phi}_i^{\mathsf{T}} \boldsymbol{\psi}_j\right|^2}{\left(\boldsymbol{\phi}_i^{\mathsf{T}} \boldsymbol{\phi}_i\right)\left(\boldsymbol{\psi}_j^{\mathsf{T}} \boldsymbol{\psi}_j\right)},$$

thereby supporting a transparent calibration loop on stiffness distribution, mass assumptions and boundary conditions. In this way, the integrated survey provided a continuous pipeline—from acquisition to drawing-based validation and dynamic re-analysis—capable of informing conservation priorities and risk-aware management of the structure [62, 63].

The three-dimensional model was interrogated through targeted measurements to quantify: (i) the centre-to-centre spacing of the arches, (ii) the heights of relevant elements with respect to a local geodetic datum, and (iii) the thickness of the arches. Arch spacing was computed as the Euclidean distance between consecutive arch centrelines extracted by spline fitting through intrados key points along the crest alignment,

$$s_k = \left\| \mathbf{c}_{k+1} - \mathbf{c}_k \right\|, \qquad k = 1, \ldots, n-1.$$

Element heights were derived as vertical differences between crown/intrados points and a reference plane/datum, while thickness was obtained by orthogonal sampling between intrados and extrados surfaces along selected generatrices; for each arch, the minimum, mean, and maximum values were recorded.

To enhance legibility at the intended drawing scale (1:50), results were translated into a two-dimensional representation based on planimetric contour lines at a constant vertical interval Δz. Contours were generated from the DSM/mesh after mild smoothing to suppress high-frequency noise and with enforced breaklines along structural edges; the resulting plate enables a synoptic reading of height gradients, bay spacing, and thickness variations. An example of the output is shown in Figure 12, where the constant-step contour map is overlaid with the measured centrelines and annotated values.

Measurement uncertainty was propagated from the registration residuals and the native point spacing of the datasets; reported values include conservative bounds of $\pm\varepsilon$, with $\varepsilon \approx \max\{\mathrm{RMSE}_{\mathrm{reg}},\ 2 \cdot \mathrm{GSD}\}$, ensuring consistency between the 3D interrogation and the 2D graphical communication.

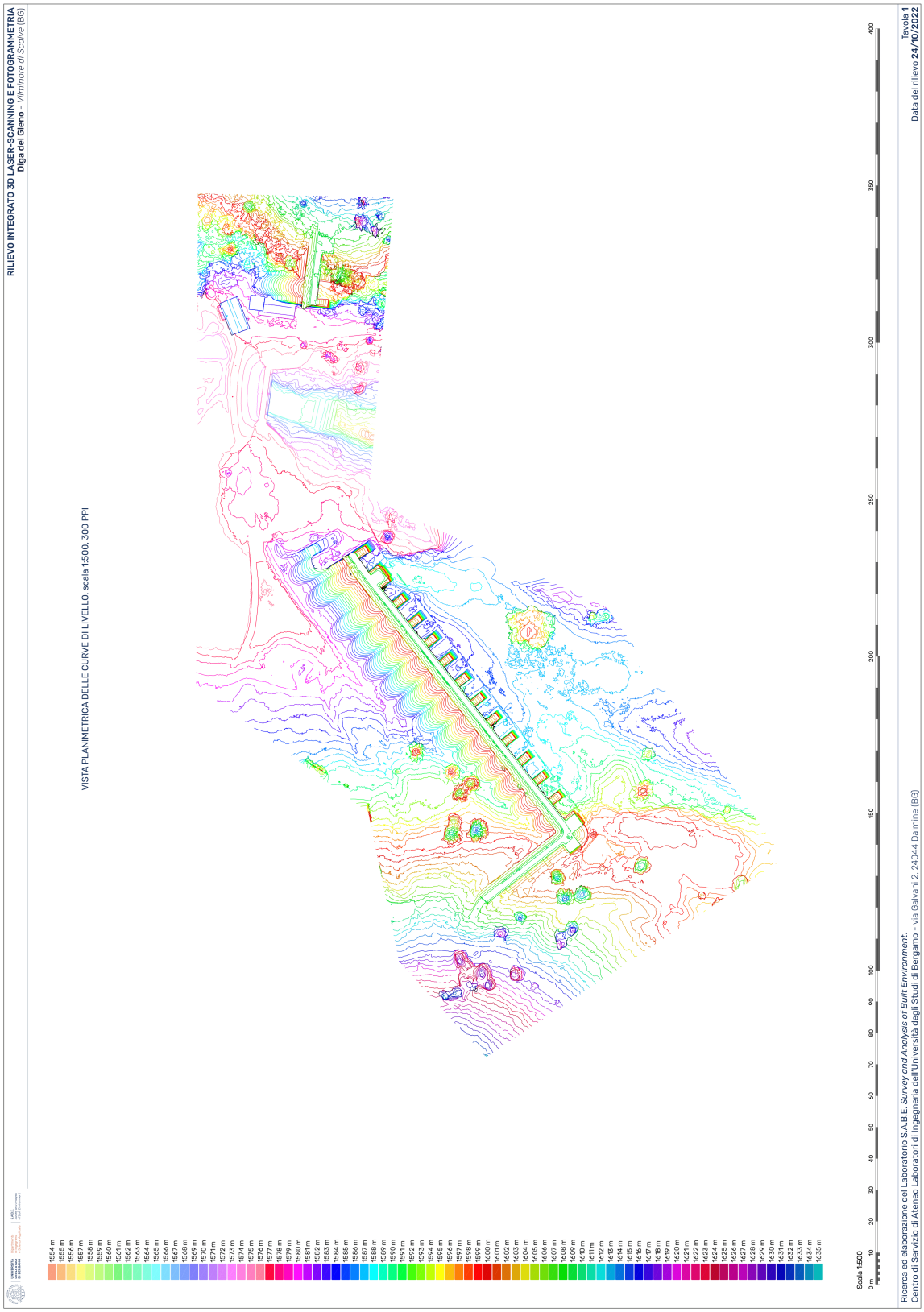

Figure 12: Gleno Dam: planimetric contour map from the integrated survey. Planimetric view of constant–interval contour lines derived from the co-registered TLS–UAS Digital Elevation Model (DEM) and composed for print at 1:500 (300 ppi). The plate provides a synoptic reading of local relief around the breached structure, highlighting terrace breaks, drainage lines, and the longitudinal alignment of the arches. *Image credit: author*.

2.4.4 UAS multispectral survey of the basin: vegetation indices and bio-colonisation cues

To extend the analysis from the structure to its environmental setting, a multispectral UAS campaign was carried out over the entire basin. The flight plan prioritised uniform ground sampling and bidirectional coverage to mitigate anisotropy and topographic shadowing typical of alpine valleys. From the calibrated reflectance products, vegetation indices (VIs) were derived to support a synoptic reading of vegetation dynamics and to highlight potential bio-colonisation near and on the dam.

NDVI parenthesis. The Normalised Difference Vegetation Index (NDVI) was computed as

$$\mathrm{NDVI} = \frac{\mathrm{NIR} - \mathrm{Red}}{\mathrm{NIR} + \mathrm{Red}},$$

and complemented by indices sensitive to chlorophyll content and canopy structure: the Enhanced Vegetation Index (EVI), Soil-Adjusted Vegetation Index (SAVI), Green NDVI (GNDVI), and the Red-Edge NDVI (NDRE). The selection reflects well-documented trade-offs among illumination sensitivity, soil/background effects, and saturation at high biomass, particularly relevant in shadowed or steep terrains [64, 65]. Operationally, index maps were produced at the same spatial resolution and co-registered with the photogrammetric orthophoto to enable direct overlay and joint interpretation [66, 67].

Outcomes. The multispectral layer yielded (i) a basin-wide vegetation map for context and access planning and (ii) qualitative—rather than quantitative—indications of biological patina on exposed concrete surfaces. In practice, local positive excursions of VIs at the structure (not attributable to cast shadows or adjacent vegetation) flagged areas with likely biological colonisation (algae/lichen biofilms), a phenomenon typically associated with moisture retention and increased surface porosity. This interpretation is consistent with the literature on biogenic colonisation and deterioration of cementitious materials, where biofilms and associated microbial activity correlate with chemical and physical degradation mechanisms on porous matrices [68]. These cues, while not a substitute for material testing, provide a rapid screening layer to prioritise closer inspection and to cross-reference with the elevation plates derived from the geometric survey [67, 66].

2.4.5 Discussion of the main outcomes and contributions to heritage knowledge

The integrated campaign produced a metrically controlled, high-resolution 3D record of the breach morphology, surveying arch fragments, and adjacent slopes, overcoming occlusions and scale gaps that single-technique surveys typically encounter in gorge contexts. The UAS workflow, anchored to the geodetic frame, yielded orthophotos and DSMs suitable for planimetric analysis of arch sequences, buttress alignments, erosion traces, and residual deformation cues; elevation plates supported tex-

ture reading of concrete surfaces and identification of discontinuities and out-of-plane irregularities. Comparative processing confirmed the advantages of RAW-based imagery in shaded or high-contrast conditions, notably clearer edge definition and reduced voids in occluded regions [61].
Methodologically, the case consolidates good practice for linear infrastructures: (i) RTK-UAS direct georeferencing can, in specific configurations, reduce GCP fieldwork while retaining centimetric accuracy; and (ii) when GCPs are required, corridor-aware layouts improve stability and reduce systematic trends in bundle adjustment [60, 59]. The disciplined 3D-to-2D translation (orthophotos, orthogonal elevations, sections) preserves metric legibility for drawing-based analysis and publication, facilitating reproducible documentation and risk-informed interpretation of the site [58].

2.4.6 Experimental evaluation of RAW vs. RGB photogrammetry

To assess how image formation affects geometric reliability and textural interpretability, we replicated—on a subset of the survey—an A/B experiment inspired by the protocol in [61]. The core question is whether workflows that ingest sensor-level data (RAW) provide measurable advantages over pipelines based on rendered images (RGB, e.g., in-camera JPEG or externally converted TIFF) when all other acquisition and processing parameters are held constant.

Design and controls. Two reconstruction branches were produced from the same flight lines, camera poses and exposure settings: (i) a *RAW-driven* branch in which native files were demosaiced and radiometrically linearised for direct use in the photogrammetric software; and (ii) an *RGB-driven* branch using rendered images exported with fixed conversion settings (gamma/contrast sharpening disabled; colour profile recorded). Both branches shared the identical control network (GCPs/QCPs), camera model (fixed interior orientation with the same distortion coefficients), alignment parameters, and dense-matching settings. White-balance normalisation was derived from a colour target photographed at the start of each mission. This strict pairing isolates the contribution of image formation from acquisition geometry or control quality [61].

Evaluation metrics. Following [61], performance was appraised with: (a) image alignment robustness (number of images aligned; average tie-points per image; distribution of reprojection errors); (b) geometric fidelity (RMSE at QCPs; cloud-to-cloud deviations on planar test patches; residual bowing); (c) completeness and density (valid points per square metre within a fixed ROI; percentage of filled pixels in orthomosaics); and (d) texture quality proxies (edge acuity on high-contrast targets; local entropy as a measure of textural richness).

Findings. Consistently with [61], the RAW-driven reconstructions exhibited higher keypoint stability and a more favourable reprojection-error distribution, especially in oblique views and shadowed/high-contrast areas typical of alpine settings. These gains propagated to (i) fewer alignment failures, (ii) marginally lower QCP residuals, (iii) reduced voids in occluded or low-texture regions, and (iv) crisper orthophoto edges suitable for drawing-scale outputs (1:50). From an interpretative standpoint, RAW-based textures improved the legibility of surface discontinuities and material changes without additional radiometric post-processing.

Implications for the present case. On the basis of these results, the project adopted a *RAW-first* processing policy for mission-critical blocks (e.g., the dam crest and abutments), with RGB-rendered imagery retained for redundancy and rapid previews. The measured differences—though bounded by the common geodetic frame—justify RAW ingestion whenever lighting dynamics are pronounced or when the intended outputs include fine-scale, metrically constrained drawings. The methodological choice aligns with the evidence reported by [61] and supports the reproducibility of the integrated TLS–UAS workflow described in Sections 2.4.2 and 2.4.4.

2.5 Case study 4: Geotechnical centrifugal test surveying

A geotechnical centrifuge reproduces prototype self-weight stresses in a reduced model by rotating it at an enhanced acceleration g^* so that effective stresses match those at full scale. For a geometric scale $L^* = L_m/L_p$ (model/prototype), **stress similitude** with $\sigma' = \rho g H$ requires:

$$\sigma'^* = \frac{\sigma'_m}{\sigma'_p} = 1 \quad \Rightarrow \quad g^* = \frac{1}{L^*}.$$

Thus, a 1:50 model should be tested at $\sim 50g$.

In dynamic problems (e.g. seismic input), kinematic similitude demands:

$$a^* = g^* = \frac{1}{L^*}, \qquad T^* = L^*, \qquad f^* = \frac{1}{L^*},$$

so model time contracts and frequency expands by $1/L^*$. With these scalings, in-flight shakers can impose earthquake records that preserve the dependence of soil response on confining stress, enabling realistic simulation of shear strain localization, cyclic mobility, and liquefaction.

For slopes, centrifuge modelling captures crest settlement, runout, and the formation of basal slip surfaces under transient acceleration fields. Typical outputs include displacement fields, the excess pore-pressure ratio

$$r_u = \frac{\Delta u}{\sigma'_{v0}},$$

and spectral amplification along the slope. Care is required to minimize acceleration-field gradients, container boundary effects, and to respect instrumentation bandwidth in the shortened model time T^*. Where relevant, consolidation or diffusion processes must be checked against their characteristic times so that rate effects remain prototypical.

In Italy, the ISMGEO seismic geotechnical centrifuge in Seriate (Bergamo) is a beam-type facility equipped with an in-flight shaker, widely used for earthquake slope stability and countermeasure studies. Published applications include dynamic tests on landslide stabilization systems and on embankment seismic response. The centrifuge thus provides a powerful bridge between mechanism discovery and the validation of numerical analyses for earthquake-induced slope failure and soil–structure interaction [69, 70, 71] (Figure 13).

The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLVIII-2/W8-2024
8th International ISPRS Workshop LowCost 3D - Sensors, Algorithms, Applications, 12–13 December 2024, Brescia, Italy

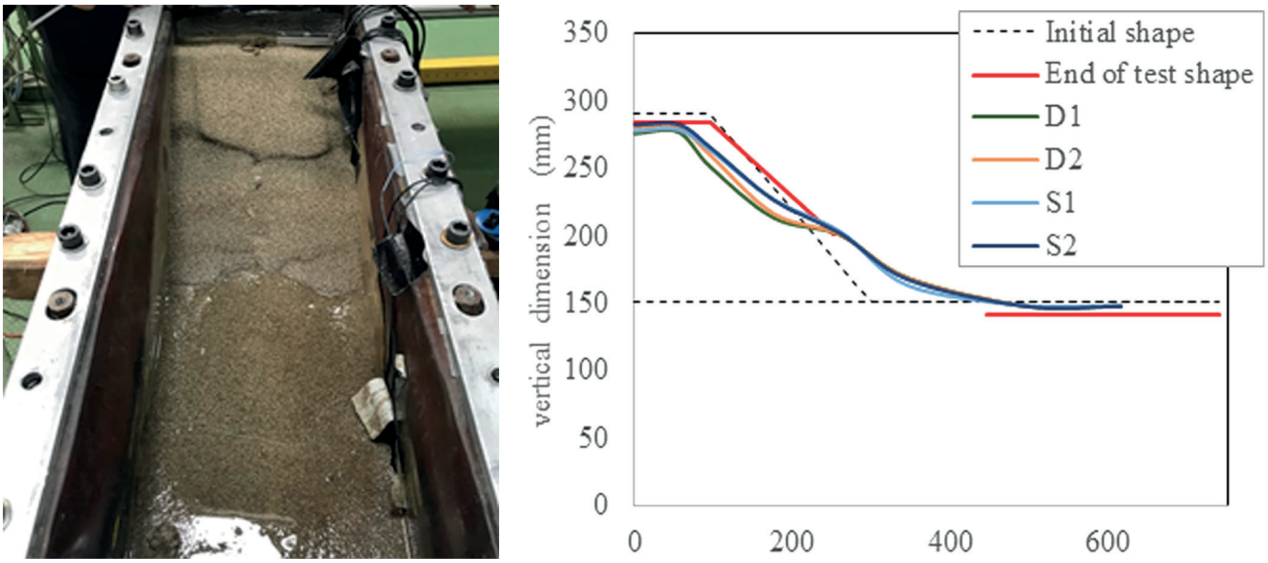

Figure 13: ESB container and photogrammetric context. Source: Cardaci, Giretti & Azzola (2024), CC BY 4.0.

2.5.1 Aim of the survey workflow

This case study establishes a *initial/final* close-range, multi-view photogrammetric survey to quantify surface changes on a sand slope subjected to dynamic loading in a geotechnical centrifuge. Instead of relying on single-plane in-flight image velocimetry, the method acquires complete three-dimensional (3D) surface geometry *before* and *after* shaking and reconstructs metrically controlled meshes via an SfM–MVS pipeline. The two epochs are strictly co-registered in a *single* reference frame so that

displacement maps, slope/gradient fields, surface normals, and longitudinal sections can be derived consistently across the entire free surface (Figure 14).

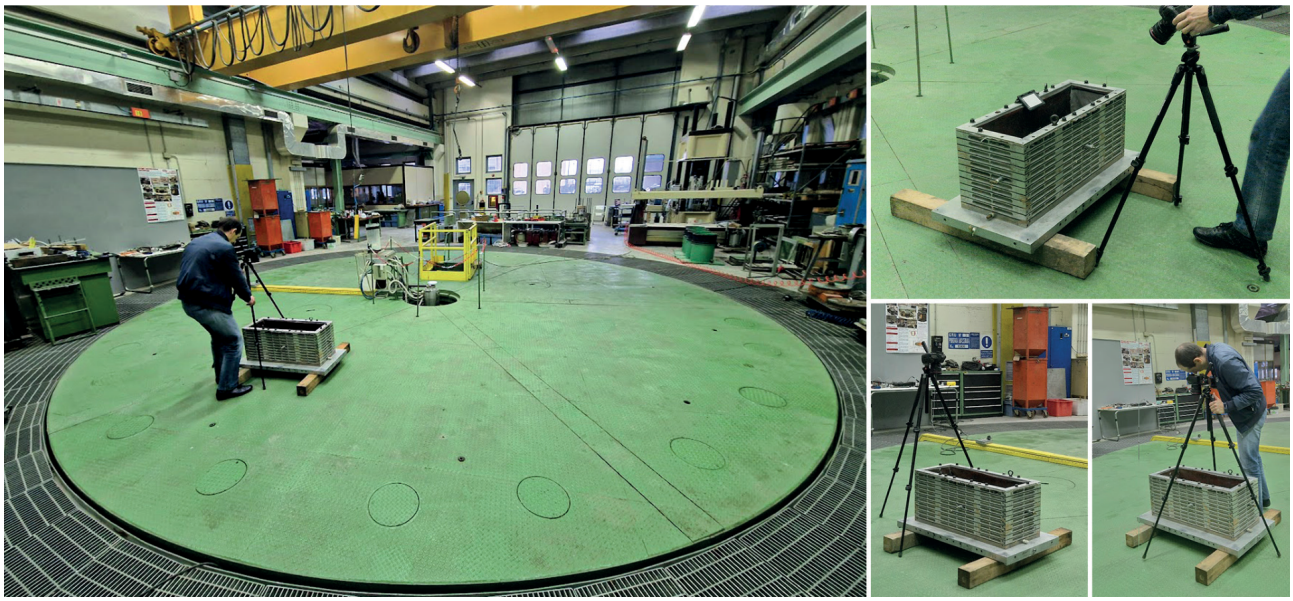

Figure 14: The photogrammetry data acquisition of the sample and the ESB support. Source: Cardaci, Giretti & Azzola (2024), CC BY 4.0.

2.5.2 Image acquisition protocol (numerical settings)

Cameras and exposure. The capture strategy assumes full-frame sensors with a native resolution of at least 18 MP. Depth of field is maximised by fixing the aperture at f/22 in order to verify the depth of field, and using ISO 100; exposure is biased by about $-2/3$ EV to reduce specular saturation on metallic parts. A tripod and remote release are used to mitigate vibration.

View network and coverage. Stations progress from frontal to oblique and near-zenithal viewpoints while keeping the optical axis oriented to the container centre. Image overlap is constrained within $75\% \leq \text{overlap} \leq 90\%$. Working distances are selected by hyperfocal checks in the $\sim$ 0.5–5 m range to achieve a ground sampling distance (GSD) on the order of 0.1–1 mm.

Lighting and timing. Both acquisitions are performed on a rigid stand near the centrifuge, immediately before and after the run, under controlled artificial lighting (window-free room with diffuse ceiling lights). This minimises handling deformation and promotes tonal/sharpness consistency.

Referencing. Stable container features or coded targets define the common frame for the two epochs, enabling rigid alignment and defensible differencing.

2.5.3 Processing pipeline and quality control thresholds

The workflow is intentionally standardised so that different operators and specimens yield comparable reconstructions.

Sparse alignment (SfM). Key/tie-point limits are set in the range of KP/TP $\approx$ 10,000/100,000 to 50,000/500,000 for the given container/optics. Initial alignment typically produces a sparse cloud exceeding 5.5×10^5 points, with reprojection error targets of RMS < 0.65 px and max ≈ 35 px prior to refinement. After introducing ~ 1 mm control points and applying gradual selection (reprojection, reconstruction uncertainty, projection accuracy), the refined sparse cloud contains on the order of 1.0×10^5 points with RMS < 0.35 px and max $\approx$ 3–5 px.

Dense reconstruction and meshing. The dense cloud is generated at high quality, yielding approximately 6.5×10^7 points (control point error ~ 0.5 px). Meshing enforces a minimum triangle size of about 2.5$\times$ the mean GSD, resulting in roughly 3.5×10^6 faces and 1.7×10^6 vertices. These thresholds prevent over-meshing beneath the resolvable scale and stabilise subsequent differential analyses.

Standardisation across tests. Because the ESB container size is fixed, the number of images, station geometry, and processing thresholds are kept constant across tests. At least two complete photo campaigns are recommended to validate a test, recognising that images cannot be substituted ex post without breaching protocol.

2.5.4 Derived products of the survey

Orthographic views and qualitative reading. Pre/post orthographic renderings provide an immediate morphological comparison, revealing areas of substantial deformation prior to quantitative differencing.

Cloud-to-cloud displacement. Distances between the two point clouds supply a full-field displacement estimate across the entire exposed surface, eliminating the section-only limitation inherent to single-window PIV (Figure 15).

Normals, slopes, curvature. Surface normals and their derivatives support gradient/slope mapping and curvature-based reading, assisting in the identification of anticlinal/synclinal patterns and the localisation of shear bands consistent with instability mechanisms.

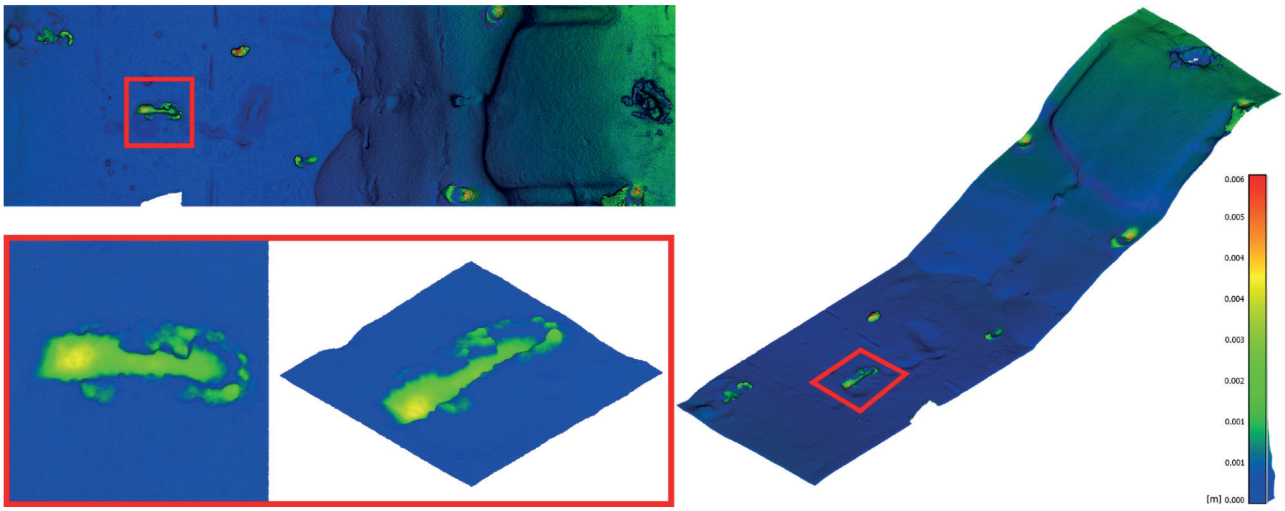

Figure 15: Point cloud distances before and after the test, in correspondency of local deformations. Source: Cardaci, Giretti & Azzola (2024), CC BY 4.0.

Subsampling for efficiency. Grid subsampling at 10 mm and at 1 mm is adopted to balance computational load with diagnostic power: the coarser level supports rapid exploration and sectioning, whereas the finer level is near the accuracy bound of the measurement (Figure 16).

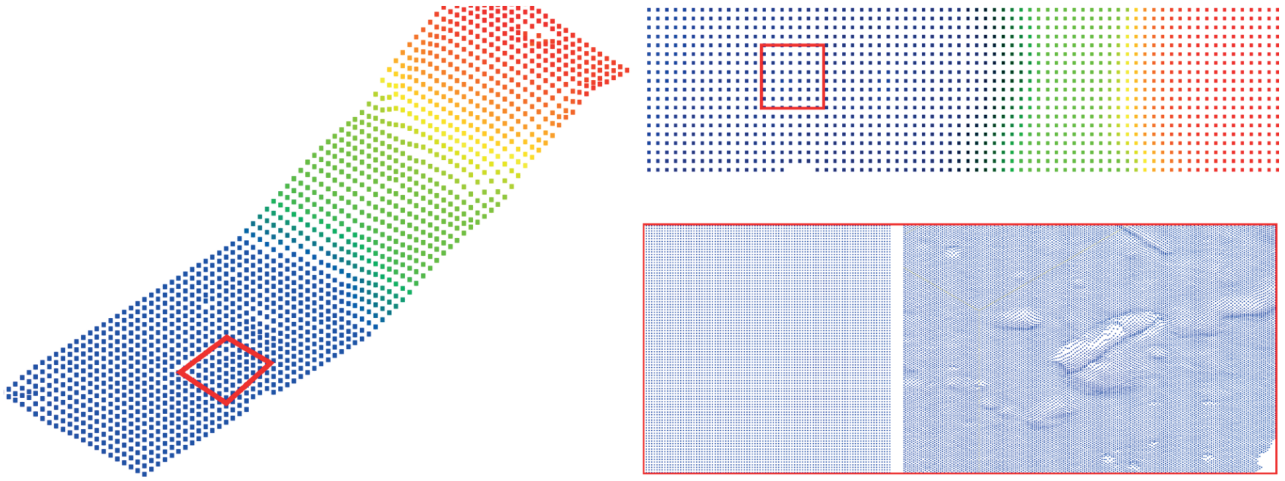

Figure 16: Cloud-to-cloud displacement map (cropped). Source: Cardaci, Giretti & Azzola (2024), CC BY 4.0.

2.5.5 Test scale and geometry (context for survey settings)

The demonstrative slope is modelled at $N = 50$ and tested at $50g$. The slope height is approximately 140 mm (prototype ~ 7 m); crown and base widths are about 95 mm and 300 mm (prototype ~ 4.75 m and 15 m), with a nominal inclination near 34°. The loose foundation layer height is approximately 151 mm (prototype ~ 7.55 m). The ESB container envelope is roughly 337×250×750 mm (H×W×L). These dimensions motivate the chosen GSD targets, distance band, and station layout.

2.5.6 Practical considerations for replication

In order to ensure methodological replicability, the experimental setup requires the explicit definition and controlled consideration of the following test conditions.

- Lock key capture settings (f/22, ISO 100, $-2/3$ EV, overlap, distance band, viewpoint orbit) and avoid run-to-run variation without updating QC targets.
- Plan redundancy: perform at least two complete pre/post photo sets; images cannot be replaced later without violating protocol.
- Control lighting (no windows, diffuse ceiling lights) and minimise handling between epochs.
- Enforce a single coordinate frame via fixed markers/features; this underpins honest differencing.
- Use numeric QC: aim for RMS reprojection < 0.35–0.65 px, dense counts near 6.5×10^7 points, and set mesh triangle size $\gtrsim 2.5\times$ GSD.
- Employ multiscale subsampling (1 mm / 10 mm) to manage computation while preserving diagnostic detail.

2.5.7 Limitations

Dense reconstruction is computationally intensive, and the pre/post strategy does not provide time histories; it therefore *complements*—rather than replaces—accelerometers, pore-pressure sensors, and displacement transducers. The strength of the approach lies in standardised acquisition and processing that yield reproducible, metrically reliable surface models for differential analysis in earthquake-induced slope instability studies.

2.6 Comparative discussion and lessons learned

The main outcomes of the case studies highlight the effectiveness of graphical representations and drawings in conveying measurements, metric features, and both local and global surface trends. Image-based representation enables fast and effective communication, while also allowing for subsequent re-elaboration and modification through image-editing methods. By working with multiple layers that can be superimposed and combined, it becomes possible to achieve flexible and versatile outputs (Figure 17).

Across the case studies, recurrent methodological workflows can be identified, often repetitive in nature, aimed at producing such representations. Terrestrial laser scanning consistently plays a

fundamental role, as it provides metrically reliable references for validating point clouds. At the same time, terrestrial photogrammetry has proven to be a robust and dependable survey technique, provided that it is tied to an appropriately defined network of reference vertices. The integration of laser scanning and photogrammetric data is especially valuable in situations where a single method alone cannot fully capture the complexity of a structure. A notable example is the survey of the bridge over the Carso stream, which was successfully documented through an integrated approach.

Common challenges emerge in achieving full coverage of built heritage assets, due to occlusions, inaccessible areas, and the need to optimize the balance between acquisition time and the desired level of detail. Regarding graphical outputs, image-based representations reveal significant limitations, particularly in establishing the correct balance between image resolution and the intended representation scale. In many cases, adjustments are required during the workflow, leading to the necessity of exporting precautionary high-resolution images. This practice, however, inevitably increases processing time and computational demands.

Managing data through layered editing in software such as Adobe Illustrator or Adobe Photoshop requires considerable computational resources, which are not always available in every working context. Nonetheless, best practices can be defined to prevent data loss throughout the workflow—from the initial images to the final processed outputs. For instance, within the Adobe suite (Lightroom, Photoshop, Illustrator), this can be achieved by saving Lightroom catalogs in proprietary formats, using PSD package files with external references, and managing multi-layered .ai files. At the end of this process, the elaborated outputs can be exported into user-defined image formats (e.g., TIFF, PNG, JPEG).

Point cloud processing software (e.g., Faro Scene, Agisoft Metashape) further expands these possibilities, enabling navigation within 3D models and the extraction of both 3D and 2D representations. These tools assist users in managing resolution and image size during the export phase. The resulting products are highly valuable across disciplines: from structural engineering, where accurate measurement of structural elements is required, to conservation and restoration, where textured orthographic projections continue to serve as the primary reference for mapping surface decay.

Looking ahead, there is a clear need to advance beyond mere representation of point clouds toward their analytical treatment, particularly in terms of evaluating the spatial positioning of individual points. The overarching goal of this research is to emphasize the potential applications and the critical insights that analytical data processing—ultimately oriented toward representation—can provide across the various disciplines involved in the documentation, conservation, and valorization of built heritage.

2.7 Comparative synthesis of the case studies

The survey campaigns described in this chapter were conducted in heterogeneous environments and under different operational constraints. In order to facilitate a systematic comparison between the four case studies, the main acquisition conditions, survey strategies, processing workflows, and analytical outputs are summarised in the following tables.

The purpose of this synthesis is to standardise the description of the survey campaigns and to highlight the parameters that most strongly influence the geometric reliability and analytical usability of the resulting datasets.

Case study overview

Table 2: Overview of the case studies

Case study	Environment	Structure type	Main objective
Filon dei Mot	Alpine ridge	Military settlement	Architectural documentation
Nembro bridge	Narrow gorge	Reinforced concrete arch bridge	Structural geometry analysis
Gleno Dam	Alpine valley	Dam remains	Morphological documentation
Centrifuge model	Laboratory	Geotechnical test model	Deformation measurement

Survey objectives

Table 3: Primary survey objectives

Case study	Main survey goals
Filon dei Mot	Documentation of architectural remains and wall deformation
Nembro bridge	Geometric assessment of arch structure
Gleno Dam	Documentation of structural remains and surrounding terrain
Centrifuge model	Quantitative analysis of pre/post-test deformation

Environmental constraints

Table 4: Operational constraints affecting survey design

Case study	Main constraints
Filon dei Mot	High altitude, harsh weather, logistical difficulty
Nembro bridge	Severe occlusions, narrow valley geometry
Gleno Dam	Large scale site, unstable terrain
Centrifuge model	Controlled laboratory environment

Survey instrumentation

Table 5: Instrumentation used in the survey campaigns

Case study	Sensors	Platform
Filon dei Mot	TLS scanner + RGB camera	Tripod + UAV
Nembro bridge	TLS scanner + RGB camera	Tripod + UAV
Gleno Dam	GNSS + RGB camera	UAV
Centrifuge model	DSLR RGB camera	Fixed rig

Acquisition parameters

Table 6: Main acquisition parameters

Case study	Acquisition distance	Resolution / GSD	Dataset size
Filon dei Mot	10–30 m	5–10 mm / 1–2 cm	41 scans / 1150 images
Nembro bridge	15–40 m	5–10 mm / 2–3 cm	47 scans / 895 images
Gleno Dam	30–80 m	2–4 cm	11 Points / 912 images
Centrifuge model	0.5–2 m	sub-mm	56 + 56 images

Georeferencing strategy

Table 7: Georeferencing strategies adopted

Case study	Control strategy
Filon dei Mot	GNSS network with ground control points
Nembro bridge	Total station control network
Gleno Dam	GNSS ground control points
Centrifuge model	Scale bars and laboratory reference frame

Dataset characteristics

Table 8: Point clouds dataset characteristics

Case study	Dataset type	Approximate data volume
Filon dei Mot	TLS point clouds + photogrammetry	235 million points
Nembro bridge	TLS + photogrammetric model	118 million points
Gleno Dam	Photogrammetric model	155 million points
Centrifuge model	Photogrammetric mesh models	3 million points

Table 9: Processing pipelines

Case study	Registration method	Filtering	Software
Filon dei Mot	ICP + GCP alignment	Statistical filtering	CloudCompare, Metashape
Nembro bridge	TLS registration + SfM	Radius filtering	CloudCompare, Metashape
Gleno Dam	SfM bundle adjustment	Dense cloud filtering	CloudCompare, Metashape
Centrifuge model	Photogrammetric alignment	Outlier removal	CloudCompare, Metashape

Processing workflow

Computational effort

Table 10: Computational requirements

Case study	Alignment time	Reconstruction time	Hardware
Filon dei Mot	56 min	85 min	CPU/GPU workstation
Nembro bridge	49 min	78 min	CPU/GPU workstation
Gleno Dam	42 min	95 min	CPU/GPU workstation
Centrifuge model	51 min	48 min	CPU/GPU workstation

Analytical outputs

Table 11: Analytical outputs derived from the surveys

Case study	Analysis type	Outputs
Filon dei Mot	Morphological analysis	Orthophotos, sections
Nembro bridge	Structural geometry analysis	Profiles and sections
Gleno Dam	Morphological documentation	Orthophotos and maps
Centrifuge model	Deformation analysis	Displacement fields

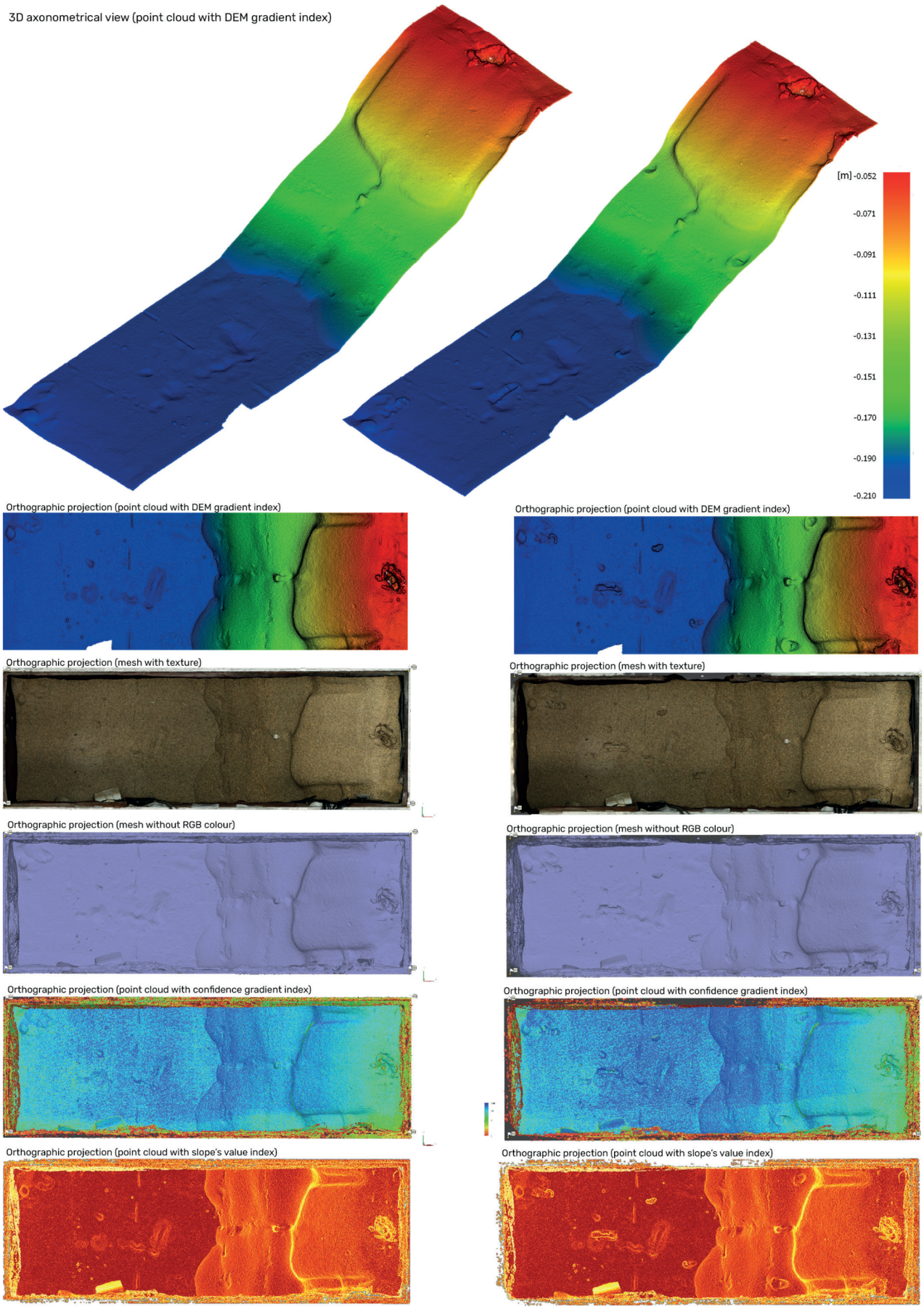

Figure 17: Pre/post 3D views and orthographic projections (cropped). Source: Cardaci, Giretti & Azzola (2024), CC BY 4.0.

Chapter 3. Methodological framework

3.1 Curvature-based geomorphometric workflow

3.1.1 Protocol design and data homogenization

The proposed methodological framework emphasizes a rigorous, reproducible workflow for transforming raw geometric data into behavioural indicators. The initial phase involves the standardization of point clouds derived from diverse sources, such as Terrestrial Laser Scanning (TLS) and Unmanned Aerial Vehicles (UAV) [33]. This process includes resampling to ensure a uniform spatial distribution and the removal of non-spatial attributes (colour, reflectance) to focus strictly on morphological variations [33].

3.1.2 Numerical estimation via covariance analysis

The core of the 3D surface analysis relies on the numerical estimation of the unit normal vector ($\hat{n}$) and the subsequent derivation of curvature proxies. For a given point P and its neighborhood of n points, the local geometry is summarized by a covariance matrix (M):

$$M = \frac{1}{n}\sum_{i=1}^{n}(P_i - \bar{P}) \otimes (P_i - \bar{P})^T \tag{3.1}$$

where $\bar{P}$ denotes the centroid of the neighborhood. The eigenvectors of M provide the principal directions of the surface, and the eigenvalues ($\lambda_0 \leq \lambda_1 \leq \lambda_2$) describe the distribution of points relative to the local tangent plane. The smallest eigenvalue λ_0 acts as a proxy for the surface's "thickness" or roughness in the direction of the normal, effectively providing a descriptor of local curvature variation [33].

3.1.3 The dual-level analytical approach

The framework operates on two complementary levels to balance global diagnostic oversight with local precision [33].

Global surface analysis

The first level involves an omnidirectional interrogation of the 3D surface. By estimating the local surface attitude across the entire point cloud, researchers can generate synthetic orientation maps, such as Dip and Dip Direction [33]. This allows for the rapid identification of surface rotations, changes in alignment, and the segmentation of the model into regions of homogeneous structural behaviour.

Directional section-based analysis

The second level focuses on specific planar sections. This directional approach is preferred in structural engineering because it mimics traditional technical drawing and reduces noise from lateral elements [33]. Profiles are extracted along representative traces, and the local inclination (ω) is calculated as the first derivative of the profile function $\zeta = f(x)$:

$$\omega = \frac{d\zeta}{dx} \tag{3.2}$$

The curvature proxy (γ) is then derived from the second derivative, representing the rate of direction change along the profile:

$$\gamma \approx \frac{d^2\zeta}{dx^2} \tag{3.3}$$

This allows for the identification of inflection points and curvature inversions that may signal structural stress or deviation from design intent [33].

3.1.4 Interpretation and interdisciplinary application

The interpretation of curvature results serves as a bridge between pure metrics and structural diagnosis. High curvature values often correlate with ridges or depressions that mark zones of instability or stress concentration in civil infrastructure [33, 32]. By comparing multitemporal datasets, the workflow reveals the trajectories of deterioration over time, transforming the survey from a static record into a dynamic monitoring tool. The integration of Python for automation and MATLAB for high-precision numerical processing ensures that the results are both scientifically valid and practically applicable across disciplines such as architecture, geotechnics, and conservation [33].

3.2 Slope and curvature analysis from XYZ data

The use of differential calculus in the study of three-dimensional spatial data (XYZ) represents one of the most established and, at the same time, versatile methodologies for analyzing the shape and morphology of surfaces and profiles. The geometric information contained in coordinate datasets, when properly processed through first- and second-order derivatives, makes it possible to extract two concise yet highly meaningful parameters: *slope* and *curvature*.

These quantities are not merely abstract measures derived from the mathematical representation of forms; rather, they hold crucial significance across a wide range of applications. Slope provides a measure of the inclination of a plane or profile, with direct implications for assessing slope stability, designing built structures, or evaluating spatial accessibility. Curvature, on the other hand, identifies

points of major geometric variation, enabling the detection of discontinuities, edges, inflections, or areas subject to stress concentration.

Their applications span multiple disciplines—from geomatics to civil engineering, from architecture to conservation, from structural analysis to physical geography. In this regard, the study of slope and curvature should not be understood as a purely mathematical operation, but as an interpretative process that transforms raw positional data into morphological and constructive knowledge.

3.2.1 Mathematical definition and physical meaning

From a mathematical perspective, the slope of a curve $z = f(x)$ is defined as the first derivative:

$$\text{slope} = \frac{dz}{dx}$$

In three-dimensional contexts, slope corresponds to the magnitude of the gradient of a surface:

$$\text{slope} = \|\nabla z(x, y)\| = \sqrt{\left(\frac{\partial z}{\partial x}\right)^2 + \left(\frac{\partial z}{\partial y}\right)^2}$$

Curvature, by contrast, measures the variation of the tangent direction along a curve and is obtained through the second derivative:

$$\kappa = \frac{d^2 z}{dx^2}$$

The physical interpretation of these parameters is more immediate:

- **Slope** expresses the change in elevation relative to horizontal distance. It quantifies local inclination and, in physical terms, represents the energy required to overcome a rise, the stability of a slope, or the resistance to flow. In architectural or territorial surveys, slope serves as a key indicator of the morphology of both built and natural environments.

- **Curvature**, calculated through the second derivative, describes the rate at which a line or profile changes direction in space. It is directly related to concepts of stability and stress: high curvature values correspond to ridges, depressions, or edges, which often mark zones of instability or stress concentration. In architecture and structural engineering, curvature highlights areas of deformation and discontinuity, offering insights into the static and dynamic behaviour of built systems.

The slope measures how rapidly a surface rises or falls, while the curvature quantifies how quickly the direction of that rise or fall changes. Combined analysis of these parameters provides a more accurate interpretation of the morphology and physical behaviour of surfaces and profiles.
In architectural surveying, slope is an immediate indicator of the regularity of horizontal and vertical surfaces, allowing for the detection of unwanted settlements or inclinations. Curvature, meanwhile, makes it possible to highlight geometric anomalies or deformations in arches, vaults, and curved surfaces, with direct implications for conservation and restoration. Similarly, the generation of digital terrain models (DTM or DEM) relies on slope and curvature maps, which offer clear morphological representations and are widely used for slope classification, hydrological modelling, and geomorphological process analysis.
In structural surveying, curvature is closely linked to material behaviour: the greater the curvature of a beam or surface, the greater the deformation it undergoes. The relationship between curvature and mechanical stresses underlies elastic theory and the structural modelling of materials and systems.

3.2.2 Outcomes

The differential analysis of XYZ data, through slope and curvature, provides an effective interpretative framework for understanding the geometry and behaviour of surfaces. Slope highlights local inclinations, while curvature reveals directional changes, thereby identifying critical morphological features.
When applied together, these parameters bridge mathematical rigor with practical applicability. They enable the transition from elementary geometric data to deeper knowledge, relevant in contexts ranging from built heritage documentation to geotechnical modelling, from structural analysis to territorial management. The combined use of slope and curvature thus emerges as a methodologically robust and interdisciplinary tool, capable of supporting scientific research as well as operational needs in engineering, architecture, and conservation.
Open-source and licensed software tools compute elevation, local slope, and curvature directly on point clouds by first estimating a unit normal versor at each point from a user-defined neighborhood (k-NN or radius) [72]. In the standard least-squares plane fit, the local covariance is

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{p}_i - \bar{\mathbf{p}}) \, (\mathbf{p}_i - \bar{\mathbf{p}})^{\top},$$

and the unit normal $\hat{\mathbf{n}}$ is the eigenvector associated with the smallest eigenvalue of $\mathbf{C}$ [73]. Normal orientation (sign) can be propagated by a preferred axis, a sensor viewpoint, or MST-based strategies. Local slope is obtained from the gradient of the fitted plane, while curvature may be estimated by

local quadric fitting or eigenvalue-based indices (with principal directions given by the eigenvectors). CloudCompare exposes these parameters under *Normals* → *Compute* and *Curvature*.

3.2.3 Neighborhood-based normal estimation and surface definition

During the computation of the unit normal vector at each point of a point cloud, the algorithm evaluates an isotropic three-dimensional neighborhood around the query point, extending in all directions. This local neighborhood—typically specified either by a fixed radius or by a k-nearest-neighbors criterion—provides the data support for approximating a locally interpolating surface. Because the neighborhood is omnidirectional, this approach is particularly well suited to the definition and interpretation of surfaces, as opposed to analyses constrained along a single direction or plane.

3.2.4 Orientation descriptors in CloudCompare: dip and dip direction

Within this framework, it is useful to recall the orientation representations adopted in *CloudCompare*. The software reports (i) *Dip*, which encodes the inclination of the computed normals, and (ii) *Dip Direction*, which encodes their azimuthal orientation around the vertical axis. In the adopted convention, the rotation is measured with respect to the positive ordinate direction taken as reference. Together, these two attributes offer a compact, quantitative description of local surface attitude and are effective for visual diagnostics on complex geometries.

3.2.5 Motivation for planar sectioning alongside 3D visualization

Despite the advantages of full 3D visualization, there are many situations in which a planar representation of model sections is indispensable and, in practice, more effective. Planar sections enable targeted metric readings, direct profile-to-profile comparisons, and presentation of results in a form that is immediately interpretable in engineering and architectural workflows.

3.2.6 An assisted, GUI-based workflow for section generation

For these reasons, an automated workflow—exposed through a graphical user interface (GUI)—is developed to support users who may not have advanced programming expertise. The goal is to obtain standardized representations of sections extracted from point clouds generated by laser scanning and photogrammetry. The interface guides the user through the selection of input data, the definition of section traces, and the export of figures and tabular outputs suited to subsequent analysis and reporting.

3.2.7 Software integration and standardized management of point clouds

This research activity relies on the integration of heterogeneous software tools to enable semi-automatic workflow management, with particular emphasis on MATLAB and Python. The following chapters present a method for standardized point-cloud handling designed to extract consistent information from models composed of point clouds and meshes. Achieving such results from general survey datasets requires the ability to orchestrate code in multiple languages and to maintain stable interfaces between processing stages; therefore, a set of principles and conventions is introduced to ensure repeatability and clarity.

3.2.8 Implementation strategy: Python-driven MATLAB output for advanced plotting

The implementation is centered on Python, which programmatically compiles a MATLAB Live Script (or MATLAB code) to produce plots and derived representations of the surveyed models that would otherwise be cumbersome—or not readily accessible—within the native software employed to construct the point cloud. This division of labor leverages Python for data mediation and automation, and MATLAB for robust, publication-grade visualization and numerical post-processing.

3.2.9 Directional estimation of slope and curvature on user-defined sections

In parallel with the omnidirectional neighborhood analysis used to compute normals, the subsequent development focuses on estimating slope and curvature along specified directions, i.e., constrained to user-defined planar sections. The aim is to obtain additional, section-specific results that are not influenced by variations occurring in adjacent slices of the cloud. This condition finds useful practical applications in cases where the morphology of the point clouds requires focusing attention on thin sections, at the sides of which there may be accessory disturbing elements that we want to exclude from the evaluation.

By restricting the analysis to the chosen section, the method yields profiles and curvature readings that directly reflect the local, section-aligned behaviour of the object—thereby complementing the surface-oriented information obtained from the 3D neighborhood of each point.

3.2.10 Outlook

Taken together, these components—(i) omnidirectional normal estimation and orientation descriptors, (ii) rigorous planar sectioning, (iii) a GUI-assisted workflow, and (iv) a Python–MATLAB integration for standardized processing and plotting—provide a coherent framework for extracting both surface-

based and section-based metrics from point-cloud surveys, with an emphasis on clarity, reproducibility, and accessibility for users across surveying, architecture, and engineering domains.

3.3 A MATLAB script for analytical analysis of point cloud sections

The developed MATLAB code is in listing A.2 and it is illustrated in the following steps

3.3.1 Purpose

The first application of this experimentation is carried out within the case study of the geotechnical test, which consists of a comparison between two point clouds: one representing the slope surface before the test and the other representing it after the test. The script compares two surveyed longitudinal sections ("Initial" and "Final") exported from CloudCompare as ASCII tables with three numeric columns $[X\ Y\ Z]$. It selects a thin band around a target plane at constant Y, fits a smoothed cubic spline $z = s(x)$ for each epoch, evaluates the first and second derivatives along the section (slope and a curvature proxy), visualizes profiles and derivatives, exports spline samples to simple `.poly` files, and writes structured results to an Excel workbook, a consolidated solution, although sometimes limiting, but which is often requested and considered useful by operators and teams who intend to evaluate significant results.

3.3.2 Inputs and key parameters

The first essential step consists in defining the fundamental parameters of the MATLAB script. These parameters establish the operational framework within which the subsequent computational procedures are performed. In particular, they determine the structure of the input data, the numerical settings adopted for processing, and the variables required for the analysis. The correct specification of these elements ensures the reproducibility and stability of the workflow, while providing a clear reference for the interpretation of the results. The following section therefore introduces and describes the main parameters that govern the implementation of the script.

- **File paths:** `filename_initial`, `filename_final` are absolute paths to the section files produced in CloudCompare (three columns X, Y, Z, whitespace-separated, no headers).
- **Section midline:** $y_0 = 0.165\,\text{m}$ is the nominal Y-coordinate of the target section.
- **Half-thickness:** $\delta = 2 \times 10^{-4}\,\text{m}$ controls the selection band around y_0. A point is retained if

$$y_0 - \delta < Y < y_0 + \delta.$$

- **Smoothing parameter:** $p = 0.9999999$ sets the strength of the cubic smoothing spline (`csaps`); values near 1 behave nearly interpolatory, values closer to 0 increase smoothing.

Smoothing parameter in MATLAB `csaps`

The `csaps` function (Cubic Smoothing Spline) in MATLAB constructs a cubic smoothing spline from sampled data, balancing two main objectives:

1. **Data fidelity**: the spline remains close to the observed data points.
2. **Smoothness**: the spline is as regular as possible, avoiding unwanted oscillations.

The smoothing parameter $p \in [0, 1]$ controls the trade-off between these two aspects [74, 75]:

- $p \approx 1$: the spline behaves almost like an interpolant, passing very close to the data points.
- $p \approx 0$: the spline is strongly smoothed, producing a regular curve while sacrificing pointwise adherence.

The criterion being minimized is

$$p \cdot \sum_{i=1}^{n} w_i \, (y_i - s(x_i))^2 \; + \; (1 - p) \int (s''(x))^2 \, dx,$$

where:

- y_i are the observed data,
- $s(x)$ is the spline,
- w_i are weights,
- the first term measures the approximation error,
- the second penalizes curvature (smoothness).

As shown in Figure 18 and Listing 3.1, the parameter P was represented using MATLAB code.

Interpretation of $p = 0.9999999$. With a value so close to 1, the spline tends to follow the data almost exactly, behaving nearly interpolatory. This is useful when the data are very regular and almost noise-free, but it may reproduce measurement noise as well.

Choice of p.

- $p \approx 1$: suitable for high-precision survey data (e.g., laser scanning, photogrammetry).

- $p \approx 0$: recommended when the data are highly noisy, to obtain a globally smooth curve.

- Intermediate values (e.g. between 0.01 and 0.9): good compromise in case of moderately noisy real-world data.

Listing 3.1: MATLAB code illustrating the effect of the smoothing parameter p in `csaps`

```
x = linspace(0,10,100);
y = sin(x) + 0.1*randn(size(x));   % noisy data

% very smooth spline
pp1 = csaps(x,y,0.01);
% almost interpolatory spline
pp2 = csaps(x,y,0.9999999);

xx = linspace(0,10,500);
yy1 = fnval(pp1,xx);
yy2 = fnval(pp2,xx);

plot(x,y,'o'), hold on
plot(xx,yy1,'r','LineWidth',2)
plot(xx,yy2,'g','LineWidth',2)
legend('Data','p=0.01 (smooth)','p\approx1 (interpolatory)')
%%
```

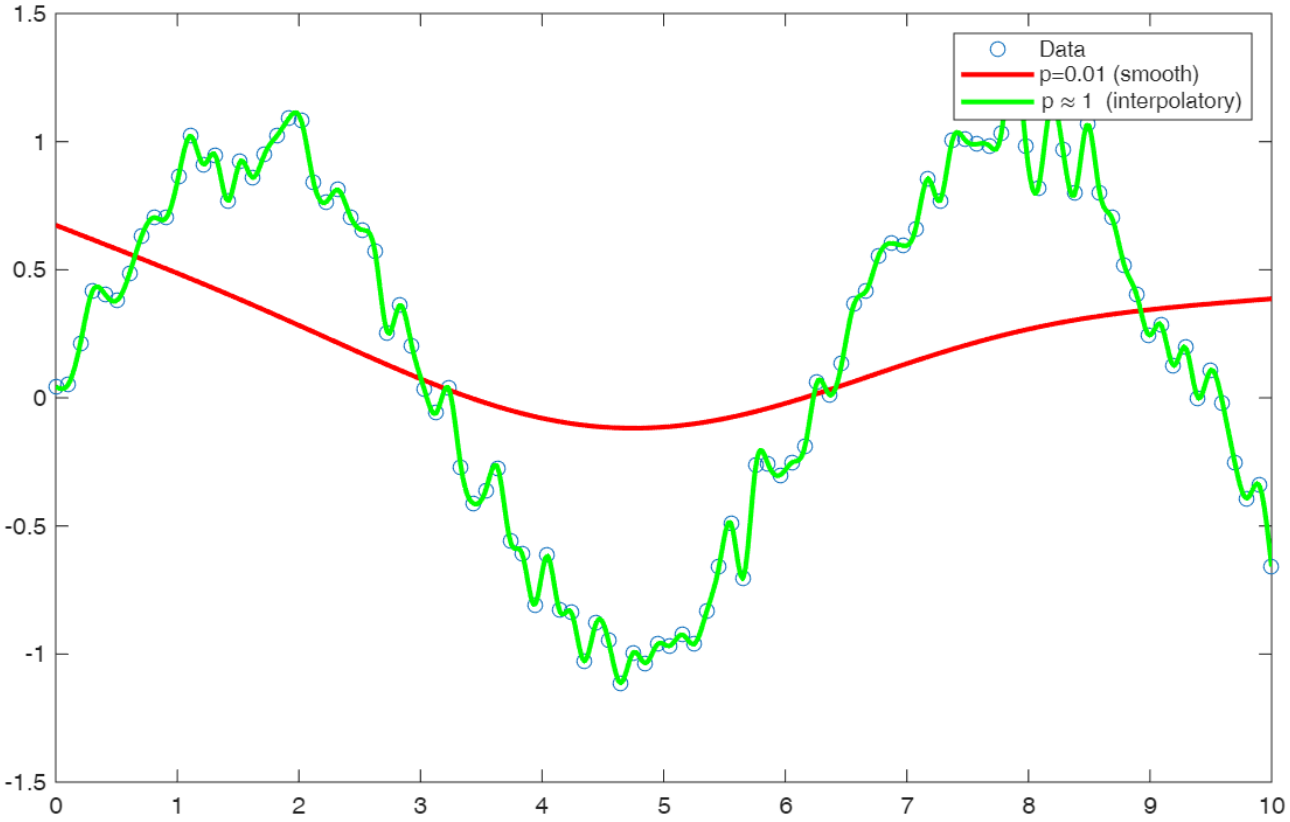

Figure 18: Effect of the smoothing parameter p in MATLAB's `csaps`. Noisy data (blue circles) are fitted with two cubic smoothing splines: with $p = 0.01$ (red), producing a very smooth curve, and with $p \approx 1$ (green), yielding an almost interpolatory fit.

Following this approach, experimental trials were performed to evaluate the representation of longitudinal sections under different values of the smoothing parameter p (Figures 19,20,21,22). This approach allows the generation of representative examples illustrating the different influence of the smoothing parameter p on the smoothness of the function period. Low smoothing values result in reduced sensitivity and lower noise levels; however, they may also lead to a less reliable or less detailed representation of the data. Conversely, higher values of the smoothing parameter, approaching unity, produce a significantly more sensitive measurement, although at the expense of increased noise amplification.

It is worth noting that the first configuration (low smoothing values) is particularly suitable for large-scale structures, where minor geometric defects or local irregularities should not dominate the overall interpretation. In contrast, higher smoothing indices are commonly adopted in fields such as geotechnical engineering, where both macroscopic and microscopic trends must be captured, potentially down to the scale of individual soil grains.

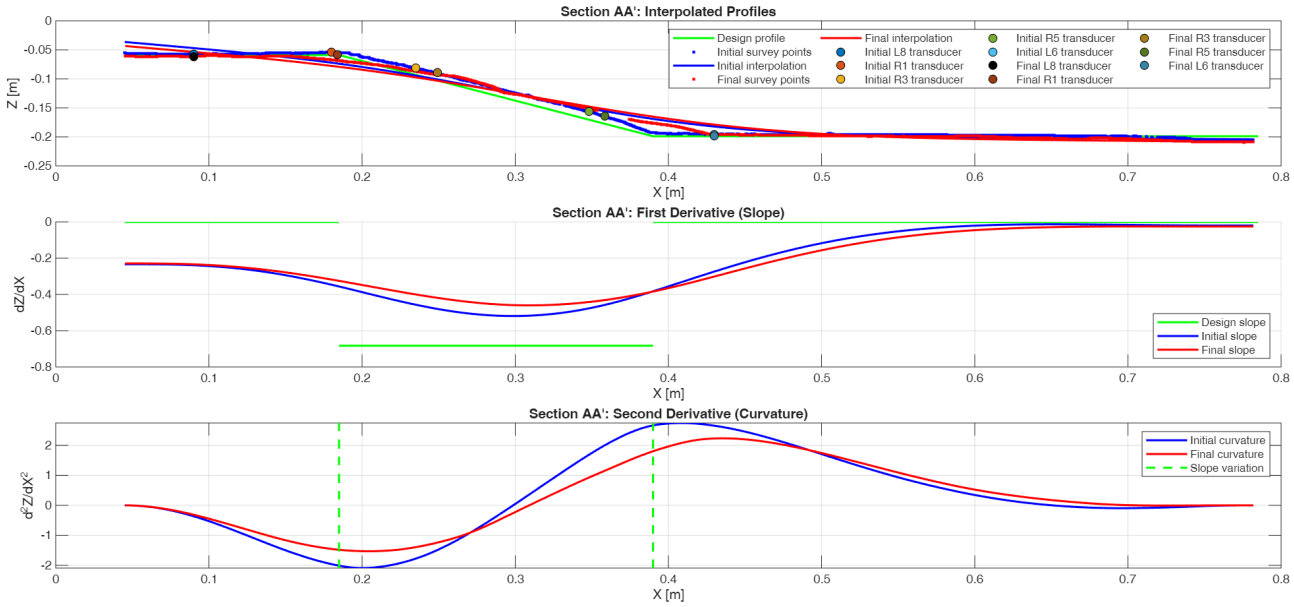

Figure 19: Example of high smoothing parameter with $p = 0.9$ in MATLAB's `csaps`, in geotechnical test, producing a very smooth curve.

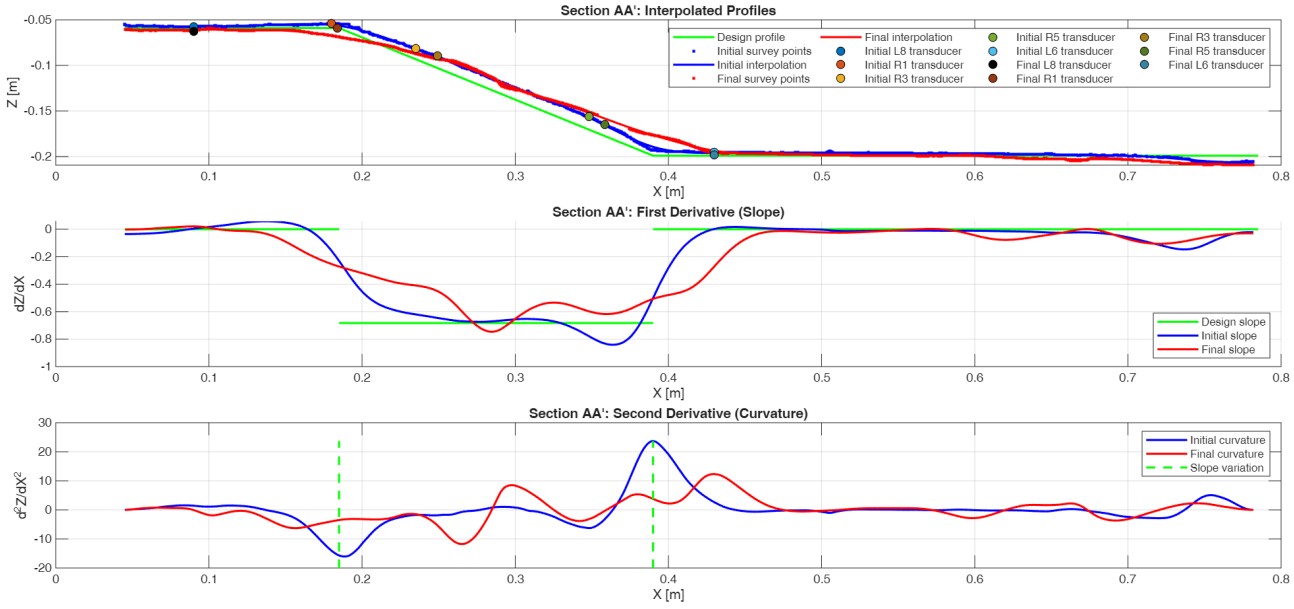

Figure 20: Example of smoothing parameter with $p = 0.9999$ in MATLAB's `csaps`, in geotechnical test, producing a smooth curve.

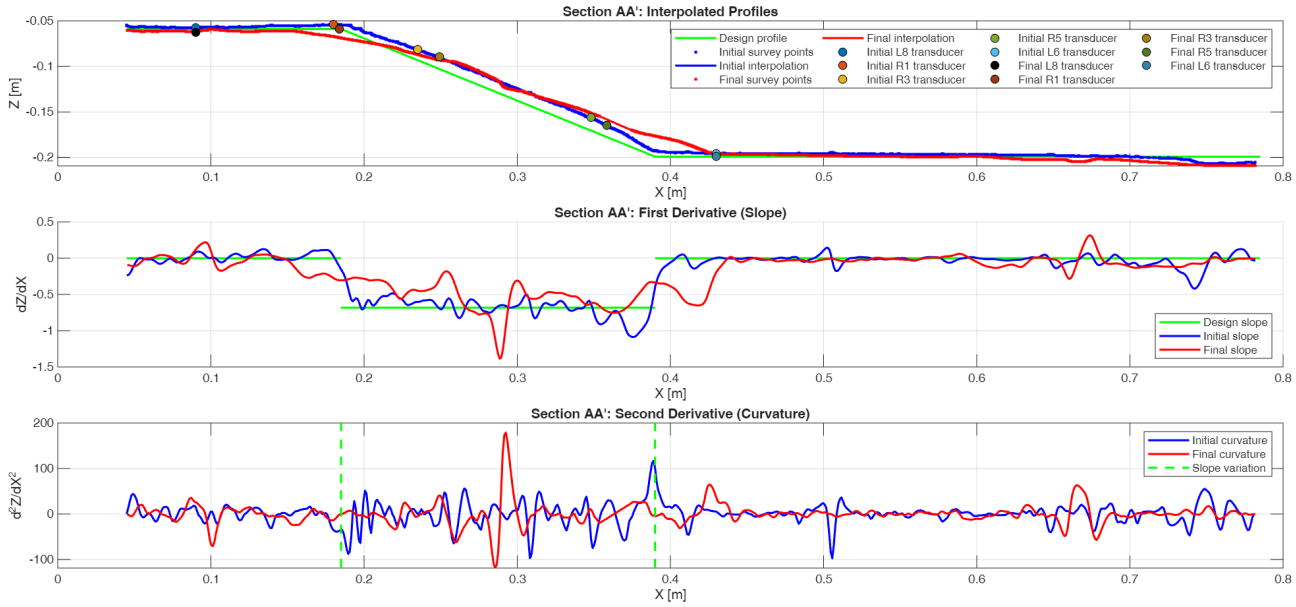

Figure 21: Example of low smoothing parameter with $p = 0.9999999$ in MATLAB's `csaps`, in geotechnical test, producing an interpolating curve.

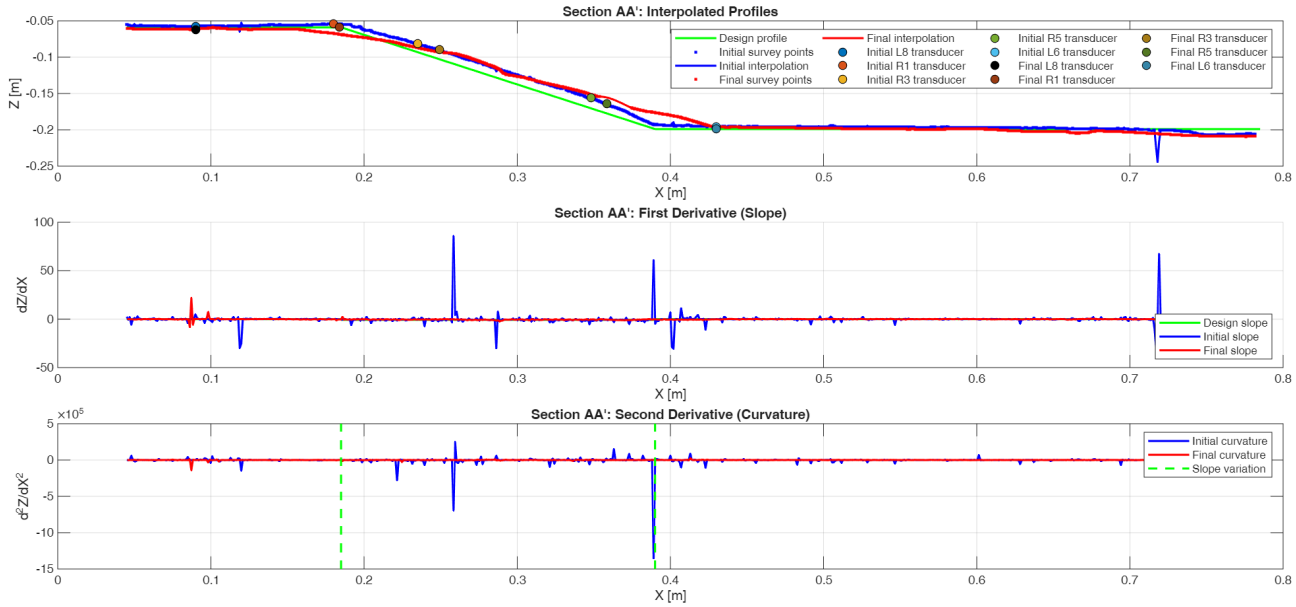

Figure 22: Example of smoothing with parameter with $p = 1$ in MATLAB's `csaps`, in geotechnical test, producing an excessive non-interpolating broken line.

3.3.3 Core routine: `process_section`

All operations for one dataset are encapsulated in

$$[x_q,\ z_q,\ dz,\ ddz,\ x_{\text{unique}},\ z_{\text{unique}},\ sp] = \texttt{process\_section}(\text{filename},\ y_0,\ \delta,\ p).$$

The MATLAB script is organised into the following successive processing stages.

(1) Data import

- `data = load(filename)` reads the ASCII table into an $N \times 3$ matrix.
- Columns are mapped as x = `data(:,1)`, y = `data(:,2)`, z = `data(:,3)`.

(2) Band selection at constant Y

- Build a logical mask `idx` = $(y > y_0 - \delta)$ & $(y < y_0 + \delta)$.
- Extract the section strip: `x_sec` = x(`idx`), `z_sec` = z(`idx`).
- This enforces a thin, quasi-planar profile suited to 1D fitting.

(3) Sorting and de-duplication along x

- Sort by abscissa: [`x_sorted`, `sort_idx`] = `sort`(`x_sec`).
- Reorder ordinates: `z_sorted` = `z_sec`(`sort_idx`).
- Remove repeated abscissae: [`x_unique`, `ia`] = `unique`(`x_sorted`), then `z_unique` = `z_sorted`(`ia`).
- Rationale: spline fitting assumes a single-valued mapping $z = f(x)$; duplicates in x would make the mapping ill-defined.

(4) Cubic smoothing spline fit

- Fit $s(x)$ = `csaps`(`x_unique`, `z_unique`, p), stored in the spline structure `sp`.
- The parameter $p \in [0, 1]$ balances fidelity and smoothness; near 1 yields a curve close to the data, lower values damp noise and oscillations.

(5) Uniform sampling and derivatives

- Build a uniform query grid on the observed domain:

$$x_q = \texttt{linspace}(\min x_{\text{unique}},\ \max x_{\text{unique}},\ 740).$$

- Evaluate profile and derivatives:

$$z_q = s(x_q), \qquad dz = s'(x_q) = \frac{dZ}{dX}, \qquad ddz = s''(x_q) = \frac{d^2Z}{dX^2}.$$

- Implementation uses `fnval(sp, xq)` and `fnder(sp,1/2)` for first/second derivatives.

3.3.4 Applying the routine to both epochs

The function is called on `filename_initial` and `filename_final`, producing two sets of outputs:

$$(x_{q,\text{initial}}, z_{q,\text{initial}}, dz_{\text{initial}}, ddz_{\text{initial}}), \qquad (x_{q,\text{final}}, z_{q,\text{final}}, dz_{\text{final}}, ddz_{\text{final}}),$$

as well as the cleaned input points $(x_{\text{unique}}, z_{\text{unique}})$ for each epoch and the corresponding spline objects `sp_initial`, `sp_final`.

3.3.5 Reference polyline and piecewise slopes

A design/reference profile is defined by node arrays $\{x_{\text{ref}}, z_{\text{ref}}\}$. The segment slopes are computed by finite differences

$$m_k = \frac{z_{\text{ref},k+1} - z_{\text{ref},k}}{x_{\text{ref},k+1} - x_{\text{ref},k}}, \qquad k = 1, \ldots, n-1,$$

and later plotted as a piecewise-constant function for comparison with measured slopes dz_{initial} and dz_{final}.

3.3.6 Visualization: three diagnostic panels

1. **Interpolated profiles (top subplot).** The design polyline is drawn in green. The de-duplicated initial/final points are shown as blue/red markers, and their spline profiles as blue/red lines. A set of "transducer" markers (coloured filled circles) is overlaid to anchor instrument locations in initial and final states.

2. **First derivative (middle subplot).** The measured slopes dZ/dX for initial/final are drawn as continuous blue/red curves. The design slope appears as a green, piecewise-constant overlay built from $\{m_k\}$.

3. **Second derivative (bottom subplot).** The curvature proxy d^2Z/dX^2 for initial/final is shown as blue/red curves. Vertical dashed lines at interior abscissae $x_{\text{ref}}(i)$ highlight design breakpoints where slope changes by construction.

3.3.7 Spline sampling and `.poly` export

The datasets obtained through the smoothing and sampling procedures are subsequently exported as three-dimensional polyline files. These polylines provide a simplified geometric representation of the processed profiles, enabling direct inspection of their spatial configuration. The export format is designed to ensure compatibility with external visualization tools, such as CloudCompare, where the resulting geometries can be analysed and displayed in a three-dimensional environment. In this way,

the numerical outputs of the MATLAB workflow can be readily integrated into standard point-cloud processing platforms for further visual verification and interpretation.

The helper `writePoly(filename, sp, y0, N)` uniformly samples the spline across its domain $[\texttt{sp.breaks}(1), \texttt{sp.breaks}(\text{end})]$, evaluates $z = s(x)$, and writes N vertices one per line in the format `X Y Z` with a constant $Y = y_0$. Two files are produced:

`AAsp_initial.poly`, `AAsp_final.poly`.

This simple XYZ polyline is convenient for downstream tools that accept vertex lists without headers.

3.3.8 Excel export: sheets and variables

In addition to geometric and visual outputs, the results of the numerical processing are also organised in tabular form. The structured export of data into spreadsheet environments preserves the effectiveness and accessibility of the results, particularly in multidisciplinary contexts where three-dimensional visualisation tools may not always be available or familiar to all users. Tabular representations allow the numerical values associated with the analysed profiles to be inspected, compared, and reused in a transparent manner. For this reason, the processed datasets are exported to an Excel workbook in which the different variables and intermediate results are organised into dedicated sheets.

The workbook `SectionAA_profiles.xlsx` collects:

- **Reference:** $(X_{\text{ref}}, Z_{\text{ref}})$ (design polyline).
- **initial_unique:** the de-duplicated initial points used for fitting.
- **initial_interp:** the initial query grid $(X_{q,\text{initial}}, Z_{q,\text{initial}})$ with dZ_{initial} and $d^2Z_{\text{initial}}/dX^2$.
- **final_unique:** the de-duplicated final points used for fitting.
- **final_interp:** the final query grid $(X_{q,\text{final}}, Z_{q,\text{final}})$ with dZ_{final} and d^2Z_{final}/dX^2.

Before writing, all numeric values are rounded to four decimals for stable presentation in spreadsheets.

3.3.9 Interpretation of the derivatives

Given the fitted spline $s(x)$,

$$\text{profile } z(x) = s(x), \qquad \text{slope } \frac{dZ}{dX}(x) = s'(x), \qquad \text{curvature proxy } \frac{d^2Z}{dX^2}(x) = s''(x).$$

The first derivative highlights local steepening/flattening with respect to x; the second derivative emphasizes changes in slope (inflection and curvature-like behaviour). Because derivatives amplify measurement noise, p controls the trade-off between detail and stability.

3.3.10 Assumptions and practical notes

Before presenting the operational details of the workflow, it is useful to clarify a number of working assumptions and practical considerations adopted throughout the analysis. These notes define the conventions used for units, data sampling, and spline fitting, ensuring consistency in the interpretation of the computed derivatives. They also highlight several implementation choices that may influence the robustness and clarity of the results.

- **Units:** coordinates are in meters; dZ/dX is dimensionless; d^2Z/dX^2 has units m/m^2.
- **No extrapolation:** the query grid x_q spans the observed domain only $[\min x_{\text{unique}}, \max x_{\text{unique}}]$.
- **Band width:** choose δ small enough to avoid mixing out-of-plane geometry, but large enough to retain sufficient points for a stable fit.
- **Smoothing:** for profiles intended mainly for visualization, near-interpolatory p is acceptable; for derivative analysis, moderate smoothing (slightly smaller p) often improves robustness.
- **Name reuse caution:** the symbol `dz` is used twice in the script (first as reference Δz for segment slopes, later overwritten as derivative arrays). This is harmless locally but can be confusing; consider renaming one of them in future revisions for clarity.

3.3.11 Representation of transducers measurements along section AA'

Along section AA′, the outcomes of the displacement and rotation transducers installed directly in the soil specimen are represented. These measuring devices, positioned at predefined control points, allow the comparison between the numerical reference profile and the deformed geometries obtained from the surveys, before and after the test. In the graphical output, each transducer is rendered as a discrete marker with a specific colour code, enabling the distinction between initial and final measurements. Their location along the profile corresponds to the actual installation depth and horizontal position within the tested sample, thus ensuring that the measured response is not abstract but physically related to the specimen geometry (Figure 36).

The representation highlights the correspondence between the interpolated spline profiles of the pre- and post-test sections and the punctual data acquired from the transducers. In particular, displacements normal to the reference surface are plotted in direct association with the interpolated curves, while angular variations are encoded through marker differentiation. This strategy makes it possible to visualise, within the same drawing, both the global deformation trend of the soil section and the local response at instrumented points. As a result, the profile of section AA′ becomes not only a geometric

reconstruction of the surveyed surfaces, but also a synthetic map of the in-situ instrumentation results, bridging the numerical analysis of curvature and slope with the experimental evidence provided by the transducers.

3.4 Integration with Python scripting: automating data acquisition strategy

This research activity is based on the integration of various software programs for semi-automatic workflow management, particularly Matlab and Python. The following chapters will present a method for standardized point cloud management aimed at obtaining information using models consisting of point clouds and meshes. Obtaining such results from generic survey data requires the ability to manage codes in various languages. For this reason, some principles are illustrated below.

3.4.1 Installing PyAutoGUI and Python environment

To automate workflow processes using PyAutoGUI, the first step is to install an up-to-date Python environment equipped with an efficient package manager (PIP). To ensure compatibility with PyAutoGUI and other modern packages, it is recommended to install Python 3.10, a version compatible with most libraries currently used in automation tasks. During the installation:

- Download the Python 3.10 installer from the official website: `https://www.python.org/downloads/release/python-3100/`.
- Run the installer and make sure to select **"Add Python 3.10 to PATH"** before proceeding with the installation. This option allows the operating system to recognize Python from any terminal or command prompt.
- Verify that PIP is included in the installation by selecting the "Install pip" checkbox during the guided process.

Once the installation is complete, you can verify its proper functioning by opening the terminal (Command Prompt or PowerShell) and executing the following commands:

```
python --version
pip --version
```

Both commands should return the installed version without errors.

3.4.2 Installing PyAutoGUI

With Python and PIP correctly installed, PyAutoGUI can be installed in a few simple steps:

1. Open the terminal

2. Execute the command:

```
pip install pyautogui
```

PIP will automatically download and install PyAutoGUI along with its dependencies. Upon completion, it is advisable to test the installation by opening a Python session:

```
python
>>> import pyautogui
>>> pyautogui.size()
```

If the command returns the screen size (e.g., '(1920, 1080)'), the installation was successful.
By following these simple steps, you can set up a modern Python working environment with PyAutoGUI, a fundamental tool for automating multi-software operations through user input simulation. The provided video guide offers a helpful step-by-step visual walkthrough during the installation process.

3.5 Python scripting for dimensional analysis with MATLAB

To streamline the preparation of the MATLAB script, we developed a GUI-based Python utility that automatically generates the `.m` file and pre-populates it with user-supplied parameters. The complete source code and usage notes are provided in Appendix, see Listing A.12.

3.6 Step-by-step explanation of the Python wizard (PyAutoGUI + Tkinter)

3.6.1 Goal and scope

The script implements an interactive *wizard* that gathers user inputs (files, folders, and numeric parameters) and prepares data needed to emit a MATLAB `.m` script for a Section AA workflow. It favours point-and-click dialogs (Tkinter) when available and falls back to PyAutoGUI prompts when Tkinter is missing. The intended platform is Windows ("native" Python; not WSL), and the final MATLAB script is meant to be saved at `path_ SectionAA_script.m`.

3.6.2 Dependencies, imports, and environment

- **Libraries.** The code imports:
  - `pyautogui` as `pg` for alerts, prompts, and confirmations;
  - `sys`, `os` for process control and filesystem;
  - `datetime` (reserved for time-stamping; not used in the snippet);
  - `Template` from `string` (reserved for later code templating; not used in the snippet).
- **Optional GUI backend.** It tries to import `tkinter` and `filedialog`; on failure, it sets `tk = None` to signal that only PyAutoGUI prompts are available.
- **PyAutoGUI safety.** `pg.PAUSE = 0.2` inserts a short delay after each call (improves stability); `pg.FAILSAFE = True` lets the user abort by moving the mouse to the top-left corner.

3.6.3 GUI utility functions (file and folder pickers, numeric input, yes/no)

`pick_file(title, initialdir)` Presents a file-open dialog. If Tkinter is unavailable, it alerts the user and falls back to a text prompt requesting a full path. It returns a string (possibly empty). Internally:

1. If `tk is None`, show `pg.alert(...)` then `pg.prompt(...)`.
2. Else, create a hidden Tk root, call `filedialog.askopenfilename(...)` with the given title and `initialdir`, then destroy the root.

`pick_dir(title, initialdir)` Same pattern for directory selection via `filedialog.askdirectory(...)`. If Tkinter is not present, it prompts the user to type a directory path (e.g. `D:/output`).

`ask_float(msg, default)` Prompts the user for a floating-point value with an input box:

1. Shows a `pg.prompt` with the message and a default (converted to string).
2. If the user cancels, a confirmation dialog asks whether to abort the wizard (`sys.exit(0)`) or continue.
3. On input, the string is trimmed, commas are converted to dots (locale-friendly), and `float(...)` is attempted.

4. On `ValueError`, an error alert appears and the loop repeats.

This loop ensures a valid float or a clean, explicit exit.

`ask_int(msg, default)` Identical logic to `ask_float` but enforcing integer conversion via `int(...)`.

`yesno(msg, default_yes=True)` Shows a two-button confirmation dialog ("Sì"/"No") and returns `True` if the user clicks "Sì". (The `default_yes` flag is not used in the current implementation but may guide future defaults.)

3.6.4 Wizard start and input files (Initial/Final)

Welcome alert. An initial `pg.alert` explains that the wizard will create a MATLAB `.m` file for section processing (Figure 23).

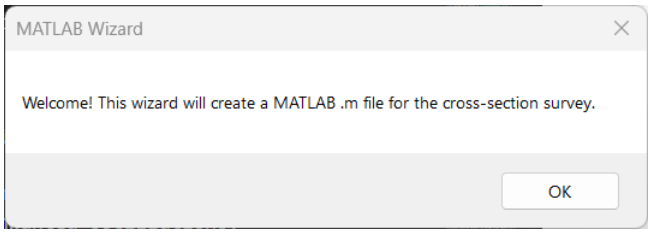

Figure 23: Welcome dialog of the MATLAB Wizard, which guides the user in creating a `.m` file for the cross-section survey.

Selecting the "INITIAL" file.

1. Call `pre_path = pick_file("Seleziona file INITIAL (PRE)")`.
2. While the path is empty or does not exist (`not os.path.exists(pre_path)`), ask the user if they want to retry via `yesno(...)`.
3. If the user declines, exit the program; otherwise re-open the file picker.

Selecting the "FINAL" file. The same loop is applied for `post_path`, ensuring a valid path or a clean exit.

3.6.5 Primary numeric parameters (validated prompts)

The wizard collects the core parameters needed by the MATLAB workflow:

- **`y0`** (*section midline in Y*): requested with example `0.165`; validated as float (Figure 24).

- **`delta`** (*half-thickness in Y*): example `0.0002`; validated as float (Figure 25).

- **`p`** (*smoothing parameter,* 0..1): example `0.9999999`; validated as float (Figure 26).

- **`n_poly`** (*number of points for* `.poly` *export*): example `1000`; validated as integer (Figure 27).

Each input uses the `ask_float`/`ask_int` loops (numeric parsing, comma-to-dot conversion, and explicit abort logic).

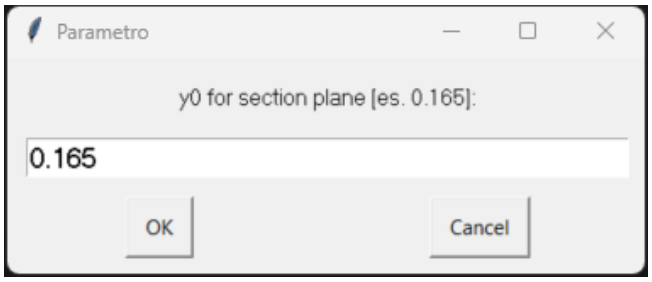

Figure 24: Input dialog for specifying the parameter y_0 corresponding to the section plane (example: $y_0 = 0.165$).

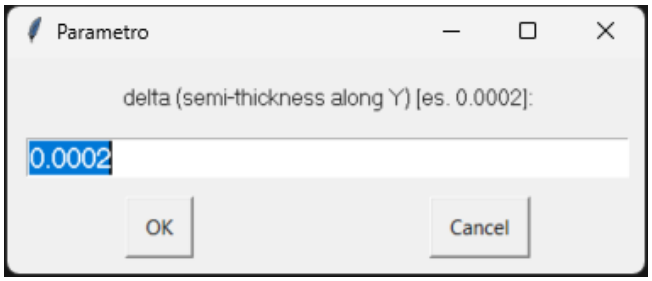

Figure 25: Input dialog for defining the parameter δ, representing the semi-thickness along the Y axis (example: $\delta = 0.0002$).

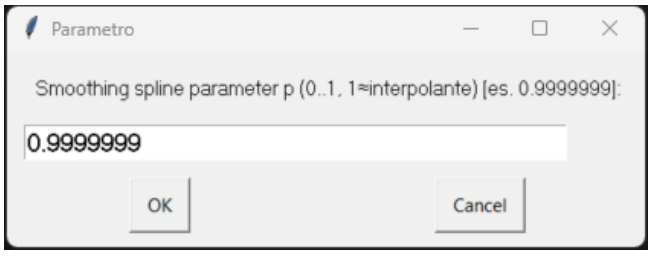

Figure 26: Input dialog for setting the smoothing spline parameter p ($0 \leq p \leq 1$), where values close to 1 behave nearly interpolatory (example: $p = 0.9999999$).

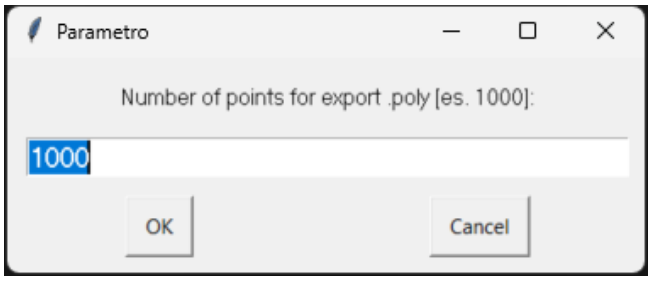

Figure 27: Input dialog for specifying the number of points to be used for the export in `.poly` format.

3.6.6 Reference polyline (default vs. manual)

Choosing between default and custom vertices. A yes/no dialog offers a pre-defined polyline (Figure):

$$\texttt{x\_ref} = [0.045, 0.185, 0.390, 0.785], \quad \texttt{z\_ref} = [-0.059, -0.059, -0.199, -0.199].$$

- If the user accepts, these arrays are assigned directly.
- Otherwise, the wizard prompts for space-separated lists (one for x, one for z), with defaults shown.

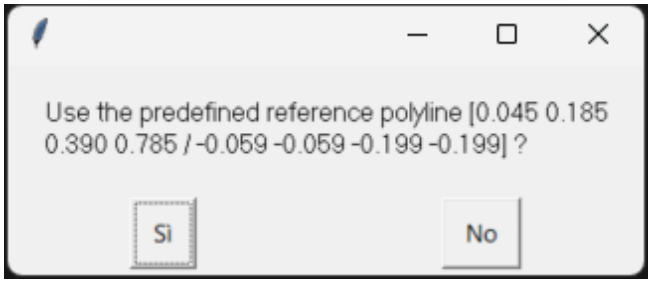

Figure 28: Dialog asking whether to use a predefined reference polyline, with the coordinates explicitly listed in the message.

Parsing and validation of manual input. When the user types custom lists:

1. Each string is split on whitespace; each token replaces comma with dot and is converted to `float`.
2. Consistency checks: the two lists must have the same length and at least two points.
3. On any parsing or validation error, the wizard alerts the user and silently falls back to the default polyline.

3.6.7 Output names and folder (robust path handling)

Base names. The wizard asks for:

- `excel_name` (default `SectionAA_profiles.xlsx`);
- `poly_pre` (default `AAsp_pre.poly`);
- `poly_post` (default `AAsp_post.poly`).

If the user cancels or submits an empty string, the default is enforced via the `or "Default.ext"` pattern.

Output folder. A directory picker asks for the (absolute) folder to be used later by MATLAB for Excel and `.poly` outputs. If the user leaves it empty, a loop asks whether to type a path manually (e.g. `D:/output`) or to proceed without setting it (in which case paths can remain relative) (Figure 29).

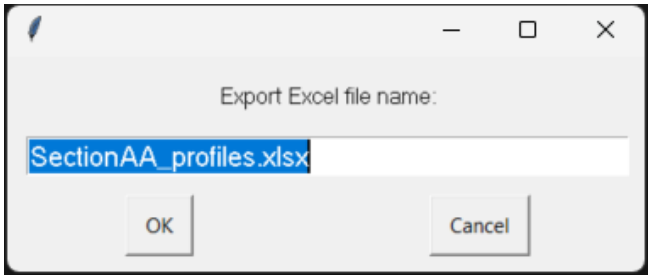

Figure 29: Dialog for specifying the name of the Excel file to be exported (e.g., `SectionAA_profiles.xlsx`).

MATLAB-friendly path join. Function `mpath_join(folder, name)` composes a path with `os.path.join` and then replaces backslashes with forward slashes:

- If `folder` is empty, it returns `name` unchanged, allowing relative paths;
- Otherwise, it returns a slash-normalized absolute path (preferred by MATLAB on Windows).

Three paths are produced: `excel_path`, `poly_pre_p`, `poly_post_p`.

3.6.8 Formatting numeric vectors for MATLAB injection

`vec_repr(v)` (pretty printing). This helper creates a MATLAB-like literal for a numeric vector:

1. For each element `val`:
   - if $|\texttt{val}| \geq 10^{-6}$, format with three decimals (`{:.3f}`) and then strip trailing zeros and any terminal dot;

- else, format with six decimals (`{:.6f}`) to preserve small magnitudes.

2. Join the tokens with single spaces and wrap them with `[` and `]` to obtain, e.g., `"[0.045 0.185 0.39 0.785]"`.

The function is used to compute the strings `x_ref_repr` and `z_ref_repr`, which are suitable for direct insertion into MATLAB code (e.g. a template).

3.6.9 User-experience and robustness notes

- **Locale tolerance.** Numeric prompts accept decimal commas by replacing them with dots before parsing.
- **Explicit exits.** On cancellations, the user is asked whether to abort. Choosing "Sì" exits via `sys.exit(0)`; otherwise the prompt reappears.
- **Graceful degradation.** If Tkinter is not available, the wizard still runs using PyAutoGUI prompts; only the native file/folder pickers are replaced by typed paths.
- **Safety.** `pg.FAILSAFE = True` lets users abort instantly by moving the mouse to the top-left screen corner (PyAutoGUI feature).
- **Forward slashes.** MATLAB on Windows accepts forward slashes in paths; normalizing here avoids common escaping issues with backslashes.

3.6.10 What follows downstream (context)

Although not shown in this snippet, two imports hint at the next steps:

- `datetime` (for time-stamping the generated MATLAB script);
- `Template` from `string` (to interpolate all collected variables into a MATLAB `.m` template and save it at `D:/aaa/SectionAA_script.m`).

At this point, all essential inputs are validated and formatted, so the script can safely render the final MATLAB file and proceed with subsequent processing.

3.6.11 Dimensional verification against design/reference models

Chapter 4. Experimental applications and case studies

4.1 Description of selected case studies

The selected case studies include both structures and infrastructures located in urban and natural areas of Northern Italy. Their selection is arbitrary and based on the analysis of different types of structures, with the aim of examining various application scenarios.

The case studies presented in the following sections represent multi-scalar examples drawn from different disciplinary contexts. The objective of this discussion is to demonstrate the effectiveness of morphological analysis based on differential methods and curvature evaluation across a range of applications. In particular, the proposed approach enables the comparison of different point clouds, supporting the monitoring and interpretation of deformations at multiple scales. This framework allows both the investigation of highly detailed local phenomena—typical of small-scale case studies—and the assessment of the overall structural behaviour in large-scale analyses, where the mitigation of background noise becomes a critical factor for reliable interpretation.

4.2 Point cloud processing and dimensional analysis workflows

Processing the point clouds determined so far begins with identifying the rotational translation operations to be applied to the elements. The goal is to prepare the model for easy and precise evaluation, identifying a pair of reference points that represent a useful direction for subsequent analysis (Figure 56). This process is done in CloudCompare, a software suitable for managing point clouds and optimized to easily process and visualize large elements.

The next step involves calculating the element's rotational transformation values around the vertical axis. CloudCompare offers the ability to determine the rotation angle directly through the Pick Points command, selecting two points and obtaining information about the segment connecting them. To generalize this procedure, the Python code for determining the angle was developed using a graphical interface that requires the input of point coordinates (Listing 4.1, Figure 30).

Listing 4.1: Python code for calculating the Z-axis rotation angle.

```
1 import numpy as np
2 import tkinter as tk
3 from tkinter import messagebox
4
5 def calculate_rotation():
```

```
    try:
        x1 = float(entry_x1.get())
        y1 = float(entry_y1.get())
        x2 = float(entry_x2.get())
        y2 = float(entry_y2.get())

        dx = x2 - x1
        dy = y2 - y1

        theta = np.arctan2(dy, dx)  # angle in radians
        rotation_angle = -theta
        rotation_angle_degrees = np.degrees(rotation_angle)

        result = (f"Rotation angle (radians): {rotation_angle:.6f}\n"
                  f"Rotation angle (degrees): {rotation_angle_degrees:.2f}deg")
        messagebox.showinfo("Rotation Angle", result)
    except ValueError:
        messagebox.showerror("Input Error", "Please enter valid numeric values.")

# Create the main window
root = tk.Tk()
root.title("Z-Axis Rotation Angle Calculator")

# Labels and entries
tk.Label(root, text="x1:").grid(row=0, column=0, padx=5, pady=5, sticky='e')
entry_x1 = tk.Entry(root)
entry_x1.grid(row=0, column=1, padx=5, pady=5)

tk.Label(root, text="y1:").grid(row=1, column=0, padx=5, pady=5, sticky='e')
entry_y1 = tk.Entry(root)
entry_y1.grid(row=1, column=1, padx=5, pady=5)
```

```
37
38 tk.Label(root, text="x2:").grid(row=2, column=0, padx=5, pady=5, sticky='
       e')
39 entry_x2 = tk.Entry(root)
40 entry_x2.grid(row=2, column=1, padx=5, pady=5)
41
42 tk.Label(root, text="y2:").grid(row=3, column=0, padx=5, pady=5, sticky='
       e')
43 entry_y2 = tk.Entry(root)
44 entry_y2.grid(row=3, column=1, padx=5, pady=5)
45
46 # Button
47 calculate_button = tk.Button(root, text="Calculate Rotation Angle",
       command=calculate_rotation)
48 calculate_button.grid(row=4, column=0, columnspan=2, pady=10)
49
50 root.mainloop()
```

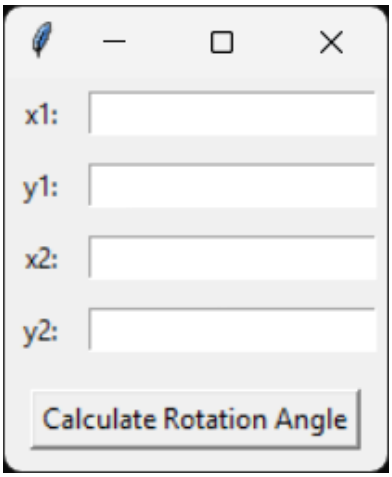

Figure 30: Welcome dialog of the MATLAB Wizard, which guides the user in creating a `.m` file for the cross-section survey.

4.3 Case study of a geotechnical application

Our first case study builds upon the geotechnical analysis example previously discussed in Section 2.5 (p. 52).

In this first geotechnical application, a model based on an ideal design geometry is represented (Figure 31). This was determined by surveying the initial geometry (Figure 32) and the final situation (Figure 33). Vertical section positions are represented in Figure 34 and 35.

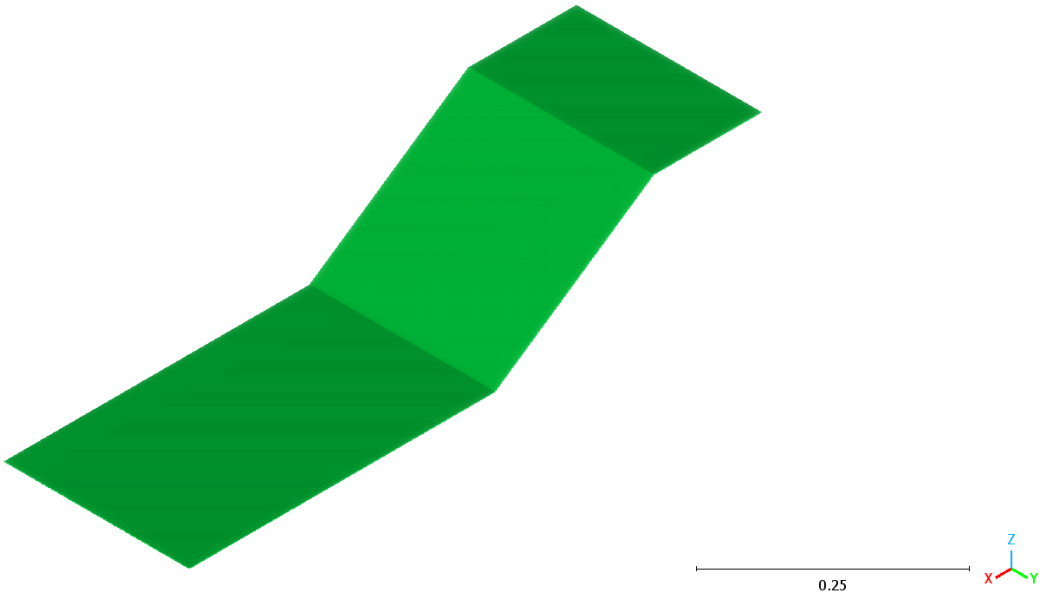

Figure 31: Representation of the design model.

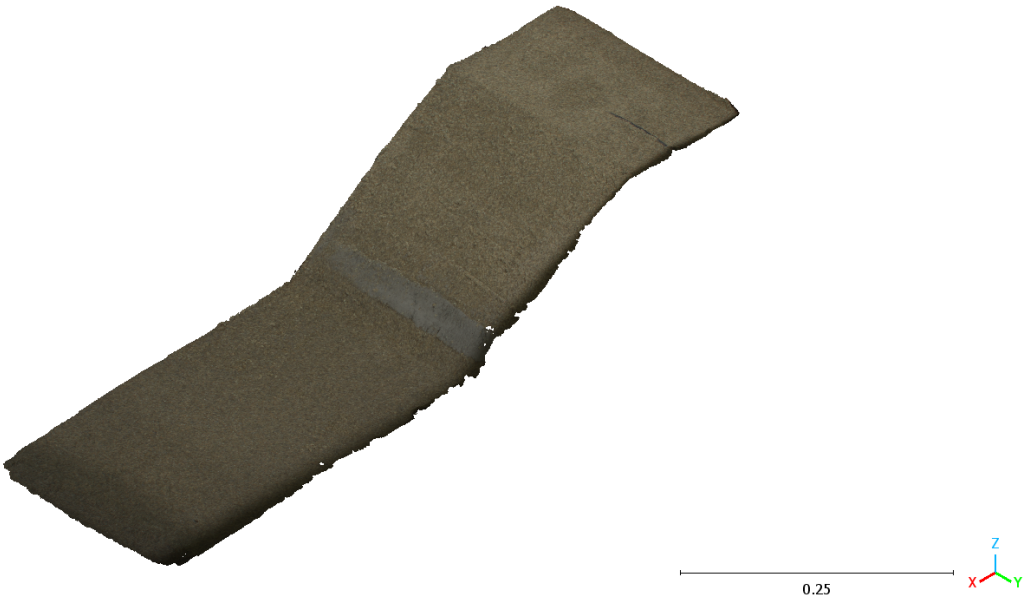

Figure 32: Representation of the initial point cloud model.

Figure 33: Representation of the final point cloud model.

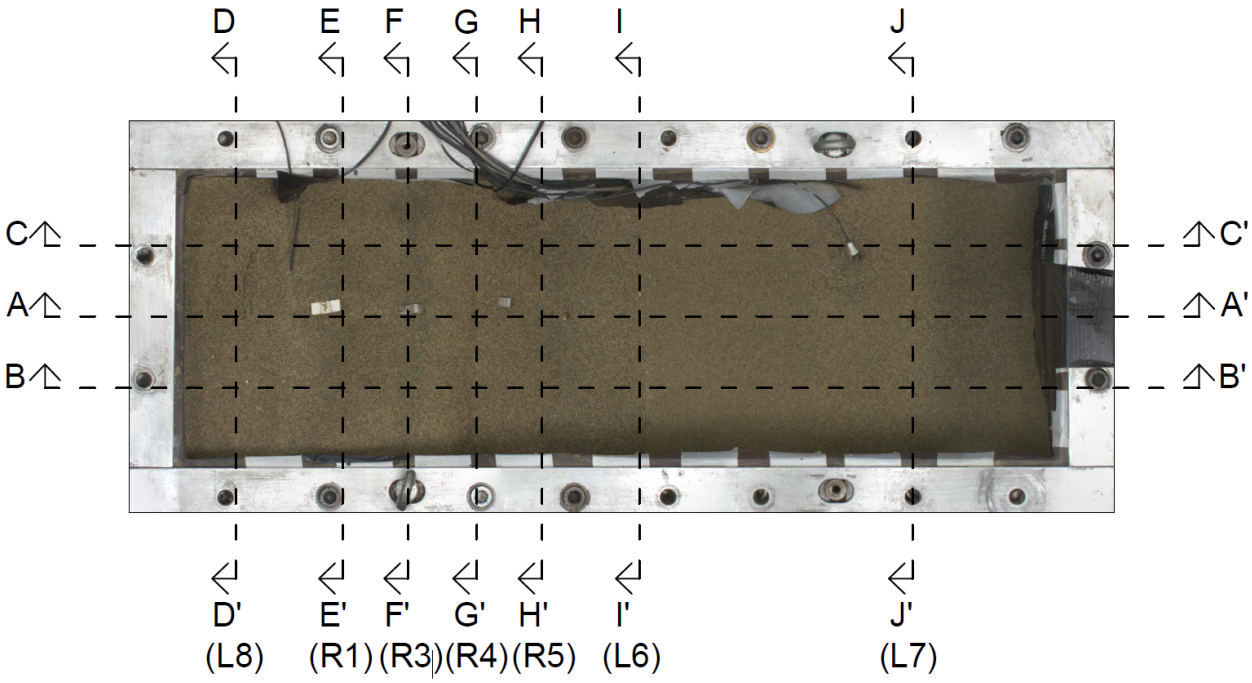

Figure 34: Representation of vertical sections.

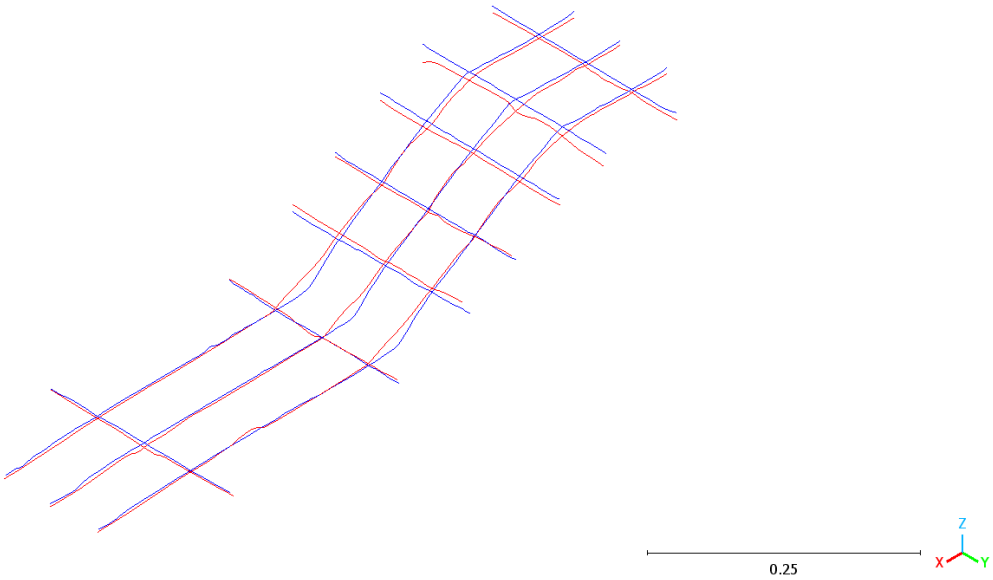

Figure 35: Representation of 3D model vertical sections.

Results obtained by the MATLAB code are presented in the following figures and tables. All the required longitudinal and transverse section profiles were automatically extracted from the reconstructed datasets in order to enable a consistent geometric analysis of the surveyed surfaces.

In particular, Section AA' (Figure 36) includes the positions of the displacement transducers together with the corresponding measured values recorded during the experiment, allowing a direct comparison between the geometrical profiles derived from the point cloud and the instrumental readings.

The smoothing parameter P was calibrated and optimised in order to highlight discontinuities occurring at the soil failure points, enhancing the identification of slope changes and local breaks in the terrain profiles.

Vertical Section AA'. See figure 36 and listing A.2 in appendix A.

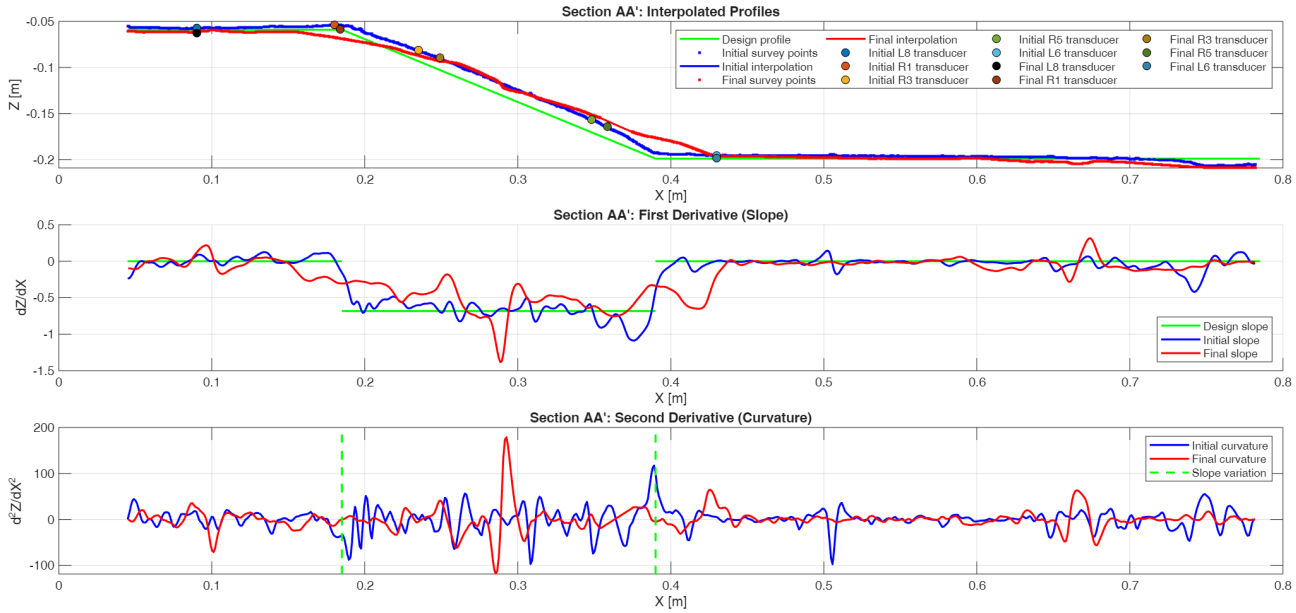

Figure 36: Vertical section AA'.

Cross section	X position [m]	Z design [m]	Z initial [m]	Z final [m]
DD' (L8 transducer)	0.090	-0.059	-0.057	-0.061
EE' (R1 transducer)	0.180	-0.059	-0.054	-0.067
FF' (R3 transducer)	0.235	-0.093	-0.082	-0.087
GG' (R4 transducer)	0.293	-0.133	-0.120	-0.123
HH' (R5 transducer)	0.348	-0.170	-0.156	-0.151
II' (L6 transducer)	0.430	-0.199	-0.195	-0.196
JJ' (L7 transducer)	0.660	-0.199	-0.198	-0.203

Table 12: AA' vertical section z values.

Vertical Section BB'. See figure 37 and listing A.3 in appendix A.

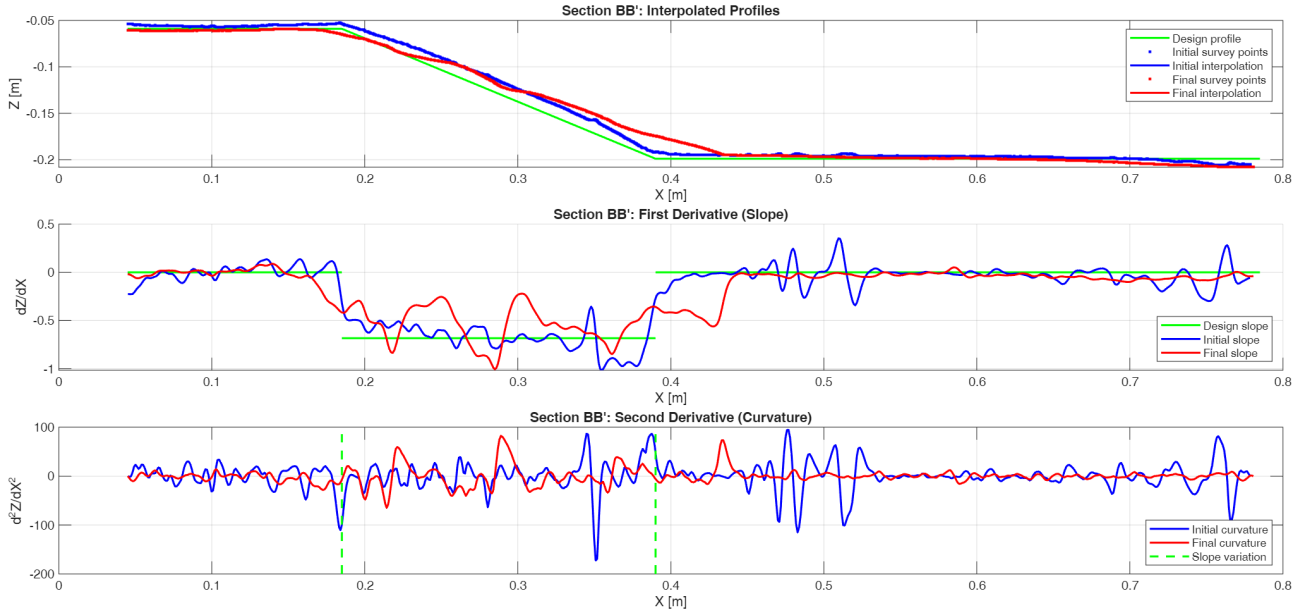

Figure 37: Vertical section BB'.

Cross section	X position [m]	Z design [m]	Z initial [m]	Z final [m]
DD’	0.090	-0.059	-0.057	-0.061
EE’	0.180	-0.059	-0.054	-0.067
FF’	0.235	-0.093	-0.082	-0.087
GG’	0.293	-0.133	-0.120	-0.123
HH’	0.348	-0.170	-0.156	-0.151
II’	0.430	-0.199	-0.195	-0.196
JJ’	0.660	-0.199	-0.198	-0.203

Table 13: BB’ vertical section z values.

Vertical Section CC’. See figure 38 and listing A.4 in appendix A.

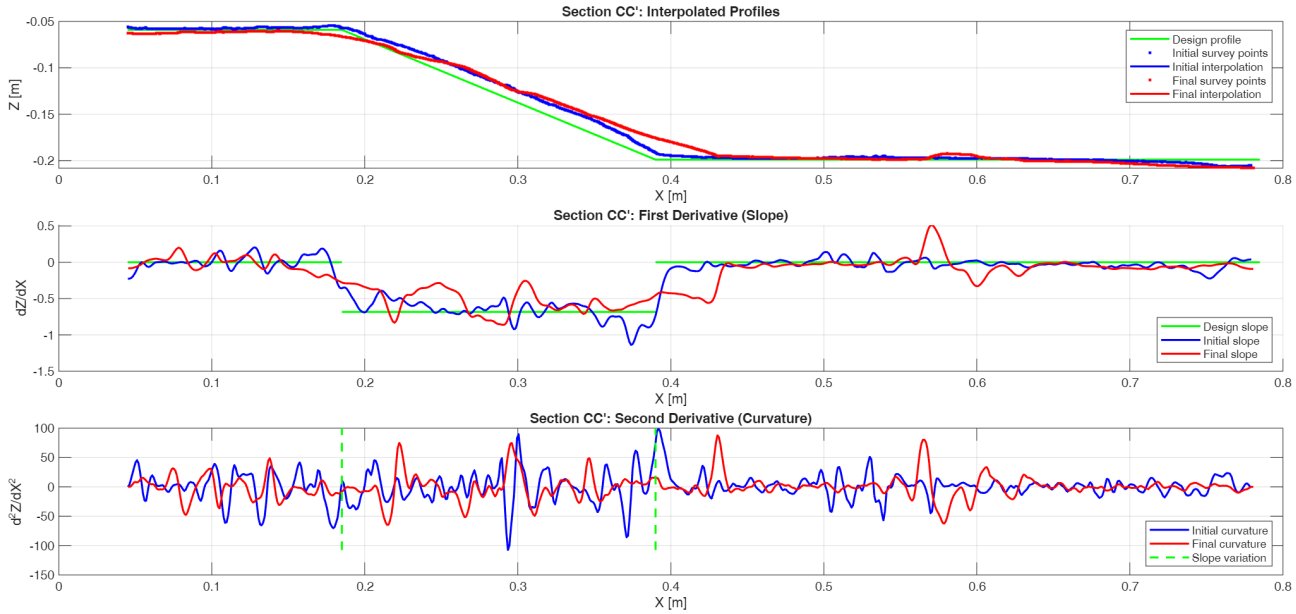

Figure 38: Vertical section CC’.

Cross section	X position [m]	Z design [m]	Z initial [m]	Z final [m]
DD’	0.090	-0.059	-0.058	-0.062
EE’	0.180	-0.059	-0.055	-0.065
FF’	0.235	-0.093	-0.082	-0.089
GG’	0.293	-0.133	-0.120	-0.121
HH’	0.348	-0.170	-0.155	-0.151
II’	0.430	-0.199	-0.197	-0.194
JJ’	0.660	-0.199	-0.199	-0.201

Table 14: CC’ vertical section z values.

Vertical Section DD’. See figure 39 and listing A.5 in appendix A.

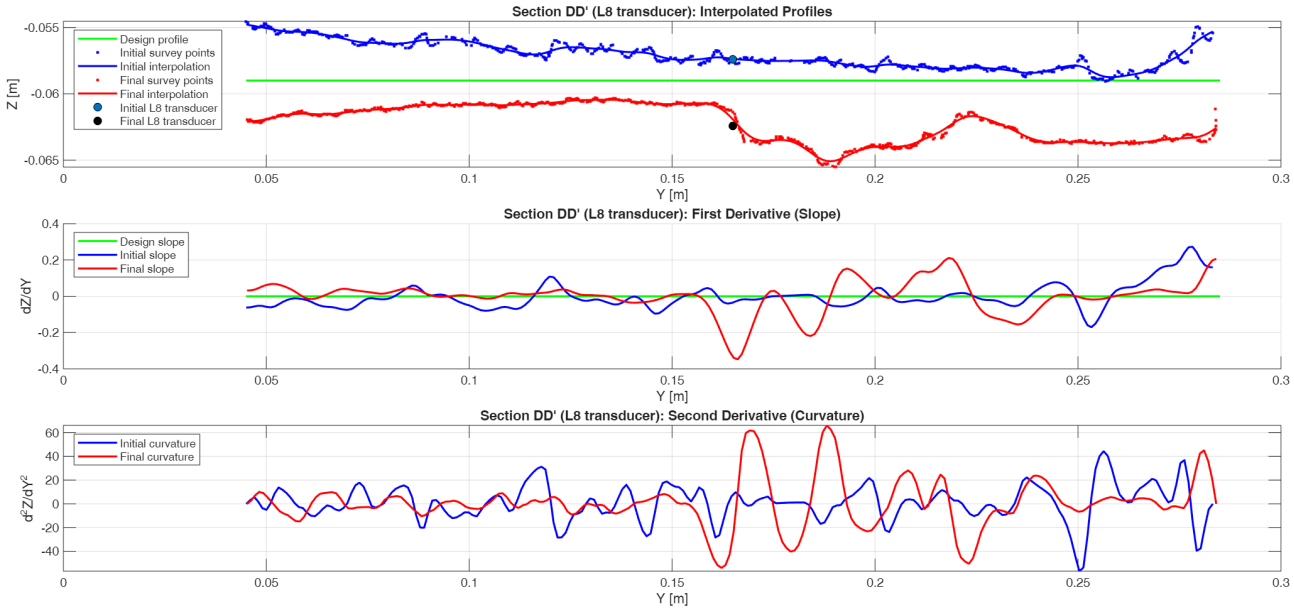

Figure 39: Vertical section DD'.

Cross section	Y position [m]	Z design [m]	Z initial [m]	Z final [m]
BB'	0.105	-0.059	-0.056	-0.061
AA'	0.165	-0.059	-0.057	-0.062
CC'	0.225	-0.059	-0.058	-0.062

Table 15: DD' vertical section z values.

Vertical Section EE'. See figure 40 and listing A.6 in appendix A.

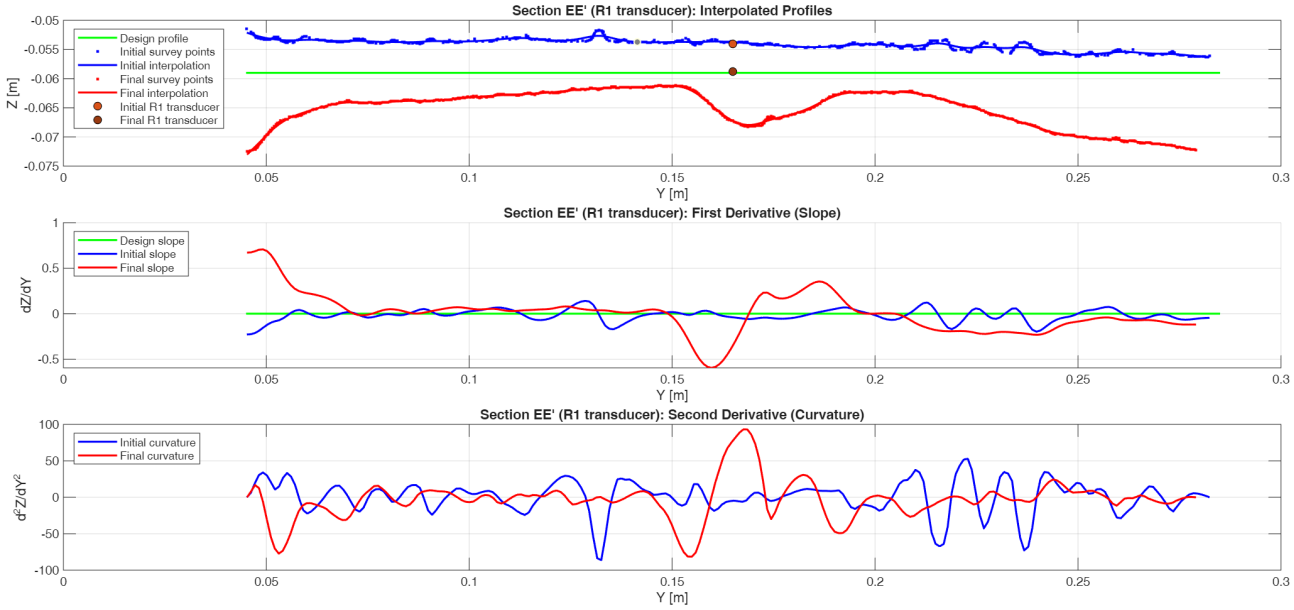

Figure 40: Vertical section EE'.

Vertical Section FF'. See figure 41 and listing A.7 in appendix A.

Cross section	Y position [m]	Z design [m]	Z initial [m]	Z final [m]
BB’	0.105	-0.059	-0.053	-0.063
AA’	0.165	-0.059	-0.054	-0.067
CC’	0.225	-0.059	-0.055	-0.065

Table 16: EE’ vertical section z values.

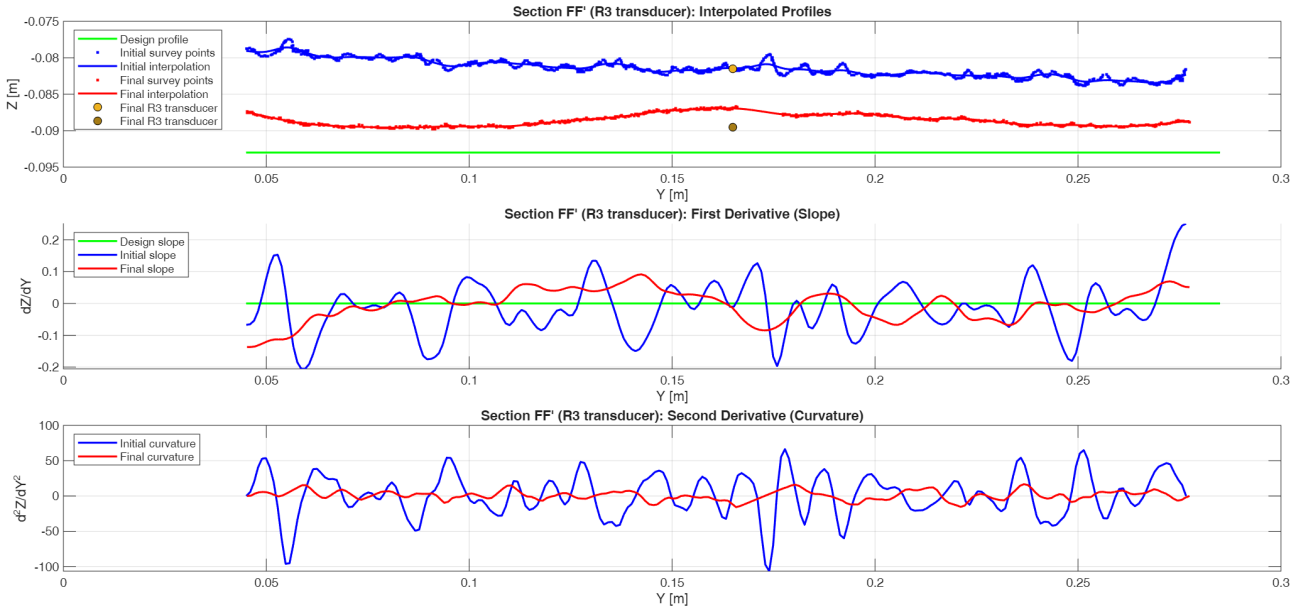

Figure 41: Vertical section FF’.

Cross section	Y position [m]	Z design [m]	Z initial [m]	Z final [m]
BB’	0.105	-0.093	-0.081	-0.089
AA’	0.165	-0.093	-0.081	-0.087
CC’	0.225	-0.093	-0.082	-0.089

Table 17: FF’ vertical section z values.

Vertical Section GG’. See figure 42 and listing A.8 in appendix A.

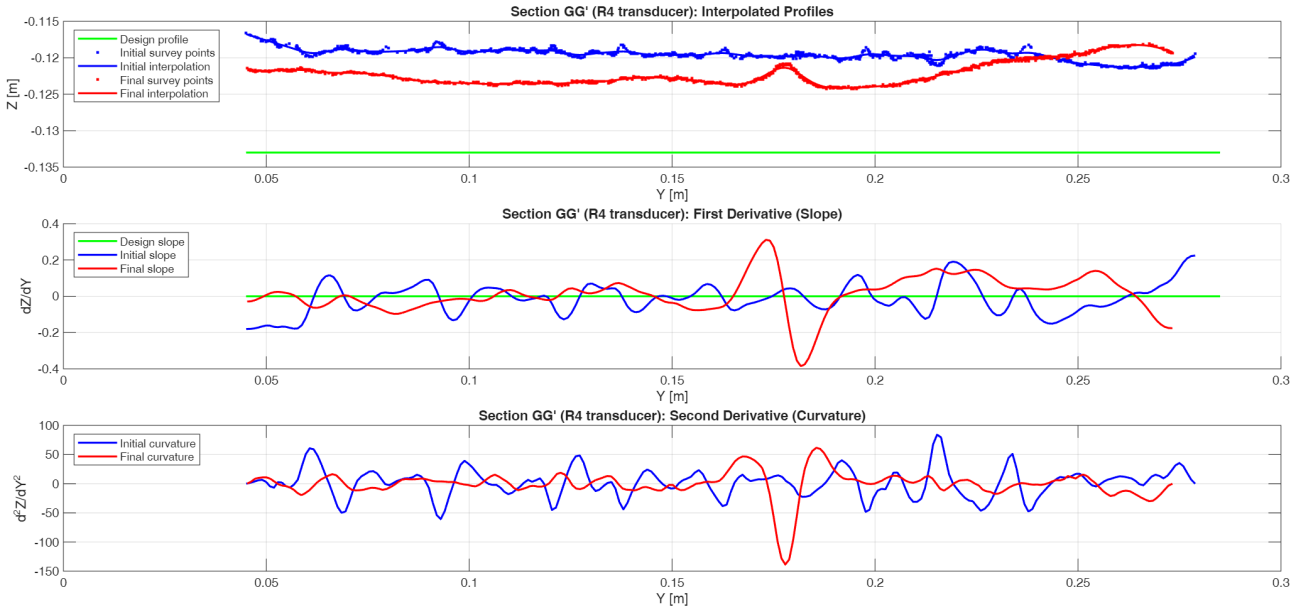

Figure 42: Vertical section GG'.

Cross section	Y position [m]	Z design [m]	Z initial [m]	Z final [m]
BB'	0.105	-0.133	-0.119	-0.124
AA'	0.165	-0.133	-0.119	-0.124
CC'	0.225	-0.133	-0.119	-0.121

Table 18: GG' vertical section z values.

Vertical Section HH'. See figure 43 and listing A.9 in appendix A.

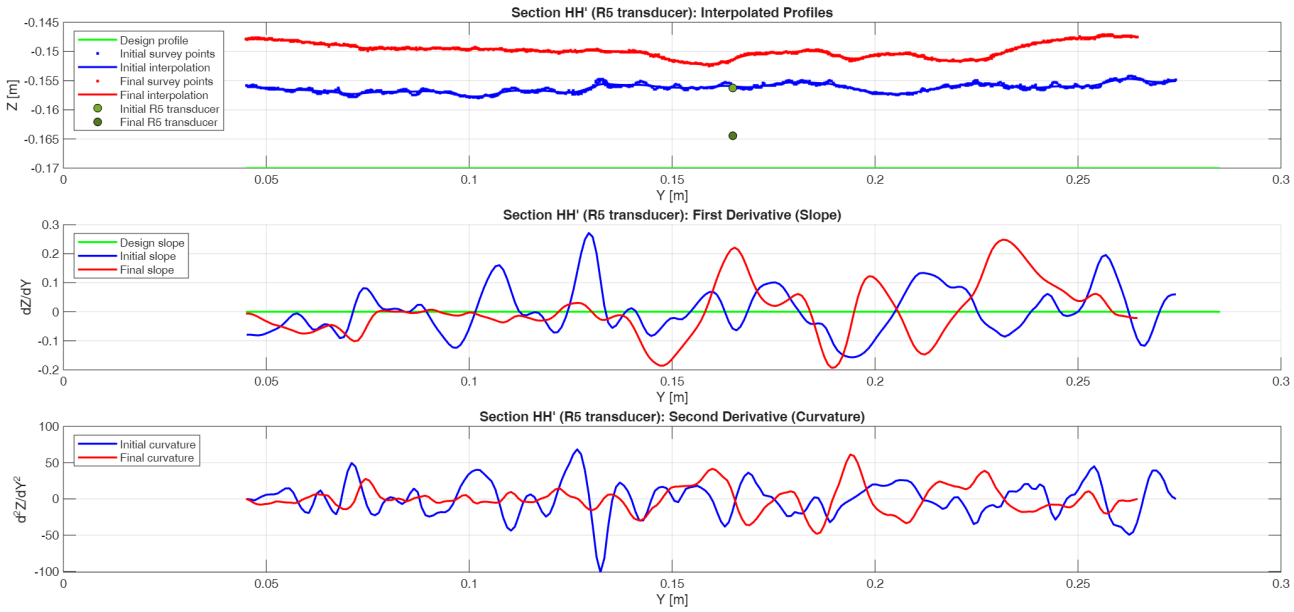

Figure 43: Vertical section HH'.

Vertical Section II'. See figure 44 and listing A.10 in appendix A.

Cross section	Y position [m]	Z design [m]	Z initial [m]	Z final [m]
BB’	0.105	-0.170	-0.157	-0.150
AA’	0.165	-0.170	-0.156	-0.151
CC’	0.225	-0.170	-0.156	-0.151

Table 19: HH’ vertical section z values.

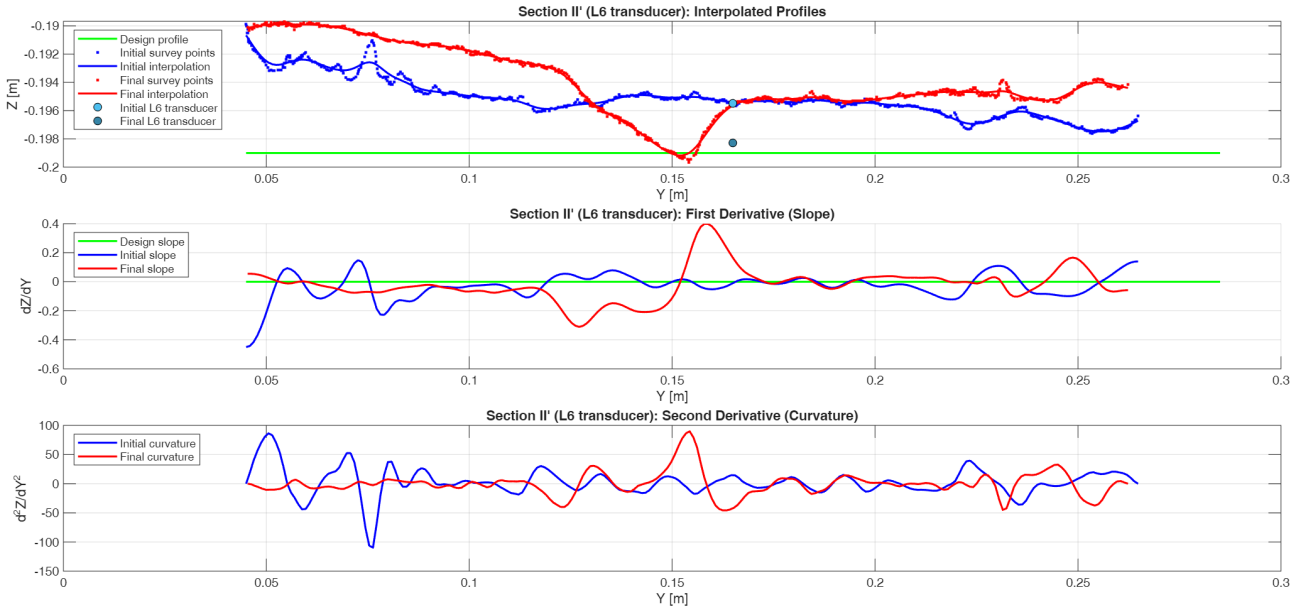

Figure 44: Vertical section II’.

Cross section	Y position [m]	Z design [m]	Z initial [m]	Z final [m]
BB’	0.105	-0.199	-0.195	-0.192
AA’	0.165	-0.199	-0.195	-0.196
CC’	0.225	-0.199	-0.197	-0.195

Table 20: II’ vertical section z values.

Vertical Section JJ’. See figure 45 and listing A.11 in appendix A.

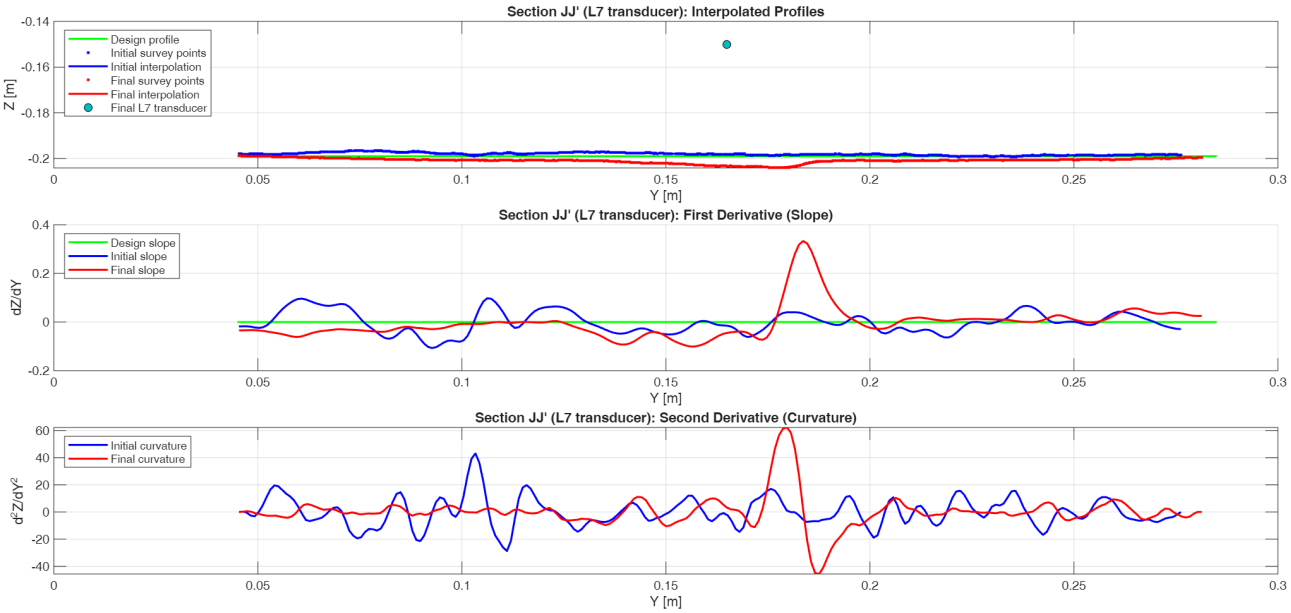

Figure 45: Vertical section JJ'.

Cross section	Y position [m]	Z design [m]	Z initial [m]	Z final [m]
BB'	0.105	-0.199	-0.198	-0.201
AA'	0.165	-0.199	-0.198	-0.203
CC'	0.225	-0.199	-0.199	-0.201

Table 21: JJ' vertical section z values.

The next step was to perform a top-down orthographic representation of the previously obtained numerical values both through a 2D representation with false-color mapping and through a 3D representation of the obtained functions. See figures 46, 47, 48, 49, 50, 51, 52, 53, 54, 55 and listing A.13 in appendix A.

The code aims to rigorously compare two three-dimensional surveys, defined respectively as *Initial* and *Final*, obtained from point clouds each containing coordinates (x, y, z). Once the source files are specified, the user selects the direction along which topographic change will be analyzed. This direction can be expressed either as an angle (in degrees, measured from the x-axis and rotated counterclockwise toward y^+) or as an explicit directional vector. The program normalizes this choice by constructing a unit vector $\mathbf{u} = [u_x, u_y]$, which then becomes the reference for all subsequent derivative computations.

A common regular grid is then created in the xy plane. The bounds of this grid are derived from the intersection of the bounding boxes of the Initial and Final point clouds. A user-defined margin (padding) is applied. On this uniform grid, each dataset is interpolated using a chosen method (for instance, "natural"), resulting in two rasterized surfaces $Z_{Initial}$ and Z_{Final}. Regions where either surface is undefined are masked out with a logical mask.

For both surfaces, first derivatives with respect to x and y, as well as second and mixed derivatives, are computed using finite differences on the regular grid. The mixed derivative is symmetrized to ensure the numerical symmetry of the Hessian matrix, a mathematically necessary property for surface modelling. From these gradients, the *first-order directional derivative* is obtained as the scalar product between the gradient and **u**, while the *second-order directional derivative* corresponds to the curvature of the surface along **u**, calculated by applying the quadratic form of the Hessian along **u**.

The same procedure is repeated for the Final surface, yielding four directional fields in total: slope and curvature for both Initial and Final states. By subtracting the two, the code derives maps of differences: elevation (ΔZ), directional slope ($\Delta D1$), and directional curvature ($\Delta D2$). These are fundamental measures for assessing how the terrain surface has changed between the two time steps and are widely used in geomorphometric studies to characterize uplift, erosion, or deposition processes.

Visualization follows a two-dimensional scheme organized in a 3×3 tiled layout. The first row displays the Initial surface, the Final surface, and their difference in elevation. The second row shows the directional slope fields for Initial, Final, and their difference, while the third row presents the corresponding directional curvature fields. Each panel is equipped with a color scale, oriented Cartesian axes, and symmetric color limits to make comparisons direct and visually meaningful.

The program also provides interactive three-dimensional renderings. The surfaces $Z_{Initial}$, Z_{Final}, ΔZ, and likewise the $D1$ and $D2$ fields are represented as 3D meshes in which both height and color correspond to the variable of interest. Lighting, shading, and an oblique viewing angle enhance perception of surface morphology. For elevation surfaces, equal scaling across x, y, and z axes is enforced, ensuring geometric fidelity; for derivatives and curvature, this constraint is removed to allow clearer visualization of numerical ranges.

Finally, the results are exported to Microsoft Excel. Each gridded matrix (Initial elevation, Final elevation, directional derivatives and curvatures) is converted into a long-format table with one row per grid point and three columns (x, y, value). These tables are filtered through the validity mask and written to separate sheets. This enables further analyses in external software environments, the creation of custom charts, or the integration into other computational models.

In summary, the script implements a methodologically coherent and scientifically grounded workflow: starting from two point-cloud based surfaces, it interpolates them onto a shared domain, computes directional derivatives up to second order, derives change maps between the two temporal states, visualizes results in both 2D and 3D, and exports numerical values for subsequent processing.

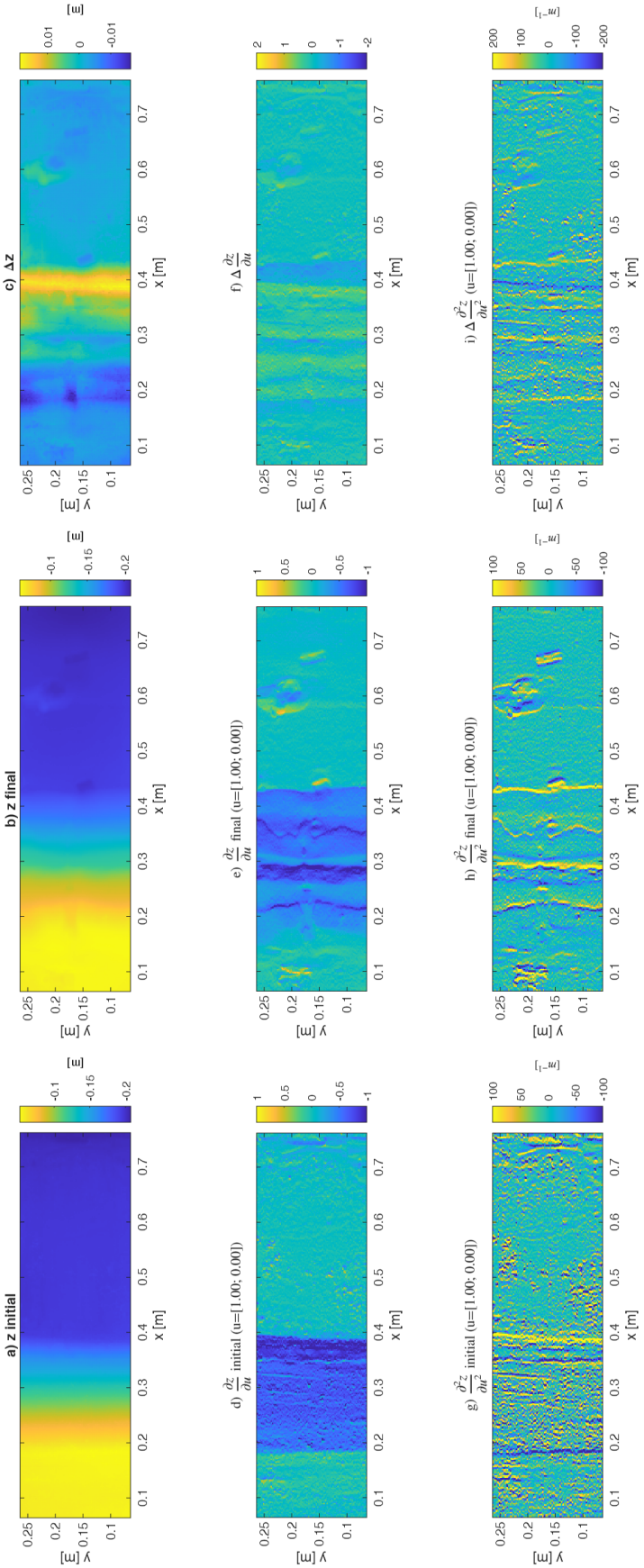

Figure 46: 2D representaiton of z, first and second derivatives.

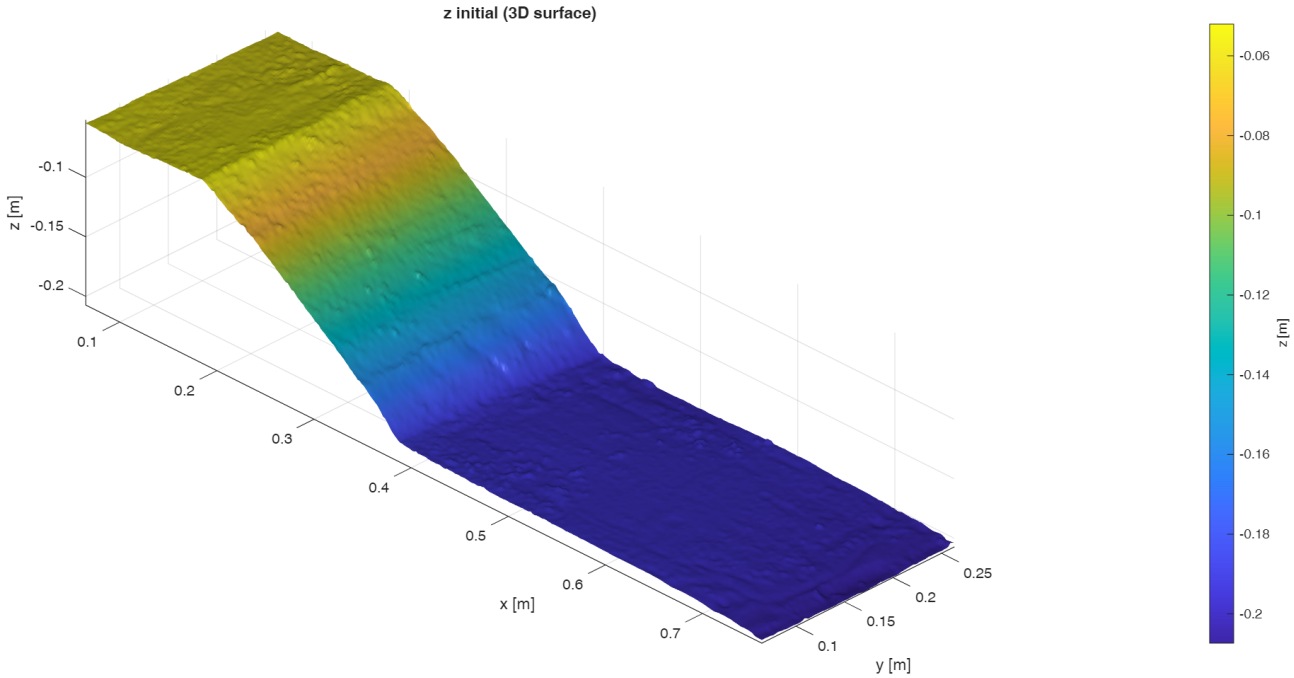

Figure 47: z initial 3D view.

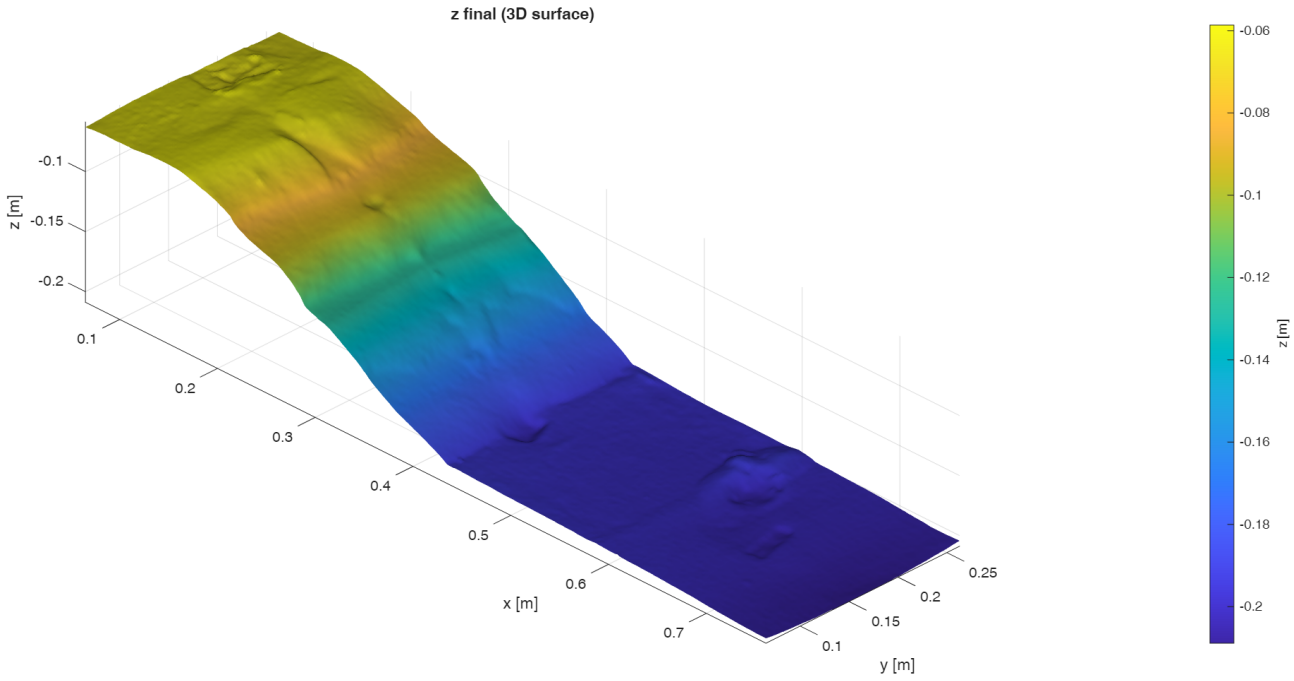

Figure 48: z final 3D view.

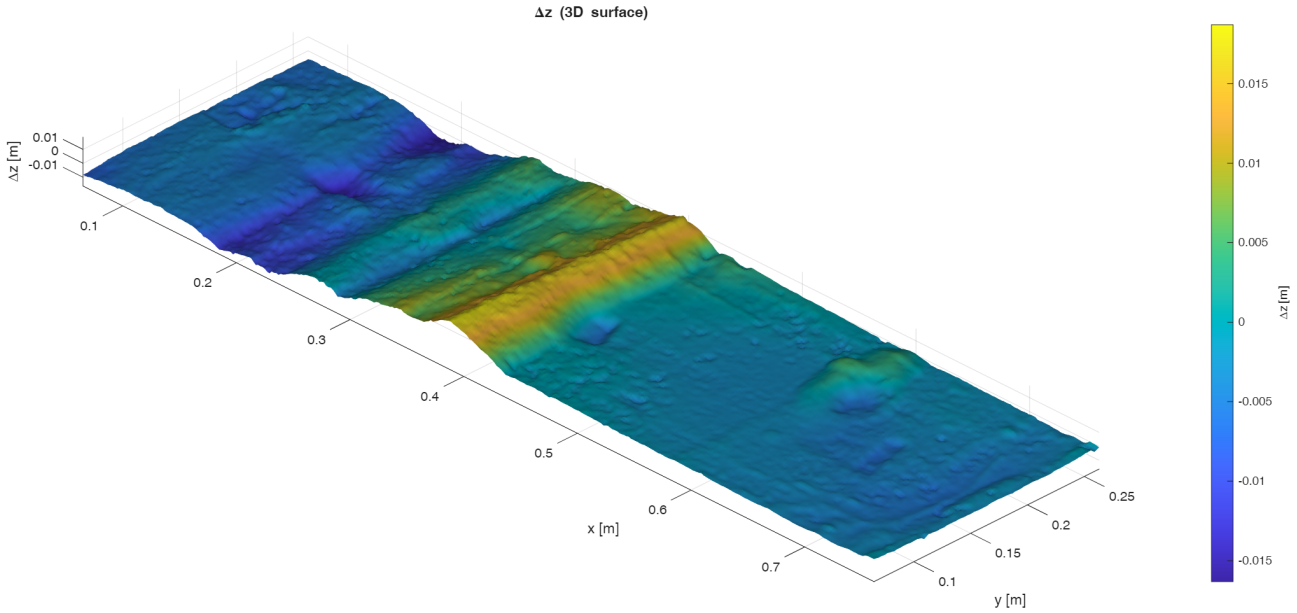

Figure 49: Difference on z 3D view.

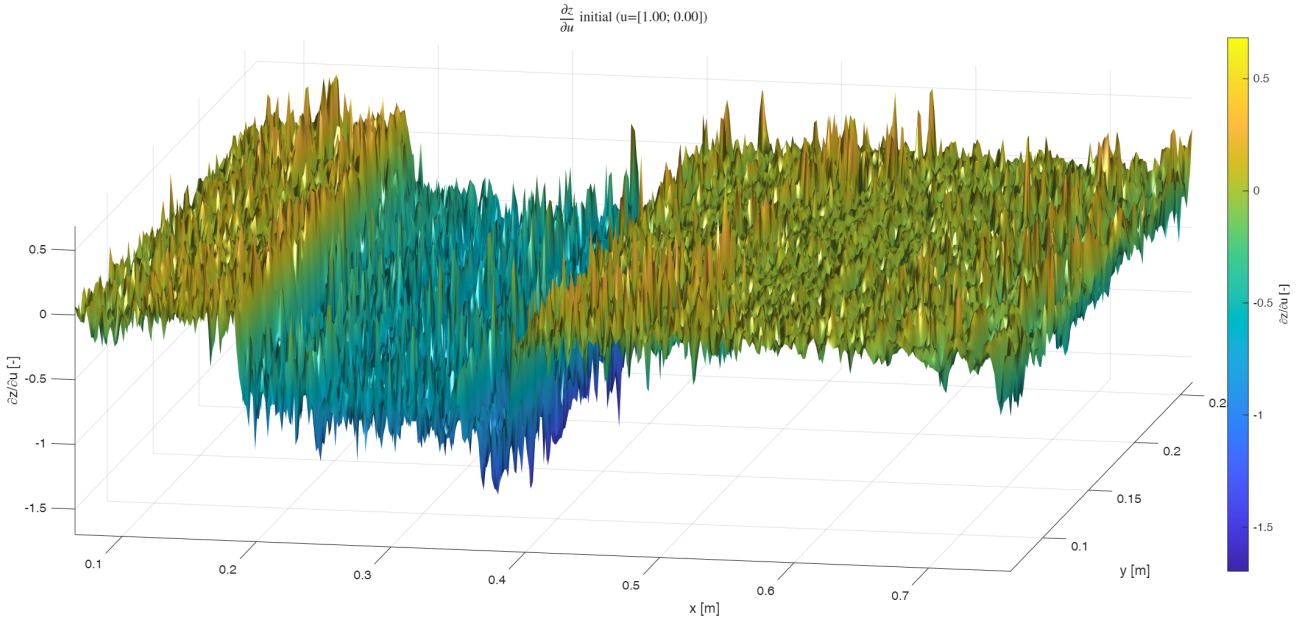

Figure 50: First derivative wrt u, initial.

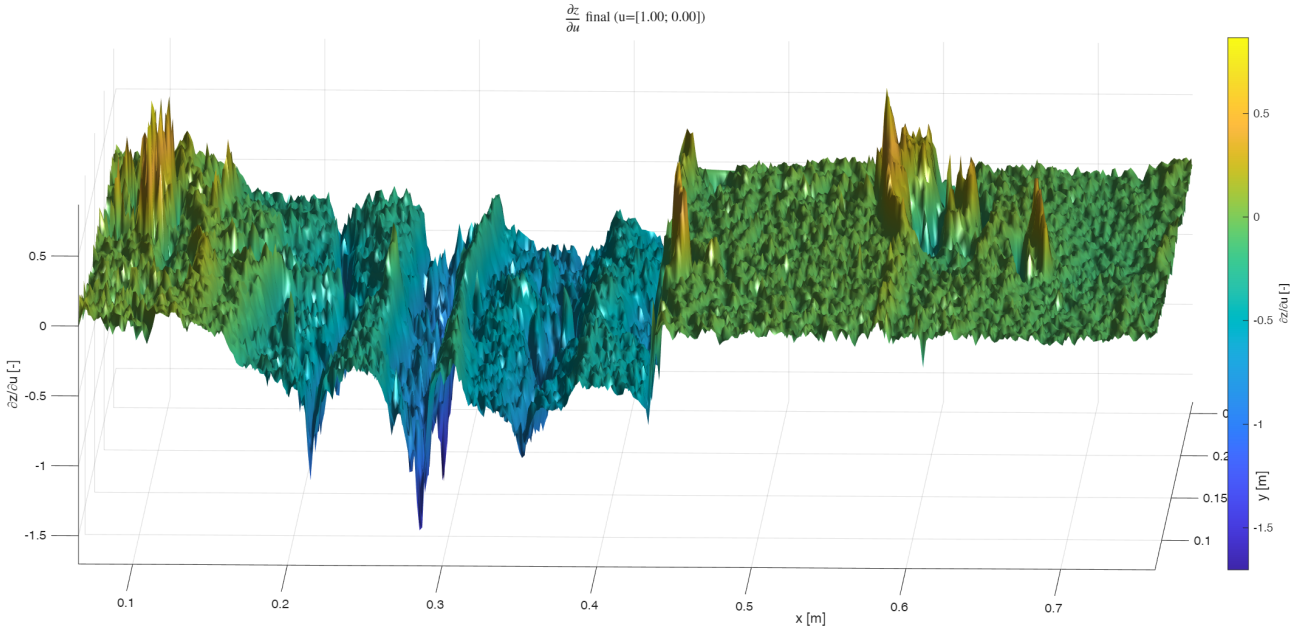

Figure 51: First derivative wrt u, final.

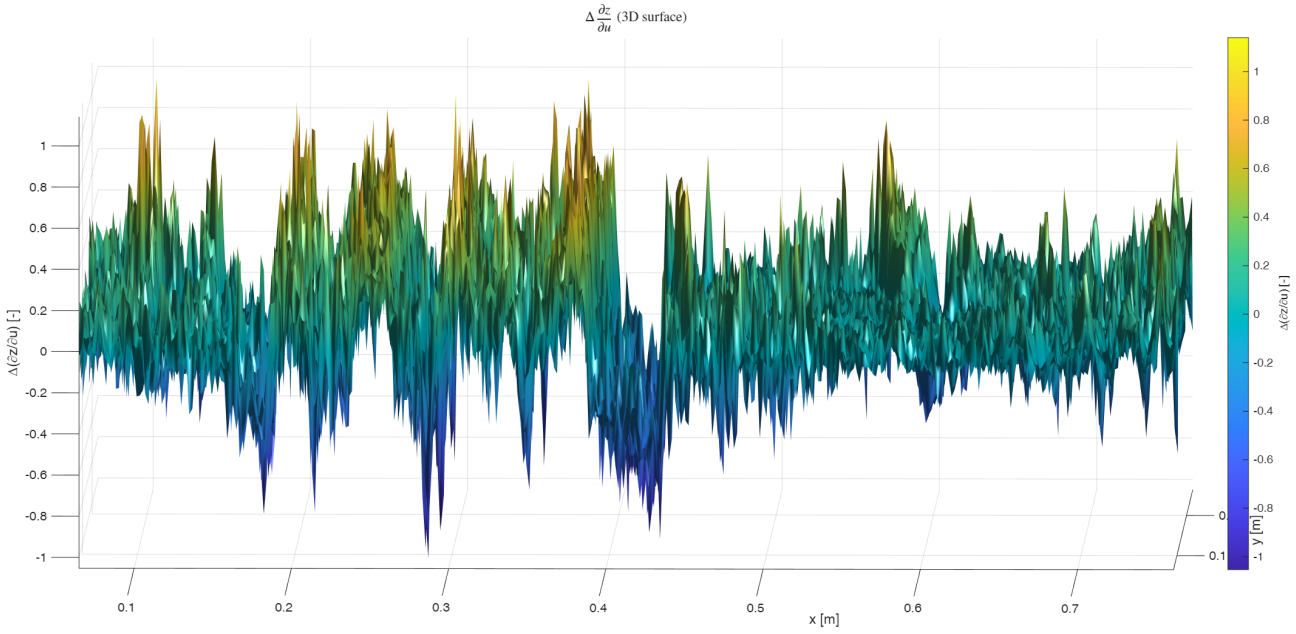

Figure 52: First derivative wrt u, difference.

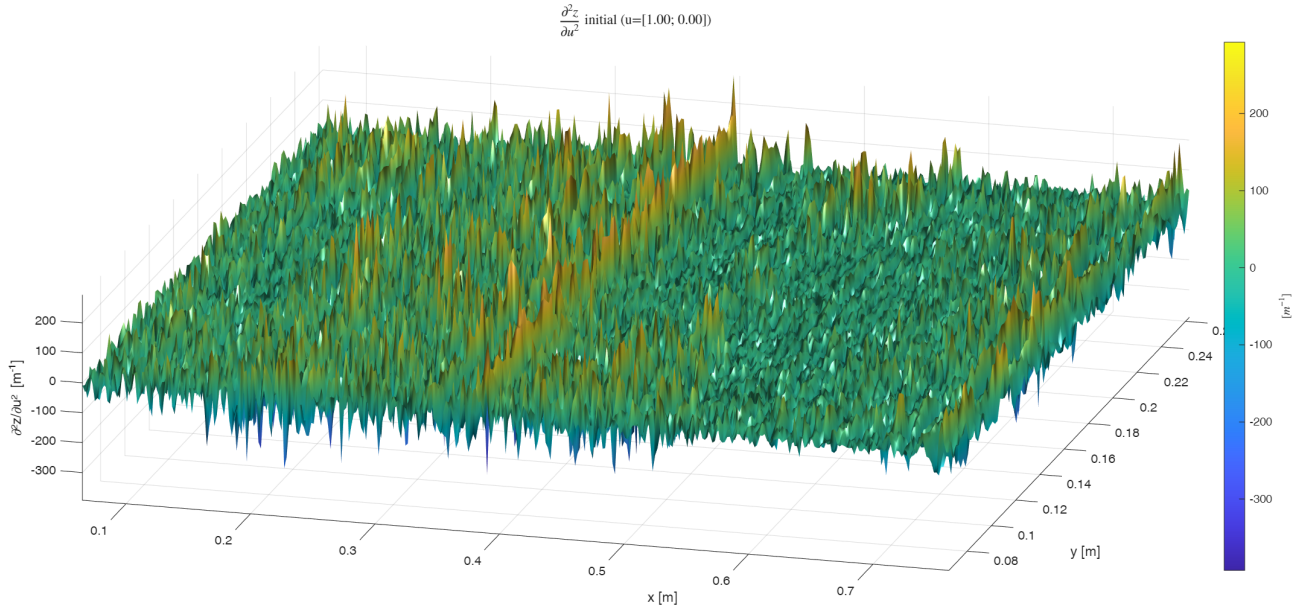

Figure 53: Second derivative wrt u, initial.

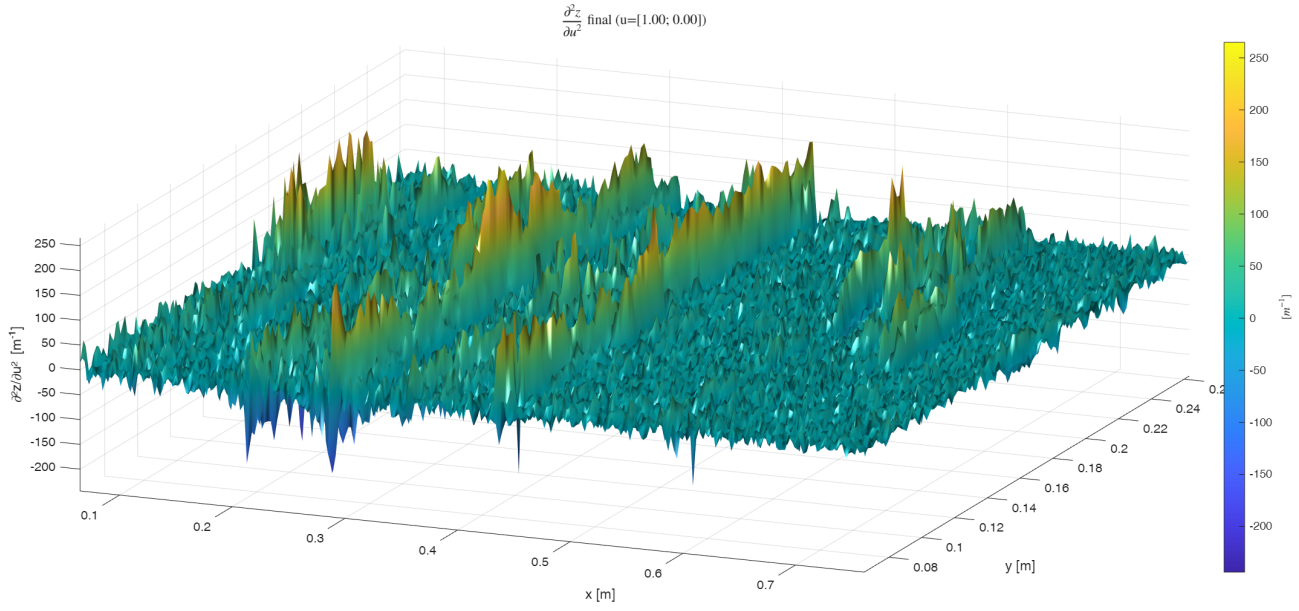

Figure 54: Second derivative wrt u, final.

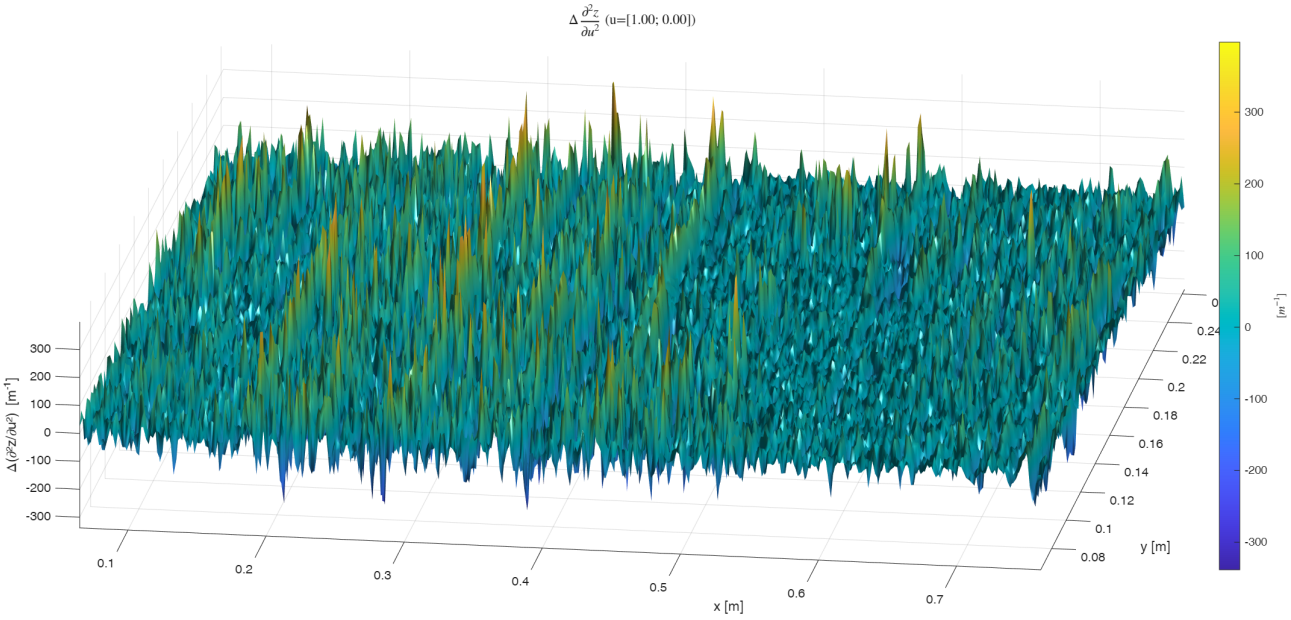

Figure 55: Second derivative wrt u, difference.

4.4 Arch bridge over the Carso River

4.4.1 Bridge arch survey: application of the algorithm

This section introduces a second application of the MATLAB-based workflow to the bridge case study previously described in Section 2.3 (p. 37). Unlike the geotechnical example discussed in Section 2.5, where two point clouds acquired at different moments were compared to detect temporal deformation, the present analysis focuses on the dimensional evaluation of structural elements within the same structure. In particular, the comparison is carried out between the point clouds representing the intrados and extrados surfaces of the reinforced-concrete arches supporting the bridge.

Following a segmentation phase performed in CloudCompare using both manual and semi-automatic procedures, the relevant portions of the point clouds were extracted while preserving the original reference system. These datasets were then analysed through an adapted version of the MATLAB script previously introduced. In this context, the objective is not the detection of microscopic surface variations—as in the geotechnical experiment—but rather the identification of construction irregularities, geometric deviations, and dimensional inconsistencies that may provide meaningful information for assessing structural behaviour and supporting Structural Health Monitoring (SHM) analyses.

The figures 57 and 58 illustrates the output obtained for the vertical longitudinal sections of the bridge arches. This example has been intentionally selected to emphasise a common condition in real-world surveys: the available data rarely provide complete coverage of the object under investigation. As a result, the interpretation of the results requires a critical assessment of data reliability. While the

algorithm supports the analytical processing of the point clouds, it does not replace the operator's judgement, which remains essential for distinguishing between trustworthy portions of the dataset and areas affected by incomplete or uncertain measurements.

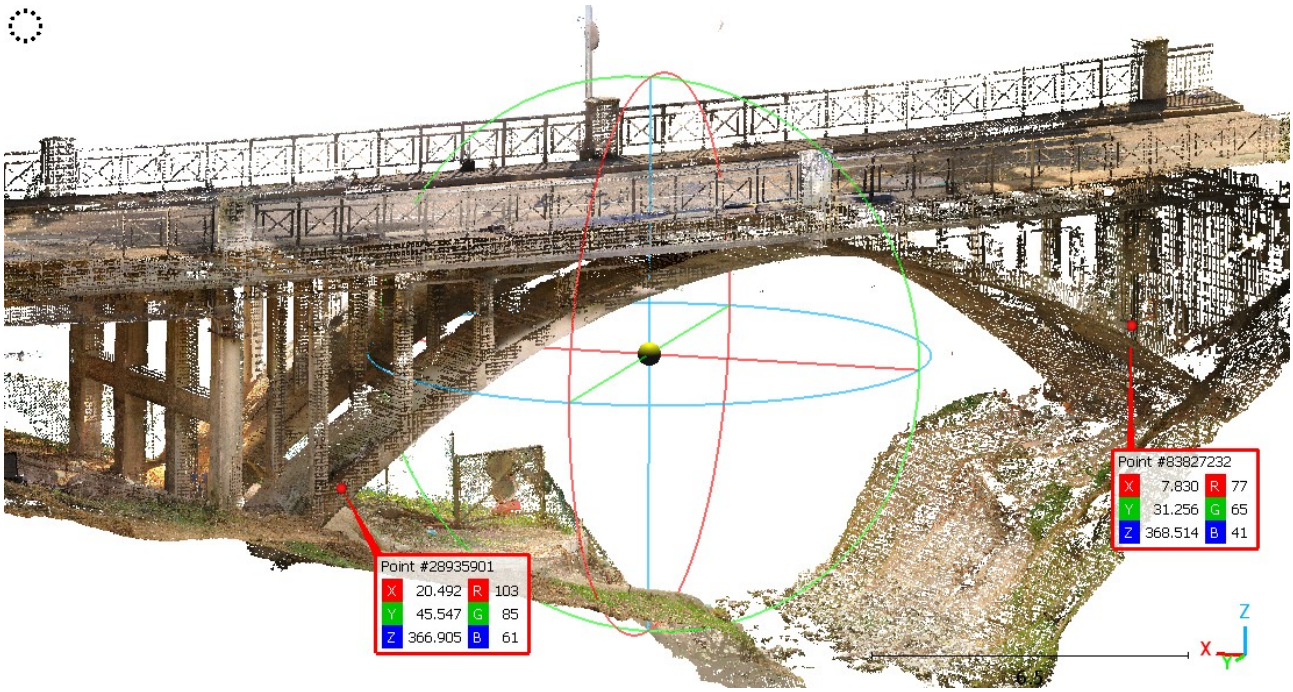

Figure 56: Finding the coordinates of the reference points for determining the rotation in a structure point cloud, in CloudCompare.

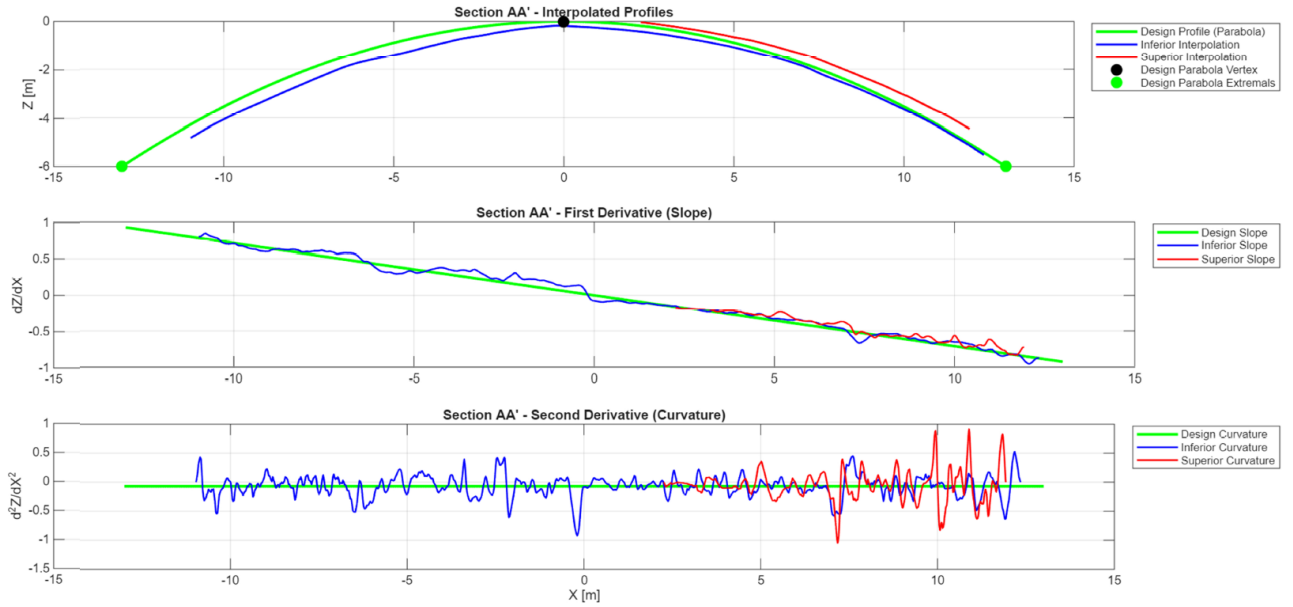

Figure 57: Vertical section on upstream arch.

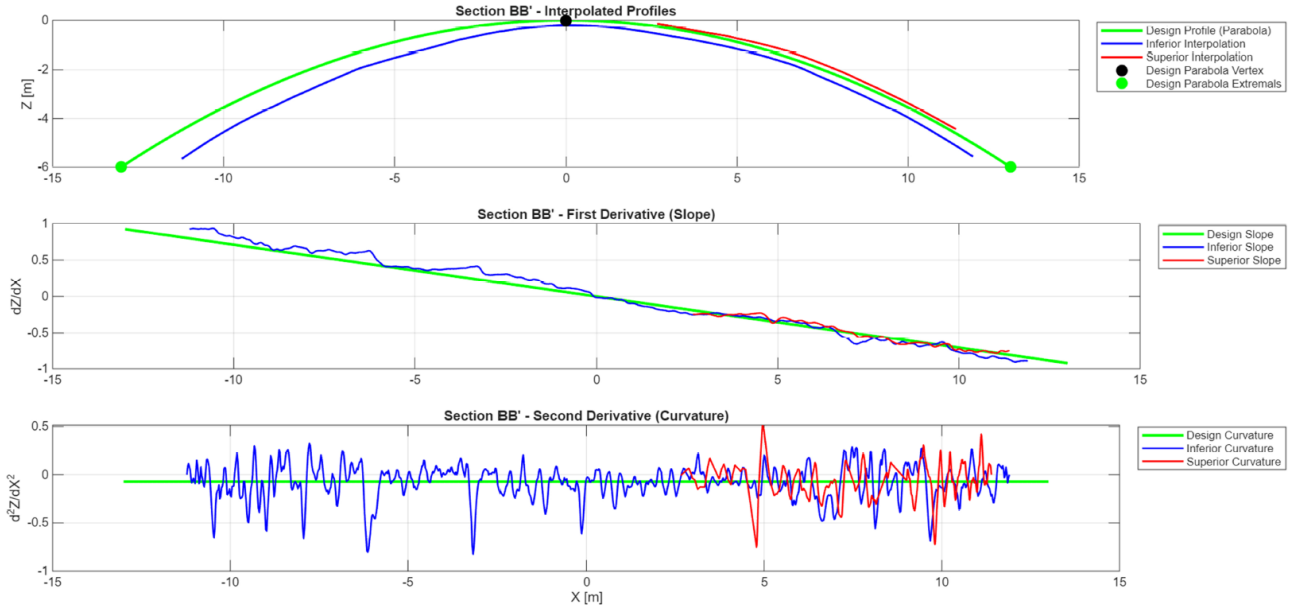

Figure 58: Vertical section on downstream arch.

Numerical analysis of the bridge arch: comparison between design geometry and as-built configuration The numerical analysis was conducted in order to evaluate the geometric discrepancies between the real structure—surveyed through terrestrial laser scanning—and the original theoretical design developed by the engineer Cesare Pesenti. The workflow was structured into several sequential stages, combining point-cloud processing, differential analysis, and geometric interpretation. While the previous section focused on the evaluation of the vertical thickness of the arch through a frontal section, the present analysis shifts the perspective to a planimetric investigation of the *variable width* of the structural arch.

Data processing. The first stage consisted of a systematic preprocessing of the point clouds. A subsampling procedure with a constant step of approximately 1 cm was applied to the original datasets in order to ensure numerical stability and to reduce redundancy while preserving the geometric fidelity of the surveyed surfaces. Subsequently, the relevant portions of the structure were extracted through a segmentation process performed in *CloudCompare*. Both manual and semi-automatic methods were employed to isolate the specific regions of interest, in particular the lateral surfaces of the structural arches. Unlike the previous case study—where the comparison involved the intrados and extrados surfaces—the present investigation focuses on the two lateral faces (flanks) of the arch. These point clouds were extracted while maintaining the original reference system, ensuring the metric consistency required for subsequent comparative analysis.

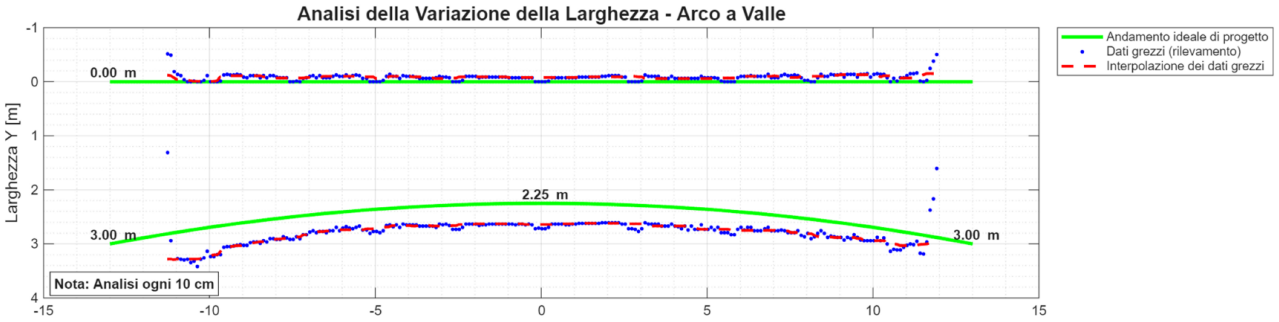

Figure 59: Analysis of the variable width of the downstream arch.

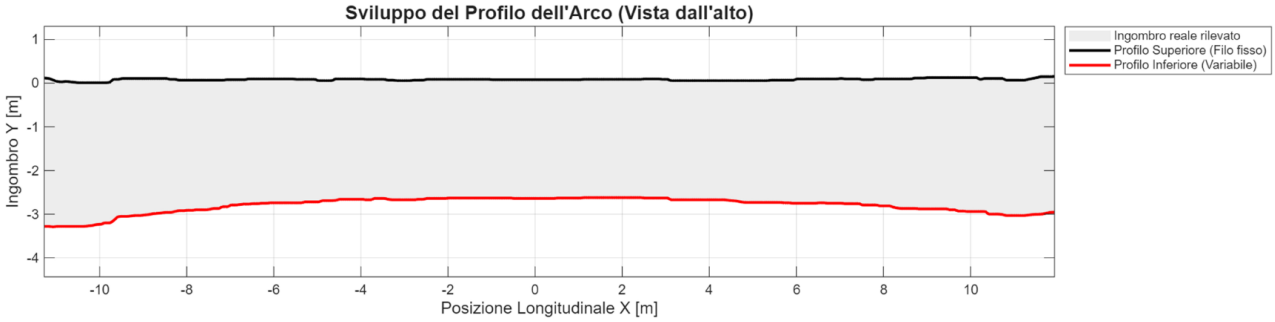

Figure 60: Planimetric development of the downstream arch profile.

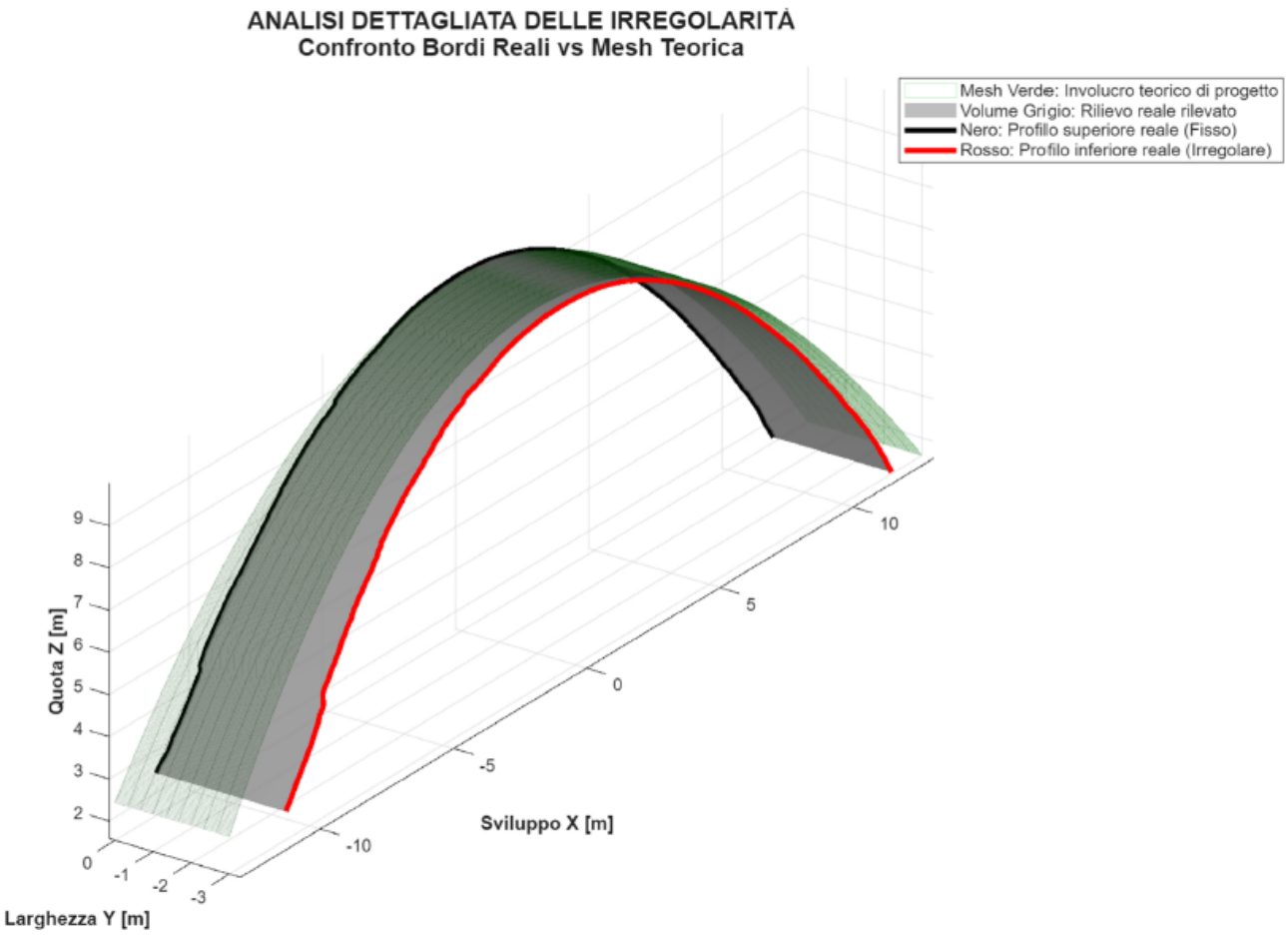

Figure 61: Three-dimensional analysis of the downstream arch intrados.

Analysis of the arch width variation. The adopted methodology allows the comparison between two different point-cloud datasets representing the lateral surfaces of the arches. These surfaces correspond to the flanks of the reinforced-concrete structure and provide a direct measurement of the variable width of the arch when observed in plan view. This characteristic was already present in the original structural design: according to the project developed by Cesare Pesenti, the downstream arch was conceived with a progressively varying width along its span. The design drawings specify the expected dimensional variation, which reflects both structural and constructional considerations.

The MATLAB-based algorithm previously introduced was adapted to this new analytical context. Instead of highlighting the microscopic surface irregularities observed in the geotechnical case study, the objective here is to evaluate the dimensional consistency of the structural geometry. In particular, the algorithm compares the extracted point clouds of the arch flanks and reconstructs the corresponding geometric profiles, allowing the calculation of local deviations from the theoretical configuration. In the graphical outputs, the segmented point clouds of the arch flanks are represented in green, forming the basis for the subsequent comparison between the ideal design geometry and the measured as-built configuration.

Comparison between theoretical and surveyed geometries. Figure 59 illustrates the comparison between the ideal geometry derived from the design documentation and the dimensions measured from the surveyed point clouds. The theoretical profile, reconstructed from the original project parameters, represents the expected structural configuration, while the point-cloud-derived geometry corresponds to the actual built condition. This comparison highlights the dimensional differences between the two configurations, allowing the identification of systematic deviations and local irregularities.

Figures 59 and 60 present the graphical outputs related to this planimetric analysis. The diagrams illustrate the variation of the arch width along its longitudinal development, revealing how the measured geometry departs from the theoretical values specified in the original design. These representations provide a clear quantitative framework for evaluating construction tolerances, geometric inconsistencies, or possible structural adaptations that occurred during the construction phase.

Three-dimensional interpretation of the results. Figure 61 extends the analysis to a fully three-dimensional representation of the investigated geometry. In this case, the curved structural form of the arch is not analysed solely in plan view but also incorporates its vertical dimension, thereby providing a comprehensive spatial interpretation of the structural configuration. The three-dimensional

visualization integrates the geometric information obtained from the segmented point clouds and reconstructs the spatial relationship between the ideal design model and the surveyed structure.

The resulting representation highlights the differences between the theoretical project geometry and the real geometry of the arch as constructed. These discrepancies may originate from construction tolerances, adaptations made during the building process, or long-term structural behaviour. By analysing the geometry in three dimensions, it becomes possible to identify patterns of deviation that would remain less evident in purely two-dimensional representations.

Interpretative implications. The combined analysis of planimetric width variation and vertical structural geometry provides valuable insights into the behaviour of the bridge arches. The comparison between ideal and real geometries allows researchers to distinguish between deviations related to construction practices and those potentially associated with structural deformation. In this context, the algorithm does not simply function as a computational tool but rather as a methodological framework that supports the interpretation of complex geometric datasets. When combined with critical evaluation by the operator, the approach can contribute to a more accurate understanding of the structural configuration and may provide relevant information for future structural assessment and Structural Health Monitoring (SHM) investigations.

Remarks on the choice of the smoothing parameter. In this case study particular attention was devoted to the calibration of the smoothing parameter used in the spline fitting procedure. The objective was to identify an optimal balance between two competing requirements: on the one hand, preserving the geometric resolution of the surveyed profile, and on the other, limiting the amplification of high-frequency noise generated by micro-variations in the measured geometry.

For this reason, a smoothing parameter slightly lower than that adopted in the previous case study was selected, corresponding to $p = 0.999999$. This value proved effective in reducing small-scale oscillations while still maintaining sufficient sensitivity to detect meaningful geometric variations. The resulting smoothed profiles allow the first and second derivatives to highlight, in a clearer and more interpretable manner, the local discontinuities associated with changes in curvature along the deck profile.

Such variations in curvature may be indicative of several underlying phenomena. These include, among others, localized structural deformations caused by mechanical loading, construction inaccuracies such as irregularities in the positioning of concrete formwork, long-term material degradation processes (e.g., corrosion), or other localized geometric anomalies introduced during construction or service life.

Consequently, the calibrated smoothing parameter plays a crucial role in ensuring that the differential analysis captures relevant structural information while filtering out non-meaningful noise.

4.5 Parametric workflow for automated analysis of arch dam sections from raster profiles

4.5.1 Aim and scope

This section presents a parametric and automated workflow, implemented in MATLAB, on a case study of the Gleno dam in Vilminore di Scalve, for the metrical analysis of arch dam cross-sections starting from raster images. The method is designed to be *reproducible*, *auditable*, and *operator-in-the-loop*: the user selects meaningful profile segments (to exclude vegetation and occlusions), while all metric computations, fits and comparative diagrams are automated. The goal is twofold: (i) interpolate ideal circular arcs for the upstream and downstream intradoses of each arch and (ii) quantify the *radial thickness* as a function of angle, together with diagnostics on how well the real pixels adhere to the ideal geometry.

The case study concerns the western trunk of an arch dam made of ten full-centered arches nominally designed with constant thickness. In practice, local variations in radius and thickness are expected. The proposed workflow quantifies those variations, compares them across arches, and supplies interpretable graphs (per-arch and global) to support construction diagnostics and monitoring.

4.5.2 Data preparation and scale control

The input is a 2D section exported from the 3D point cloud with CloudCompare as a high-resolution TIFF. Since CloudCompare can export with transparent background, an intermediate step in Adobe Photoshop replaces the transparency with a uniform white field. This simple edit has two practical effects: it increases the edge contrast for robust Canny detection and avoids spurious edges where alpha channels are present.

A key design choice is to lock the *metric scale* at export time by specifying the pixel size. In the experiments reported here, images were exported with a pixel size of 10 mm:

$$1 \text{ px} = 10 \text{ mm}.$$

The MATLAB tool requests confirmation of the pixel size (in m/px) on launch. From that point onward, all magnitudes are expressed in SI units. If future campaigns adopt a different raster resolution, only the pixel-to-meter scale needs to be updated.

4.5.3 High-level workflow

The workflow is articulated in seven phases, and the entire code is in A.14

1. **Import & scale confirmation.** The raster section is selected via File Explorer; the user confirms or overrides the pixel size (m/px).

2. **Edge extraction.** A Canny detector with light morphological cleanup produces a thin, coherent edge map of the true pixels.

3. **Interactive ROIs.** For each arch, the operator draws two freehand ROIs: one for the downstream profile and one for the upstream profile. Zoom/pan is available between drawings to frame difficult areas. This manual selection is deliberate: it filters out vegetation, overhangs, and occlusions that cannot be reliably segmented automatically in general.

4. **Robust circle interpolation.** Inside each ROI, the tool fits a circle robustly using bounded RANSAC with Taubin refinement. This returns the center and radius for the downstream and upstream ideal arcs.

5. **Radial thickness.** The thickness $t(\varphi)$ is computed along rays emitted from the downstream center over the sector $\varphi \in [-60°, +60°]$ (with 0° at the crown and angles increasing clockwise). A theoretical reference thickness (here 0.76 m) is plotted for immediate comparison.

6. **Adherence of real pixels to ideal circles.** For every real edge pixel within each ROI, the radial residual $\Delta r(\varphi)$ with respect to the fitted circle is computed and plotted against φ; per-degree medians make trends legible without suppressing outliers.

7. **Global comparisons across n arches.** The tool compiles bar charts of upstream and downstream radii (with tight autoscaling), overlays all thickness profiles, and draws the vectors $\Delta\mathbf{C}_i = \mathbf{C}_{U,i} - \mathbf{C}_{D,i}$ from the origin, visualizing the relative displacement of upstream centers with respect to downstream ones.

4.5.4 Geometric conventions

We adopt the image coordinate frame with x to the right and y downward. Angles are defined at the downstream center with:

- $\varphi = 0°$ pointing vertically upward (crown),

- φ increasing *clockwise*,

- thickness evaluated for $\varphi \in [-60°, +60°]$ to focus on the measurable and structurally relevant sector.

A unit ray at angle φ is

$$\mathbf{u}(\varphi) = \begin{bmatrix} \sin\varphi \\ \\ -\cos\varphi \end{bmatrix}.$$

Let $\mathbf{C}_D, R_D$ and $\mathbf{C}_U, R_U$ be the centers and radii (in meters) of the downstream and upstream fitted circles respectively.

4.5.5 Circle fitting: strategy and justification

Raster sections inevitably include noise, lacunae, and spurious fragments. A two-stage fitting strategy balances robustness and accuracy:

Stage 1: RANSAC with radius constraints. Random minimal samples of three points define candidate circles. A candidate is accepted if its radius R falls within plausible bounds derived from the ROI span (width and height), and if the number of inliers (points within a tolerance τ of the circle) exceeds a threshold. The inlier test uses the radial residual $| \, \|P - \mathbf{C}\| - R \, | < \tau$ in pixels. Radius bounds derived from the ROI make RANSAC efficient and suppress unrealistic solutions (e.g., degenerate very small or extremely large circles).

Stage 2: Taubin algebraic refinement. Given the RANSAC inliers, Taubin's algebraic circle fit provides a stable least-squares estimate that reduces bias with respect to geometric fits, especially under small noise. If Taubin's refined radius leaves the admissible interval, the RANSAC estimate is retained. Fallbacks include Taubin on all ROI points and a second RANSAC pass with expanded bounds, ensuring graceful degradation when the ROI is weak.

Parameterization. Default values proved effective for the present dataset:

- Canny $\sigma = 1.4$ (edge smoothing),
- RANSAC iterations $N = 3000$,
- inlier tolerance $\tau = 2$ px,
- minimum inliers $m = 30$.

These values can be relaxed for noisy, low-contrast rasters (increase σ and τ) or tightened for crisp vector-like images.

4.5.6 Radial thickness computation

Thickness is measured radially from the downstream center. A ray at angle φ is

$$\mathbf{X}(t) = \mathbf{C}_D + t\,\mathbf{u}(\varphi), \qquad t \geq 0.$$

Intersections with the upstream circle solve

$$\|\mathbf{X}(t) - \mathbf{C}_U\|^2 = R_U^2 \quad \Longleftrightarrow \quad t^2 + 2\,\mathbf{u}(\varphi)^\top(\mathbf{C}_D - \mathbf{C}_U)\,t + \|\mathbf{C}_D - \mathbf{C}_U\|^2 - R_U^2 = 0.$$

Among the two solutions $t_1(\varphi), t_2(\varphi)$, the algorithm selects the smallest *positive* intersection lying beyond the downstream radius along the same ray. Denoting the chosen root by $t^\star(\varphi)$, the radial thickness is

$$t(\varphi) = t^\star(\varphi) - R_D.$$

This definition properly handles the frequent case of *non-concentric* circles (upstream and downstream centers are displaced), which precludes measuring thickness with a naive difference of radii.

4.5.7 Pixel-to-Circle adherence: residuals by angle

To evaluate how well the real edge pixels adhere to the fitted ideal circle, for each ROI point P we compute:

$$\Delta r = \|P - \mathbf{C}\| - R, \qquad \varphi = \mathrm{wrap}_{[-180^\circ,180^\circ]}\big(\mathrm{atan2}(P_y - C_y,\ P_x - C_x) + 90^\circ\big)\,,$$

and retain only $\varphi \in [-60^\circ, +60^\circ]$. The residuals are then represented as:

- a light scatter of *all* contributing pixels $(\varphi, \Delta r)$ in meters, and
- a robust median curve computed per degree to reveal systematic deviations without erasing outliers.

This display answers two practical questions: *where* along the arch pixels deviate most from the circle, and *by how much*, with angular localization.

4.5.8 User interaction and ROI strategy

The interface first opens the raster with nearest-neighbour display (no smoothing) to make pixels crisp at high zoom. Before each ROI, the tool explicitly allows the user to pan/zoom, then requests to draw the polygonal freehand boundary. Two ROIs are drawn per arch: downstream first, then upstream. This order mirrors the subsequent computation, in which thickness rays originate from $\mathbf{C}_D$.

Selecting ROIs is not a generic tracing task: the operator should include segments that are metrically representative of the arch intrados and exclude features that are known to be unreliable (e.g., vegetation silhouettes, occlusions, strong shadows). The algorithm is tolerant to small gaps but needs sufficient angular coverage and point density to fit a stable circle.

4.5.9 Multi-arch batch and outputs

After the n pairs of ROIs are collected, the tool:

1. **Per arch:**
    - overlays the fitted circles on the raster with centers marked;
    - draws the angular reference at $\varphi = \{-60°, 0°, +60°\}$ (clockwise convention);
    - plots the thickness $t(\varphi)$ over $[-60°, +60°]$ with the theoretical reference (here 0.76 m);
    - produces residual diagrams $(\varphi, \Delta r)$ for downstream and upstream pixels.
2. **Global comparisons:**
    - bar charts of all downstream and all upstream radii with tight vertical limits (to enhance small differences), allowing a direct comparison between the curvature of each arch (Figure 73);
    - overlay of all thickness profiles $t(\varphi)$ for the ten analysed arches, enabling a direct comparison between the measured thickness distributions and the theoretical reference thickness (Figure 72);
    - vector diagram of $\Delta\mathbf{C}_i = \mathbf{C}_{U,i} - \mathbf{C}_{D,i}$, each arrow drawn from the origin in meters and annotated by arch index (Figure 74).

These outputs form a compact dashboard that is both visually interpretable and quantitatively auditable.

4.5.10 Parameter choices and sensitivity

Scale. The pixel size controls every metric quantity. With 1 px = 0.01 m, the minimum resolvable step is one centimetre. Sub-pixel edge localization is not attempted on purpose to keep the pipeline auditable; however, a future refinement could estimate sub-pixel edges (e.g., by gradient fitting) to reduce quantization.

Edge detection. Canny $\sigma = 1.4$ is a balanced default; lower values may under-smooth noisy scans (fragmented edges), higher values may over-smooth genuine details. The subsequent morphological thinning and small-object removal are intentionally light so that the residual analysis remains tied to *actual* pixel positions.

RANSAC tolerance and iterations. The inlier tolerance $\tau = 2$ px corresponds to 2 cm at the present scale; this is aligned with the typical raster aliasing and slight drawing imperfection on the section. Increasing τ makes RANSAC more permissive; decreasing it makes the fit crisper but more brittle. Three thousand iterations are sufficient for robust convergence with moderate outlier rates.

Angular sector. The interval $[-60°, +60°]$ avoids low-visibility areas near the springings and focuses on the most stable part of the arch section for both fit and thickness. If your sections expose wider coverage, the bounds can be relaxed.

4.5.11 Uncertainty and quality control

Three main contributors affect uncertainty:

1. **Scale error** δs (m/px): a uniform bias on all lengths; mitigated by careful export and verification.
2. **Edge localization** δp (px): due to raster aliasing and imperfect binarisation; manifests as random noise in residuals.
3. **ROI selection** δ_{ROI}: operator bias on which portions are considered reliable; mitigated by documenting ROI masks and repeating the fit on alternative selections for sensitivity analysis.

Two routine checks are recommended and supported by the plots:

- *Angular homogeneity of residuals:* systematic drifts in $\Delta r(\varphi)$ (e.g., consistently positive near $+60°$) suggest geometric bias or selection artefacts.
- *Consistency of center shifts:* clustering of $\Delta \mathbf{C}_i$ around a preferred direction may indicate construction logic or deformation patterns.

4.5.12 Reading the results

Empirically, the crowns ($\varphi \approx 0°$) tend to exhibit the smallest residuals, while deviations can increase toward the sector bounds. Across the ten arches analysed in the western trunk:

- each arch is well approximated by a circular arc, yet the radii differ, indicating construction variability;

- the thickness $t(\varphi)$ departs from the constant design line (here 0.76 m) by centimetric amounts, arch-dependent and angle-dependent;

- center-shift vectors $\Delta\mathbf{C}_i$ display a coherent but not identical pattern, which is compatible with non-concentric upstream/downstream intradoses and with practical tolerances at erection and finishing stages.

These findings support a nuanced conclusion: the arches are effectively semicircular, but not identical; radial thickness varies in a controlled band; differences are consistent with a plausible construction envelope (Figures 62,63,64,65,66,67,68,69,70,71,72,73,74). These trends can be observed in the graphical summaries of the analysis, including the comparison of arch radii (Figure 73), the displacement of upstream centers relative to downstream centers (Figure 74), and the overlay of thickness profiles for all arches (Figure 72).

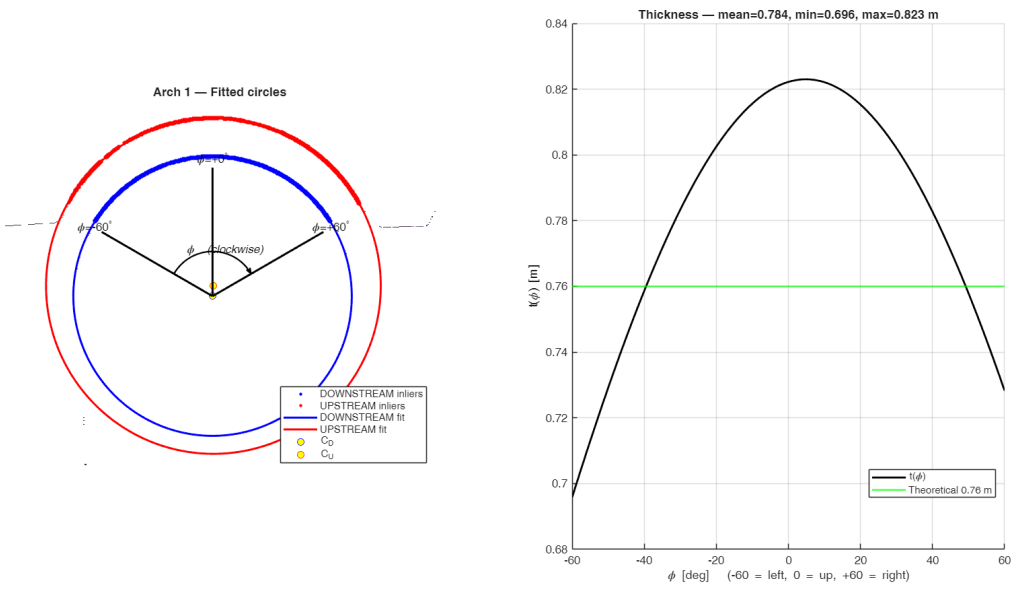

Figure 62: Metric analysis on the horizontal section of arch number 1.

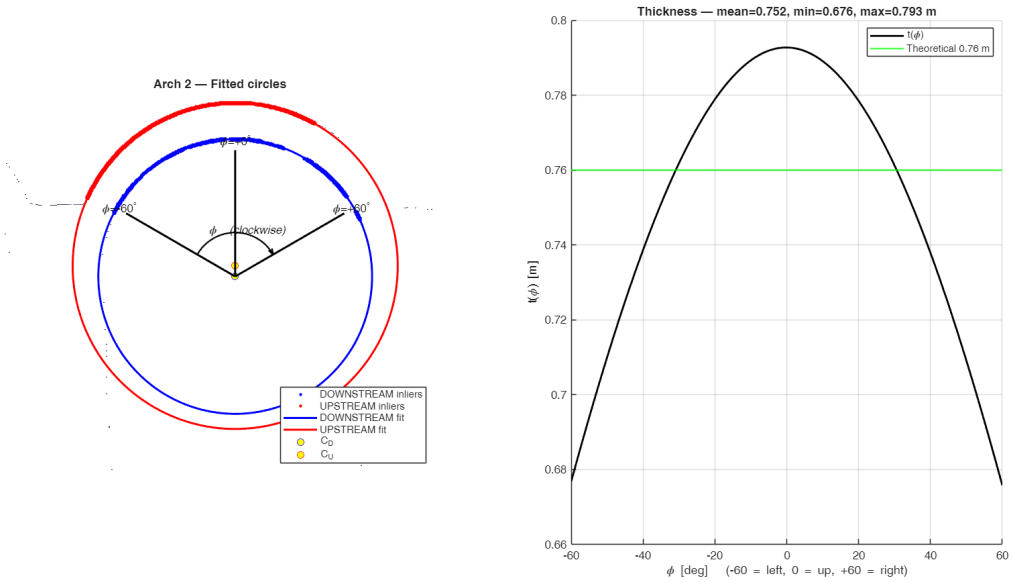

Figure 63: Metric analysis on the horizontal section of arch number 2.

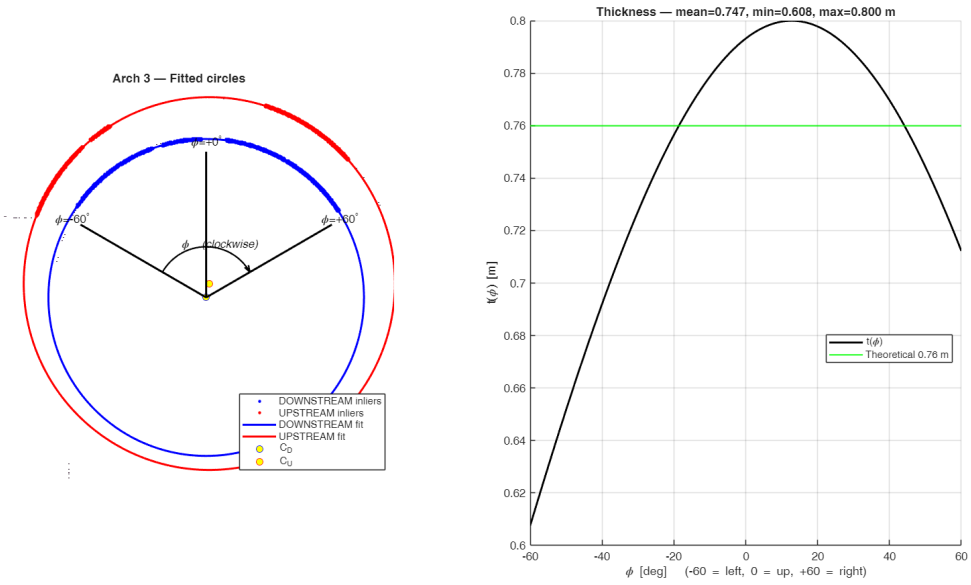

Figure 64: Metric analysis on the horizontal section of arch number 3.

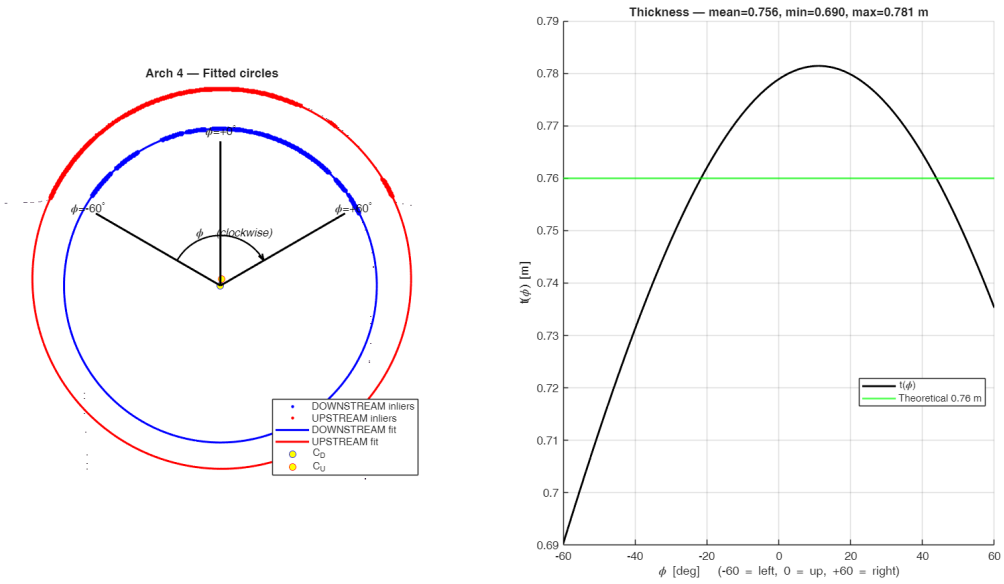

Figure 65: Metric analysis on the horizontal section of arch number 4.

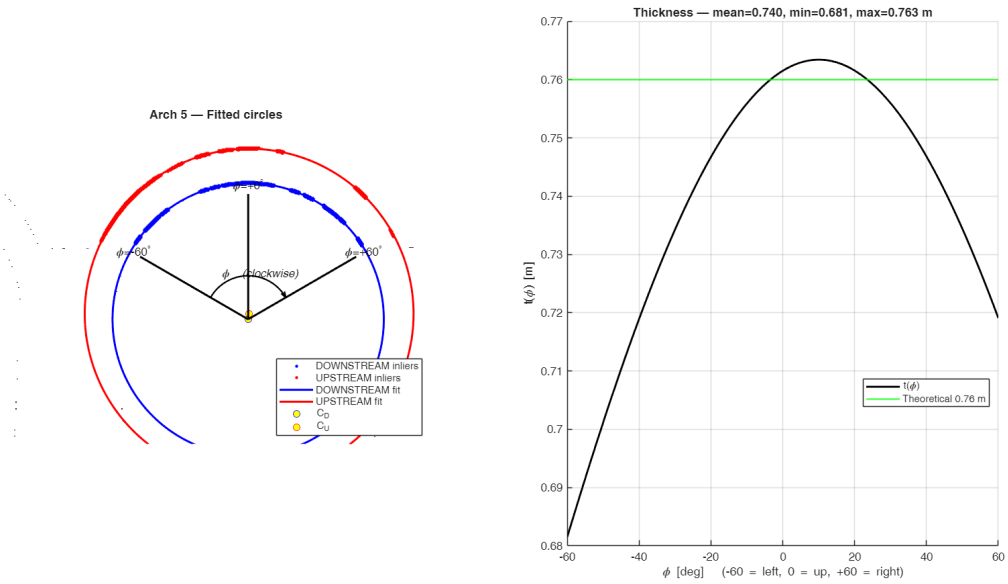

Figure 66: Metric analysis on the horizontal section of arch number 5.

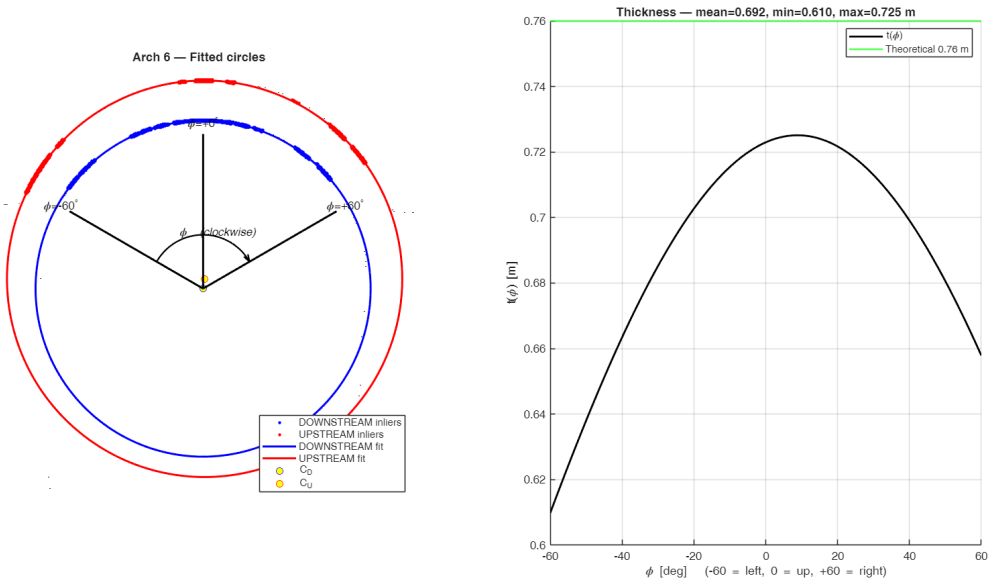

Figure 67: Metric analysis on the horizontal section of arch number 6.

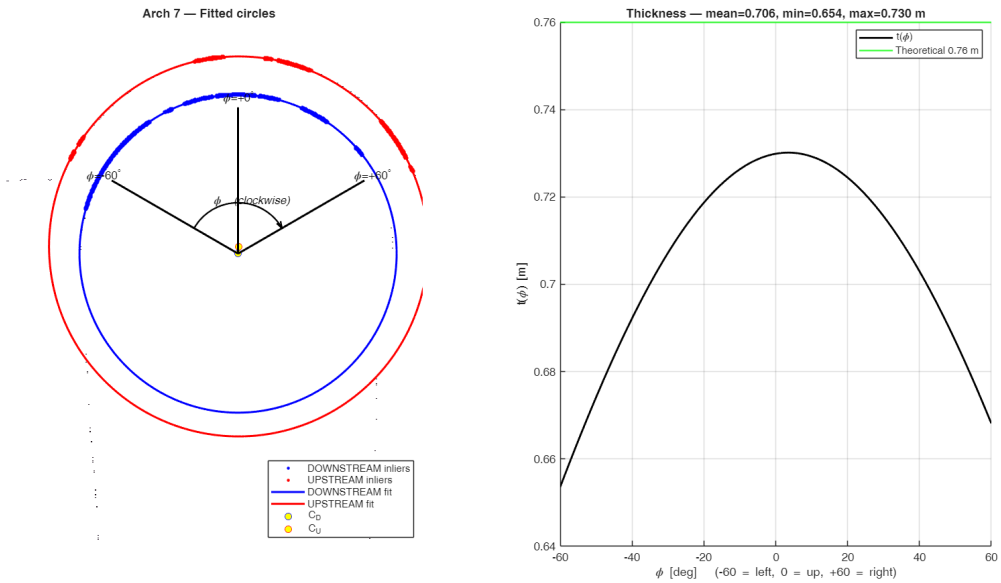

Figure 68: Metric analysis on the horizontal section of arch number 7.

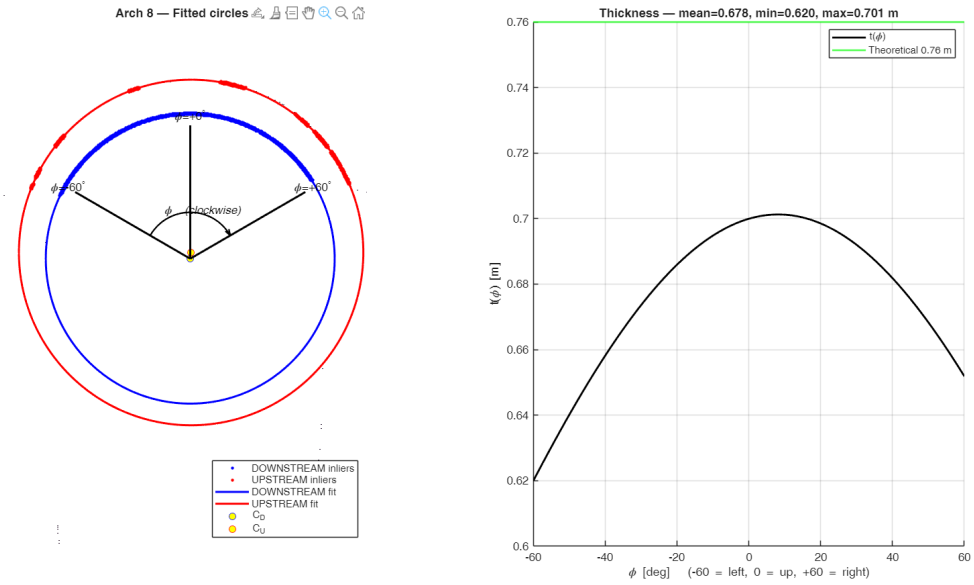

Figure 69: Metric analysis on the horizontal section of arch number 8.

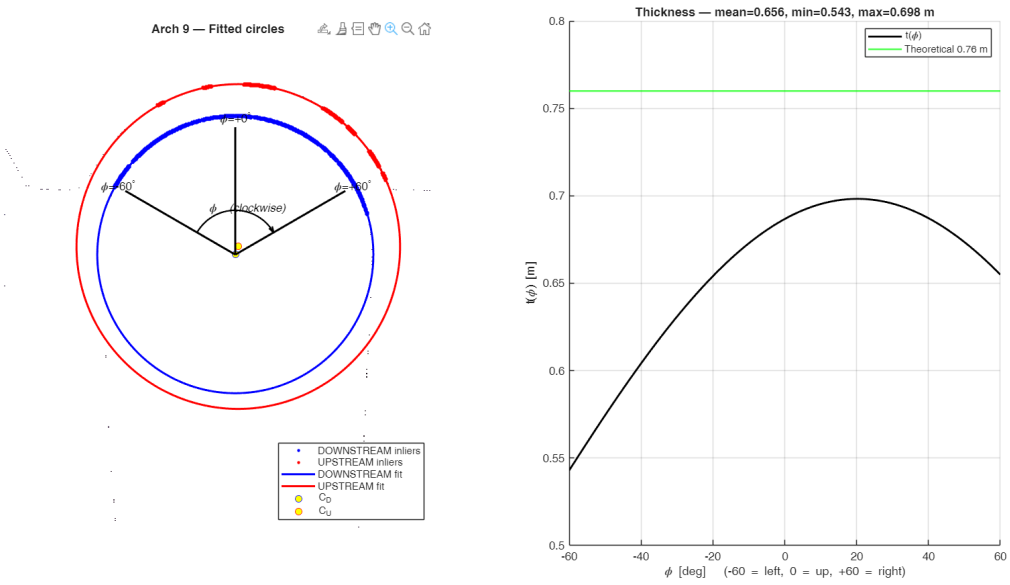

Figure 70: Metric analysis on the horizontal section of arch number 9.

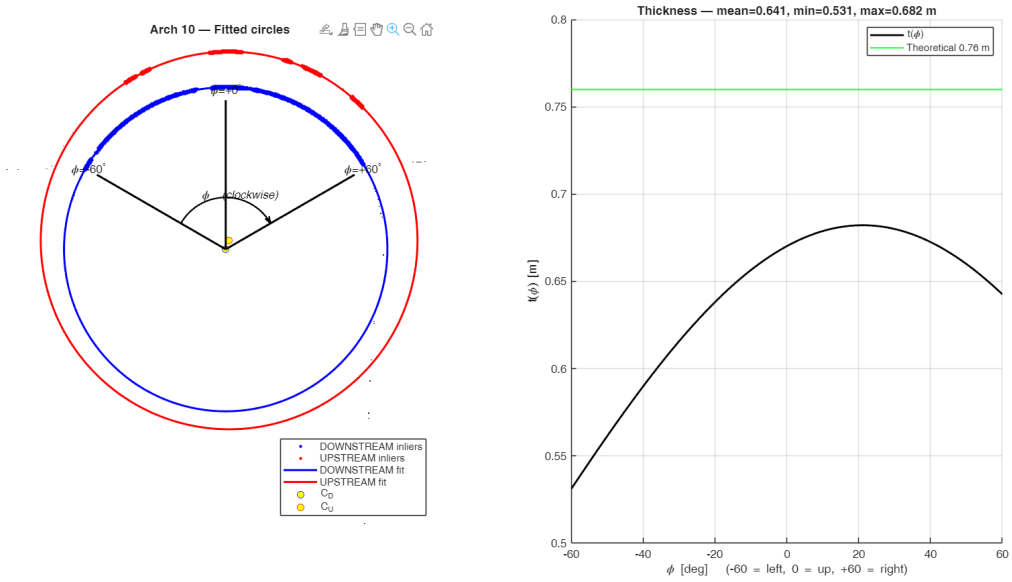

Figure 71: Metric analysis on the horizontal section of arch number 10.

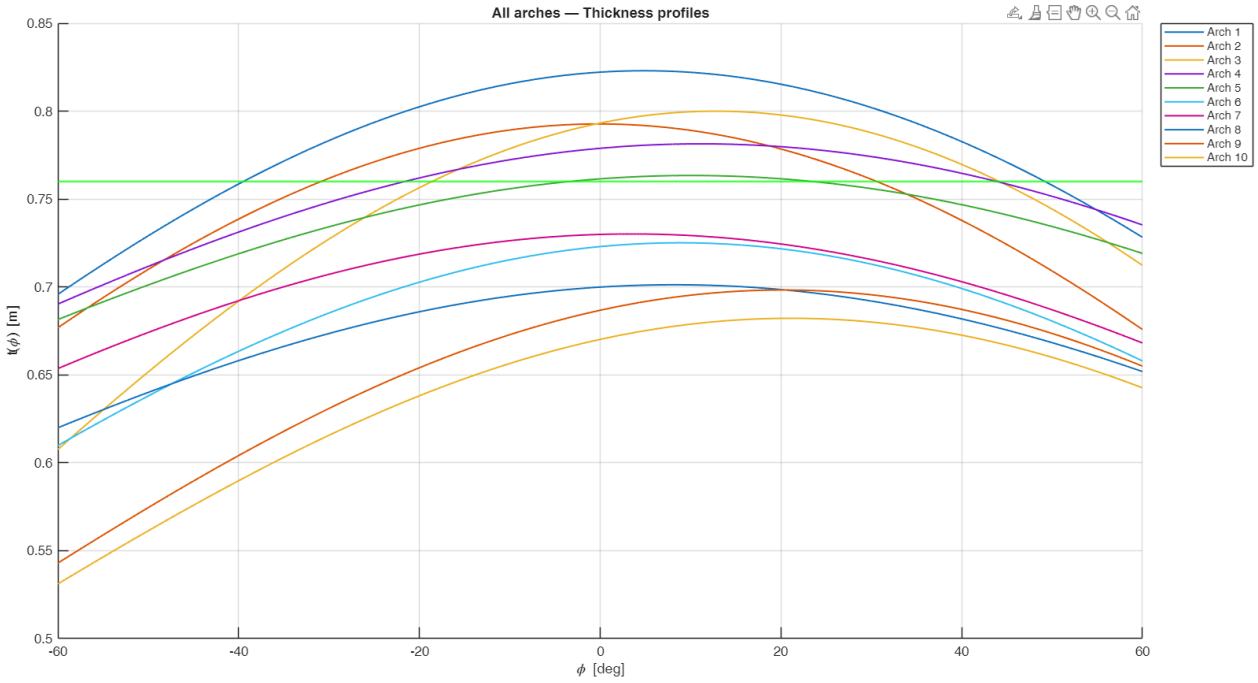

Figure 72: Metric comparison of different thicknesses of arches.

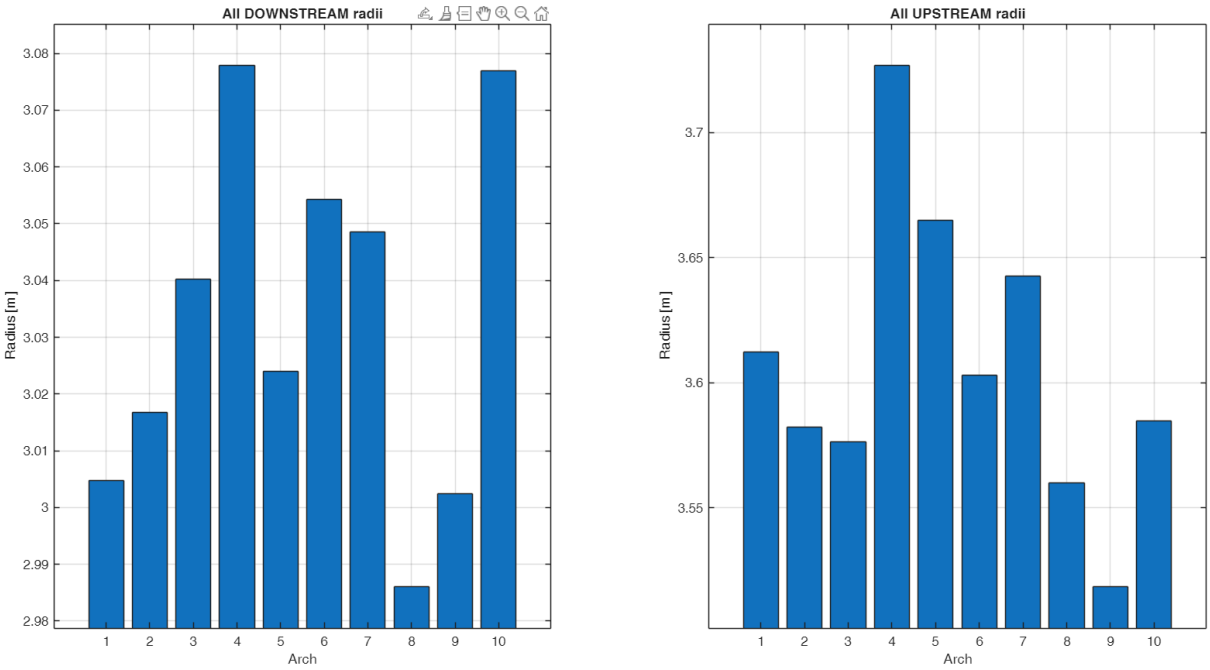

Figure 73: Radius comparison between different arches.

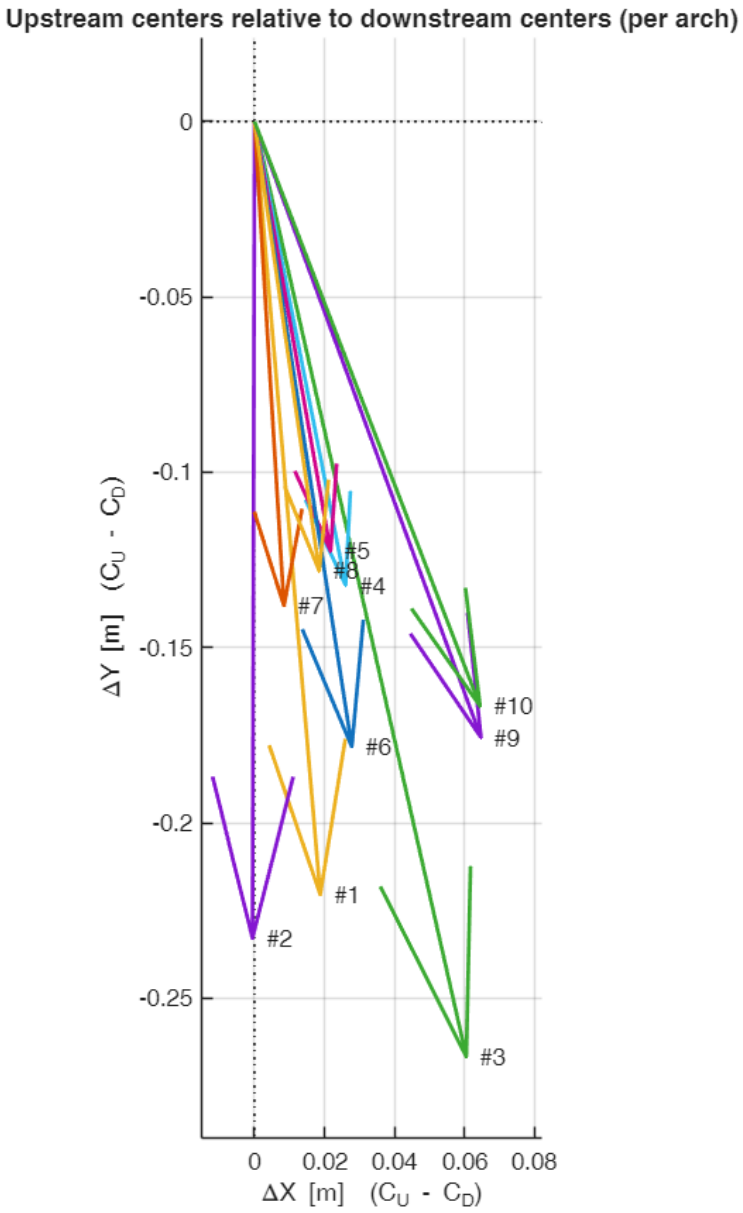

Figure 74: Center displacements of upstream and downstream Arches.

4.5.13 Limitations and extensions

The present method assumes that each intrados segment is *locally* well modelled by a circle. When larger-scale ovalisation or multi-centre geometries occur, a circular model may be insufficient. Two extensions are natural:

1. **Segment-wise modelling:** fit arcs independently in sub-intervals of φ and compare AIC/BIC against a single-circle model.

2. **Spline/ellipse models:** replace circles with constrained splines or ellipses to capture mild non-circularity; thickness can still be computed by ray–curve intersection.

A third extension is **sub-pixel** edge localization to reduce quantization in $\Delta r(\varphi)$ and gain a clearer uncertainty budget.

4.5.14 Reproducibility and data management

To ensure auditability, the following items should be stored along with the figures:

- image filename and pixel scale (m/px),
- ROI masks for each arch and side (downstream/upstream),
- fitted parameters $\{\mathbf{C}_D, R_D, \mathbf{C}_U, R_U\}$,
- thickness profiles $t(\varphi)$ with the discretisation setup,
- residual summaries per degree (median, IQR),
- global comparison arrays (radii, $\Delta\mathbf{C}$).

A simple convention is to export CSV files per arch (`arch_k_params.csv`, `arch_k_thickness.csv`, `arch_k_residuals.csv`) and a global summary (`summary_radii.csv`, `summary_centershift.csv`).

4.5.15 User protocol

1. Export the section from CloudCompare as TIFF at the desired pixel size (e.g., 10 mm/px).
2. In Photoshop, replace transparent background with uniform white; save as PNG/TIFF without downsampling.
3. Run the MATLAB tool; select the image and confirm the pixel size (m/px).
4. Enter the number of arches n to analyse.
5. For each arch:
    a) pan/zoom as needed; draw the downstream ROI and close with double-click;
    b) pan/zoom; draw the upstream ROI and close with double-click;
    c) inspect the overlaid fitted circles and the thickness plot;
    d) review the residual diagrams for both sides.
6. After all arches, review the global dashboards: radii bars, overlaid thicknesses, and $\Delta\mathbf{C}$ vector diagram.

4.5.16 Data analysis summary

The proposed raster-to-metrics workflow combines selective human judgement (ROI definition) with robust, automated fitting and metrology. It yields angle-resolved thickness measurements, adherence diagnostics tied to *real* pixels, and cross-arch comparative views. The results demonstrate that the arches are semicircular in the engineering sense but not identical, with centimetric thickness variability and non-concentric upstream/downstream centers. The methodology is lightweight, auditable, and extensible; it is thus well-suited to support conservation diagnostics, construction verification, and long-term monitoring of built infrastructure.

The complete MATLAB implementation is provided in Appendix A, including helper functions for ROI handling, robust geometric fitting, thickness ray casting.

Chapter 5. Methodological recommendations and operational guidelines

5.1 Methodological strengths and research contributions

The research framework developed in this thesis highlights several methodological and scientific aspects that are relevant for the analysis and documentation of built heritage and civil infrastructures. These aspects concern primarily the integration of surveying techniques with numerical analysis, the transferability of the proposed procedures across different disciplinary contexts, and the possibility of extending point-cloud analysis beyond purely descriptive or representational purposes.

Methodological innovation

A central aspect of the research concerns the methodological integration between three-dimensional surveying techniques and numerical analysis procedures. The proposed approach combines high-resolution surveying methods—such as terrestrial laser scanning (TLS), photogrammetry (terrestrial and UAS-based), and geodetic control networks—with computational tools for the analysis of the resulting datasets.

Within this framework, point clouds are treated not only as sources for geometric reconstruction or visualization, but as structured datasets suitable for quantitative analysis. Dedicated computational workflows have been implemented to extract geometric profiles from point clouds and to process them through spline interpolation, smoothing procedures, and differential analysis. This allows the computation of geometric indicators such as first and second derivatives, curvature variations, and local discontinuities along the surveyed surfaces.

The methodological interest of this approach lies in the possibility of transforming large three-dimensional datasets into analytical representations that can be systematically investigated. Such procedures enable the comparison between different geometric datasets, either obtained at different temporal stages or extracted from different portions of the same structure.

Scientific contribution

The analytical framework developed in this research provides tools for the quantitative investigation of geometric configurations derived from digital surveys. By combining high-resolution spatial ac-

quisition with differential analysis, the methodology enables the identification of geometric variations that may not be immediately evident through standard visualization techniques.

In the presented case studies, this approach allows the detection of variations in curvature and geometric discontinuities along structural elements such as arches, bridge decks, and other infrastructural components. These variations may be associated with several phenomena, including construction tolerances, localized deformations, or long-term material transformations.

The proposed procedures also support the comparison between theoretical design geometries and as-built configurations derived from surveying data. This type of comparison can provide useful information for structural interpretation, dimensional verification, and monitoring activities. More generally, the methodology contributes to expanding the analytical potential of point-cloud datasets in the study of built structures.

Interdisciplinary relevance

The methodological framework adopted in this thesis operates at the intersection of several disciplinary domains, including architectural surveying and representation, structural analysis, computational geometry, and geotechnical experimentation.

The case studies demonstrate the applicability of the same analytical procedures across different spatial scales and research contexts. These range from laboratory-scale experiments—such as the analysis of geotechnical models—to large-scale infrastructural and architectural surveys. In each case, the same principles of geometric extraction, numerical analysis, and differential interpretation are applied to datasets generated through different acquisition techniques.

This cross-disciplinary applicability highlights the potential of the proposed methods to serve as a common analytical framework for research areas that traditionally employ different investigative tools. The integration of digital surveying with numerical analysis therefore supports a broader dialogue between disciplines involved in the documentation, analysis, and monitoring of complex built systems.

5.2 Dimensional analysis protocols using MATLAB

Objective

The analytical procedures implemented in MATLAB provide a structured framework for the dimensional investigation of profiles extracted from point clouds. The protocol has been developed to support

the interpretation of geometric data derived from different surveying techniques, including terrestrial laser scanning, photogrammetric reconstruction, and other three-dimensional sensing systems.

A central feature of the workflow is the possibility of iteratively refining the analytical results through the adjustment of processing parameters. In particular, the pipeline allows the operator to control the level of smoothing applied to the interpolated profiles and the degree of subsampling applied to the original point cloud. These parameters make it possible to balance two competing requirements: preserving the geometric resolution of the data while limiting the influence of high-frequency noise generated by micro-variations in the measured surface.

The flexibility of this approach allows the analysis to be adapted to datasets characterized by different densities, acquisition methods, and levels of geometric complexity.

Input preparation and scale

The workflow is based on sections extracted from three-dimensional point clouds generated through standard surveying procedures. These sections may be exported from point-cloud processing environments, such as CloudCompare, as raster images or as sets of sampled points.

A fundamental requirement of the process is the preservation of the metric scale of the original dataset. When raster representations are used, the pixel size defined at the export stage determines the conversion between image coordinates and metric units. This information is verified during the initialization of the MATLAB workflow so that all subsequent measurements remain expressed in consistent metric units.

Although the analysis can be applied to raster sections, the method ultimately relies on geometric information derived from the underlying point cloud, ensuring that the numerical processing remains directly connected to the measured data.

Profile selection and data preparation

The analysis requires the selection of representative portions of the point cloud corresponding to the geometric profiles to be investigated. These segments are typically extracted through manual or semi-automatic segmentation procedures performed within point-cloud processing software.

Once the relevant regions are identified, the extracted datasets can be subsampled in order to reduce redundancy while preserving the overall geometric structure of the profile. The subsampling step allows the analyst to control the density of the points used in the numerical fitting procedures and contributes to stabilizing the subsequent interpolation and smoothing operations.

This preparatory phase ensures that the datasets used for the analysis contain only geometrically meaningful information, minimizing the influence of occlusions, noise, or irregular sampling density.

Geometric fitting and profile smoothing

After the extraction of the relevant data, the MATLAB pipeline reconstructs continuous profiles through interpolation and smoothing procedures. The smoothing parameter plays a key role in determining the behaviour of the fitted curve. Higher smoothing values tend to preserve small-scale geometric variations but may also amplify measurement noise, whereas stronger smoothing reduces noise but may suppress subtle geometric features.

The possibility of adjusting this parameter allows the analyst to explore different levels of geometric detail and to identify a configuration that provides a suitable balance between geometric fidelity and numerical stability. This exploratory process represents an important aspect of the workflow, as it enables the refinement of the analytical results according to the characteristics of the dataset under investigation.

Differential analysis of geometric variation

Once the smoothed profiles have been generated, the pipeline computes derivative-based indicators that describe the geometric behaviour of the analysed section. In particular, variations in slope and curvature are evaluated along the profile, providing quantitative information about changes in the geometry of the surveyed structure.

These indicators make it possible to highlight subtle geometric variations that may not be immediately visible in traditional graphical representations. Changes in curvature, for instance, can reveal localized geometric irregularities, construction tolerances, or structural deformations that remain difficult to detect through direct visual inspection of section drawings.

The differential diagrams generated by the workflow therefore provide an analytical perspective on the geometry of the surveyed elements, complementing the descriptive information contained in conventional drawings.

Interpretation and graphical outputs

The results of the analysis are visualized through a set of diagrams that describe the evolution of the extracted profiles and their associated geometric indicators. These graphical outputs include the reconstructed profiles, slope diagrams, and curvature diagrams, which together provide a multi-layered representation of the geometric behaviour of the structure.

It is important to emphasize that this analytical representation does not aim to replace traditional graphical outputs such as plans, sections, or elevations. Rather, it is intended to complement these conventional representations by introducing an additional layer of quantitative analysis.
By combining traditional drawings with derivative-based diagrams, the method facilitates a more comprehensive understanding of the surveyed geometry. This integrated approach supports the interpretation of structural behaviour and contributes to improving the overall knowledge of the investigated architectural or infrastructural asset.

5.3 Integration with open-source data formats and tools (interoperability)

Recommended exchange formats

- **Point clouds:** LAS/LAZ for efficient storage and streaming; E57 for multi-sensor metadata; PLY for mesh-plus-colour when simplicity suffices.
- **Raster and grids:** GeoTIFF with explicit CRS; Cloud-Optimized GeoTIFF (COG) for web streaming; NetCDF for volumetric or time-indexed fields.
- **Meshes and scenes:** OBJ/MTL for broad compatibility; glTF/GLB for lightweight web visualization; STL only for geometry when no textures or colours are needed.
- **BIM and semantic models:** IFC4 for building-scale semantics; CityGML/CityJSON for urban context; OGC GeoPackage for geospatial tables and tiles.

Interoperability practices

1. **CRS hygiene.** Record CRS at every stage; for non-georeferenced sections, document local datum and axes orientation in metadata and plots.
2. **Sidecar metadata.** Use JSON sidecars to bind pixel size, acquisition dates, instrument settings, and registration RMSE to each deliverable.
3. **Versioning and fingerprints.** Compute content hashes for exported artifacts; include them in reports and figure captions to anchor traceability.
4. **Bridging HBIM.** Where appropriate, align idealized curves and thickness fields with IFC entities via property sets, preserving both metric accuracy and provenance.

Conclusions and future research perspectives

5.4 Summary of main findings

The research presented in this thesis demonstrates how the integration of three-dimensional surveying techniques and numerical analysis tools can generate analytical representations of the built environment that are both metrically reliable and interpretatively meaningful. Within this framework, drawing and graphical representation operate not only as descriptive outputs but also as interpretive tools capable of mediating between measured data and analytical interpretation.

The computational workflow implemented in MATLAB enables the systematic analysis of geometric profiles extracted from point clouds. The experiments carried out across the different case studies show that the method can be used to compare geometric models and to identify variations in structural configurations through the analysis of curvature and slope variations. In particular, the comparison between pairs of point clouds allows the investigation of dimensional relationships between different surfaces or structural components.

The adopted approach has been applied, for example, to the comparison between the intrados and extrados geometries of bridge arches, enabling the evaluation of thickness variations and deviations from the theoretical design geometry. At the same time, the same methodology can be extended to temporal monitoring scenarios, where point clouds acquired at different moments in time are compared in order to detect structural deformations or long-term geometric changes.

Although the case studies presented in this work primarily rely on the comparison of pairs of point clouds, the analytical structure of the workflow allows the extension of the method to larger datasets involving multiple acquisitions or multiple structural elements. This enables the possibility of conducting comparative analyses across several components of the same structure or across different temporal observations of the same object.

Another relevant outcome concerns the identification of geometric reference models through the analysis of extracted sections. In the case study of the Gleno dam, for instance, circular geometries were interpolated from selected regions of interest (ROIs) extracted from raster sections derived from the point cloud. However, the same analytical procedure can be extended to other geometric configurations by adapting the fitting model used in the script. Possible alternatives include linear, polygonal, parabolic, or other parametric geometries, depending on the characteristics of the investigated structure.

More generally, the workflow is designed to remain adaptable to a wide range of analytical contexts. The MATLAB scripts developed in this research are conceived as accessible tools that can be used

even by operators who may not be specialized in programming, provided that they possess adequate expertise in surveying, metrology, and geometric interpretation. The scripts allow the user to control the analytical parameters of the processing chain while maintaining a transparent link between the numerical results and the underlying measured data.

An additional aspect concerns the integration of the point-cloud-based analysis with complementary measurement techniques. In some experimental configurations, the geometric observations derived from point clouds can be compared with independent measurements obtained from displacement transducers or other monitoring instruments. Such combined approaches allow the verification of experimentally measured deformations through their geometric manifestation in the point-cloud model.

Finally, the outputs generated by the workflow are intentionally designed to remain open and adaptable to different representation needs. The results can be exported in multiple formats, including three-dimensional polylines suitable for spatial visualization, tabular datasets (e.g., CSV files) for numerical inspection, and graphical diagrams generated within MATLAB. The graphical outputs can be customized in terms of style, resolution, and figure dimensions, allowing them to be adapted to different communication contexts.

This multiplicity of representational formats facilitates the integration of the analytical results into different research environments. In particular, it supports collaboration between disciplines that rely on different forms of data interpretation, ranging from spatial visualization in three-dimensional environments to tabular analysis and statistical evaluation. In this sense, the proposed workflow contributes to establishing a flexible analytical framework capable of supporting interdisciplinary research on the geometry, behaviour, and evolution of built structures.

5.5 Implications for the surveying and conservation community

The methodological framework developed in this research suggests several implications for the broader field of surveying, structural analysis, and heritage conservation.

First, the integration of point-cloud surveying with numerical analysis workflows enables the transformation of geometric datasets into analytical resources capable of supporting structural interpretation and monitoring. The ability to compare different point-cloud datasets—either across time or across structural components—creates new opportunities for systematic monitoring of geometric variations.

Second, the use of derivative-based indicators such as slope and curvature provides additional analytical tools for interpreting geometric behaviour. These indicators allow subtle variations in the geometry of structures to be detected and quantified, complementing traditional drawing-based documentation.

Finally, the emphasis on interoperable outputs and transparent analytical procedures contributes to improving the reproducibility of survey-based analyses. By combining graphical, tabular, and spatial representations, the workflow facilitates the exchange of information across different disciplinary domains involved in the documentation and conservation of built heritage.

5.6 Identified limitations and challenges

Despite the analytical potential of the proposed workflow, several methodological limitations should be considered.

One limitation concerns the dependence of the analysis on the selection of representative portions of the point cloud. The identification of relevant regions or profiles often requires expert judgement, and different selections may lead to slightly different analytical outcomes.

A second limitation relates to the resolution and quality of the input data. The geometric accuracy of the results ultimately depends on the precision of the original survey and on the density of the available point cloud. Measurement noise, incomplete data coverage, or registration uncertainties may influence the stability of the numerical fitting procedures.

Another challenge concerns the choice of geometric models used in the analysis. While circular or parametric models may provide a suitable approximation for many structural configurations, some geometries may require more complex representations or segmented modelling approaches.

Finally, when datasets are derived from multiple acquisition campaigns, the quality of the registration between different point clouds becomes a critical factor. Misalignment between datasets may introduce systematic deviations that must be carefully evaluated during the interpretation of the results.

5.7 Proposed future research developments

Several directions for future research emerge from the methodological framework developed in this thesis.

One possible development concerns the extension of the analytical models used for geometric fitting. Future work could explore the use of more flexible parametric models capable of describing complex structural geometries beyond simple circular or linear approximations.

Another promising direction involves the systematic integration of point-cloud analysis with time-dependent monitoring data. By combining repeated surveys with complementary measurement techniques, it may be possible to develop more comprehensive approaches for detecting and interpreting structural deformations.

Further research could also focus on improving the automation of the analytical workflow while maintaining the transparency and interpretability of the results. The development of user-friendly interfaces and modular scripts may facilitate the broader adoption of these methods in different research and professional contexts.

Finally, the integration of the analytical outputs with digital documentation systems, such as HBIM environments or geospatial information systems, could provide new opportunities for linking geometric analysis with broader processes of documentation, conservation planning, and structural assessment.

Appendices

Appendix A

Code listing

Listing A.1: Script Python di esempio

```python
import numpy as np
import tkinter as tk
from tkinter import messagebox

def calculate_rotation():
    try:
        x1 = float(entry_x1.get())
        y1 = float(entry_y1.get())
        x2 = float(entry_x2.get())
        y2 = float(entry_y2.get())

        dx = x2 - x1
        dy = y2 - y1

        theta = np.arctan2(dy, dx)  # angle in radians
        rotation_angle = -theta
        rotation_angle_degrees = np.degrees(rotation_angle)

        result = (f"Rotation angle (radians): {rotation_angle:.6f}\n"
                  f"Rotation angle (degrees): {rotation_angle_degrees:.2f}deg")
        messagebox.showinfo("Rotation Angle", result)
    except ValueError:
        messagebox.showerror("Input Error", "Please enter valid numeric values.")

# Create the main window
root = tk.Tk()
root.title("Z-Axis Rotation Angle Calculator")

```

```
# Labels and entries
tk.Label(root, text="x1:").grid(row=0, column=0, padx=5, pady=5, sticky='e')
entry_x1 = tk.Entry(root)
entry_x1.grid(row=0, column=1, padx=5, pady=5)

tk.Label(root, text="y1:").grid(row=1, column=0, padx=5, pady=5, sticky='e')
entry_y1 = tk.Entry(root)
entry_y1.grid(row=1, column=1, padx=5, pady=5)

tk.Label(root, text="x2:").grid(row=2, column=0, padx=5, pady=5, sticky='e')
entry_x2 = tk.Entry(root)
entry_x2.grid(row=2, column=1, padx=5, pady=5)

tk.Label(root, text="y2:").grid(row=3, column=0, padx=5, pady=5, sticky='e')
entry_y2 = tk.Entry(root)
entry_y2.grid(row=3, column=1, padx=5, pady=5)

# Button
calculate_button = tk.Button(root, text="Calculate Rotation Angle", command=calculate_rotation)
calculate_button.grid(row=4, column=0, columnspan=2, pady=10)

root.mainloop()
```

Listing A.2: Vertical section AA'. MATLAB listing

```
% === PARAMETERS ===
filename_pre  = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di punti/prova 3/Esportazione txt da CloudCompare/PRE prova.section.txt';
filename_post = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di punti/prova 3/Esportazione txt da CloudCompare/POST prova.section.txt';
```

```
y0 = 0.165;
delta = 0.0002;
p = 0.9999999;

% === FUNCTION TO LOAD AND PROCESS A SECTION ===
function [xq, zq, dz, ddz, x_unique, z_unique, sp] = process_section(
   filename, y0, delta, p)
    data = load(filename);
    x = data(:,1); y = data(:,2); z = data(:,3);
    idx = (y > y0 - delta) & (y < y0 + delta);
    x_sec = x(idx);
    z_sec = z(idx);
    [x_sorted, sort_idx] = sort(x_sec);
    z_sorted = z_sec(sort_idx);
    [x_unique, ia] = unique(x_sorted);
    z_unique = z_sorted(ia);
    sp = csaps(x_unique, z_unique, p);
    xq = linspace(min(x_unique), max(x_unique), 740);
    zq = fnval(sp, xq);
    dz = fnval(fnder(sp,1), xq);
    ddz = fnval(fnder(sp,2), xq);
end

% === PROCESS BOTH DATASETS ===
[xq_pre, zq_pre, dz_pre, ddz_pre, x_u_pre, z_u_pre, sp_pre] =
   process_section(filename_pre,  y0, delta, p);
[xq_post, zq_post, dz_post, ddz_post, x_u_post, z_u_post, sp_post] =
   process_section(filename_post, y0, delta, p);

% === REFERENCE POLYLINE ===
x_ref = [0.045, 0.185, 0.390, 0.785];
z_ref = [-0.059, -0.059, -0.199, -0.199];
dx = diff(x_ref);
dz = diff(z_ref);
slopes = dz ./ dx;
```

```
36
37 % === PLOTS ===
38 figure;
39
40 % === PLOT 1: INTERPOLATED PROFILES ===
41 subplot(3,1,1);
42 plot(x_ref, z_ref, 'g-', 'LineWidth', 1.5); hold on;
43 plot(x_u_pre, z_u_pre, 'b.', xq_pre, zq_pre, 'b', 'LineWidth', 1.5);
44 plot(x_u_post, z_u_post, 'r.', xq_post, zq_post, 'r', 'LineWidth', 1.5);
45 % Transducers
46 % Define colors (c1-14)
47 c1  = [0.0000 0.4470 0.7410]; % blue
48 c2  = [0.8500 0.3250 0.0980]; % orange
49 c3  = [0.9290 0.6940 0.1250]; % yellow
50 c5  = [0.4660 0.6740 0.1880]; % green
51 c6  = [0.3010 0.7450 0.9330]; % light blue
52 c8  = [0.0000 0.0000 0.0000]; % black
53 c9  = [0.8500 0.3250 0.0980]*0.7; % dark orange
54 c10 = [0.9290 0.6940 0.1250]*0.7; % ochre
55 c12 = [0.4660 0.6740 0.1880]*0.7; % olive green
56 c13 = [0.3010 0.7450 0.9330]*0.7; % dark cyan
57
58 % Plot initial transducers (c1-c7)
59 plot(0.090, -0.0574, 'ko', 'MarkerFaceColor', c1);  % initial transducer
   L8
60 plot(0.180, -0.0540, 'ko', 'MarkerFaceColor', c2);  % initial transducer
   R1
61 plot(0.235, -0.0815, 'ko', 'MarkerFaceColor', c3);  % initial transducer
   R3
62 plot(0.348, -0.1563, 'ko', 'MarkerFaceColor', c5);  % initial transducer
   R5
63 plot(0.430, -0.1955, 'ko', 'MarkerFaceColor', c6);  % initial transducer
   L6
64 plot(0.090, -0.0624, 'ko', 'MarkerFaceColor', c8);   % final transducer
   L8
```

```
65 plot(0.1838, -0.0588, 'ko', 'MarkerFaceColor', c9);   % final transducer
       R1
66 plot(0.2493, -0.0895, 'ko', 'MarkerFaceColor', c10);  % final transducer
       R3
67 plot(0.3585, -0.1645, 'ko', 'MarkerFaceColor', c12);  % final transducer
       R5
68 plot(0.430, -0.1983, 'ko', 'MarkerFaceColor', c13);  % final transducer
       L6
69
70 xlabel('X [m]'); ylabel('Z [m]');
71 title('Section AA'': Interpolated Profiles');
72 legend('Design profile','Initial survey points','Initial interpolation','
       Final survey points','Final interpolation','Initial L8 transducer','
       Initial R1 transducer','Initial R3 transducer','Initial R5 transducer'
       ,'Initial L6 transducer','Final L8 transducer','Final R1 transducer','
       Final R3 transducer','Final R5 transducer','Final L6 transducer', '
       NumColumns',4, 'Location', 'northeast');
73 grid on;
74
75 % === PLOT 2: FIRST DERIVATIVES ===
76 subplot(3,1,2);
77 hold on;
78 x1 = x_ref(1:end-1); x2 = x_ref(2:end);
79 x_seg = [x1; x2]; y_seg = [slopes; slopes];
80 hRef = plot(x_seg, y_seg, 'g-', 'LineWidth', 1.5);
81 hPre  = plot(xq_pre, dz_pre, 'b', 'LineWidth', 1.5);
82 hPost = plot(xq_post, dz_post, 'r', 'LineWidth', 1.5);
83 xlabel('X [m]'); ylabel('dZ/dX');
84 title('Section AA'': First Derivative (Slope)');
85 legend([hRef(1), hPre, hPost], {'Design slope','Initial slope','Final
       slope'}, 'NumColumns',1, 'Location', 'southeast');
86 grid on;
87
88 % === PLOT 3: SECOND DERIVATIVES ===
89 subplot(3,1,3);
```

```
plot(xq_pre, ddz_pre, 'b', 'LineWidth', 1.5); hold on;
plot(xq_post, ddz_post, 'r', 'LineWidth', 1.5);
for i = 2:length(x_ref)-1
    x_v = x_ref(i);
    plot([x_v x_v], ylim, 'g--', 'LineWidth', 1.5);
end
xlabel('X [m]'); ylabel('d^2Z/dX^2');
title('Section AA'': Second Derivative (Curvature)');
legend('Initial curvature','Final curvature','Slope variation', '
   NumColumns',1, 'Location', 'northeast');
grid on;

% ==== sp POLY EXPORT (uses current MATLAB folder) ====
N = 1000;  % number of sampled points

writePoly('AAsp_pre.poly',  sp_pre,  y0, N);
writePoly('AAsp_post.poly', sp_post, y0, N);

function writePoly(filename, sp, y0, N)
    if nargin < 4, N = 1000; end
    % Sample the spline evenly along its X range
    xq = linspace(sp.breaks(1), sp.breaks(end), N);
    zq = fnval(sp, xq);

    % Open file for writing
    fid = fopen(filename, 'w');
    assert(fid>0, 'Cannot write file: %s', filename);

    % Write each vertex in X Y Z format, with constant Y = y0
    for i = 1:numel(xq)
        fprintf(fid, '%.6f %.6f %.6f\n', xq(i), y0, zq(i));  % X Y Z
    end

    fclose(fid);
end
```

```
124
125 % === EXPORT DATA TO EXCEL ===
126 outFile = 'SectionAA_profiles.xlsx';
127
128 % Helper function to round and format to 4 decimals
129 fmt = @(x) round(x,4);
130
131 % Reference polyline
132 T_ref = table(fmt(x_ref(:)), fmt(z_ref(:)), ...
133     'VariableNames', {'X_ref [m]','Z_ref [m]'});
134 writetable(T_ref, outFile, 'Sheet','Reference');
135
136 % Pre (unique points)
137 T_pre_unique = table(fmt(x_u_pre(:)), fmt(z_u_pre(:)), ...
138     'VariableNames', {'X_unique_pre [m]','Z_unique_pre [m]'});
139 writetable(T_pre_unique, outFile, 'Sheet','Pre_unique');
140
141 % Pre (interpolated query with derivatives)
142 T_pre_interp = table(fmt(xq_pre(:)), fmt(zq_pre(:)), fmt(dz_pre(:)), fmt(
    ddz_pre(:)), ...
143     'VariableNames', {'Xq_pre [m]','Zq_pre [m]','dZ_pre','ddZ_pre'});
144 writetable(T_pre_interp, outFile, 'Sheet','Pre_interp');
145
146 % Post (unique points)
147 T_post_unique = table(fmt(x_u_post(:)), fmt(z_u_post(:)), ...
148     'VariableNames', {'X_unique_post [m]','Z_unique_post [m]'});
149 writetable(T_post_unique, outFile, 'Sheet','Post_unique');
150
151 % Post (interpolated query with derivatives)
152 T_post_interp = table(fmt(xq_post(:)), fmt(zq_post(:)), fmt(dz_post(:)),
    fmt(ddz_post(:)), ...
153     'VariableNames', {'Xq_post [m]','Zq_post [m]','dZ_post','ddZ_post'});
154 writetable(T_post_interp, outFile, 'Sheet','Post_interp');
155
156 fprintf('Excel file written: %s\n', outFile);
```

Listing A.3: Vertical section BB'. MATLAB listing

```
% === PARAMETERS ===
filename_pre  = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di punti/prova 3/Esportazione txt da CloudCompare/PRE prova.section.txt';
filename_post = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di punti/prova 3/Esportazione txt da CloudCompare/POST prova.section.txt';
y0 = 0.105;
delta = 0.0002;
p = 0.9999999;

% === FUNCTION TO LOAD AND PROCESS A SECTION ===
function [xq, zq, dz, ddz, x_unique, z_unique, sp] = process_section(filename, y0, delta, p)
    data = load(filename);
    x = data(:,1); y = data(:,2); z = data(:,3);
    idx = (y > y0 - delta) & (y < y0 + delta);
    x_sec = x(idx);
    z_sec = z(idx);
    [x_sorted, sort_idx] = sort(x_sec);
    z_sorted = z_sec(sort_idx);
    [x_unique, ia] = unique(x_sorted);
    z_unique = z_sorted(ia);
    sp = csaps(x_unique, z_unique, p);
    xq = linspace(min(x_unique), max(x_unique), 740);
    zq = fnval(sp, xq);
    dz = fnval(fnder(sp,1), xq);
    ddz = fnval(fnder(sp,2), xq);
end

% === PROCESS BOTH DATASETS ===
[xq_pre, zq_pre, dz_pre, ddz_pre, x_u_pre, z_u_pre, sp_pre] = process_section(filename_pre,  y0, delta, p);
[xq_post, zq_post, dz_post, ddz_post, x_u_post, z_u_post, sp_post] = process_section(filename_post, y0, delta, p);
```

```
29
30 % === REFERENCE POLYLINE ===
31 x_ref = [0.045, 0.185, 0.390, 0.785];
32 z_ref = [-0.059, -0.059, -0.199, -0.199];
33 dx = diff(x_ref);
34 dz = diff(z_ref);
35 slopes = dz ./ dx;
36
37 % === PLOTS ===
38 figure;
39
40 % === PLOT 1: INTERPOLATED PROFILES ===
41 subplot(3,1,1);
42 plot(x_ref, z_ref, 'g-', 'LineWidth', 1.5); hold on;
43 plot(x_u_pre, z_u_pre, 'b.', xq_pre, zq_pre, 'b', 'LineWidth', 1.5);
44 plot(x_u_post, z_u_post, 'r.', xq_post, zq_post, 'r', 'LineWidth', 1.5);
45 xlabel('X [m]'); ylabel('Z [m]');
46 title('Section BB'': Interpolated Profiles');
47 legend('Design profile','Initial survey points','Initial interpolation','
      Final survey points','Final interpolation');
48 grid on;
49
50 % === PLOT 2: FIRST DERIVATIVES ===
51 subplot(3,1,2);
52 hold on;
53 x1 = x_ref(1:end-1); x2 = x_ref(2:end);
54 x_seg = [x1; x2]; y_seg = [slopes; slopes];
55 hRef = plot(x_seg, y_seg, 'g-', 'LineWidth', 1.5);
56 hPre  = plot(xq_pre, dz_pre, 'b', 'LineWidth', 1.5);
57 hPost = plot(xq_post, dz_post, 'r', 'LineWidth', 1.5);
58 xlabel('X [m]'); ylabel('dZ/dX');
59 title('Section BB'': First Derivative (Slope)');
60 legend([hRef(1), hPre, hPost], {'Design slope','Initial slope','Final
      slope'}, 'Location', 'southeast');
61 grid on;
```

```
62
63 % === PLOT 3: SECOND DERIVATIVES ===
64 subplot(3,1,3);
65 plot(xq_pre, ddz_pre, 'b', 'LineWidth', 1.5); hold on;
66 plot(xq_post, ddz_post, 'r', 'LineWidth', 1.5);
67 for i = 2:length(x_ref)-1
68     x_v = x_ref(i);
69     plot([x_v x_v], ylim, 'g--', 'LineWidth', 1.5);
70 end
71 xlabel('X [m]'); ylabel('d^2Z/dX^2');
72 title('Section BB'': Second Derivative (Curvature)');
73 legend('Initial curvature','Final curvature','Slope variation','Location'
    , 'southeast');
74 grid on;
75
76 % ==== sp POLY EXPORT (uses current MATLAB folder) ====
77 N = 1000;  % number of sampled points
78
79 writePoly('BBsp_pre.poly',  sp_pre,  y0, N);
80 writePoly('BBsp_post.poly', sp_post, y0, N);
81
82 function writePoly(filename, sp, y0, N)
83     if nargin < 4, N = 1000; end
84     % Sample the spline evenly along its X range
85     xq = linspace(sp.breaks(1), sp.breaks(end), N);
86     zq = fnval(sp, xq);
87
88     % Open file for writing
89     fid = fopen(filename, 'w');
90     assert(fid>0, 'Cannot write file: %s', filename);
91
92     % Write each vertex in X Y Z format, with constant Y = y0
93     for i = 1:numel(xq)
94         fprintf(fid, '%.6f %.6f %.6f\n', xq(i), y0, zq(i));  % X Y Z
95     end
```

```
96
97      fclose(fid);
98  end
99
100 % === EXPORT DATA TO EXCEL ===
101 outFile = 'SectionBB_profiles.xlsx';
102
103 % Helper function to round and format to 4 decimals
104 fmt = @(x) round(x,4);
105
106 % Reference polyline
107 T_ref = table(fmt(x_ref(:)), fmt(z_ref(:)), ...
108     'VariableNames', {'X_ref [m]','Z_ref [m]'});
109 writetable(T_ref, outFile, 'Sheet','Reference');
110
111 % Pre (unique points)
112 T_pre_unique = table(fmt(x_u_pre(:)), fmt(z_u_pre(:)), ...
113     'VariableNames', {'X_unique_pre [m]','Z_unique_pre [m]'});
114 writetable(T_pre_unique, outFile, 'Sheet','Pre_unique');
115
116 % Pre (interpolated query with derivatives)
117 T_pre_interp = table(fmt(xq_pre(:)), fmt(zq_pre(:)), fmt(dz_pre(:)), fmt(
    ddz_pre(:)), ...
118     'VariableNames', {'Xq_pre [m]','Zq_pre [m]','dZ_pre','ddZ_pre'});
119 writetable(T_pre_interp, outFile, 'Sheet','Pre_interp');
120
121 % Post (unique points)
122 T_post_unique = table(fmt(x_u_post(:)), fmt(z_u_post(:)), ...
123     'VariableNames', {'X_unique_post [m]','Z_unique_post [m]'});
124 writetable(T_post_unique, outFile, 'Sheet','Post_unique');
125
126 % Post (interpolated query with derivatives)
127 T_post_interp = table(fmt(xq_post(:)), fmt(zq_post(:)), fmt(dz_post(:)),
    fmt(ddz_post(:)), ...
128     'VariableNames', {'Xq_post [m]','Zq_post [m]','dZ_post','ddZ_post'});
```

```
writetable(T_post_interp, outFile, 'Sheet','Post_interp');

fprintf('Excel file written: %s\n', outFile);
```

Listing A.4: Vertical section CC'. MATLAB listing

```
% === PARAMETERS ===
filename_pre  = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
   punti/prova 3/Esportazione txt da CloudCompare/PRE prova.section.txt';
filename_post = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
   punti/prova 3/Esportazione txt da CloudCompare/POST prova.section.txt'
   ;
y0 = 0.225;
delta = 0.0002;
p = 0.9999999;

% === FUNCTION TO LOAD AND PROCESS A SECTION ===
function [xq, zq, dz, ddz, x_unique, z_unique, sp] = process_section(
   filename, y0, delta, p)
    data = load(filename);
    x = data(:,1); y = data(:,2); z = data(:,3);
    idx = (y > y0 - delta) & (y < y0 + delta);
    x_sec = x(idx);
    z_sec = z(idx);
    [x_sorted, sort_idx] = sort(x_sec);
    z_sorted = z_sec(sort_idx);
    [x_unique, ia] = unique(x_sorted);
    z_unique = z_sorted(ia);
    sp = csaps(x_unique, z_unique, p);
    xq = linspace(min(x_unique), max(x_unique), 740);
    zq = fnval(sp, xq);
    dz = fnval(fnder(sp,1), xq);
    ddz = fnval(fnder(sp,2), xq);
end

% === PROCESS BOTH DATASETS ===
```

```
[xq_pre, zq_pre, dz_pre, ddz_pre, x_u_pre, z_u_pre, sp_pre] =
    process_section(filename_pre,  y0, delta, p);
[xq_post, zq_post, dz_post, ddz_post, x_u_post, z_u_post, sp_post] =
    process_section(filename_post, y0, delta, p);

% === REFERENCE POLYLINE ===
x_ref = [0.045, 0.185, 0.390, 0.785];
z_ref = [-0.059, -0.059, -0.199, -0.199];
dx = diff(x_ref);
dz = diff(z_ref);
slopes = dz ./ dx;

% === PLOTS ===
figure;

% === PLOT 1: INTERPOLATED PROFILES ===
subplot(3,1,1);
plot(x_ref, z_ref, 'g-', 'LineWidth', 1.5); hold on;
plot(x_u_pre, z_u_pre, 'b.', xq_pre, zq_pre, 'b', 'LineWidth', 1.5);
plot(x_u_post, z_u_post, 'r.', xq_post, zq_post, 'r', 'LineWidth', 1.5);
xlabel('X [m]'); ylabel('Z [m]');
title('Section CC'': Interpolated Profiles');
legend('Design profile','Initial survey points','Initial interpolation','
    Final survey points','Final interpolation');
grid on;

% === PLOT 2: FIRST DERIVATIVES ===
subplot(3,1,2);
hold on;
x1 = x_ref(1:end-1); x2 = x_ref(2:end);
x_seg = [x1; x2]; y_seg = [slopes; slopes];
hRef = plot(x_seg, y_seg, 'g-', 'LineWidth', 1.5);
hPre  = plot(xq_pre, dz_pre, 'b', 'LineWidth', 1.5);
hPost = plot(xq_post, dz_post, 'r', 'LineWidth', 1.5);
xlabel('X [m]'); ylabel('dZ/dX');
```

```
title('Section CC'': First Derivative (Slope)');
legend([hRef(1), hPre, hPost], {'Design slope','Initial slope','Final slope'}, 'Location', 'southeast');
grid on;

% === PLOT 3: SECOND DERIVATIVES ===
subplot(3,1,3);
plot(xq_pre, ddz_pre, 'b', 'LineWidth', 1.5); hold on;
plot(xq_post, ddz_post, 'r', 'LineWidth', 1.5);
for i = 2:length(x_ref)-1
    x_v = x_ref(i);
    plot([x_v x_v], ylim, 'g--', 'LineWidth', 1.5);
end
xlabel('X [m]'); ylabel('d^2Z/dX^2');
title('Section CC'': Second Derivative (Curvature)');
legend('Initial curvature','Final curvature','Slope variation', 'Location', 'southeast');
grid on;

% ==== sp POLY EXPORT (uses current MATLAB folder) ====
N = 1000;  % number of sampled points

writePoly('CCsp_pre.poly',  sp_pre,  y0, N);
writePoly('CCsp_post.poly', sp_post, y0, N);

function writePoly(filename, sp, y0, N)
    if nargin < 4, N = 1000; end
    % Sample the spline evenly along its X range
    xq = linspace(sp.breaks(1), sp.breaks(end), N);
    zq = fnval(sp, xq);

    % Open file for writing
    fid = fopen(filename, 'w');
    assert(fid>0, 'Cannot write file: %s', filename);

```

```
    % Write each vertex in X Y Z format, with constant Y = y0
    for i = 1:numel(xq)
        fprintf(fid, '%.6f %.6f %.6f\n', xq(i), y0, zq(i));  % X Y Z
    end

    fclose(fid);
end

% === EXPORT DATA TO EXCEL ===
outFile = 'SectionCC_profiles.xlsx';

% Helper function to round and format to 4 decimals
fmt = @(x) round(x,4);

% Reference polyline
T_ref = table(fmt(x_ref(:)), fmt(z_ref(:)), ...
    'VariableNames', {'X_ref [m]','Z_ref [m]'});
writetable(T_ref, outFile, 'Sheet','Reference');

% Pre (unique points)
T_pre_unique = table(fmt(x_u_pre(:)), fmt(z_u_pre(:)), ...
    'VariableNames', {'X_unique_pre [m]','Z_unique_pre [m]'});
writetable(T_pre_unique, outFile, 'Sheet','Pre_unique');

% Pre (interpolated query with derivatives)
T_pre_interp = table(fmt(xq_pre(:)), fmt(zq_pre(:)), fmt(dz_pre(:)), fmt(
   ddz_pre(:)), ...
    'VariableNames', {'Xq_pre [m]','Zq_pre [m]','dZ_pre','ddZ_pre'});
writetable(T_pre_interp, outFile, 'Sheet','Pre_interp');

% Post (unique points)
T_post_unique = table(fmt(x_u_post(:)), fmt(z_u_post(:)), ...
    'VariableNames', {'X_unique_post [m]','Z_unique_post [m]'});
writetable(T_post_unique, outFile, 'Sheet','Post_unique');

```

```
% Post (interpolated query with derivatives)
T_post_interp = table(fmt(xq_post(:)), fmt(zq_post(:)), fmt(dz_post(:)),
    fmt(ddz_post(:)), ...
     'VariableNames', {'Xq_post [m]','Zq_post [m]','dZ_post','ddZ_post'});
writetable(T_post_interp, outFile, 'Sheet','Post_interp');

fprintf('Excel file written: %s\n', outFile);
```

Listing A.5: Vertical section DD'. MATLAB listing

```
% === PARAMETERS ===
filename_pre  = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
    punti/prova 3/Esportazione txt da CloudCompare/PRE prova.section.txt';
filename_post = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
    punti/prova 3/Esportazione txt da CloudCompare/POST prova.section.txt'
    ;
x0 = 0.090;
delta = 0.0002;
p = 0.9999999;

% === FUNCTION TO LOAD AND PROCESS A SECTION ===
function [yq, zq, dz, ddz, y_unique, z_unique, sp] = process_section(
    filename, x0, delta, p)
     data = load(filename);
     x = data(:,1); y = data(:,2); z = data(:,3);
     idy = (x > x0 - delta) & (x < x0 + delta);
     y_sec = y(idy);
     z_sec = z(idy);
     [y_sorted, sort_idy] = sort(y_sec);
     z_sorted = z_sec(sort_idy);
     [y_unique, ia] = unique(y_sorted);
     z_unique = z_sorted(ia);
     sp = csaps(y_unique, z_unique, p);
     yq = linspace(min(y_unique), max(y_unique), 240);
     zq = fnval(sp, yq);
     dz = fnval(fnder(sp,1), yq);
```

```
    ddz = fnval(fnder(sp,2), yq);
end

% === PROCESS BOTH DATASETS ===
[yq_pre, zq_pre, dz_pre, ddz_pre, y_u_pre, z_u_pre, sp_pre] =
   process_section(filename_pre,  x0, delta, p);
[yq_post, zq_post, dz_post, ddz_post, y_u_post, z_u_post, sp_post] =
   process_section(filename_post, x0, delta, p);

% === REFERENCE POLYLINE ===
y_ref = [0.045, 0.285];
z_ref = [-0.059, -0.059];
dy = diff(y_ref);
dz = diff(z_ref);
slopes = dz ./ dy;

% === PLOTS ===
figure;

% === PLOT 1: INTERPOLATED PROFILES ===
subplot(3,1,1);
plot(y_ref, z_ref, 'g-', 'LineWidth', 1.5); hold on;
plot(y_u_pre, z_u_pre, 'b.', yq_pre, zq_pre, 'b', 'LineWidth', 1.5);
plot(y_u_post, z_u_post, 'r.', yq_post, zq_post, 'r', 'LineWidth', 1.5);
% color assignment (RGB)
c1  = [0.0000 0.4470 0.7410]; % blue
c8  = [0.0000 0.0000 0.0000]; % black
% trasducer
plot(0.165, -0.0574, 'ko', 'MarkerFaceColor', c1); % initial transducer
   L8
plot(0.165, -0.0624, 'ko', 'MarkerFaceColor', c8); % final transducer L8
xlabel('Y [m]'); ylabel('Z [m]');
title('Section DD'' (L8 transducer): Interpolated Profiles');
legend('Design profile', 'Initial survey points', 'Initial interpolation'
   , ...
```

```
54        'Final survey points', 'Final interpolation', 'Initial L8
              transducer', 'Final L8 transducer', ...
55        'Location', 'northwest');
56  grid on;
57
58  % === PLOT 2: FIRST DERIVATIVES ===
59  subplot(3,1,2);
60  hold on;
61  y1 = y_ref(1:end-1); y2 = y_ref(2:end);
62  y_seg = [y1; y2]; x_seg = [slopes; slopes];
63  hRef = plot(y_seg, x_seg, 'g-', 'LineWidth', 1.5);
64  hPre  = plot(yq_pre, dz_pre, 'b', 'LineWidth', 1.5);
65  hPost = plot(yq_post, dz_post, 'r', 'LineWidth', 1.5);
66  xlabel('Y [m]'); ylabel('dZ/dY');
67  title('Section DD'' (L8 transducer): First Derivative (Slope)');
68  legend([hRef(1), hPre, hPost], {'Design slope','Initial slope','Final
        slope'}, 'Location', 'northwest');
69  grid on;
70
71  % === PLOT 3: SECOND DERIVATIVES ===
72  subplot(3,1,3);
73  plot(yq_pre, ddz_pre, 'b', 'LineWidth', 1.5); hold on;
74  plot(yq_post, ddz_post, 'r', 'LineWidth', 1.5);
75  for i = 2:length(y_ref)-1
76      y_v = y_ref(i);
77      plot([y_v y_v], xlim, 'g--', 'LineWidth', 1.5);
78  end
79  xlabel('Y [m]'); ylabel('d^2Z/dY^2');
80  title('Section DD'' (L8 transducer): Second Derivative (Curvature)');
81  legend('Initial curvature', 'Final curvature', 'Slope variation', '
        Location', 'northwest');
82
83  grid on;
84
85  % ==== sp POLY EXPORT (uses current MATLAB folder) ====
```

```
N = 1000;  % number of sampled points

writePoly('DDsp_pre.poly',  x0,  sp_pre, N);
writePoly('DDsp_post.poly', x0, sp_post, N);

function writePoly(filename, x0, sp, N)
    if nargin < 4, N = 1000; end
    % Sample the spline evenly along its Y range
    yq = linspace(sp.breaks(1), sp.breaks(end), N);
    zq = fnval(sp, yq);

    % Open file for writing
    fid = fopen(filename, 'w');
    assert(fid>0, 'Cannot write file: %s', filename);

    % Write each vertex in X Y Z format, with constant X = x0
    for i = 1:numel(yq)
        fprintf(fid, '%.6f %.6f %.6f\n', x0, yq(i), zq(i));  % X Y Z
    end

    fclose(fid);
end

% === EXPORT DATA TO EXCEL ===
outFile = 'SectionDD_profiles.xlsx';

% Helper function to round and format to 4 decimals
fmt = @(x) round(x,4);

% Reference polyline
T_ref = table(fmt(y_ref(:)), fmt(z_ref(:)), ...
    'VariableNames', {'Y_ref [m]','Z_ref [m]'});
writetable(T_ref, outFile, 'Sheet','Reference');

% Pre (unique points)
```

```
T_pre_unique = table(fmt(y_u_pre(:)), fmt(z_u_pre(:)), ...
    'VariableNames', {'Y_unique_pre [m]','Z_unique_pre [m]'});
writetable(T_pre_unique, outFile, 'Sheet','Pre_unique');

% Pre (interpolated query with derivatives)
T_pre_interp = table(fmt(yq_pre(:)), fmt(zq_pre(:)), fmt(dz_pre(:)), fmt(
    ddz_pre(:)), ...
    'VariableNames', {'Yq_pre [m]','Zq_pre [m]','dZ_pre','ddZ_pre'});
writetable(T_pre_interp, outFile, 'Sheet','Pre_interp');

% Post (unique points)
T_post_unique = table(fmt(y_u_post(:)), fmt(z_u_post(:)), ...
    'VariableNames', {'Y_unique_post [m]','Z_unique_post [m]'});
writetable(T_post_unique, outFile, 'Sheet','Post_unique');

% Post (interpolated query with derivatives)
T_post_interp = table(fmt(yq_post(:)), fmt(zq_post(:)), fmt(dz_post(:)),
    fmt(ddz_post(:)), ...
    'VariableNames', {'Yq_post [m]','Zq_post [m]','dZ_post','ddZ_post'});
writetable(T_post_interp, outFile, 'Sheet','Post_interp');

fprintf('Excel file written: %s\n', outFile);
```

Listing A.6: Vertical section EE'. MATLAB listing

```
% === PARAMETERS ===
filename_pre  = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
    punti/prova 3/Esportazione txt da CloudCompare/PRE prova.section.txt';
filename_post = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
    punti/prova 3/Esportazione txt da CloudCompare/POST prova.section.txt'
    ;
x0 = 0.180;
delta = 0.0002;
p = 0.9999999;

% === FUNCTION TO LOAD AND PROCESS A SECTION ===
```

```
function [yq, zq, dz, ddz, y_unique, z_unique, sp] = process_section(
    filename, x0, delta, p)
    data = load(filename);
    x = data(:,1); y = data(:,2); z = data(:,3);
    idy = (x > x0 - delta) & (x < x0 + delta);
    y_sec = y(idy);
    z_sec = z(idy);
    [y_sorted, sort_idy] = sort(y_sec);
    z_sorted = z_sec(sort_idy);
    [y_unique, ia] = unique(y_sorted);
    z_unique = z_sorted(ia);
    sp = csaps(y_unique, z_unique, p);
    yq = linspace(min(y_unique), max(y_unique), 240);
    zq = fnval(sp, yq);
    dz = fnval(fnder(sp,1), yq);
    ddz = fnval(fnder(sp,2), yq);
end

% === PROCESS BOTH DATASETS ===
[yq_pre, zq_pre, dz_pre, ddz_pre, y_u_pre, z_u_pre, sp_pre] =
    process_section(filename_pre,  x0, delta, p);
[yq_post, zq_post, dz_post, ddz_post, y_u_post, z_u_post, sp_post] =
    process_section(filename_post, x0, delta, p);

% === REFERENCE POLYLINE ===
y_ref = [0.045, 0.285];
z_ref = [-0.059, -0.059];
dy = diff(y_ref);
dz = diff(z_ref);
slopes = dz ./ dy;

% === PLOTS ===
figure;

% === PLOT 1: INTERPOLATED PROFILES ===
```

```
subplot(3,1,1);
plot(y_ref, z_ref, 'g-', 'LineWidth', 1.5); hold on;
plot(y_u_pre, z_u_pre, 'b.', yq_pre, zq_pre, 'b', 'LineWidth', 1.5);
plot(y_u_post, z_u_post, 'r.', yq_post, zq_post, 'r', 'LineWidth', 1.5);
% color assignment (RGB)
c2  = [0.8500 0.3250 0.0980]; % orange
c9  = [0.8500 0.3250 0.0980]*0.7; % dark orange
% transducer
plot(0.165, -0.0540, 'ko', 'MarkerFaceColor', c2); % Transducer R1
plot(0.165, -0.0588, 'ko', 'MarkerFaceColor', c9); % Transducer R1
xlabel('Y [m]'); ylabel('Z [m]');
title('Section EE'' (R1 transducer): Interpolated Profiles');
legend('Design profile', 'Initial survey points', 'Initial interpolation'
   , ...
       'Final survey points', 'Final interpolation','Initial R1
          transducer', 'Final R1 transducer', ...
       'Location', 'northwest');
grid on;

% === PLOT 2: FIRST DERIVATIVES ===
subplot(3,1,2);
hold on;
y1 = y_ref(1:end-1); y2 = y_ref(2:end);
y_seg = [y1; y2]; x_seg = [slopes; slopes];
hRef = plot(y_seg, x_seg, 'g-', 'LineWidth', 1.5);
hPre  = plot(yq_pre, dz_pre, 'b', 'LineWidth', 1.5);
hPost = plot(yq_post, dz_post, 'r', 'LineWidth', 1.5);
xlabel('Y [m]'); ylabel('dZ/dY');
title('Section EE'' (R1 transducer): First Derivative (Slope)');
legend([hRef(1), hPre, hPost], {'Design slope','Initial slope','Final
   slope'}, 'Location', 'northwest');
grid on;

% === PLOT 3: SECOND DERIVATIVES ===
subplot(3,1,3);
```

```
plot(yq_pre, ddz_pre, 'b', 'LineWidth', 1.5); hold on;
plot(yq_post, ddz_post, 'r', 'LineWidth', 1.5);
for i = 2:length(y_ref)-1
    y_v = y_ref(i);
    plot([y_v y_v], xlim, 'g--', 'LineWidth', 1.5);
end
xlabel('Y [m]'); ylabel('d^2Z/dY^2');
title('Section EE'' (R1 transducer): Second Derivative (Curvature)');
legend('Initial curvature', 'Final curvature', 'Slope variation', '
   Location', 'northwest');

grid on;

% ==== sp POLY EXPORT (uses current MATLAB folder) ====
N = 1000;  % number of sampled points

writePoly('EEsp_pre.poly',  x0,  sp_pre, N);
writePoly('EEsp_post.poly', x0, sp_post, N);

function writePoly(filename, x0, sp, N)
    if nargin < 4, N = 1000; end
    % Sample the spline evenly along its Y range
    yq = linspace(sp.breaks(1), sp.breaks(end), N);
    zq = fnval(sp, yq);

    % Open file for writing
    fid = fopen(filename, 'w');
    assert(fid>0, 'Cannot write file: %s', filename);

    % Write each vertex in X Y Z format, with constant X = x0
    for i = 1:numel(yq)
        fprintf(fid, '%.6f %.6f %.6f\n', x0, yq(i), zq(i));  % X Y Z
    end

    fclose(fid);
```

```
end

% === EXPORT DATA TO EXCEL ===
outFile = 'SectionEE_profiles.xlsx';

% Helper function to round and format to 4 decimals
fmt = @(x) round(x,4);

% Reference polyline
T_ref = table(fmt(y_ref(:)), fmt(z_ref(:)), ...
    'VariableNames', {'Y_ref [m]','Z_ref [m]'});
writetable(T_ref, outFile, 'Sheet','Reference');

% Pre (unique points)
T_pre_unique = table(fmt(y_u_pre(:)), fmt(z_u_pre(:)), ...
    'VariableNames', {'Y_unique_pre [m]','Z_unique_pre [m]'});
writetable(T_pre_unique, outFile, 'Sheet','Pre_unique');

% Pre (interpolated query with derivatives)
T_pre_interp = table(fmt(yq_pre(:)), fmt(zq_pre(:)), fmt(dz_pre(:)), fmt(
   ddz_pre(:)), ...
    'VariableNames', {'Yq_pre [m]','Zq_pre [m]','dZ_pre','ddZ_pre'});
writetable(T_pre_interp, outFile, 'Sheet','Pre_interp');

% Post (unique points)
T_post_unique = table(fmt(y_u_post(:)), fmt(z_u_post(:)), ...
    'VariableNames', {'Y_unique_post [m]','Z_unique_post [m]'});
writetable(T_post_unique, outFile, 'Sheet','Post_unique');

% Post (interpolated query with derivatives)
T_post_interp = table(fmt(yq_post(:)), fmt(zq_post(:)), fmt(dz_post(:)),
   fmt(ddz_post(:)), ...
    'VariableNames', {'Yq_post [m]','Zq_post [m]','dZ_post','ddZ_post'});
writetable(T_post_interp, outFile, 'Sheet','Post_interp');

```

```
fprintf('Excel file written: %s\n', outFile);
```

Listing A.7: Vertical section FF'. MATLAB listing

```
% === PARAMETERS ===
filename_pre  = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di punti/prova 3/Esportazione txt da CloudCompare/PRE prova.section.txt';
filename_post = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di punti/prova 3/Esportazione txt da CloudCompare/POST prova.section.txt';
x0 = 0.235;
delta = 0.0002;
p = 0.9999999;

% === FUNCTION TO LOAD AND PROCESS A SECTION ===
function [yq, zq, dz, ddz, y_unique, z_unique, sp] = process_section(filename, x0, delta, p)
    data = load(filename);
    x = data(:,1); y = data(:,2); z = data(:,3);
    idy = (x > x0 - delta) & (x < x0 + delta);
    y_sec = y(idy);
    z_sec = z(idy);
    [y_sorted, sort_idy] = sort(y_sec);
    z_sorted = z_sec(sort_idy);
    [y_unique, ia] = unique(y_sorted);
    z_unique = z_sorted(ia);
    sp = csaps(y_unique, z_unique, p);
    yq = linspace(min(y_unique), max(y_unique), 240);
    zq = fnval(sp, yq);
    dz = fnval(fnder(sp,1), yq);
    ddz = fnval(fnder(sp,2), yq);
end

% === PROCESS BOTH DATASETS ===
[yq_pre, zq_pre, dz_pre, ddz_pre, y_u_pre, z_u_pre, sp_pre] = process_section(filename_pre,  x0, delta, p);
```

```
[yq_post, zq_post, dz_post, ddz_post, y_u_post, z_u_post, sp_post] = process_section(filename_post, x0, delta, p);

% === REFERENCE POLYLINE ===
y_ref = [0.045, 0.285];
z_ref = [-0.093, -0.093];
dy = diff(y_ref);
dz = diff(z_ref);
slopes = dz ./ dy;

% === PLOTS ===
figure;

% === PLOT 1: INTERPOLATED PROFILES ===
subplot(3,1,1);
plot(y_ref, z_ref, 'g-', 'LineWidth', 1.5); hold on;
plot(y_u_pre, z_u_pre, 'b.', yq_pre, zq_pre, 'b', 'LineWidth', 1.5);
plot(y_u_post, z_u_post, 'r.', yq_post, zq_post, 'r', 'LineWidth', 1.5);
% color assignment (RGB)
c3  = [0.9290 0.6940 0.1250]; % yellow
c10 = [0.9290 0.6940 0.1250]*0.7; % ochre
% transducers
plot(0.165, -0.0815, 'ko', 'MarkerFaceColor', c3); % Initial Transducer R3
plot(0.165, -0.0895, 'ko', 'MarkerFaceColor', c10); % Final Transducer R3
xlabel('Y [m]'); ylabel('Z [m]');
title('Section FF'' (R3 transducer): Interpolated Profiles');
legend('Design profile', 'Initial survey points', 'Initial interpolation', ...
    'Final survey points', 'Final interpolation', 'Final R3 transducer', 'Final R3 transducer', ...
    'Location', 'northwest');
grid on;

% === PLOT 2: FIRST DERIVATIVES ===
```

```
subplot(3,1,2);
hold on;
y1 = y_ref(1:end-1); y2 = y_ref(2:end);
y_seg = [y1; y2]; x_seg = [slopes; slopes];
hRef = plot(y_seg, x_seg, 'g-', 'LineWidth', 1.5);
hPre  = plot(yq_pre, dz_pre, 'b', 'LineWidth', 1.5);
hPost = plot(yq_post, dz_post, 'r', 'LineWidth', 1.5);
xlabel('Y [m]'); ylabel('dZ/dY');
title('Section FF'' (R3 transducer): First Derivative (Slope)');
legend([hRef(1), hPre, hPost], {'Design slope','Initial slope','Final
    slope'}, 'Location', 'northwest');
grid on;

% === PLOT 3: SECOND DERIVATIVES ===
subplot(3,1,3);
plot(yq_pre, ddz_pre, 'b', 'LineWidth', 1.5); hold on;
plot(yq_post, ddz_post, 'r', 'LineWidth', 1.5);
for i = 2:length(y_ref)-1
    y_v = y_ref(i);
    plot([y_v y_v], xlim, 'g--', 'LineWidth', 1.5);
end
xlabel('Y [m]'); ylabel('d^2Z/dY^2');
title('Section FF'' (R3 transducer): Second Derivative (Curvature)');
legend('Initial curvature', 'Final curvature', 'Slope variation', '
    Location', 'northwest');

grid on;

% ==== sp POLY EXPORT (uses current MATLAB folder) ====
N = 1000;  % number of sampled points

writePoly('FFsp_pre.poly',  x0,  sp_pre, N);
writePoly('FFsp_post.poly', x0, sp_post, N);

function writePoly(filename, x0, sp, N)
```

```
92      if nargin < 4, N = 1000; end
93      % Sample the spline evenly along its Y range
94      yq = linspace(sp.breaks(1), sp.breaks(end), N);
95      zq = fnval(sp, yq);
96
97      % Open file for writing
98      fid = fopen(filename, 'w');
99      assert(fid>0, 'Cannot write file: %s', filename);
100
101     % Write each vertex in X Y Z format, with constant X = x0
102     for i = 1:numel(yq)
103         fprintf(fid, '%.6f %.6f %.6f\n', x0, yq(i), zq(i));  % X Y Z
104     end
105
106     fclose(fid);
107 end
108
109 % === EXPORT DATA TO EXCEL ===
110 outFile = 'SectionFF_profiles.xlsx';
111
112 % Helper function to round and format to 4 decimals
113 fmt = @(x) round(x,4);
114
115 % Reference polyline
116 T_ref = table(fmt(y_ref(:)), fmt(z_ref(:)), ...
117     'VariableNames', {'Y_ref [m]','Z_ref [m]'});
118 writetable(T_ref, outFile, 'Sheet','Reference');
119
120 % Pre (unique points)
121 T_pre_unique = table(fmt(y_u_pre(:)), fmt(z_u_pre(:)), ...
122     'VariableNames', {'Y_unique_pre [m]','Z_unique_pre [m]'});
123 writetable(T_pre_unique, outFile, 'Sheet','Pre_unique');
124
125 % Pre (interpolated query with derivatives)
```

```
T_pre_interp = table(fmt(yq_pre(:)), fmt(zq_pre(:)), fmt(dz_pre(:)), fmt(
    ddz_pre(:)), ...
    'VariableNames', {'Yq_pre [m]','Zq_pre [m]','dZ_pre','ddZ_pre'});
writetable(T_pre_interp, outFile, 'Sheet','Pre_interp');

% Post (unique points)
T_post_unique = table(fmt(y_u_post(:)), fmt(z_u_post(:)), ...
    'VariableNames', {'Y_unique_post [m]','Z_unique_post [m]'});
writetable(T_post_unique, outFile, 'Sheet','Post_unique');

% Post (interpolated query with derivatives)
T_post_interp = table(fmt(yq_post(:)), fmt(zq_post(:)), fmt(dz_post(:)),
    fmt(ddz_post(:)), ...
    'VariableNames', {'Yq_post [m]','Zq_post [m]','dZ_post','ddZ_post'});
writetable(T_post_interp, outFile, 'Sheet','Post_interp');

fprintf('Excel file written: %s\n', outFile);
```

Listing A.8: Vertical section GG'. MATLAB listing

```
% === PARAMETERS ===
filename_pre  = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
    punti/prova 3/Esportazione txt da CloudCompare/PRE prova.section.txt';
filename_post = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
    punti/prova 3/Esportazione txt da CloudCompare/POST prova.section.txt'
    ;
x0 = 0.293;
delta = 0.0002;
p = 0.9999999;

% === FUNCTION TO LOAD AND PROCESS A SECTION ===
function [yq, zq, dz, ddz, y_unique, z_unique, sp] = process_section(
    filename, x0, delta, p)
    data = load(filename);
    x = data(:,1); y = data(:,2); z = data(:,3);
    idy = (x > x0 - delta) & (x < x0 + delta);
```

```
    y_sec = y(idy);
    z_sec = z(idy);
    [y_sorted, sort_idy] = sort(y_sec);
    z_sorted = z_sec(sort_idy);
    [y_unique, ia] = unique(y_sorted);
    z_unique = z_sorted(ia);
    sp = csaps(y_unique, z_unique, p);
    yq = linspace(min(y_unique), max(y_unique), 240);
    zq = fnval(sp, yq);
    dz = fnval(fnder(sp,1), yq);
    ddz = fnval(fnder(sp,2), yq);
end

% === PROCESS BOTH DATASETS ===
[yq_pre, zq_pre, dz_pre, ddz_pre, y_u_pre, z_u_pre, sp_pre] =
   process_section(filename_pre,  x0, delta, p);
[yq_post, zq_post, dz_post, ddz_post, y_u_post, z_u_post, sp_post] =
   process_section(filename_post, x0, delta, p);

% === REFERENCE POLYLINE ===
y_ref = [0.045, 0.285];
z_ref = [-0.133, -0.133];
dy = diff(y_ref);
dz = diff(z_ref);
slopes = dz ./ dy;

% === PLOTS ===
figure;

% === PLOT 1: INTERPOLATED PROFILES ===
subplot(3,1,1);
plot(y_ref, z_ref, 'g-', 'LineWidth', 1.5); hold on;
plot(y_u_pre, z_u_pre, 'b.', yq_pre, zq_pre, 'b', 'LineWidth', 1.5);
plot(y_u_post, z_u_post, 'r.', yq_post, zq_post, 'r', 'LineWidth', 1.5);
xlabel('Y [m]'); ylabel('Z [m]');
```

```
title('Section GG'' (R4 transducer): Interpolated Profiles');
legend('Design profile', 'Initial survey points', 'Initial interpolation', ...
       'Final survey points', 'Final interpolation', 'Final R4 transducer', ...
       'Location', 'northwest');
grid on;

% === PLOT 2: FIRST DERIVATIVES ===
subplot(3,1,2);
hold on;
y1 = y_ref(1:end-1); y2 = y_ref(2:end);
y_seg = [y1; y2]; x_seg = [slopes; slopes];
hRef = plot(y_seg, x_seg, 'g-', 'LineWidth', 1.5);
hPre  = plot(yq_pre, dz_pre, 'b', 'LineWidth', 1.5);
hPost = plot(yq_post, dz_post, 'r', 'LineWidth', 1.5);
xlabel('Y [m]'); ylabel('dZ/dY');
title('Section GG'' (R4 transducer): First Derivative (Slope)');
legend([hRef(1), hPre, hPost], {'Design slope','Initial slope','Final slope'}, 'Location', 'northwest');
grid on;

% === PLOT 3: SECOND DERIVATIVES ===
subplot(3,1,3);
plot(yq_pre, ddz_pre, 'b', 'LineWidth', 1.5); hold on;
plot(yq_post, ddz_post, 'r', 'LineWidth', 1.5);
for i = 2:length(y_ref)-1
    y_v = y_ref(i);
    plot([y_v y_v], xlim, 'g--', 'LineWidth', 1.5);
end
xlabel('Y [m]'); ylabel('d^2Z/dY^2');
title('Section GG'' (R4 transducer): Second Derivative (Curvature)');
legend('Initial curvature', 'Final curvature', 'Slope variation', 'Location', 'northwest');
grid on;
```

```
77
78  % ==== sp POLY EXPORT (uses current MATLAB folder) ====
79  N = 1000;  % number of sampled points
80
81  writePoly('GGsp_pre.poly',  x0,  sp_pre, N);
82  writePoly('GGsp_post.poly', x0, sp_post, N);
83
84  function writePoly(filename, x0, sp, N)
85      if nargin < 4, N = 1000; end
86      % Sample the spline evenly along its Y range
87      yq = linspace(sp.breaks(1), sp.breaks(end), N);
88      zq = fnval(sp, yq);
89
90      % Open file for writing
91      fid = fopen(filename, 'w');
92      assert(fid>0, 'Cannot write file: %s', filename);
93
94      % Write each vertex in X Y Z format, with constant X = x0
95      for i = 1:numel(yq)
96          fprintf(fid, '%.6f %.6f %.6f\n', x0, yq(i), zq(i));  % X Y Z
97      end
98
99      fclose(fid);
100 end
101
102 % === EXPORT DATA TO EXCEL ===
103 outFile = 'SectionGG_profiles.xlsx';
104
105 % Helper function to round and format to 4 decimals
106 fmt = @(x) round(x,4);
107
108 % Reference polyline
109 T_ref = table(fmt(y_ref(:)), fmt(z_ref(:)), ...
110     'VariableNames', {'Y_ref [m]','Z_ref [m]'});
111 writetable(T_ref, outFile, 'Sheet','Reference');
```

```
112
113 % Pre (unique points)
114 T_pre_unique = table(fmt(y_u_pre(:)), fmt(z_u_pre(:)), ...
115     'VariableNames', {'Y_unique_pre [m]','Z_unique_pre [m]'});
116 writetable(T_pre_unique, outFile, 'Sheet','Pre_unique');
117
118 % Pre (interpolated query with derivatives)
119 T_pre_interp = table(fmt(yq_pre(:)), fmt(zq_pre(:)), fmt(dz_pre(:)), fmt(
       ddz_pre(:)), ...
120     'VariableNames', {'Yq_pre [m]','Zq_pre [m]','dZ_pre','ddZ_pre'});
121 writetable(T_pre_interp, outFile, 'Sheet','Pre_interp');
122
123 % Post (unique points)
124 T_post_unique = table(fmt(y_u_post(:)), fmt(z_u_post(:)), ...
125     'VariableNames', {'Y_unique_post [m]','Z_unique_post [m]'});
126 writetable(T_post_unique, outFile, 'Sheet','Post_unique');
127
128 % Post (interpolated query with derivatives)
129 T_post_interp = table(fmt(yq_post(:)), fmt(zq_post(:)), fmt(dz_post(:)),
       fmt(ddz_post(:)), ...
130     'VariableNames', {'Yq_post [m]','Zq_post [m]','dZ_post','ddZ_post'});
131 writetable(T_post_interp, outFile, 'Sheet','Post_interp');
132
133 fprintf('Excel file written: %s\n', outFile);
```

Listing A.9: Vertical section HH'. MATLAB listing

```
1 % === PARAMETERS ===
2 filename_pre  = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
      punti/prova 3/Esportazione txt da CloudCompare/PRE prova.section.txt';
3 filename_post = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
      punti/prova 3/Esportazione txt da CloudCompare/POST prova.section.txt'
      ;
4 x0 = 0.348;
5 delta = 0.0002;
6 p = 0.9999999;
```

```
7
8  % === FUNCTION TO LOAD AND PROCESS A SECTION ===
9  function [yq, zq, dz, ddz, y_unique, z_unique, sp] = process_section(
       filename, x0, delta, p)
10     data = load(filename);
11     x = data(:,1); y = data(:,2); z = data(:,3);
12     idy = (x > x0 - delta) & (x < x0 + delta);
13     y_sec = y(idy);
14     z_sec = z(idy);
15     [y_sorted, sort_idy] = sort(y_sec);
16     z_sorted = z_sec(sort_idy);
17     [y_unique, ia] = unique(y_sorted);
18     z_unique = z_sorted(ia);
19     sp = csaps(y_unique, z_unique, p);
20     yq = linspace(min(y_unique), max(y_unique), 240);
21     zq = fnval(sp, yq);
22     dz = fnval(fnder(sp,1), yq);
23     ddz = fnval(fnder(sp,2), yq);
24 end
25
26 % === PROCESS BOTH DATASETS ===
27 [yq_pre, zq_pre, dz_pre, ddz_pre, y_u_pre, z_u_pre, sp_pre] =
       process_section(filename_pre,  x0, delta, p);
28 [yq_post, zq_post, dz_post, ddz_post, y_u_post, z_u_post, sp_post] =
       process_section(filename_post, x0, delta, p);
29
30 % === REFERENCE POLYLINE ===
31 y_ref = [0.045, 0.285];
32 z_ref = [-0.170, -0.170];
33 dy = diff(y_ref);
34 dz = diff(z_ref);
35 slopes = dz ./ dy;
36
37 % === PLOTS ===
38 figure;
```

```
39
40 % === PLOT 1: INTERPOLATED PROFILES ===
41 subplot(3,1,1);
42 plot(y_ref, z_ref, 'g-', 'LineWidth', 1.5); hold on;
43 plot(y_u_pre, z_u_pre, 'b.', yq_pre, zq_pre, 'b', 'LineWidth', 1.5);
44 plot(y_u_post, z_u_post, 'r.', yq_post, zq_post, 'r', 'LineWidth', 1.5);
45 % color assignment (RGB)
46 c5  = [0.4660 0.6740 0.1880]; % green
47 c12 = [0.4660 0.6740 0.1880]*0.7; % olive green
48 % transducers
49 plot(0.165, -0.1563, 'ko', 'MarkerFaceColor', c5); % Initial Transducer
      R5
50 plot(0.165, -0.1645, 'ko', 'MarkerFaceColor', c12); % Final Transducer R5
51 xlabel('Y [m]'); ylabel('Z [m]');
52 title('Section HH'' (R5 transducer): Interpolated Profiles');
53 legend('Design profile', 'Initial survey points', 'Initial interpolation'
      , ...
54        'Final survey points', 'Final interpolation', 'Initial R5
             transducer', 'Final R5 transducer', ...
55        'Location', 'northwest');
56 grid on;
57
58 % === PLOT 2: FIRST DERIVATIVES ===
59 subplot(3,1,2);
60 hold on;
61 y1 = y_ref(1:end-1); y2 = y_ref(2:end);
62 y_seg = [y1; y2]; x_seg = [slopes; slopes];
63 hRef = plot(y_seg, x_seg, 'g-', 'LineWidth', 1.5);
64 hPre  = plot(yq_pre, dz_pre, 'b', 'LineWidth', 1.5);
65 hPost = plot(yq_post, dz_post, 'r', 'LineWidth', 1.5);
66 xlabel('Y [m]'); ylabel('dZ/dY');
67 title('Section HH'' (R5 transducer): First Derivative (Slope)');
68 legend([hRef(1), hPre, hPost], {'Design slope','Initial slope','Final
      slope'}, 'Location', 'northwest');
69 grid on;
```

```
70
71  % === PLOT 3: SECOND DERIVATIVES ===
72  subplot(3,1,3);
73  plot(yq_pre, ddz_pre, 'b', 'LineWidth', 1.5); hold on;
74  plot(yq_post, ddz_post, 'r', 'LineWidth', 1.5);
75  for i = 2:length(y_ref)-1
76      y_v = y_ref(i);
77      plot([y_v y_v], xlim, 'g--', 'LineWidth', 1.5);
78  end
79  xlabel('Y [m]'); ylabel('d^2Z/dY^2');
80  title('Section HH'' (R5 transducer): Second Derivative (Curvature)');
81  legend('Initial curvature', 'Final curvature', 'Slope variation', '
       Location', 'northwest');
82
83  grid on;
84
85  % ==== sp POLY EXPORT (uses current MATLAB folder) ====
86  N = 1000;  % number of sampled points
87
88  writePoly('HHsp_pre.poly',  x0,  sp_pre, N);
89  writePoly('HHsp_post.poly', x0, sp_post, N);
90
91  function writePoly(filename, x0, sp, N)
92      if nargin < 4, N = 1000; end
93      % Sample the spline evenly along its Y range
94      yq = linspace(sp.breaks(1), sp.breaks(end), N);
95      zq = fnval(sp, yq);
96
97      % Open file for writing
98      fid = fopen(filename, 'w');
99      assert(fid>0, 'Cannot write file: %s', filename);
100
101     % Write each vertex in X Y Z format, with constant X = x0
102     for i = 1:numel(yq)
103         fprintf(fid, '%.6f %.6f %.6f\n', x0, yq(i), zq(i));  % X Y Z
```

```
    end

    fclose(fid);
end

% === EXPORT DATA TO EXCEL ===
outFile = 'SectionHH_profiles.xlsx';

% Helper function to round and format to 4 decimals
fmt = @(x) round(x,4);

% Reference polyline
T_ref = table(fmt(y_ref(:)), fmt(z_ref(:)), ...
    'VariableNames', {'Y_ref [m]','Z_ref [m]'});
writetable(T_ref, outFile, 'Sheet','Reference');

% Pre (unique points)
T_pre_unique = table(fmt(y_u_pre(:)), fmt(z_u_pre(:)), ...
    'VariableNames', {'Y_unique_pre [m]','Z_unique_pre [m]'});
writetable(T_pre_unique, outFile, 'Sheet','Pre_unique');

% Pre (interpolated query with derivatives)
T_pre_interp = table(fmt(yq_pre(:)), fmt(zq_pre(:)), fmt(dz_pre(:)), fmt(
   ddz_pre(:)), ...
    'VariableNames', {'Yq_pre [m]','Zq_pre [m]','dZ_pre','ddZ_pre'});
writetable(T_pre_interp, outFile, 'Sheet','Pre_interp');

% Post (unique points)
T_post_unique = table(fmt(y_u_post(:)), fmt(z_u_post(:)), ...
    'VariableNames', {'Y_unique_post [m]','Z_unique_post [m]'});
writetable(T_post_unique, outFile, 'Sheet','Post_unique');

% Post (interpolated query with derivatives)
T_post_interp = table(fmt(yq_post(:)), fmt(zq_post(:)), fmt(dz_post(:)),
   fmt(ddz_post(:)), ...
```

```
    'VariableNames', {'Yq_post [m]','Zq_post [m]','dZ_post','ddZ_post'});
writetable(T_post_interp, outFile, 'Sheet','Post_interp');

fprintf('Excel file written: %s\n', outFile);
```

Listing A.10: Vertical section II'. MATLAB listing

```
% === PARAMETERS ===
filename_pre  = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
   punti/prova 3/Esportazione txt da CloudCompare/PRE prova.section.txt';
filename_post = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
   punti/prova 3/Esportazione txt da CloudCompare/POST prova.section.txt'
   ;
x0 = 0.430;
delta = 0.0002;
p = 0.9999999;

% === FUNCTION TO LOAD AND PROCESS A SECTION ===
function [yq, zq, dz, ddz, y_unique, z_unique, sp] = process_section(
   filename, x0, delta, p)
    data = load(filename);
    x = data(:,1); y = data(:,2); z = data(:,3);
    idy = (x > x0 - delta) & (x < x0 + delta);
    y_sec = y(idy);
    z_sec = z(idy);
    [y_sorted, sort_idy] = sort(y_sec);
    z_sorted = z_sec(sort_idy);
    [y_unique, ia] = unique(y_sorted);
    z_unique = z_sorted(ia);
    sp = csaps(y_unique, z_unique, p);
    yq = linspace(min(y_unique), max(y_unique), 240);
    zq = fnval(sp, yq);
    dz = fnval(fnder(sp,1), yq);
    ddz = fnval(fnder(sp,2), yq);
end

```

```
% === PROCESS BOTH DATASETS ===
[yq_pre, zq_pre, dz_pre, ddz_pre, y_u_pre, z_u_pre, sp_pre] =
    process_section(filename_pre,  x0, delta, p);
[yq_post, zq_post, dz_post, ddz_post, y_u_post, z_u_post, sp_post] =
    process_section(filename_post, x0, delta, p);

% === REFERENCE POLYLINE ===
y_ref = [0.045, 0.285];
z_ref = [-0.199, -0.199];
dy = diff(y_ref);
dz = diff(z_ref);
slopes = dz ./ dy;

% === PLOTS ===
figure;

% === PLOT 1: INTERPOLATED PROFILES ===
subplot(3,1,1);
plot(y_ref, z_ref, 'g-', 'LineWidth', 1.5); hold on;
plot(y_u_pre, z_u_pre, 'b.', yq_pre, zq_pre, 'b', 'LineWidth', 1.5);
plot(y_u_post, z_u_post, 'r.', yq_post, zq_post, 'r', 'LineWidth', 1.5);
% color assignment (RGB)
c6  = [0.3010 0.7450 0.9330]; % light blue
c13 = [0.3010 0.7450 0.9330]*0.7; % dark cyan
% transducers
plot(0.165, -0.1955, 'ko', 'MarkerFaceColor', c6); % Initial Transducer
    L6
plot(0.165, -0.1983, 'ko', 'MarkerFaceColor', c13); % Final Transducer L6
xlabel('Y [m]'); ylabel('Z [m]');
title('Section II'' (L6 transducer): Interpolated Profiles');
legend('Design profile', 'Initial survey points', 'Initial interpolation'
    , ...
       'Final survey points', 'Final interpolation', 'Initial L6
          transducer', 'Final L6 transducer', ...
       'Location', 'northwest');
```

```
grid on;

% === PLOT 2: FIRST DERIVATIVES ===
subplot(3,1,2);
hold on;
y1 = y_ref(1:end-1); y2 = y_ref(2:end);
y_seg = [y1; y2]; x_seg = [slopes; slopes];
hRef = plot(y_seg, x_seg, 'g-', 'LineWidth', 1.5);
hPre  = plot(yq_pre, dz_pre, 'b', 'LineWidth', 1.5);
hPost = plot(yq_post, dz_post, 'r', 'LineWidth', 1.5);
xlabel('Y [m]'); ylabel('dZ/dY');
title('Section II'' (L6 transducer): First Derivative (Slope)');
legend([hRef(1), hPre, hPost], {'Design slope','Initial slope','Final
   slope'}, 'Location', 'northwest');
grid on;

% === PLOT 3: SECOND DERIVATIVES ===
subplot(3,1,3);
plot(yq_pre, ddz_pre, 'b', 'LineWidth', 1.5); hold on;
plot(yq_post, ddz_post, 'r', 'LineWidth', 1.5);
for i = 2:length(y_ref)-1
    y_v = y_ref(i);
    plot([y_v y_v], xlim, 'g--', 'LineWidth', 1.5);
end
xlabel('Y [m]'); ylabel('d^2Z/dY^2');
title('Section II'' (L6 transducer): Second Derivative (Curvature)');
legend('Initial curvature', 'Final curvature', 'Slope variation', '
   Location', 'northwest');

grid on;

% ==== sp POLY EXPORT (uses current MATLAB folder) ====
N = 1000;  % number of sampled points

writePoly('IIsp_pre.poly',  x0,  sp_pre, N);
```

```
writePoly('IIsp_post.poly', x0, sp_post, N);

function writePoly(filename, x0, sp, N)
    if nargin < 4, N = 1000; end
    % Sample the spline evenly along its Y range
    yq = linspace(sp.breaks(1), sp.breaks(end), N);
    zq = fnval(sp, yq);

    % Open file for writing
    fid = fopen(filename, 'w');
    assert(fid>0, 'Cannot write file: %s', filename);

    % Write each vertex in X Y Z format, with constant X = x0
    for i = 1:numel(yq)
        fprintf(fid, '%.6f %.6f %.6f\n', x0, yq(i), zq(i));  % X Y Z
    end

    fclose(fid);
end

% === EXPORT DATA TO EXCEL ===
outFile = 'SectionII_profiles.xlsx';

% Helper function to round and format to 4 decimals
fmt = @(x) round(x,4);

% Reference polyline
T_ref = table(fmt(y_ref(:)), fmt(z_ref(:)), ...
    'VariableNames', {'Y_ref [m]','Z_ref [m]'});
writetable(T_ref, outFile, 'Sheet','Reference');

% Pre (unique points)
T_pre_unique = table(fmt(y_u_pre(:)), fmt(z_u_pre(:)), ...
    'VariableNames', {'Y_unique_pre [m]','Z_unique_pre [m]'});
writetable(T_pre_unique, outFile, 'Sheet','Pre_unique');
```

```
124
125 % Pre (interpolated query with derivatives)
126 T_pre_interp = table(fmt(yq_pre(:)), fmt(zq_pre(:)), fmt(dz_pre(:)), fmt(
       ddz_pre(:)), ...
127     'VariableNames', {'Yq_pre [m]','Zq_pre [m]','dZ_pre','ddZ_pre'});
128 writetable(T_pre_interp, outFile, 'Sheet','Pre_interp');
129
130 % Post (unique points)
131 T_post_unique = table(fmt(y_u_post(:)), fmt(z_u_post(:)), ...
132     'VariableNames', {'Y_unique_post [m]','Z_unique_post [m]'});
133 writetable(T_post_unique, outFile, 'Sheet','Post_unique');
134
135 % Post (interpolated query with derivatives)
136 T_post_interp = table(fmt(yq_post(:)), fmt(zq_post(:)), fmt(dz_post(:)),
       fmt(ddz_post(:)), ...
137     'VariableNames', {'Yq_post [m]','Zq_post [m]','dZ_post','ddZ_post'});
138 writetable(T_post_interp, outFile, 'Sheet','Post_interp');
139
140 fprintf('Excel file written: %s\n', outFile);
```

Listing A.11: Vertical section JJ'. MATLAB listing

```
1 % === PARAMETERS ===
2 filename_pre  = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
      punti/prova 3/Esportazione txt da CloudCompare/PRE prova.section.txt';
3 filename_post = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di
      punti/prova 3/Esportazione txt da CloudCompare/POST prova.section.txt'
      ;
4 x0 = 0.660;
5 delta = 0.0002;
6 p = 0.9999999;
7
8 % === FUNCTION TO LOAD AND PROCESS A SECTION ===
9 function [yq, zq, dz, ddz, y_unique, z_unique, sp] = process_section(
      filename, x0, delta, p)
10     data = load(filename);
```

```
    x = data(:,1); y = data(:,2); z = data(:,3);
    idy = (x > x0 - delta) & (x < x0 + delta);
    y_sec = y(idy);
    z_sec = z(idy);
    [y_sorted, sort_idy] = sort(y_sec);
    z_sorted = z_sec(sort_idy);
    [y_unique, ia] = unique(y_sorted);
    z_unique = z_sorted(ia);
    sp = csaps(y_unique, z_unique, p);
    yq = linspace(min(y_unique), max(y_unique), 240);
    zq = fnval(sp, yq);
    dz = fnval(fnder(sp,1), yq);
    ddz = fnval(fnder(sp,2), yq);
end

% === PROCESS BOTH DATASETS ===
[yq_pre, zq_pre, dz_pre, ddz_pre, y_u_pre, z_u_pre, sp_pre] =
   process_section(filename_pre,  x0, delta, p);
[yq_post, zq_post, dz_post, ddz_post, y_u_post, z_u_post, sp_post] =
   process_section(filename_post, x0, delta, p);

% === REFERENCE POLYLINE ===
y_ref = [0.045, 0.285];
z_ref = [-0.199, -0.199];
dy = diff(y_ref);
dz = diff(z_ref);
slopes = dz ./ dy;

% === PLOTS ===
figure;

% === PLOT 1: INTERPOLATED PROFILES ===
subplot(3,1,1);
plot(y_ref, z_ref, 'g-', 'LineWidth', 1.5); hold on;
plot(y_u_pre, z_u_pre, 'b.', yq_pre, zq_pre, 'b', 'LineWidth', 1.5);
```

```
plot(y_u_post, z_u_post, 'r.', yq_post, zq_post, 'r', 'LineWidth', 1.5);
xlabel('Y [m]'); ylabel('Z [m]');
title('Section JJ'' (L7 transducer): Interpolated Profiles');
legend('Design profile', 'Initial survey points', 'Initial interpolation', ...
    'Final survey points', 'Final interpolation', ...
    'Location', 'northwest');
grid on;

% === PLOT 2: FIRST DERIVATIVES ===
subplot(3,1,2);
hold on;
y1 = y_ref(1:end-1); y2 = y_ref(2:end);
y_seg = [y1; y2]; x_seg = [slopes; slopes];
hRef = plot(y_seg, x_seg, 'g-', 'LineWidth', 1.5);
hPre  = plot(yq_pre, dz_pre, 'b', 'LineWidth', 1.5);
hPost = plot(yq_post, dz_post, 'r', 'LineWidth', 1.5);
xlabel('Y [m]'); ylabel('dZ/dY');
title('Section JJ'' (L7 transducer): First Derivative (Slope)');
legend([hRef(1), hPre, hPost], {'Design slope','Initial slope','Final slope'}, 'Location', 'northwest');
grid on;

% === PLOT 3: SECOND DERIVATIVES ===
subplot(3,1,3);
plot(yq_pre, ddz_pre, 'b', 'LineWidth', 1.5); hold on;
plot(yq_post, ddz_post, 'r', 'LineWidth', 1.5);
for i = 2:length(y_ref)-1
    y_v = y_ref(i);
    plot([y_v y_v], xlim, 'g--', 'LineWidth', 1.5);
end
xlabel('Y [m]'); ylabel('d^2Z/dY^2');
title('Section JJ'' (L7 transducer): Second Derivative (Curvature)');
legend('Initial curvature', 'Final curvature', 'Slope variation', 'Location', 'northwest');
```

```
76
77  grid on;
78
79  % ==== sp POLY EXPORT (uses current MATLAB folder) ====
80  N = 1000;  % number of sampled points
81
82  writePoly('JJsp_pre.poly',  x0,  sp_pre, N);
83  writePoly('JJsp_post.poly', x0, sp_post, N);
84
85  function writePoly(filename, x0, sp, N)
86      if nargin < 4, N = 1000; end
87      % Sample the spline evenly along its Y range
88      yq = linspace(sp.breaks(1), sp.breaks(end), N);
89      zq = fnval(sp, yq);
90
91      % Open file for writing
92      fid = fopen(filename, 'w');
93      assert(fid>0, 'Cannot write file: %s', filename);
94
95      % Write each vertex in X Y Z format, with constant X = x0
96      for i = 1:numel(yq)
97          fprintf(fid, '%.6f %.6f %.6f\n', x0, yq(i), zq(i));  % X Y Z
98      end
99
100     fclose(fid);
101 end
102
103 % === EXPORT DATA TO EXCEL ===
104 outFile = 'SectionJJ_profiles.xlsx';
105
106 % Helper function to round and format to 4 decimals
107 fmt = @(x) round(x,4);
108
109 % Reference polyline
110 T_ref = table(fmt(y_ref(:)), fmt(z_ref(:)), ...
```

```
    'VariableNames', {'Y_ref [m]','Z_ref [m]'});
writetable(T_ref, outFile, 'Sheet','Reference');

% Pre (unique points)
T_pre_unique = table(fmt(y_u_pre(:)), fmt(z_u_pre(:)), ...
    'VariableNames', {'Y_unique_pre [m]','Z_unique_pre [m]'});
writetable(T_pre_unique, outFile, 'Sheet','Pre_unique');

% Pre (interpolated query with derivatives)
T_pre_interp = table(fmt(yq_pre(:)), fmt(zq_pre(:)), fmt(dz_pre(:)), fmt(
   ddz_pre(:)), ...
    'VariableNames', {'Yq_pre [m]','Zq_pre [m]','dZ_pre','ddZ_pre'});
writetable(T_pre_interp, outFile, 'Sheet','Pre_interp');

% Post (unique points)
T_post_unique = table(fmt(y_u_post(:)), fmt(z_u_post(:)), ...
    'VariableNames', {'Y_unique_post [m]','Z_unique_post [m]'});
writetable(T_post_unique, outFile, 'Sheet','Post_unique');

% Post (interpolated query with derivatives)
T_post_interp = table(fmt(yq_post(:)), fmt(zq_post(:)), fmt(dz_post(:)),
   fmt(ddz_post(:)), ...
    'VariableNames', {'Yq_post [m]','Zq_post [m]','dZ_post','ddZ_post'});
writetable(T_post_interp, outFile, 'Sheet','Post_interp');

fprintf('Excel file written: %s\n', outFile);
```

Listing A.12: Python scripting for MATLAB vertical section compiling.

```
"""
Wizard PyAutoGUI per comporre lo script MATLAB di sezione AA.
- Prompt/alert/confirm con pyautogui
- Dialog di apertura/salvataggio file e scelta cartelle con tkinter (se
   disponibile)
- Salvataggio forzato del file MATLAB in D:\\aaa\\SectionAA_script.m

```

```
Requisiti: pip install pyautogui
Consigliato su Windows: esegui Python "nativo", non WSL.
"""
import pyautogui as pg
import sys
import os
import datetime
from string import Template

# Dialog via tkinter (se disponibile)
try:
    import tkinter as tk
    from tkinter import filedialog
except Exception:
    tk = None

pg.PAUSE = 0.2
pg.FAILSAFE = True

# ---------- Utility GUI ----------
def pick_file(title="Select file", initialdir=os.path.expanduser("~")):
    if tk is None:
        pg.alert("tkinter non disponibile: inserisci il percorso a mano
           nel prompt successivo.", "Info")
        path = pg.prompt(text="Percorso completo del file:", title=title,
            default="")
        return path or ""
    root = tk.Tk(); root.withdraw()
    path = filedialog.askopenfilename(title=title, initialdir=initialdir)
    root.destroy()
    return path or ""

def pick_dir(title="Seleziona cartella di output", initialdir=os.path.
   expanduser("~")):
    if tk is None:
```

```
        d = pg.prompt(text="Percorso cartella di output (es. D:/output):", title=title, default="")
        return d or ""
    root = tk.Tk(); root.withdraw()
    d = filedialog.askdirectory(title=title, initialdir=initialdir)
    root.destroy()
    return d or ""

def ask_float(msg, default):
    while True:
        s = pg.prompt(text=msg, title="Parametro", default=str(default))
        if s is None:
            if pg.confirm("Annullare il wizard?", buttons=["Yes","No"]) == "Yes":
                sys.exit(0)
            else:
                continue
        s = s.strip().replace(",", ".")
        try:
            return float(s)
        except ValueError:
            pg.alert("Valore non valido. Inserisci un numero.", "Errore")

def ask_int(msg, default):
    while True:
        s = pg.prompt(text=msg, title="Parametro", default=str(default))
        if s is None:
            if pg.confirm("Annullare il wizard?", buttons=["Yes","No"]) == "Yes":
                sys.exit(0)
            else:
                continue
        try:
            return int(s)
        except ValueError:
```

```
            pg.alert("Valore non valido. Inserisci un intero.", "Errore")

def yesno(msg, default_yes=True):
    ans = pg.confirm(msg, buttons=["Yes","No"])
    return ans == "Yes"

# ---------- Inizio wizard ----------
pg.alert("Benvenuto! Questo wizard crea un file MATLAB .m per il rilievo di sezione.", "MATLAB Wizard")

# File Initial/Final
pre_path  = pick_file("Seleziona file INITIAL (PRE)")
while not pre_path or not os.path.exists(pre_path):
    if not yesno("File INITIAL non valido. Riprovare?"):
        sys.exit(0)
    pre_path = pick_file("Seleziona file INITIAL (PRE)")

post_path = pick_file("Seleziona file FINAL (POST)")
while not post_path or not os.path.exists(post_path):
    if not yesno("File FINAL non valido. Riprovare?"):
        sys.exit(0)
    post_path = pick_file("Seleziona file FINAL (POST)")

# Parametri principali
y0     = ask_float("y0 (quota sezione centrale) [es. 0.165]:", 0.165)
delta  = ask_float("delta (semi-spessore fascia in Y) [es. 0.0002]:", 0.0002)
p      = ask_float("Parametro smoothing spline p (0..1, 1 ~ interpolante) [es. 0.9999999]:", 0.9999999)
n_poly = ask_int("N punti per export .poly [es. 1000]:", 1000)

# Polilinea di riferimento
use_default_ref = yesno("Usare polilinea di riferimento predefinita [0.045 0.185 0.390 0.785 / -0.059 -0.059 -0.199 -0.199] ?", True)
if use_default_ref:
```

```
    x_ref = [0.045, 0.185, 0.390, 0.785]
    z_ref = [-0.059, -0.059, -0.199, -0.199]
else:
    xr = pg.prompt("Inserisci X_ref separati da spazi (es: 0.045 0.185 0.390 0.785):", default="0.045 0.185 0.390 0.785") or ""
    zr = pg.prompt("Inserisci Z_ref separati da spazi (es: -0.059 -0.059 -0.199 -0.199):", default="-0.059 -0.059 -0.199 -0.199") or ""
    try:
        x_ref = [float(s.replace(",",".")) for s in xr.split()]
        z_ref = [float(s.replace(",",".")) for s in zr.split()]
        if len(x_ref) != len(z_ref) or len(x_ref) < 2:
            raise ValueError
    except Exception:
        pg.alert("Valori non validi: uso la polilinea di default.", "Avviso")
        x_ref = [0.045, 0.185, 0.390, 0.785]
        z_ref = [-0.059, -0.059, -0.199, -0.199]

# Nomi di base
excel_name = pg.prompt("Nome file Excel di export:", default="SectionAA_profiles.xlsx") or "SectionAA_profiles.xlsx"
poly_pre   = pg.prompt("Nome file .poly INITIAL:", default="AAsp_pre.poly") or "AAsp_pre.poly"
poly_post  = pg.prompt("Nome file .poly FINAL:",  default="AAsp_post.poly") or "AAsp_post.poly"

# Cartella di output (assoluta) per Excel e .poly
out_dir = pick_dir("Seleziona cartella per Excel e .poly (usata dal codice MATLAB)")
while not out_dir:
    if yesno("Nessuna cartella selezionata. Vuoi inserirne una manualmente?"):
        out_dir = pg.prompt("Percorso cartella di output (es. D:/output):", default="D:/output") or ""
    else:
```

```
        break

# Normalizzazione e composizione percorsi MATLAB-friendly (forward slash)
def mpath_join(folder, name):
    if not folder:
        return name  # rimane relativo
    return os.path.join(folder, name).replace("\\", "/")

excel_path = mpath_join(out_dir, excel_name)
poly_pre_p = mpath_join(out_dir, poly_pre)
poly_post_p= mpath_join(out_dir, poly_post)

# Vettori in sintassi MATLAB
def vec_repr(v):
    parts = []
    for val in v:
        if abs(val) >= 1e-6:
            s = ("{:.3f}".format(val)).rstrip("0").rstrip(".")
        else:
            s = "{:.6f}".format(val)
        parts.append(s)
    return "[" + " ".join(parts) + "]"

x_ref_repr = vec_repr(x_ref)
z_ref_repr = vec_repr(z_ref)

# Template MATLAB (usa $variabile per evitare conflitti con { })
tmpl = Template(u"""% Auto-generated by PyAutoGUI wizard on $date
% === PARAMETERS ===
filename_pre  = '$pre';
filename_post = '$post';
y0    = $y0;
delta = $delta;
p     = $p;

```

```
% === FUNCTION TO LOAD AND PROCESS A SECTION ===
function [xq, zq, dz, ddz, x_unique, z_unique, sp] = process_section(filename, y0, delta, p)
    try, data = load(filename); catch, data = readmatrix(filename); end
    x = data(:,1); y = data(:,2); z = data(:,3);
    idx = (y > y0 - delta) & (y < y0 + delta);
    x_sec = x(idx); z_sec = z(idx);
    if numel(x_sec) < 5, error('Not enough points in section window: %d', numel(x_sec)); end
    [x_sorted, sort_idx] = sort(x_sec);
    z_sorted = z_sec(sort_idx);
    [x_unique, ia] = unique(x_sorted);
    z_unique = z_sorted(ia);
    sp = csaps(x_unique, z_unique, p);
    xq = linspace(min(x_unique), max(x_unique), 740);
    zq = fnval(sp, xq);
    dz  = fnval(fnder(sp,1), xq);
    ddz = fnval(fnder(sp,2), xq);
end

% === PROCESS BOTH DATASETS ===
[xq_pre,  zq_pre,  dz_pre,  ddz_pre,  x_u_pre,  z_u_pre,  sp_pre ] = process_section(filename_pre,  y0, delta, p);
[xq_post, zq_post, dz_post, ddz_post, x_u_post, z_u_post, sp_post] = process_section(filename_post, y0, delta, p);

% === REFERENCE POLYLINE ===
x_ref = $x_ref;
z_ref = $z_ref;
dx = diff(x_ref); dzr = diff(z_ref); slopes = dzr ./ dx;

% === PLOTS ===
figure('Color','w');

% === PLOT 1: INTERPOLATED PROFILES ===
```

```
subplot(3,1,1); hold on;
plot(x_ref, z_ref, 'g-', 'LineWidth', 1.5);
plot(x_u_pre,  z_u_pre,  'b.', xq_pre,  zq_pre,  'b', 'LineWidth', 1.5);
plot(x_u_post, z_u_post, 'r.', xq_post, zq_post, 'r', 'LineWidth', 1.5);
xlabel('X [m]'); ylabel('Z [m]');
title('Section AA'': Interpolated Profiles');
legend('Design profile','Initial survey points','Initial interpolation','
   Final survey points','Final interpolation', ...
       'NumColumns',3,'Location','northeast'); grid on;

% === PLOT 2: FIRST DERIVATIVES ===
subplot(3,1,2); hold on;
x1 = x_ref(1:end-1); x2 = x_ref(2:end);
for k = 1:numel(x1)
    plot([x1(k) x2(k)], [slopes(k) slopes(k)], 'g-', 'LineWidth', 1.5);
end
plot(xq_pre, dz_pre, 'b', 'LineWidth', 1.5);
plot(xq_post, dz_post,'r', 'LineWidth', 1.5);
xlabel('X [m]'); ylabel('dZ/dX');
title('Section AA'': First Derivative (Slope)');
legend('Design slope','Initial slope','Final slope', 'Location','
   southeast'); grid on;

% === PLOT 3: SECOND DERIVATIVES ===
subplot(3,1,3); hold on;
plot(xq_pre,  ddz_pre,  'b', 'LineWidth', 1.5);
plot(xq_post, ddz_post, 'r', 'LineWidth', 1.5);
if exist('xline','file') == 2
    for i = 2:length(x_ref)-1, xline(x_ref(i),'g--','LineWidth',1.2); end
else
    yl = ylim; for i = 2:length(x_ref)-1, plot([x_ref(i) x_ref(i)], yl, '
       g--','LineWidth',1.2); end
end
xlabel('X [m]'); ylabel('d^2Z/dX^2');
title('Section AA'': Second Derivative (Curvature)');
```

```
226 legend('Initial curvature','Final curvature', 'Location','northeast');
    grid on;
227
228 % ==== sp POLY EXPORT ====
229 N = $N;
230 writePoly('$poly_pre',  sp_pre,  y0, N);
231 writePoly('$poly_post', sp_post, y0, N);
232
233 function writePoly(filename, sp, y0, N)
234     if nargin < 4, N = 1000; end
235     xq = linspace(sp.breaks(1), sp.breaks(end), N);
236     zq = fnval(sp, xq);
237     fid = fopen(filename, 'w', 'n', 'UTF-8');
238     assert(fid>0, 'Cannot write file: %s', filename);
239     for i = 1:numel(xq)
240         fprintf(fid, '%.6f %.6f %.6f\\n', xq(i), y0, zq(i));
241     end
242     fclose(fid);
243 end
244
245 % === EXPORT DATA TO EXCEL ===
246 outFile = '$excel';
247 fmt = @(x) round(x,4);
248 T_ref = table(fmt(x_ref(:)), fmt(z_ref(:)), 'VariableNames', {'X_ref [m
    ]','Z_ref [m]'}); writetable(T_ref, outFile, 'Sheet','Reference');
249 T_ini_unique = table(fmt(x_u_pre(:)), fmt(z_u_pre(:)), 'VariableNames',
    {'X_unique_initial [m]','Z_unique_initial [m]'}); writetable(
    T_ini_unique, outFile, 'Sheet','Initial_unique');
250 T_ini_interp = table(fmt(xq_pre(:)), fmt(zq_pre(:)), fmt(dz_pre(:)), fmt(
    ddz_pre(:)), 'VariableNames', {'Xq_initial [m]','Zq_initial [m]','
    dZ_initial','ddZ_initial'}); writetable(T_ini_interp, outFile, 'Sheet
    ','Initial_interp');
251 T_fin_unique = table(fmt(x_u_post(:)), fmt(z_u_post(:)), 'VariableNames',
     {'X_unique_final [m]','Z_unique_final [m]'}); writetable(T_fin_unique
    , outFile, 'Sheet','Final_unique');
```

```
T_fin_interp = table(fmt(xq_post(:)), fmt(zq_post(:)), fmt(dz_post(:)),
    fmt(ddz_post(:)), 'VariableNames', {'Xq_final [m]','Zq_final [m]','
    dZ_final','ddZ_final'}); writetable(T_fin_interp, outFile, 'Sheet','
    Final_interp');
fprintf('Excel file written: %s\\n', outFile);
""")

# Sostituzione variabili nel template
matlab_code = tmpl.substitute(
    date=datetime.datetime.now().strftime("%Y-%m-%d %H:%M"),
    pre=pre_path.replace("\\","/"),
    post=post_path.replace("\\","/"),
    y0=y0, delta=delta, p=p, N=n_poly,
    x_ref=x_ref_repr, z_ref=z_ref_repr,
    poly_pre=poly_pre_p, poly_post=poly_post_p,
    excel=excel_path
)

# ---------- Salvataggio forzato in D:\aaa\SectionAA_script.m ----------
fixed_out_dir  = r"D:\aaa"
fixed_out_file = r"D:\aaa\SectionAA_script.m"

try:
    if not os.path.isdir(fixed_out_dir):
        os.makedirs(fixed_out_dir, exist_ok=True)
except Exception:
    # Fallback per Python vecchi
    if not os.path.isdir(fixed_out_dir):
        os.makedirs(fixed_out_dir)

try:
    with open(fixed_out_file, "w", encoding="utf-8") as f:
        f.write(matlab_code)
    pg.alert("Creato file MATLAB:\n{}".format(fixed_out_file), "Fatto!")
except Exception:
```

```
    # Fallback se l'arg encoding non e' supportato
    with open(fixed_out_file, "wb") as f:
        f.write(matlab_code.encode("utf-8"))
    pg.alert("Creato file MATLAB (fallback encoding):\n{}".format(
       fixed_out_file), "Fatto!")
```

Listing A.13: 2D and 3D representations of functions and derivatives. MATLAB listing

```
%% === INPUT ===
file_pre  = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di punti/
   prova 3/Esportazione txt da CloudCompare/PRE prova.section.txt';   %
   columns: x y z
file_post = 'D:/1_Projects/20240723_Geotecnica e rilievo/nuvole di punti/
   prova 3/Esportazione txt da CloudCompare/POST prova.section.txt';  %
   columns: x y z

% Direction of derivation: choose ONE of the two options
dir_as_angle_deg = 0;           % e.g. 20 deg from X axis (CCW towards Y
   +)
dir_as_vector    = [];          % or [ux, uy]; leave [] if using angle

% Grid resolution
grid_dx = 0.002;                % step in X [m]
grid_dy = 0.002;                % step in Y [m]
grid_padding = 0.00;            % margin beyond bounding box [m]

interp_method = 'natural';      % 'linear' | 'natural' | 'nearest'

% (optional) restrict the region of interest
xlim_roi = []; % [xmin xmax] or []
ylim_roi = []; % [ymin ymax] or []

%% === LOAD DATA ===
Ppre  = load(file_pre);   xpre = Ppre(:,1);  ypre = Ppre(:,2);  zpre =
   Ppre(:,3);
```

```
Ppost = load(file_post);  xpost= Ppost(:,1); ypost= Ppost(:,2); zpost=
    Ppost(:,3);

%% === UNIT DIRECTION VECTOR u = [ux, uy] ===
if ~isempty(dir_as_vector)
    u = dir_as_vector(:).';
    u = u / norm(u);
else
    th = deg2rad(dir_as_angle_deg);
    u = [cos(th), sin(th)];
end
ux = u(1); uy = u(2);

%% === COMMON GRID ===
xmin = max([min(xpre),  min(xpost)]) - grid_padding;
xmax = min([max(xpre),  max(xpost)]) + grid_padding;
ymin = max([min(ypre),  min(ypost)]) - grid_padding;
ymax = min([max(ypre),  max(ypost)]) + grid_padding;

if ~isempty(xlim_roi), xmin = max(xmin, xlim_roi(1)); xmax = min(xmax,
    xlim_roi(2)); end
if ~isempty(ylim_roi), ymin = max(ymin, ylim_roi(1)); ymax = min(ymax,
    ylim_roi(2)); end

xv = xmin:grid_dx:xmax;
yv = ymin:grid_dy:ymax;
[X, Y] = meshgrid(xv, yv);

%% === INTERPOLATE TO GRID ===
Fpre  = scatteredInterpolant(xpre,  ypre,  zpre,  interp_method, 'none');
Fpost = scatteredInterpolant(xpost, ypost, zpost, interp_method, 'none');
Zpre  = Fpre(X, Y);
Zpost = Fpost(X, Y);

% Validity mask (only where both surfaces are defined)
```

```
M = isfinite(Zpre) & isfinite(Zpost);

%% === FIRST AND SECOND DERIVATIVES (PRE) ===
[Zx_pre, Zy_pre] = gradient(Zpre, grid_dx, grid_dy);  % z_x, z_y
[Zxx_pre, Zxy_pre] = gradient(Zx_pre, grid_dx, grid_dy);
[Zyx_pre, Zyy_pre] = gradient(Zy_pre, grid_dx, grid_dy);
Zxy_pre = 0.5*(Zxy_pre + Zyx_pre); % symmetrize mixed derivative
D1_pre = ux.*Zx_pre + uy.*Zy_pre;
D2_pre = ux.*(ux.*Zxx_pre + uy.*Zxy_pre) + ...
         uy.*(ux.*Zxy_pre + uy.*Zyy_pre);

%% === FIRST AND SECOND DERIVATIVES (POST) ===
[Zx_post, Zy_post] = gradient(Zpost, grid_dx, grid_dy);
[Zxx_post, Zxy_post] = gradient(Zx_post, grid_dx, grid_dy);
[Zyx_post, Zyy_post] = gradient(Zy_post, grid_dx, grid_dy);
Zxy_post = 0.5*(Zxy_post + Zyx_post);
D1_post = ux.*Zx_post + uy.*Zy_post;
D2_post = ux.*(ux.*Zxx_post + uy.*Zxy_post) + ...
          uy.*(ux.*Zxy_post + uy.*Zyy_post);

%% === DIFFERENCES POST - PRE ===
dZ  = Zpost - Zpre;
dD1 = D1_post - D1_pre;
dD2 = D2_post - D2_pre;

%% === 2D PLOTS ===
figure('Name','Directional derivatives');
tiledlayout(3,3,'Padding','compact','TileSpacing','compact');

%1
nexttile;
imagesc(xv, yv, Zpre);
set(gca,'YDir','normal');
cb = colorbar('eastoutside');
ylabel(cb, '[m]','Interpreter','none');
```

```
89  title('z initial');
90  xlabel('x [m]'); ylabel('y [m]'); axis image;
91
92  %2
93  nexttile;
94  imagesc(xv, yv, Zpost);
95  set(gca,'YDir','normal');
96  cb = colorbar('eastoutside');
97  ylabel(cb, '[m]','Interpreter','none');
98  title('z final');
99  xlabel('x [m]'); ylabel('y [m]'); axis image;
100
101 %3
102 nexttile;
103 imagesc(xv, yv, dZ);
104 set(gca,'YDir','normal');
105 cb = colorbar('eastoutside');
106 ylabel(cb, '[m]','Interpreter','none');
107 title('\Deltaz');
108 xlabel('x [m]'); ylabel('y [m]'); axis image;
109
110 %4
111 nexttile;
112 imagesc(xv, yv, D1_pre, [-1 1]);
113 set(gca,'YDir','normal');
114 cb = colorbar('eastoutside');
115 ylabel(cb, '','Interpreter','none');
116 title(sprintf('$\\frac{\\partial z}{\\partial u}$ initial (u=[%.2f; %.2f
       ])', ux, uy),'Interpreter','latex');
117 xlabel('x [m]'); ylabel('y [m]'); axis image;
118
119 %5
120 nexttile;
121 imagesc(xv, yv, D1_post, [-1 1]);
122 set(gca,'YDir','normal');
```

```
cb = colorbar('eastoutside');
ylabel(cb, '','Interpreter','none');
title(sprintf('$\\frac{\\partial z}{\\partial u}$ final (u=[%.2f; %.2f])'
   , ux, uy),'Interpreter','latex');
xlabel('x [m]'); ylabel('y [m]'); axis image;

%6
nexttile;
imagesc(xv, yv, dD1, [-2 2]);
set(gca,'YDir','normal');
cb = colorbar('eastoutside');
title('$\Delta \frac{\partial z}{\partial u}$','Interpreter','latex');
ylabel(cb, '','Interpreter','none');
xlabel('x [m]'); ylabel('y [m]'); axis image;

%7
nexttile;
imagesc(xv, yv, D2_pre, [-100 100]);
set(gca,'YDir','normal');
cb = colorbar('eastoutside');
cb.Label.Interpreter = 'latex';
cb.Label.String = '$[m^{-1}]$';
ylabel('[m^{-1}]');
title(sprintf('$\\frac{\\partial^2 z}{\\partial u^2}$ initial (u=[%.2f;
   %.2f])', ux, uy),'Interpreter','latex');
xlabel('x [m]'); ylabel('y [m]'); axis image;

%8
nexttile;
imagesc(xv, yv, D2_post, [-100 100]);
set(gca,'YDir','normal');
cb = colorbar('eastoutside');
cb.Label.Interpreter = 'latex';
cb.Label.String = '$[m^{-1}]$';
ylabel('[m^{-1}]');
```

```
156 title(sprintf('$\\frac{\\partial^2 z}{\\partial u^2}$ final (u=[%.2f; %.2
       f])', ux, uy),'Interpreter','latex');
157 xlabel('x [m]'); ylabel('y [m]'); axis image;
158
159 %9
160 nexttile;
161 imagesc(xv, yv, dD2, [-200 200]);
162 set(gca,'YDir','normal');
163 cb = colorbar('eastoutside');
164 cb.Label.Interpreter = 'latex';
165 cb.Label.String = '$[m^{-1}]$';
166 ylabel('[m^{-1}]');
167 title(sprintf('$\\Delta \\frac{\\partial^2 z}{\\partial u^2}$ (u=[%.2f;
       %.2f])', ux, uy),'Interpreter','latex');
168 xlabel('x [m]'); ylabel('y [m]'); axis image;
169
170 %% === 3D PLOTS ===
171
172 %1 z initial (equal scale)
173 Zpre3 = Zpre;
174 Zpre3(~M) = NaN;   % mask outside domain
175 figure('Name','z initial (3D surface)','Color','w');
176 surf(X, Y, Zpre3, Zpre3, 'EdgeColor','none');   % height = color = Zpre
177 shading interp;
178 colormap parula;
179 cb = colorbar('eastoutside');
180 ylabel(cb, 'z [m]', 'Interpreter','none');
181 xlabel('x [m]');
182 ylabel('y [m]');
183 zlabel('z [m]');
184 title('z initial (3D surface)');
185 axis tight;
186 axis equal;   % enforce equal scaling for x, y, z
187 view(45,30);
188 camlight headlight;
```

```
189 lighting gouraud;
190 rotate3d on;
191
192 %2 z final (equal scale)
193 Zpost3 = Zpost;
194 Zpost3(~M) = NaN;   % mask outside domain
195 figure('Name','z final (3D surface)','Color','w');
196 surf(X, Y, Zpost3, Zpost3, 'EdgeColor','none');   % height = color =
       Zpost
197 shading interp;
198 colormap parula;
199 cb = colorbar('eastoutside');
200 ylabel(cb, 'z [m]', 'Interpreter','none');
201 xlabel('x [m]');
202 ylabel('y [m]');
203 zlabel('z [m]');
204 title('z final (3D surface)');
205 axis tight;
206 axis equal;   % enforce equal scaling for x, y, z
207 view(45,30);
208 camlight headlight;
209 lighting gouraud;
210 rotate3d on;
211
212 %3 Delta z (equal scale)
213 dZ3 = dZ;
214 dZ3(~M) = NaN;   % mask outside domain
215 figure('Name','Delta z (3D surface)','Color','w');
216 surf(X, Y, dZ3, dZ3, 'EdgeColor','none');   % height = color = Delta Z
217 shading interp;
218 colormap parula;
219 cb = colorbar('eastoutside');
220 ylabel(cb, '$\Delta z$ [m]', 'Interpreter','latex');
221 xlabel('x [m]');
222 ylabel('y [m]');
```

```
zlabel('$\Delta z$ [m]','Interpreter','latex');
title('\Deltaz (3D surface)','Interpreter','tex');
axis tight;
axis equal;   % enforce equal scaling for x, y, z
view(45,30);
camlight headlight;
lighting gouraud;
rotate3d on;

%4 D1 initial (no equal scale)
D1pre3 = D1_pre;
D1pre3(~M) = NaN;   % mask outside domain
figure('Name','D1 initial (3D surface)','Color','w');
surf(X, Y, D1pre3, D1pre3, 'EdgeColor','none');   % height = color =
   D1_pre
shading interp; colormap parula;
cb = colorbar('eastoutside');
ylabel(cb, '$\partial z/\partial u$ [-]','Interpreter','latex');
xlabel('x [m]'); ylabel('y [m]');
zlabel('$\partial z/\partial u$ [-]','Interpreter','latex');
title(sprintf('$\\frac{\\partial z}{\\partial u}$ initial (u=[%.2f; %.2f
   ])', ux, uy),'Interpreter','latex');
axis tight;
view(45,30); camlight headlight; lighting gouraud; rotate3d on;

%5 D1 final (no equal scale)
D1post3 = D1_post;
D1post3(~M) = NaN;
figure('Name','D1 final (3D surface)','Color','w');
surf(X, Y, D1post3, D1post3, 'EdgeColor','none');   % height = color =
   D1_post
shading interp; colormap parula;
cb = colorbar('eastoutside');
ylabel(cb, '$\partial z/\partial u$ [-]','Interpreter','latex');
xlabel('x [m]'); ylabel('y [m]');
```

```
zlabel('$\partial z/\partial u$ [-]','Interpreter','latex');
title(sprintf('$\\frac{\\partial z}{\\partial u}$ final (u=[%.2f; %.2f])', ux, uy),'Interpreter','latex');
axis tight;
view(45,30); camlight headlight; lighting gouraud; rotate3d on;

%6 Delta D1 (no equal scale)
dD13 = dD1;
dD13(~M) = NaN;
figure('Name','Delta D1 (3D surface)','Color','w');
surf(X, Y, dD13, dD13, 'EdgeColor','none');   % height = color = Delta D1
shading interp; colormap parula;
cb = colorbar('eastoutside');
ylabel(cb, '$\Delta(\partial z/\partial u)$ [-]','Interpreter','latex');
xlabel('x [m]'); ylabel('y [m]');
zlabel('$\Delta(\partial z/\partial u)$ [-]','Interpreter','latex');
title('$\Delta \frac{\partial z}{\partial u}$ (3D surface)','Interpreter','latex');
axis tight;
view(45,30); camlight headlight; lighting gouraud; rotate3d on;

%7 D2 initial (no equal scale)
D2pre3 = D2_pre;
D2pre3(~M) = NaN;
figure('Name','D2 initial (3D surface)','Color','w');
surf(X, Y, D2pre3, D2pre3, 'EdgeColor','none');   % height = color = D2_pre
shading interp; colormap parula;
cb = colorbar('eastoutside'); cb.Label.Interpreter='latex'; cb.Label.String='$[m^{-1}]$';
xlabel('x [m]'); ylabel('y [m]');
zlabel('$\partial^2 z/\partial u^2$ [m^{-1}]','Interpreter','latex');
title(sprintf('$\\frac{\\partial^2 z}{\\partial u^2}$ initial (u=[%.2f; %.2f])', ux, uy),'Interpreter','latex');
axis tight;
```

```
285 view(45,30); camlight headlight; lighting gouraud; rotate3d on;
286
287 %8 D2 final (no equal scale)
288 D2post3 = D2_post;
289 D2post3(~M) = NaN;
290 figure('Name','D2 final (3D surface)','Color','w');
291 surf(X, Y, D2post3, D2post3, 'EdgeColor','none');   % height = color =
       D2_post
292 shading interp; colormap parula;
293 cb = colorbar('eastoutside'); cb.Label.Interpreter='latex'; cb.Label.
       String='$[m^{-1}]$';
294 xlabel('x [m]'); ylabel('y [m]');
295 zlabel('$\partial^2 z/\partial u^2$ [m^{-1}]','Interpreter','latex');
296 title(sprintf('$\\frac{\\partial^2 z}{\\partial u^2}$ final (u=[%.2f; %.2
       f])', ux, uy),'Interpreter','latex');
297 axis tight;
298 view(45,30); camlight headlight; lighting gouraud; rotate3d on;
299
300 %9 Delta D2 (no equal scale)
301 dD23 = dD2;
302 dD23(~M) = NaN;
303 figure('Name','Delta D2 (3D surface)','Color','w');
304 surf(X, Y, dD23, dD23, 'EdgeColor','none');   % height = color = Delta D2
305 shading interp; colormap parula;
306 cb = colorbar('eastoutside'); cb.Label.Interpreter='latex'; cb.Label.
       String='$[m^{-1}]$';
307 xlabel('x [m]'); ylabel('y [m]');
308 zlabel('$\Delta(\partial^2 z/\partial u^2)$ [m^{-1}]','Interpreter','
       latex');
309 title(sprintf('$\\Delta \\frac{\\partial^2 z}{\\partial u^2}$ (u=[%.2f;
       %.2f])', ux, uy),'Interpreter','latex');
310 axis tight;
311 view(45,30); camlight headlight; lighting gouraud; rotate3d on;
312
313 %% === EXPORT TO EXCEL ===
```

```
outXlsx = 'DirectionalDerivatives_maps.xlsx';

toLong = @(X,Y,V,name) table(X(:), Y(:), V(:), 'VariableNames', {'X','Y', name});

T_Zpre   = toLong(X, Y, Zpre,   'Zpre');   T_Zpre   = T_Zpre(M(:),:);
T_Zpost  = toLong(X, Y, Zpost,  'Zpost');  T_Zpost  = T_Zpost(M(:),:);
T_D1pre  = toLong(X, Y, D1_pre, 'D1_pre'); T_D1pre  = T_D1pre(M(:),:);
T_D1post = toLong(X, Y, D1_post,'D1_post');T_D1post = T_D1post(M(:),:);
T_D2pre  = toLong(X, Y, D2_pre, 'D2_pre'); T_D2pre  = T_D2pre(M(:),:);
T_D2post = toLong(X, Y, D2_post,'D2_post');T_D2post = T_D2post(M(:),:);

writetable(T_Zpre,   outXlsx, 'Sheet','Z_pre');
writetable(T_Zpost,  outXlsx, 'Sheet','Z_post');
writetable(T_D1pre,  outXlsx, 'Sheet','D1_pre');
writetable(T_D1post, outXlsx, 'Sheet','D1_post');
writetable(T_D2pre,  outXlsx, 'Sheet','D2_pre');
writetable(T_D2post, outXlsx, 'Sheet','D2_post');

fprintf('Excel file written: %s\n', outXlsx);
```

Listing A.14: Gleno dam Arch analysis. MATLAB listing

```
function multi_arch_thickness_analysis()
% MULTI_ARCH_THICKNESS_ANALYSIS - upstream & downstream, no residuals
%
    ------------------------------------------------------------------------

% Requested changes:
%  - Angular reference on the blue circle: phi measured CLOCKWISE, with three rays
%    at -60 deg, 0 deg, +60 deg + a circular arrow.
%  - Thickness on interval [-60 deg, +60 deg].
%  - Center shift: single vector diagram dC = C_U - C_D from origin.
%  - Bar charts of radii with "tight" autoscaling.
```

```
%
    ------------------------------------------------------------------------


    % ===== User parameters (unchanged where not needed)
       ==================
    default_scale_m_per_px = 0.01;     % default: 1 px = 0.01 m
    theoretical_thickness  = 0.76;     % green reference [m]
    edgeSigma   = 1.4;                 % Canny smoothing
    minInliers  = 30;                  % RANSAC min inliers
    ransacIters = 3000;                % RANSAC iterations
    inlierTol   = 2.0;                 % inlier tolerance [px]

    % ===== 1) Pick image
       ================================================
    [fn, fp] = uigetfile({'*.png;*.jpg;*.jpeg;*.tif;*.tiff','Images'},
       ...
                         'Select the section image to analyze');
    if isequal(fn,0), error('No image selected. Aborting.'); end
    imgPath = fullfile(fp, fn);
    I  = imread(imgPath);
    if size(I,3)>1, Ig = rgb2gray(I); else, Ig = I; end
    Ig = im2double(Ig);

    % ===== 2) Scale (m/px)
       ================================================
    fprintf('\nPixel scale confirmation\n');
    fprintf('Default: 1 px = %.6f m (%.2f cm)\n', default_scale_m_per_px,
        default_scale_m_per_px*100);
    s = input('Press ENTER to accept, or type a new value in meters/pixel
       : ','s');
    if isempty(s)
        scale = default_scale_m_per_px;
    else
```

```
36          val = str2double(s); assert(isfinite(val)&&val>0,'Invalid scale')
               ; scale = val;
37      end
38      fprintf('Using: 1 px = %.6f m (%.2f cm)\n\n', scale, scale*100);
39
40      % ===== 3) Global edge map (REAL pixels to work with)
           ================
41      BW = edge(Ig,'Canny',[],edgeSigma);
42      BW = bwmorph(BW,'clean');
43      BW = bwmorph(BW,'thin',1);
44      BW = bwareaopen(BW,30);
45
46      % ===== 4) Number of arches ========================================
47      nArches = input('How many arches to analyze? (e.g., 10): ');
48      assert(isfinite(nArches)&&nArches>=1&&mod(nArches,1)==0,'Enter a
           positive integer.');
49
50      % ===== Storage
           ========================================================
51      radiiDown     = nan(nArches,1);   % DOWNSTREAM radii [m]
52      radiiUp       = nan(nArches,1);   % UPSTREAM   radii [m]
53      centersDown   = nan(nArches,2);   % DOWNSTREAM centers [m]
54      centersUp     = nan(nArches,2);   % UPSTREAM   centers [m]
55      profilesPhiDeg= cell(nArches,1);  % phi support [deg]
56      profilesT     = cell(nArches,1);  % thickness t(phi) [m]
57
58      % ===== 5) Base window: single axes reused
           ============================
59      hBase = figure('Color','w');
60      axBase = axes('Parent',hBase);
61      imshow(I, 'Parent', axBase, 'InitialMagnification','fit');
62      axis(axBase,'image'); hold(axBase,'on');
63      title(axBase, {'Zoom/pan as needed.','Press ENTER to start ROIs for
           Arch #1.'});
```

```
64      zoom(hBase,'on'); pan(hBase,'on'); pause; zoom(hBase,'off'); pan(
           hBase,'off');
65
66      % ===== 6) Per-arch loop
           ============================================
67      for a = 1:nArches
68          fprintf('\n================  ARCH %d / %d  ================\n', a
               , nArches);
69
70          % ---- DOWNSTREAM ROI
71          figure(hBase); axes(axBase);
72          title(axBase, sprintf('Arch %d - Draw DOWNSTREAM arc ROI (double-
               click to finish)', a));
73          maskDown = safe_drawfreehand_mask(axBase);
74          P_D_px   = mask2edgePoints(BW, maskDown);
75          assert(size(P_D_px,1) >= 10, 'DOWNSTREAM ROI too sparse.');
76
77          % ---- UPSTREAM ROI
78          figure(hBase); axes(axBase);
79          title(axBase, sprintf('Arch %d - Draw UPSTREAM arc ROI (double-
               click to finish)', a));
80          maskUp = safe_drawfreehand_mask(axBase);
81          P_U_px = mask2edgePoints(BW, maskUp);
82          assert(size(P_U_px,1) >= 10, 'UPSTREAM ROI too sparse.');
83
84          % ---- Robust circle fits (RANSAC + Taubin)
85          [C_D_px, R_D_px, inlD, statusD] = fitCircleRobustWithFallback(
               ...
86              P_D_px, ransacIters, inlierTol, minInliers);
87          [C_U_px, R_U_px, inlU, statusU] = fitCircleRobustWithFallback(
               ...
88              P_U_px, ransacIters, inlierTol, minInliers);
89          fprintf('DOWNSTREAM: %s | inliers=%d | R=%.2f px\n', statusD, nnz
               (inlD), R_D_px);
```

```
90          fprintf('UPSTREAM  : %s | inliers=%d | R=%.2f px\n', statusU, nnz
               (inlU), R_U_px);
91
92          % ---- px -> m
93          C_D = C_D_px*scale;  R_D = R_D_px*scale;
94          C_U = C_U_px*scale;  R_U = R_U_px*scale;
95          centersDown(a,:) = C_D; radiiDown(a) = R_D;
96          centersUp(a,:)   = C_U; radiiUp(a)   = R_U;
97
98          % ---- Thickness t(phi) on [-60 deg, +60 deg], 0 deg upward
99          phi  = linspace(-deg2rad(60), +deg2rad(60), 361);  % radians
100         Udir = [sin(phi).', -cos(phi).'];                  % x->right, y->
               down (image axes)
101         dC   = C_D - C_U;
102         aC   = ones(numel(phi),1);
103         bC   = 2*(Udir*dC.').';
104         cC   = sum(dC.^2) - R_U^2;
105         disc = bC.^2 - 4*aC*cC;
106
107         T = nan(numel(phi),1);
108         for k = 1:numel(phi)
109             if disc(k) >= 0
110                 t1 = (-bC(k) - sqrt(disc(k))) / (2*aC(k));
111                 t2 = (-bC(k) + sqrt(disc(k))) / (2*aC(k));
112                 cand = sort([t1 t2]);
113                 tk   = cand(find(cand > R_D + 1e-9, 1, 'first')); % first
                        beyond downstream radius
114                 if ~isempty(tk), T(k) = tk - R_D; end
115             end
116         end
117         if any(isnan(T)) && any(~isnan(T)), T = fillmissing(T,'linear');
               end
118         phi_deg = rad2deg(phi);
119         profilesPhiDeg{a} = phi_deg; profilesT{a} = T;
120
```

```
        % ---- Per-arch plots (overlay + thickness)
        f = figure('Color','w','Position',[60 60 1200 560]); %#ok<NASGU>
        % (Left) overlay
        subplot(1,2,1); imshow(I,[],'InitialMagnification','fit'); hold
            on; axis image;
        plot(P_D_px(inlD,1), P_D_px(inlD,2), 'b.', 'MarkerSize', 8);
        plot(P_U_px(inlU,1), P_U_px(inlU,2), 'r.', 'MarkerSize', 8);
        th = linspace(0,2*pi,720);
        plot( (C_D(1)+R_D*cos(th))/scale, (C_D(2)-R_D*sin(th))/scale, 'b-
            ', 'LineWidth', 1.8 );
        plot( (C_U(1)+R_U*cos(th))/scale, (C_U(2)-R_U*sin(th))/scale, 'r-
            ', 'LineWidth', 1.8 );
        plot( C_D(1)/scale, C_D(2)/scale, 'bo', 'MarkerFaceColor','y' );
        plot( C_U(1)/scale, C_U(2)/scale, 'ro', 'MarkerFaceColor','y' );

        % >>> Angular reference for phi measured CLOCKWISE from 0 deg up
        drawPhiReferenceClockwise(C_D, R_D, scale);

        title(sprintf('Arch %d - Fitted circles', a));
        legend({'DOWNSTREAM inliers','UPSTREAM inliers','DOWNSTREAM fit',
            'UPSTREAM fit','C_D','C_U'},...
                'Location','southeast');

        % (Right) thickness profile (strictly [-60 deg, +60 deg])
        subplot(1,2,2); hold on; grid on;
        plot(phi_deg, T, 'k', 'LineWidth', 1.7);
        yline(theoretical_thickness, 'g-', 'LineWidth', 1.5);
        xlim([-60 60]);
        xlabel('\phi [deg]   (-60 = left, 0 = up, +60 = right)');
        ylabel('t(\phi) [m]');
        title(sprintf('Thickness - mean=%.3f, min=%.3f, max=%.3f m', ...
              mean(T,'omitnan'), min(T,[],'omitnan'), max(T,[],'omitnan')
                 ));
        legend({'t(\phi)','Theoretical 0.76 m'}, 'Location','best');
    end
```

```
151
152     % === Residual analysis per arch (two panels: downstream & upstream)
            ===
153     phiMax = 60;  % degrees
154
155     % Downstream residuals
156     [phiD, drD, phiGrid, medD] = residualsByPhi(P_D_px, C_D_px, R_D_px,
            scale, phiMax);
157     % Upstream residuals
158     [phiU, drU, ~,    medU]    = residualsByPhi(P_U_px, C_U_px, R_U_px,
            scale, phiMax);
159
160     figure('Color','w','Position',[70 70 1100 450]);
161     subplot(1,2,1);
162     plotResidualDiagram(phiD, drD, phiGrid, medD, ...
163        sprintf('Arch %d - Downstream residuals', a), [0 0.4470 0.7410],
               [0 0.25 0.55]);
164
165     subplot(1,2,2);
166     plotResidualDiagram(phiU, drU, phiGrid, medU, ...
167        sprintf('Arch %d - Upstream residuals', a), [0.8500 0.3250
               0.0980], [0.65 0.20 0.05]);
168
169     % ===== 7) Global comparisons
            ========================================
170     % (A) Radii with tight Y autoscale
171     figure('Color','w','Position',[60 60 1000 420]);
172     subplot(1,2,1); bar(radiiDown); grid on; ylabel('Radius [m]'); xlabel
            ('Arch'); title('All DOWNSTREAM radii');
173     ylim(tight_ylim(radiiDown));
174     subplot(1,2,2); bar(radiiUp);   grid on; ylabel('Radius [m]'); xlabel
            ('Arch'); title('All UPSTREAM radii');
175     ylim(tight_ylim(radiiUp));
176
177     % (B) All thickness profiles
```

```
    figure('Color','w','Position',[80 80 1000 500]); hold on; grid on;
    for a=1:nArches
        plot(profilesPhiDeg{a}, profilesT{a}, 'LineWidth', 1.2);
    end
    yline(theoretical_thickness,'g-','LineWidth',1.5);
    xlim([-60 60]);
    xlabel('\phi [deg]'); ylabel('t(\phi) [m]');
    title('All arches - Thickness profiles');
    legend(arrayfun(@(k) sprintf('Arch %d',k), 1:nArches, 'UniformOutput', false), ...
           'Location','bestoutside');

    % (C) Center shift: plot dC_i = C_U_i - C_D_i from origin (0,0)
    dC = centersUp - centersDown;              % Nx2, meters
    figure('Color','w','Position',[90 90 600 560]); hold on; grid on; axis equal;
    % crosshair axes
    xl = max(abs(dC(:,1))); yl = max(abs(dC(:,2)));
    pad = max(1e-3, 0.1*max(xl,yl));
    xlim([-xl-pad, xl+pad]); ylim([-yl-pad, yl+pad]);
    plot(xlim, [0 0], 'k:'); plot([0 0], ylim, 'k:');
    % arrows from origin to each dC
    for a=1:size(dC,1)
        quiver(0,0, dC(a,1), dC(a,2), 0, 'LineWidth',1.6, 'MaxHeadSize',0.6);
        text(dC(a,1), dC(a,2), sprintf('  #%d',a), 'VerticalAlignment','middle');
    end
    xlabel('\DeltaX [m]  (C_U - C_D)'); ylabel('\DeltaY [m]  (C_U - C_D)');
    title('Upstream centers relative to downstream centers (per arch)');
end

% ============================== HELPERS ==================================
```

```
207
208 function drawPhiReferenceClockwise(CD_m, RD_m, scale)
209 % Draw phi reference on the DOWNSTREAM circle:
210 % - three black rays at phi = -60 deg, 0 deg, +60 deg (phi measured
    CLOCKWISE from upward)
211 % - an angular dimension arc from -60 deg to +60 deg (clockwise) with
    arrow
212 % - labels: phi = -60 deg, phi = 0 deg, phi = +60 deg
213 %
214 % Inputs are in METERS (center CD_m, radius RD_m). We convert by /scale
    for plotting.
215
216     % Geometry settings (tweak if the image is very large/small)
217     rayLen   = 0.92 * RD_m;     % length of reference rays
218     arcRad   = 0.32 * RD_m;     % radius of the angular dimension arc
219     arcW     = 1.4;             % line width for the arc
220     rayW     = 1.8;             % line width for the rays
221     labelOff = 0.06 * RD_m;     % outward label offset
222     arrowLen = 0.045 * RD_m;    % size of arrow head
223     arrowAng = 18;              % arrow head half-angle [deg]
224
225     % Helper to convert a phi [deg] into a direction (image axes: x->
        right, y->down)
226     dirOf = @(deg) [sind(deg), -cosd(deg)];
227
228     % 1) Rays at -60, 0, +60 deg + labels
229     phiList = [-60 0 60];
230     for kk = 1:numel(phiList)
231         ph = phiList(kk);
232         d  = dirOf(ph);                          % unit direction (1x2)
233         P1 = CD_m;                               % center
234         P2 = CD_m + rayLen * d;                  % ray tip
235
236         plot([P1(1) P2(1)]/scale, [P1(2) P2(2)]/scale, 'k-', 'LineWidth',
                rayW);
```

```
237
238         % place label slightly beyond the tip in the same direction
239         Lpos = CD_m + (rayLen + labelOff) * d;
240         text(Lpos(1)/scale, Lpos(2)/scale, ...
241             sprintf('\\phi=%+d^\\circ', ph), ...
242             'Color','k','FontSize',10,'HorizontalAlignment','center');
243     end
244
245     % 2) Angular arc from -60 to +60 (CLOCKWISE)
246     % Increasing phi is clockwise in our convention, so sweep from -60 ->
        +60
247     tDeg = linspace(-60, +60, 121);
248     A = CD_m + arcRad * [sind(tDeg(:))  -cosd(tDeg(:))];   % points on
        arc
249     plot(A(:,1)/scale, A(:,2)/scale, 'k-', 'LineWidth', arcW);
250
251     % 3) Arrow head at +60 deg (end of the clockwise sweep)
252     % Tangent direction at +60 deg for increasing phi (clockwise):
253     % derivative of [sin(phi), -cos(phi)] wrt phi is [cos(phi), sin(phi)]
254     phEnd = 60;                             % deg
255     tDir  = [cosd(phEnd), sind(phEnd)];   % tangent
256     tDir  = tDir / norm(tDir);
257
258     % Tip point on the arc
259     Ptip = CD_m + arcRad * dirOf(phEnd);
260
261     % Build two arrow flanks rotated by +/- arrowAng backwards along the
        arc
262     R1 = rot2D(-arrowAng);  % rotate clockwise
263     R2 = rot2D(+arrowAng);  % rotate counterclockwise
264     flank1 = Ptip - (R1 * (tDir(:)*arrowLen)).';   % small segment
        backward & rotated
265     flank2 = Ptip - (R2 * (tDir(:)*arrowLen)).';
266
```

```
    plot([Ptip(1) flank1(1)]/scale, [Ptip(2) flank1(2)]/scale, 'k-', '
        LineWidth', arcW);
    plot([Ptip(1) flank2(1)]/scale, [Ptip(2) flank2(2)]/scale, 'k-', '
        LineWidth', arcW);

    % Optional label "phi (clockwise)" near the arc, centered at +15 deg
    phLbl = 15;
    Plbl  = CD_m + (arcRad + 0.02*RD_m) * dirOf(phLbl);
    text(Plbl(1)/scale, Plbl(2)/scale, '\phi  (clockwise)', ...
        'Color','k','FontAngle','italic','FontSize',10,'
            HorizontalAlignment','center');
end

% small 2D rotation helper (degrees, anticlockwise positive)
function R = rot2D(angDeg)
    a = deg2rad(angDeg);
    R = [cos(a) -sin(a); sin(a) cos(a)];
end

function yl = tight_ylim(v)
% Return a tight ylim around the data vector v with a minimal pad.
    v = v(~isnan(v));
    if isempty(v), yl = [0 1]; return; end
    vmin = min(v); vmax = max(v); rng = vmax - vmin;
    if rng < 1e-3, rng = 1e-3; end
    pad = 0.08 * rng;
    yl = [vmin - pad, vmax + pad];
end

function mask = safe_drawfreehand_mask(ax)
% Freehand ROI (double-click to finish), no UI mode changes here.
    if nargin < 1 || isempty(ax), ax = gca; end
    fig = ancestor(ax,'figure');
    try, zoom(fig,'off'); pan(fig,'off'); rotate3d(fig,'off'); end %#ok<
        TRYNC>
```

```
298     mask = [];
299     try
300         if exist('drawfreehand','file') == 2
301             h = drawfreehand(ax,'Color',[0 0.8 1],'LineWidth',1.5,'
                    FaceAlpha',0.15,'DrawingArea',ax);
302             wait(h); mask = createMask(h); if ishghandle(h), delete(h);
                    end; return;
303         end
304     catch
305         % fallback legacy
306     end
307     h = imfreehand(ax,'Closed',true);
308     mask = h.createMask(); if ishghandle(h), delete(h); end
309 end
310 
311 function P = mask2edgePoints(BW, mask)
312 % Edge pixels intersect ROI, return [x y] in pixels.
313     E = BW & mask;
314     E = imdilate(E, strel('disk',1));
315     [ry, rx] = find(E);
316     P = [rx, ry];
317 end
318 
319 function [C_px, R_px, inliers, status] = fitCircleRobustWithFallback(P_px
        , iters, tol_px, minInliers)
320 % RANSAC bounded + Taubin refine + fallbacks.
321     status  = "RANSAC+Taubin";
322     inliers = false(size(P_px,1),1);
323     C_px    = [NaN NaN];
324     R_px    = NaN;
325 
326     rB = radiusBoundsFromROI2D(P_px);  % [rMin rMax] in pixels
327     [C0,R0,inl0] = ransacCircle_bounded(P_px, iters, tol_px, minInliers,
            rB);
328 
```

```
    if nnz(inl0) >= minInliers && isfinite(R0) && R0 > 0
        [Ct,Rt] = circleFitTaubin(P_px(inl0,:));
        if isfinite(Rt) && Rt > 0 && Rt >= rB(1) && Rt <= rB(2)
            C_px = Ct; R_px = Rt; inliers = inl0; status = "RANSAC OK + Taubin OK"; return;
        else
            C_px = C0; R_px = R0; inliers = inl0; status = "RANSAC OK (Taubin out-of-bounds)"; return;
        end
    end

    [CtAll,RtAll] = circleFitTaubin(P_px);
    if isfinite(RtAll) && RtAll > 0 && RtAll >= rB(1) && RtAll <= rB(2)
        C_px = CtAll; R_px = RtAll; inliers = true(size(P_px,1),1);
           status = "Taubin ALL points"; return;
    end

    rB2 = [0.5*rB(1), 2.0*rB(2)];
    [C2,R2,inl2] = ransacCircle_bounded(P_px, iters, tol_px, minInliers,
       rB2);
    if nnz(inl2) >= minInliers && isfinite(R2) && R2 > 0
        C_px = C2; R_px = R2; inliers = inl2; status = "RANSAC with
           expanded bounds"; return;
    end

    status = "FAILED (no reliable model)";
end

function rB = radiusBoundsFromROI2D(P)
% Plausible radius bounds from ROI width & height (pixels).
    x = P(:,1); y = P(:,2);
    w = max(x) - min(x);
    h = max(y) - min(y);
    spanMin = max(min(w,h), 10);
    spanMax = max(w,h);
```

```
    rB = [0.3*spanMin, 15*spanMax];
end

function [C, R, inliers] = ransacCircle_bounded(P, iters, tol_px,
   minInliers, rBounds)
% RANSAC circle fit constrained by radius bounds (pixels).
    n = size(P,1);
    bestInl = false(n,1); C = [NaN NaN]; R = NaN;
    if n < 3, inliers = bestInl; return; end
    rMin = rBounds(1); rMax = rBounds(2);
    for t = 1:iters
        idx = randperm(n,3); Q = P(idx,:);
        if isCollinear(Q), continue; end
        [xc,yc,Rt] = circleFrom3(Q);
        if ~isfinite(Rt) || Rt<=0 || Rt<rMin || Rt>rMax, continue; end
        d   = vecnorm(P - [xc yc], 2, 2);
        res = abs(d - Rt);
        inl = res < tol_px;
        if nnz(inl) > nnz(bestInl)
            bestInl = inl; C = [xc yc]; R = Rt;
        end
    end
    if nnz(bestInl) < minInliers
        warning('RANSAC: only %d inliers (<%d). Fit may be weak.', nnz(
           bestInl), minInliers);
    end
    inliers = bestInl;
end

function tf = isCollinear(Q)
% True if the three points are almost collinear.
    A = Q(2,:) - Q(1,:);
    B = Q(3,:) - Q(1,:);
    tf = abs(A(1)*B(2) - A(2)*B(1)) < 1e-9;
end
```

```
392
393 function [xc,yc,R] = circleFrom3(Q)
394 % Circle through 3 non-collinear points (pixels).
395     x1=Q(1,1); y1=Q(1,2);
396     x2=Q(2,1); y2=Q(2,2);
397     x3=Q(3,1); y3=Q(3,2);
398     A = 2*[x2-x1, y2-y1; x3-x1, y3-y1];
399     b = [x2^2-x1^2 + y2^2-y1^2;  x3^2-x1^2 + y3^2-y1^2];
400     c = A\b; xc = c(1); yc = c(2);
401     R = hypot(x1-xc, y1-yc);
402 end
403
404 function [C, R] = circleFitTaubin(P)
405 % Algebraic circle fit (Taubin, 1991) on 2D pixels.
406     x = P(:,1); y = P(:,2);
407     mx = mean(x); my = mean(y);
408     x = x - mx; y = y - my;
409     Z = [x.^2 + y.^2, x, y, ones(size(x))];
410     M = Z'*Z; H = diag([2 1 1 0]); epsl = 1e-12;
411     [vec,val] = eig(M + epsl*eye(4), H);
412     [~,id] = min(real(diag(val))); v = real(vec(:,id));
413     a=v(1); b=v(2); c=v(3); d=v(4);
414     xc = -b/(2*a) + mx; yc = -c/(2*a) + my;
415     R  = sqrt((b^2 + c^2)/(4*a^2) - d/a);
416     C  = [xc yc];
417 end
418
419 function [phi_deg, dr_m, phi_grid, med_dr] = residualsByPhi(P_px, C_px,
       R_px, scale, phiMaxDeg)
420 % RESIDUALSBYPHI
421 % Compute radial residuals Delta r = r - R for REAL edge pixels vs fitted
       circle.
422 % phi convention: CLOCKWISE, 0 deg at crown (up). We keep phi in [-
       phiMaxDeg, +phiMaxDeg].
423 %
```

```
% Inputs:
%   P_px       [N x 2] real edge pixels (inside ROI), columns = [x y] in PIXELS
%   C_px       [1 x 2] fitted circle center in PIXELS
%   R_px       scalar  fitted circle radius in PIXELS
%   scale      meters per pixel
%   phiMaxDeg  scalar, e.g. 60 (we will keep [-60, +60])
%
% Outputs:
%   phi_deg    [M x 1] phi of each kept pixel (deg)
%   dr_m       [M x 1] residual Delta r for each kept pixel (meters)
%   phi_grid   [-phiMaxDeg:phiMaxDeg] degree grid
%   med_dr     [length(phi_grid) x 1] per-degree median Delta r (meters)

    % Convert to meters
    P_m = P_px * scale;
    C_m = C_px * scale;
    R_m = R_px * scale;

    % Vectors and radii
    d  = P_m - C_m;                 % [N x 2]
    r  = vecnorm(d,2,2);            % [N x 1] meters
    % phi: clockwise with 0 deg at crown: atan2d(dy,dx) gives 0 deg to the right;
    % shift +90 deg to put 0 deg at "up", then wrap to [-180, 180]
    phi = atan2d(d(:,2), d(:,1)) + 90;
    phi = wrapTo180(phi);

    % Keep only pixels in the requested sector
    keep = (phi >= -phiMaxDeg) & (phi <= +phiMaxDeg);
    phi_deg = phi(keep);
    dr_m    = r(keep) - R_m;

    % Per-degree median (robust trend)
    phi_grid = (-phiMaxDeg:+phiMaxDeg).';
```

```
    med_dr   = nan(size(phi_grid));
    for k = 1:numel(phi_grid)
        idx = abs(phi_deg - phi_grid(k)) < 0.5;    % bin width ~ 1 deg
        if any(idx)
            med_dr(k) = median(dr_m(idx),'omitnan');
        end
    end
end

function v = wrapTo180(v)
% Ensure angles are within [-180, 180]
    v = mod(v + 180, 360) - 180;
end

function plotResidualDiagram(phi_deg, dr_m, phi_grid, med_dr, titleStr,
   colorPts, colorMed)
% Scatter of all pixel residuals + median per-degree curve
    hold on; grid on;
    % light scatter of all points
    plot(phi_deg, dr_m, '.', 'Color', [colorPts 0.25], 'MarkerSize', 8, '
       DisplayName','pixels');
    % median curve
    plot(phi_grid, med_dr, '-', 'Color', colorMed, 'LineWidth', 1.8, '
       DisplayName','median per deg');
    xlim([min(phi_grid) max(phi_grid)]);
    xlabel('\phi [deg]   (-60 = left, 0 = up, +60 = right)');
    ylabel('\Delta r(\phi) [m]');
    title(titleStr);
    yline(0,'k-','HandleVisibility','off');
    legend('Location','best');
end
```

List of Figures

Figure 1: Stelvio Pass: overview of high-altitude military works 31

Figure 2: Filon dei Mot: oblique and planimetric views from the integrated survey 33

Figure 3: Filon dei Mot: orthographic elevation and masonry reading 35

Figure 4: Filon dei Mot: planimetric orthophoto and comparative metric analysis 36

Figure 5: Nembro bridge: 3D model 38

Figure 6: Nembro bridge: design vs construction 41

Figure 7: Nembro bridge: 3D reconstruction from drawings 42

Figure 8: Nembro bridge: as-built vs Pesenti scheme 43

Figure 9: Gleno Dam: photogrammetric model with photo capture points 44

Figure 10: Gleno Dam: elevation-coded views of the photogrammetric 3D model 45

Figure 11: Gleno Dam: CloudCompare DIP rendering 45

Figure 12: Gleno Dam: planimetric contour map from the integrated survey 49

Figure 13: ESB container and photogrammetric context. Source: Cardaci, Giretti & Azzola (2024), CC BY 4.0. 53

Figure 14: The photogrammetry data acquisition of the sample and the ESB support. Source: Cardaci, Giretti & Azzola (2024), CC BY 4.0. 54

Figure 15: Point cloud distances before and after the test, in correspondency of local deformations. Source: Cardaci, Giretti & Azzola (2024), CC BY 4.0. 56

Figure 16: Cloud-to-cloud displacement map (cropped). Source: Cardaci, Giretti & Azzola (2024), CC BY 4.0. 56

Figure 17: Pre/post 3D views and orthographic projections (cropped). Source: Cardaci, Giretti & Azzola (2024), CC BY 4.0. 62

Figure 18: Effect of the smoothing parameter p in MATLAB's `csaps`. Noisy data (blue circles) are fitted with two cubic smoothing splines: with $p = 0.01$ (red), producing a very smooth curve, and with $p \approx 1$ (green), yielding an almost interpolatory fit. 72

Figure 19: Example of high smoothing parameter with $p = 0.9$ in MATLAB's `csaps`, in geotechnical test, producing a very smooth curve. 73

Figure 20: Example of smoothing parameter with $p = 0.9999$ in MATLAB's `csaps`, in geotechnical test, producing a smooth curve. 73

Figure 21: Example of low smoothing parameter with $p = 0.9999999$ in MATLAB's `csaps`, in geotechnical test, producing an interpolating curve. 74
Figure 22: Example of smoothing with parameter with $p = 1$ in MATLAB's `csaps`, in geotechnical test, producing an excessive non-interpolating broken line. 74
Figure 23: Welcome dialog of the MATLAB Wizard, which guides the user in creating a `.m` file for the cross-section survey. 82
Figure 24: Input dialog for specifying the parameter y_0 corresponding to the section plane (example: $y_0 = 0.165$). 83
Figure 25: Input dialog for defining the parameter δ, representing the semi-thickness along the Y axis (example: $\delta = 0.0002$). 83
Figure 26: Input dialog for setting the smoothing spline parameter p ($0 \leq p \leq 1$), where values close to 1 behave nearly interpolatory (example: $p = 0.9999999$). 83
Figure 27: Input dialog for specifying the number of points to be used for the export in `.poly` format. 84
Figure 28: Dialog asking whether to use a predefined reference polyline, with the coordinates explicitly listed in the message. 84
Figure 29: Dialog for specifying the name of the Excel file to be exported (e.g., `SectionAA_profiles.xlsx`). 8
Figure 30: Welcome dialog of the MATLAB Wizard, which guides the user in creating a `.m` file for the cross-section survey. 89
Figure 31: Representation of the design model. 90
Figure 32: Representation of the initial point cloud model. 90
Figure 33: Representation of the final point cloud model. 91
Figure 34: Representation of vertical sections. 91
Figure 35: Representation of 3D model vertical sections. 92
Figure 36: Vertical section AA'. 93
Figure 37: Vertical section BB'. 93
Figure 38: Vertical section CC'. 94
Figure 39: Vertical section DD'. 95
Figure 40: Vertical section EE'. 95
Figure 41: Vertical section FF'. 96
Figure 42: Vertical section GG'. 97
Figure 43: Vertical section HH'. 97

Figure 44: Vertical section II'. 98
Figure 45: Vertical section JJ'. 99
Figure 46: 2D representaiton of z, first and second derivatives. 101
Figure 47: z initial 3D view. 102
Figure 48: z final 3D view. 102
Figure 49: Difference on z 3D view. 103
Figure 50: First derivative wrt u, initial. 103
Figure 51: First derivative wrt u, final. 104
Figure 52: First derivative wrt u, difference. 104
Figure 53: Second derivative wrt u, initial. 105
Figure 54: Second derivative wrt u, final. 105
Figure 55: Second derivative wrt u, difference. 106
Figure 56: Finding the coordinates of the reference points for determining the rotation in a structure point cloud, in CloudCompare. 107
Figure 57: Vertical section on upstream arch. 107
Figure 58: Vertical section on downstream arch. 108
Figure 59: Analysis of the variable width of the downstream arch. 109
Figure 60: Planimetric development of the downstream arch profile. 109
Figure 61: Three-dimensional analysis of the downstream arch intrados. 109
Figure 62: Metric analysis on the horizontal section of arch number 1. 118
Figure 63: Metric analysis on the horizontal section of arch number 2. 119
Figure 64: Metric analysis on the horizontal section of arch number 3. 119
Figure 65: Metric analysis on the horizontal section of arch number 4. 120
Figure 66: Metric analysis on the horizontal section of arch number 5. 120
Figure 67: Metric analysis on the horizontal section of arch number 6. 121
Figure 68: Metric analysis on the horizontal section of arch number 7. 121
Figure 69: Metric analysis on the horizontal section of arch number 8. 122
Figure 70: Metric analysis on the horizontal section of arch number 9. 122
Figure 71: Metric analysis on the horizontal section of arch number 10. 123
Figure 72: Metric comparison of different thicknesses of arches. 123
Figure 73: Radius comparison between different arches. 124
Figure 74: Center displacements of upstream and downstream Arches. 125

Transparency Statement

The author declares that artificial intelligence tools NotebookLM (Google), ChatGPT (OpenAI) and DeepL Write (DeepL) were used solely for minor language editing (grammar, spelling, and readability), and that all changes were reviewed and approved by the author. No scientific content, analyses, results, or interpretations were generated by AI. The author takes full responsibility for the integrity and accuracy of the manuscript.

Bibliography

[1] Fabio Remondino and Alessandro Rizzi. “Reality-based 3D documentation of natural and cultural heritage sites—techniques, problems, and examples”. In: *Applied Geomatics* 2.3 (2010), pp. 85–100. DOI: 10.1007/s12518-010-0025-X.

[2] Young Hoon Jo and Seonghyuk Hong. “Three-Dimensional Digital Documentation of Cultural Heritage Site Based on the Convergence of Terrestrial Laser Scanning and Unmanned Aerial Vehicle Photogrammetry”. In: *ISPRS International Journal of Geo-Information* 8.2 (2019), p. 53. DOI: 10.3390/ijgi8020053.

[3] Elisa Grilli and Fabio Remondino. “Machine Learning Generalisation across Different 3D Architectural Heritage Point Clouds”. In: *ISPRS International Journal of Geo-Information* 10.5 (2021), p. 303. DOI: 10.3390/ijgi10050303.

[4] Junshan Liu, Salman Azhar, Danielle Willkens, and Botao Li. “Static Terrestrial Laser Scanning (TLS): A Review of Applications and Accuracy Assessment in Heritage Documentation”. In: *Virtual Worlds* 2.2 (2023), pp. 90–114. DOI: 10.3390/virtualworlds2020006.

[5] Yanzong Zhang, Guibo Nie, and Duozhi Wang. “Structural health monitoring based on three-dimensional point cloud technology: A systematic review”. In: *Results in Engineering* 27 (2025), p. 106552. DOI: 10.1016/j.rineng.2025.106552.

[6] Qian Wang, Yi Tan, and Zhongya Mei. “Computational Methods of Acquisition and Processing of 3D Point Cloud Data for Construction Applications”. In: *Archives of Computational Methods in Engineering* 27 (2020), pp. 479–499. DOI: 10.1007/s11831-019-09320-4.

[7] Marco Puliti, Giovanni Montaggioli, and Alessandro Sabato. “Automated subsurface defects’ detection using point cloud reconstruction from infrared images”. In: *Automation in Construction* 129 (2021), p. 103829. DOI: 10.1016/j.autcon.2021.103829.

[8] Elise Kaartinen, Kyle Dunphy, and Ayan Sadhu. “LiDAR-Based Structural Health Monitoring: Applications in Civil Infrastructure Systems”. In: *Sensors* 22.12 (2022), p. 4610. DOI: 10.3390/s22124610.

[9] Linglong Zhou, Guoxin Wu, Yunbo Zuo, Xuanyu Chen, and Hongle Hu. “A Comprehensive Review of Vision-Based 3D Reconstruction Methods”. In: *Sensors* 24.7 (2024), p. 2314. DOI: 10.3390/s24072314.

[10] Ali Ghadimzadeh Alamdari and Arvin Ebrahimkhanlou. “A multi-scale robotic approach for precise crack measurement in concrete structures”. In: *Automation in Construction* 158 (2024), p. 105215. DOI: 10.1016/j.autcon.2023.105215.

[11] H. C. Jo, H. G. Sohn, and Y. M. Lim. “A LiDAR Point Cloud Data-Based Method for Evaluating Strain on a Curved Steel Plate Subjected to Lateral Pressure”. In: *Sensors* 20.3 (2020), p. 721. DOI: 10.3390/s20030721.

[12] Mohammad Ebrahim Mohammadi, Richard L. Wood, and Christine E. Wittich. “Non-Temporal Point Cloud Analysis for Surface Damage in Civil Structures”. In: *ISPRS International Journal of Geo-Information* 8.12 (2019), p. 527. DOI: 10.3390/ijgi8120527.

[13] Chang Xu, Wen Xiong, Pingbo Tang, and C. S. Cai. “Automated flatness assessment for large quantities of full-scale precast beams using laser scanning”. In: *Computer-Aided Civil and Infrastructure Engineering* 39.12 (2024), pp. 1868–1885. DOI: 10.1111/mice.13162.

[14] Jingdao Chen and Yong Kwon Cho. “CrackEmbed: Point feature embedding for crack segmentation from disaster site point clouds with anomaly detection”. In: *Advanced Engineering Informatics* 52 (2022), p. 101550. DOI: 10.1016/j.aei.2022.101550.

[15] Rosella Alessia Galantucci and Fabio Fatiguso. “Advanced damage detection techniques in historical buildings using digital photogrammetry and 3D surface analysis”. In: *Journal of Cultural Heritage* 36 (2019), pp. 51–62. DOI: 10.1016/j.culher.2018.09.014.

[16] Mehdi M. Akhlaghi, Supratik Bose, M. Ebrahim Mohammadi, Babak Moaveni, Andreas Stavridis, and Richard L. Wood. “Post-earthquake damage identification of an RC school building in Nepal using ambient vibration and point cloud data”. In: *Engineering Structures* 227 (2021), p. 111413. DOI: 10.1016/j.engstruct.2020.111413.

[17] Mohamed Abdelazeem, Ahmed Elamin, Akram Afifi, and Ahmed El-Rabbany. “Multi-sensor point cloud data fusion for precise 3D mapping”. In: *The Egyptian Journal of Remote Sensing and Space Science* 24 (June 2021). DOI: 10.1016/j.ejrs.2021.06.002.

[18] Dmitry Gura, Ekaterina Karamysheva, and Saida Pshidatok. “Using CloudCompare Software Editing Tools for Processing a Three-Dimensional Point Cloud of an Urban Development Site”. In: *ITM Web of Conferences* 59 (2024), p. 02008. DOI: 10.1051/itmconf/20245902008.

[19] Johannes L. Schönberger and Jan-Michael Frahm. “Structure-from-Motion Revisited”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4104–4113. DOI: 10.1109/CVPR.2016.445.

[20] Francesco Nex and Fabio Remondino. “UAV for 3D Mapping Applications: A Review”. In: *Applied Geomatics* 14 (2022), pp. 1–15. DOI: 10.1007/s12518-021-00379-3.

[21] Fabio Remondino, Emanuele Grilli, and Andrea Torresani. “3D Digitisation of Cultural Heritage: Scientific Challenges and Future Perspectives”. In: *Remote Sensing* 13.14 (2021), p. 2817. DOI: 10.3390/rs13142817.

[22] Emanuele Grilli and Fabio Remondino. “Machine Learning Generalisation across Different 3D Architectural Heritage”. In: *ISPRS International Journal of Geo-Information* 10.2 (2021), p. 52. DOI: 10.3390/ijgi10020052.

[23] Qian Wang and Myung-Keun Kim. “Scan-to-BIM for Infrastructure: A Review”. In: *Automation in Construction* 132 (2021), p. 103932. DOI: 10.1016/j.autcon.2021.103932.

[24] Emanuele Farella, Andrea Torresani, and Fabio Remondino. “Mesh Processing and Analysis for Cultural Heritage Documentation”. In: *Remote Sensing* 14 (2022), p. 3092. DOI: 10.3390/rs14133092.

[25] Luis Sánchez-Aparicio, Federico del Blanco García, David Mencías-Carrizosa, Paula Villanueva, Jose Aira-Zunzunegui, David Arauz, Roberto Pierdicca, Javier Pinilla, and Jesús María García Gago. “Detection of damage in heritage constructions based on 3D point clouds. A systematic review”. In: *Journal of Building Engineering* 77 (Oct. 2023). DOI: 10.1016/j.jobe.2023.107440.

[26] Cecilia Bolognesi and Daniela Aiello. “Monitoring Cultural Heritage through Multi-Temporal 3D Surveying”. In: *Heritage Science* 9 (2021), p. 94. DOI: 10.1186/s40494-021-00575-7.

[27] Raffaella Brumana, Daniela Oreni, and Andrea Raimondi. “HBIM Generation from Point Clouds for Historical Building Documentation”. In: *ISPRS International Journal of Geo-Information* 10.6 (2021), p. 409. DOI: 10.3390/ijgi10060409.

[28] Alessio Cardaci and Pietro Azzola. “Differential Geometry for Morphological Analysis of the Built Environment: a Reproducible Workflow to Standardise the Interpretation of Point Clouds”. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 48 (2026), pp. 81–88. DOI: 10.5194/isprs-archives-XLVIII-2-W12-2026-81-2026.

[29] C. F. Gauss. *Disquisitiones generales circa superficies curvas*. Vol. 1. Dieterich, 1828.

[30] B. Riemann. “Über die Hypothesen, welche der Geometrie zu Grunde liegen”. In: *Abhandlungen der Königlichen Gesellschaft der Wissenschaften zu Göttingen* 13 (1867), pp. 133–152.

[31] F. Klein. “Vergleichende Betrachtungen über neuere geometrische Forschungen”. In: *Vergleichende Betrachtungen über neuere geometrische Forschungen* (1872).

[32] L. Xiong, S. Li, G. Tang, and J. Strobl. “Geomorphometry and terrain analysis: data, methods, platforms and applications”. In: *Earth-Science Reviews* 233 (2022), p. 104191. DOI: 10.1016/j.earscirev.2022.104191.

[33] Alessio Cardaci and Pietro Azzola. “Differential Geometry for Morphological Analysis of the Built Environment: a Reproducible Workflow to Standardise the Interpretation of Point Clouds”. In: *ISPRS Archives* (2025). Primary Source. DOI: 10.5194/isprs-archives-XLVIII-2-W12-2026-81-2026.

[34] J. Minár, I. S. Evans, and M. Jenčo. “A comprehensive system of geomorphometric variables”. In: *Geomorphology* 350 (2020), p. 107031. DOI: 10.1016/j.geomorph.2020.107031.

[35] G. Sofia. “Geomorphometry: Variable, Scale and Methods”. In: *Earth Surface Processes and Landforms* 45.1 (2020), pp. 147–155. DOI: 10.1002/esp.4880.

[36] Arli Llabani and Freskida Abazaj. “3D documentation of cultural heritage using terrestrial laser scanning”. In: *Journal of Applied Engineering Science* 22 (June 2024), pp. 1–5. DOI: 10.5937/jaes0-50414.

[37] Yongkang Xing, Shengxiang Yang, Conor Fahy, Tracy Harwood, and Jethro Shell. “Capturing the Past, Shaping the Future: A Scoping Review of Photogrammetry in Cultural Building Heritage”. In: *Electronics* 14.18 (2025). DOI: 10.3390/electronics14183666.

[38] Yahya Alshawabkeh, Mohammad El Khalili, Eyad Almasri, Fadi Bal’awi, and Amaal Al-Massarweh. “Heritage documentation using laser scanner and photogrammetry. The case study of Qasr Al-Abidit, Jordan”. In: *Digital Applications in Archaeology and Cultural Heritage* 16 (Mar. 2020), pp. 133–141. DOI: 10.1016/j.daach.2019.e00133.

[39] Czesław Suchocki, Sebastian Okrój, and Wioleta Błaszczak-Bąk. “Methodology for the measurement and 3D modelling of cultural heritage: a case study of the Monument to the Polish Diaspora Bond with the Homeland”. In: *Reports on Geodesy and Geoinformatics* 116 (Aug. 2023), pp. 1–8. DOI: 10.2478/rgg-2023-0005.

[40] Jakub Markiewicz, Patryk Kot, Łukasz Markiewicz, and Magomed Muradov. “The evaluation of hand-crafted and learned-based features in TLS-SfM indoor point cloud registration: the case study of cultural heritage objects and public interiors”. In: *Heritage Science* 11.1 (2023), p. 254. DOI: 10.1186/s40494-023-01099-9.

[41] J. Liu. “Developing a Practice-Based Guide to Terrestrial Laser Scanning and Photogrammetry for Heritage Documentation”. In: *Heritage* (2025). DOI: 10.3390/heritage8080313.

[42] Alessio Cardaci, Pietro Azzola, and Versaci Antonella. “High-Altitude Architecture and Landscape: a Survey for the Conservation of Military Works at the Stelvio Pass”. eng. In: *DISEGNO 12* (2023), pp. 195–208. DOI: 10.26375/disegno.12.2023.20.

[43] Cristiana Achille, Andrea Adami, Silvia Chiarini, Stefano Cremonesi, Francesco Fassi, Luigi Fregonese, and Laura Taffurelli. “UAV-Based Photogrammetry and Integrated Technologies for Architectural Applications—Methodological Strategies for the After-Quake Survey of Vertical Structures in Mantua (Italy)”. en. In: *Sensors* 15.7 (July 2015). Publisher: Multidisciplinary Digital Publishing Institute, pp. 15520–15539. DOI: 10.3390/s150715520.

[44] T. Luhmann, M. Chizhova, D. Gorkovchuk, H. Hastedt, N. Chachava, and N. Lekveishvili. “Combination of terrestrial laser scanning, UAV and close-range photogrammetry for 3D reconstruction of complex churches in Georgia”. English. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2-W11 (May 2019), pp. 753–761. DOI: 10.5194/isprs-archives-XLII-2-W11-753-2019.

[45] Francesco Di Stefano, Stefano Chiappini, Fabio Piccinini, and Roberto Pierdicca. “Integration and Assessment Between 3D Data from Different Geomatics Techniques. Case Study: The Ancient City Walls of San Ginesio (Italy)”. en. In: *R3 in Geomatics: Research, Results and Review*. Ed. by Claudio Parente, Salvatore Troisi, and Antonio Vettore. Cham: Springer International Publishing, 2020, pp. 186–197. DOI: 10.1007/978-3-030-62800-0_15.

[46] Pesci Arianna, Giordano Teza, Loddo Fabiana, Massimo Fabris, Michele Monego, and Sara Amoroso. “Studio di possibili effetti sistematici nelle nuvole di punti SfM da APR: confronti con TLS, distorsioni e metodi di mitigazione. Evaluation of possible systematic effects in SfM UAV based point clouds: TLS and surface variations for error mitigation methods.” In: *Quaderni di Geofisica* 177 (Apr. 2022). DOI: 10.13127/qdg/177.

[47] S. Tanaka, K. Hasegawa, N. Okamoto, R. Umegaki, S. Wang, M. Uemura, A. Okamoto, and K. Koyamada. “See-through imaging of laser-scanned 3D cultural heritage objects based on stochastic rendering of large-scale point clouds”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* III-5 (2016), pp. 73–80. DOI: 10.5194/isprs-annals-III-5-73-2016.

[48] A. Scianna, G. F. Gaglio, and M. La Guardia. “Digital photogrammetry, TLS survey and 3D modelling for VR and AR applications in CH”. English. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLIII-B2-2020 (Aug. 2020), pp. 901–909. DOI: 10.5194/isprs-archives-XLIII-B2-2020-901-2020.

[49] Alessio Cardaci, Pietro Azzola, and Antonella Versaci. “Rilievo 3D e Ricostruzione Digitale della Fortezza di Bergamo: la tenaglia di Sant’Agostino e la cannoniera di San Michele”. In: *Defensive Architecture of the Mediterranean Vol. XX*. Vol. 20. DADI Press/edUPV, 2025, pp. 925–932.

[50] Alessio Cardaci, Antonella Versaci, and Pietro Azzola. “Mathematics and Geometry in the Nembro Reinforced Concrete Arch Bridge”. en. In: *Nexus Netw J* 25.S1 (June 2023), pp. 351–358. DOI: 10.1007/s00004-023-00712-5.

[51] Alessio Cardaci, Andrea Belleri, Paolo Riva, Pietro Azzola, and Luca Rota. “Un ponte tra storia e scienza. L’attraversamento del Carso a Nembro: approcci multidisciplinari per l’analisi delle infrastrutture”. In: *COSTRUENDO* 1.1 (2024), pp. 25–27.

[52] Irene Aicardi, Filiberto Chiabrando, Nives Grasso, Andrea Lingua, Francesca Noardo, and Antonia Spano. “UAV photogrammetry with oblique images: first analysis on data acquisition and processing”. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLI-B1 (June 2016), pp. 835–842. DOI: 10.5194/isprsarchives-XLI-B1-835-2016.

[53] Michele Russo and Anna Maria Manferdini. “Integration of image and range-based techniques for surveying complex architectures”. eng. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (2014), pp. 305–312. DOI: 10.5194/isprsannals-II-5-305-2014.

[54] M. Bolognesi, A. Furini, V. Russo, A. Pellegrinelli, and P. Russo. “Accuracy of cultural heritage 3D models by RPAS and terrestrial photogrammetry”. English. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XL-5 (June 2014), pp. 113–119. DOI: 10.5194/isprsarchives-XL-5-113-2014.

[55] F. Remondino, I. Toschi, M. Gerke, F. Nex, D. Holland, A. McGill, J. Talaya Lopez, and A. Magarinos. “Oblique aerial imagery for NMA: some best practices”. English. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences: XXIII ISPRS Congress*. International Society for Photogrammetry and Remote Sensing (ISPRS), June 2016, pp. 639–645. DOI: 10.5194/isprs-archives-XLI-B4-639-2016.

[56] Bahadir Ergun, Cumhur Şahin, and Furkan Bilücan. “Level of detail (LoD) geometric analysis of relief mapping employing 3D modeling via UAV images in cultural heritage studies”. In: *Heritage Science* 11 (Sept. 2023). DOI: 10.1186/s40494-023-01041-z.

[57] Cesare Pesenti. *Il cemento armato e la sua applicazione pratica. Formole, Tavole grafiche, Tabelle numeriche ed esempi pratici per il calcolo di costruzioni in cemento armato*. Società Italiana dei Cementi e delle Calci Idrauliche, 1913.

[58] Marco Pilotti, Andrea Maranzoni, Massimo Tomirotti, and Giulia Valerio. “1923 Gleno Dam Break: Case Study and Numerical Modeling”. In: *Journal of Hydraulic Engineering* 137 (Apr. 2011). DOI: 10.1061/(ASCE)HY.1943-7900.0000327.

[59] Ezequiel Ferrer, Francisco Agüera-Vega, Fernando Carvajal-Ramírez, and Patricio Martínez-Carricondo. “UAV Photogrammetry Accuracy Assessment for Corridor Mapping Based on the Number and Distribution of Ground Control Points”. In: *Remote Sensing* 12 (July 2020), p. 2447. DOI: 10.3390/rs12152447.

[60] Z. Niu, H. Xia, P. Tao, and T. Ke. “Accuracy Assessment of UAV Photogrammetry System with RTK Measurements for Direct Georeferencing”. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* X-1-2024 (2024), pp. 169–176. DOI: 10.5194/isprs-annals-X-1-2024-169-2024.

[61] Alessio Cardaci, Pietro Azzola, Michele Bianchessi, Ruggero Folli, and Simone Rapelli. “Comparative Analysis Among Photogrammetric 3D Models RAW Data vs RGB Images”. en. In: *Geomatics and Geospatial Technologies*. Ed. by Enrico Borgogno-Mondino and Paola Zamperlin. Cham: Springer International Publishing, 2022, pp. 271–282. DOI: 10.1007/978-3-030-94426-1_20.

[62] Michele Bianchessi, Simone Rapelli, Ruggero Folli, Pietro Azzola, Denny Coffetti, Monica Resmini, Alessio Cardaci, and Andrea Belleri. “La diga del Gleno: storia, rilievo, diagnostica e analisi strutturali nel centenario dal disastro”. In: *A partire da quel che resta. Il disastro del Gleno tra storia e paesaggio, memoria e futuro (1923-2023)*. FrancoAngeli editore, 2023, pp. 115–130.

[63] Simone Castelli, Andrea Belleri, Michele Persico, Luca Rota, Paolo Riva, Pietro Azzola, and Alessio Cardaci. “Preliminary structural assessment of a corroded RC beam in a Maillart bridge”. In: *Capacity Assessment of Corroded Reinforced Concrete Structures-CACRCS (2023)-Proceedings*. CTE, 2023, pp. 137–140.

[64] Jinru Xue and Baofeng Su. “Significant Remote Sensing Vegetation Indices: A Review of Developments and Applications”. In: *Journal of Sensors* 2017.1 (2017), p. 1353691. DOI: https://doi.org/10.1155/2017/1353691.

[65] Jonathan P. Dash, Grant D. Pearse, and Michael S. Watt. “UAV Multispectral Imagery Can Complement Satellite Data for Monitoring Forest Health”. en. In: *Remote Sensing* 10.8 (Aug. 2018). Publisher: Multidisciplinary Digital Publishing Institute, p. 1216. DOI: 10.3390/rs10081216.

[66] V. Santoro, G. Patrucco, A. Lingua, and A. Spanò. “Multispectral UAV Data Enhancing the Knowledge of Landscape Heritage”. English. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLVIII-M-2-2023 (June 2023), pp. 1419–1426. DOI: 10.5194/isprs-archives-XLVIII-M-2-2023-1419-2023.

[67] Maria Alicandro, Camilla Mileto, and José Luis Lerma. “Material Inspection of Historical Built Heritage with Multi-Band Images: A Case Study of the Serranos Towers in Valencia”. en. In: *Remote Sensing* 16.17 (Jan. 2024). Publisher: Multidisciplinary Digital Publishing Institute, p. 3167. DOI: 10.3390/rs16173167.

[68] Christine C. Gaylarde and Benjamin Otto Ortega-Morales. “Biodeterioration and Chemical Corrosion of Concrete in the Marine Environment: Too Complex for Prediction”. en. In: *Microorganisms* 11.10 (Oct. 2023). Publisher: Multidisciplinary Digital Publishing Institute, p. 2438. DOI: 10.3390/microorganisms11102438.

[69] S Airoldi, V Fioravante, and D Giretti. “The ISMGEO Seismic Geotechnical Centrifuge”. en. In: *EUROFUGE 2016*. Thorel, Bretschneider, Blanc and Escoffier eds, 2016, pp. 67–72.

[70] V Fioravante, D Giretti, C Prearo, and C G Lai. “Static and dynamic centrifuge modeling of landslide stabilization with large-diameter shafts”. en. In: *Proceedings of the 15th World Conference on Earthquake Engineering* (2012).

[71] A N Schofield. *Dynamic and Earthquake Geotechnical Centrifuge Modelling*. en. University of Missouri-Rolla, Rolla, Missouri, 1981.

[72] Gergana Antova and Ivan Peev. “Comparison on Commercial and Free Software for Point Cloud Processing”. EN. In: *Inżynieria Mineralna* Vol. 1.no. 1 (2024). DOI: 10.29227/IM-2024-01-57.

[73] Niloy J Mitra and An Nguyen. “Estimating Surface Normals in Noisy Point Cloud Data”. en. In: *Proceedings of the nineteenth annual symposium on Computational geometry* (2003), pp. 322–328.

[74] Xiao Wang, Jinglai Shen, and David Ruppert. *Local Asymptotics of P-Spline Smoothing*. arXiv:0912.1824 [math]. Dec. 2009. DOI: 10.48550/arXiv.0912.1824.

[75] Peter H. Broberg, Esben Lindgaard, Asbjørn M. Olesen, Simon M. Jensen, Niklas K. K. Stagsted, Rasmus L. Bjerg, Riccardo Grosselle, Iñigo Urcelay Oca, and Brian L. V. Bak. "HISAPS: High-order smoothing spline with automatic parameter selection and shape constraints". In: *SoftwareX* 29 (Feb. 2025), p. 102049. DOI: 10.1016/j.softx.2025.102049.