DIGITAL HERITAGE (2025)
S. Campana, D. Ferdani, H. Graf, G. Guidi, Z. Hegarty, S. Pescarin, and F. Remondino (Editors)

NL-2-SPARQL: Ontology-Based Natural Language Querying over 3D Point Cloud Knowledge Graphs

Matteo Codiglione, Fabio Remondino

3D Optical Metrology (3DOM) unit, Bruno Kessler Foundation (FBK), Trento, Italy
Email: (mcodiglione, remondino)@fbk.eu - Web: https://3dom.fbk.eu

KEYWORDS: Ontology, point clouds, LLM, SPARQL, natural language query, NL, Neuro-symbolic

ABSTRACT:

Point clouds are an asset in a wide range of applications, including heritage monitoring and conservation. However, querying capabilities over such 3D datasets, a fundamental task for their practical usability, is often highly challenging. In particular, difficulties arise when it comes to natural language querying, which can be in turn highly useful in lowering the technical barrier in inspecting 3D point clouds. The present paper explores various approaches for natural language querying over point clouds, leaning for the SPARQL-based query system allowed by the 3DOnt framework for point clouds management, recasting the problem as a natural language to SPARQL translation. After reviewing existing methodologies, we propose NL-2-SPARQL, a novel and flexible neuro-symbolic approach that integrates a Large Language Model (LLM) with a Graph Exploration and Query Building tool (GEQB). We then evaluate this method and demonstrate its application within the 3DOnt framework, highlighting its broader applicability to knowledge graphs in general, beyond this specific 3D-oriented context. A video presentation of the 3DOnt framework is available at https://3dom.fbk.eu/projects/3DOnt.

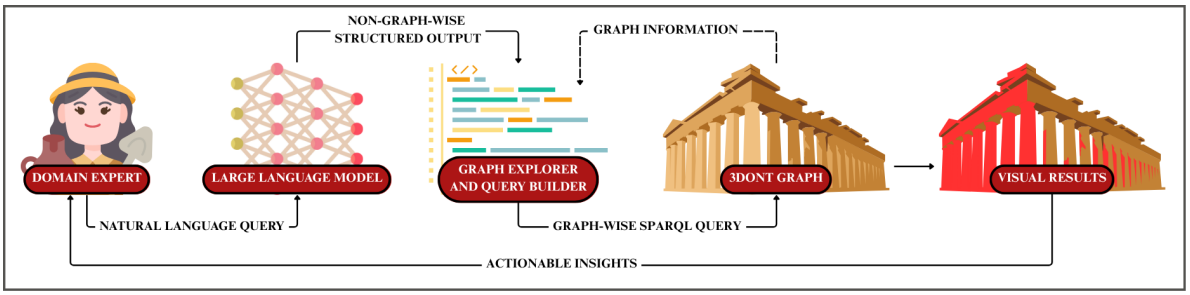


Figure 1: *The proposed NL-2-SPARQL pipeline.*

1. Introduction

3D point clouds carry a wide range of useful geometric and semantic information, which could serve a wide range of purposes. Many of them, such as heritage monitoring and preservation, urban planning and smart city related tasks, often regard a heterogeneous and complex set of data sources [RR15, AR21, LJT19, PGR*22, RC14] and are not of direct concern of the technical disciplines governing the collection of such data [SR16]. Consequently, the issue of lowering the technical barrier becomes pivotal, particularly when inspecting these datasets for the retrieval of geometric and semantic information, where the latter is often stored in the basic form of semantic classification [GDB*23, NSR*24, ZZCL19, XTZ20]. Query possibilities over point clouds [a] are often limited to a range selection over a single scalar field [Clo25], [b] technically require knowledge of some query language like SPARQL [S25, CBR24, CMR24] or [c] only rely on the direct application of NN-based feature extraction methods to point clouds in order to identify objects [HVP*23, HPZ*23, HPZ*24] or relationships [KVCN24] in an open-vocabulary yet semantically limited way. All these methods suffer from the same issue of not being able to fully answer the needs of a non-engineering employment of 3D point clouds. Such needs are:

(i) the possibility to perform queries without the knowledge of a structured query language, which is often lacking in expert of the concerned fields;
(ii) the possibility to query for relevant and highly structured semantic information, which are the most useful for tasks in such fields (e.g., the risk that a certain point develops leaching on the basis of its geometrical features and of the physical-chemical features of its material;
(iii) the possibility to query with enough syntactic complexity to express the logic required by the user's inspection needs (e.g., "show me all the columns which present an area affected by leaching which is comprised between 1 and 2 square meters);
(iv) the possibility to flexibly modify the application domain of the proposed solution in order to meet the specific needs of the specific tasks, which are often very differentiated and nuanced (i.e., the query solution should be robust to major or minor changes in the structure it queries over);
(v) the possibility to retrieve insights and explanations concerning the generation of semantic high-level features (e.g., to understand which factors were considered in determining that a certain column needed a certain invasive intervention), useful when dealing with tasks concerning irreversible operation on invaluable entities, which is often the case in cultural heritage.

As natural language querying most directly addresses the first need, this paper aims to implement such capabilities while satisfying the remaining four requirements. To do so, we first

DOI: 10.2312/dh.20253280

leverage on the 3DOnt framework [CMR24, CBR24] and on its native SPARQL [Spa25] query possibilities to recast the problem as a nl-to-SPARQL translation issue. Then, we propose a novel tool (NL-2-SPARQL) to perform such translation while preserving the discussed demands, showing its implementation into the 3DOnt framework.

2. Related works

2.1. Related works in NL querying over point clouds

In our review of recent research on natural language querying over point clouds, we identified three primary research directions. We will briefly introduce each of these approaches, referencing representative works and assessing their capability to address the last four needs outlined in the preceding paragraph.

The first research direction can be described as *direct open vocabulary nl-querying over a 3D point cloud based on the application of 2D vision-language models, where the link between 2D and 3D data is obtained by multimodal training.* The works of Hegde et al. [HVP*23], Huang et al. [HPZ*23, HPZ*24] and Peng et al. [PGJ*23] can be addressed as belonging to such a category. These approaches show themselves able in directly querying unclassified 3D point clouds in a fully open vocabulary way, enabling proper natural language querying. However, we observe that both approaches fail to address needs (ii) and (iii), as the publications suggest that their scope is limited to simple object detection and instance segmentation. Consequently, they do not support the retrieval of high-level semantic information or the expression of complex syntactic structures. Compared to other approaches, *OpenScene* [PGJ*23] demonstrates significantly broader capabilities, including the ability to query materials and nuanced features. However, it ultimately falls short of achieving truly valuable syntactic and semantic complexity. Moreover, need (iv) is only partially met by all these methods. The work of Hegde et al. faces the most significant challenges, as it relies on a training process involving the KITTY dataset [GLU12], making it unable to recognize any object categories which are not present in that dataset. In contrast, the work of Huang et al. is presented as highly responsive to specific fine-tuning, making it adaptable to the given task. However, this adaptability still relies on a dedicated fine-tuning process, which may limit its flexibility in handling previously unseen scenarios. Need (v) is completely not achieved by all the above-mentioned works.

The second research direction is related to *direct open vocabulary nl-querying on 3D point clouds based on the application of 2D vision-language models*. In this task, however, the bonding between 2D and 3D is based on the reprojection of 2D recognized objects into 3D point clouds coupled with region growing for results refinement. [AR24] is representative of such an approach. Their method manages to fully account for need (iv) and partially for need (v). The flexibility is granted by the lack of need for any multimodal learning, since the XGBoost-guided region-growing algorithm does the job of bonding 2D images and 3D point clouds and does so in a way in which the pre-training of the whole system only rely on 2D data, way more available and domain-covering than its 3D counterpart. Need (v) is only weakly and partially satisfied, since an *explainability checkpoint* is provided by the 2D recognized image, but everything before and after such *checkpoint* is entirely machine-learning based. Although the approach fails in completely accounting for needs (ii) and (iii), it is worth noting that both are eventually partially satisfied, as the system supports non-atomic queries and can incorporate additional features such as object colors.

The third and last research direction concerns *nl-query-guided graph extraction from 3D point cloud with the usage of knowledge distillation from 2D visual language models.* [KVCN24] is a representative example of this approach. Their proposal partially addresses needs (ii) and (iii), as the extracted graphs incorporate relationships between individuals, enabling a richer representation of semantics and syntax. However, this improvement remains limited, as the level of semantic and syntactic complexity achieved is too low to effectively meet the practical demands of specialized fields such as heritage preservation. Need (iv) is successfully met due to the strong zero-shot capabilities of the proposed solution, whereas need (v) remains entirely unaddressed.

2.2. 3DOnt graphs and problem recasting

All the approaches presented above mainly suffer in accounting for needs (ii) and (iii). Following this detection, we start to take into consideration a fourth approach, which is not originally oriented to NL-querying, but which strongly insists over semantic and syntactic complexity. Such approach has been presented in two subsequent papers [CBR24, CMR24], and describes an ontology-based framework (3DOnt) in which a classified 3D point cloud undergoes an inferential semantic enrichment and an object-level restructuring and becomes represented as a graph. This representation is based on a 1-to-1 correspondence between points of the point cloud and individuals (nodes) in the graph, but it then builds on this to add other semantic representational layers to the graph. Such additional levels comprise the one of the *macro-objects* and the one of the *domain knowledge*. Specifically, macro-objects - constituted by points- and domain-knowledge-related entities - degradations, materials - are modeled as individuals; low- and high-level features, both physical-chemical (e.g., solubility, porosity), geometrical (e.g., surface value and dimensions of macro-object, coordinates and color of points) are modeled as data properties. Object relationships are used to tie together entities of the same representational level (to express relationships between domain knowledge entities – e.g., *this degradation cannot affect this material* - or between objects (e.g., *this temple contains these columns*) or to bind together different levels (e.g., to link points and the objects they constitute or to the degradations they suffer from). Such graph can then be further enriched with additional data and undergo explainable reasoning processes to be supplemented with high-level features based on the low-level ones. As a graph, it is also natively prone to be queried in SPARQL, the expressivity of which – equivalent to the of Relational Algebra [AG08] – can fulfill (iii). Moreover, 3DOnt framework is generalizable, since the ontologies on which it relies regard various domains and can be adapted ad hoc by the user. 3DOnt framework already satisfies all but (i), given the technical barrier raised by the required SPARQL language knowledge. We can thus recast the main problem of the paper from a NL-to-Point Cloud issue to a NL-to-SPARQL one, more manageable and already promising in accounting for the above established needs. A proposal which performs NL-to-SPARQL query translation, implemented in 3DOnt framework in a way in which the accounting for all but (i) by such framework is preserved, would not only be a NL-to- SPARQL solution but also a NL-to-Point Cloud one.

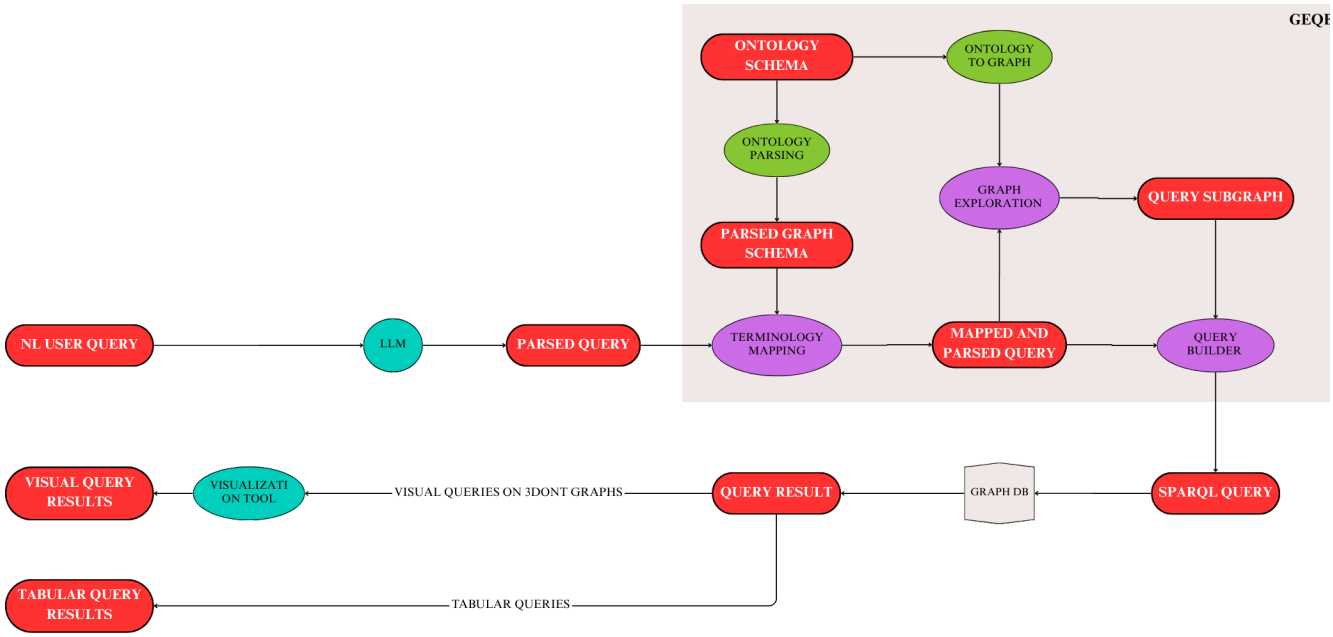


Figure 2: *NL-2-SPARQL full pipeline featuring an LLM + GEQB architecture. User NL query gets parsed by a fine-tuned LLM, which returns a non-graph-wise structured output. This is then processed by the Graph Explorer and Query Builder, which accounts for the mapping of employed terms and for the graph-exploration driven construction of the query subgraph. The graph exploration both ensures compatibility with every graph structure and prevents the user for being required to know the inner structure of the graph. Then, the proper SPARQL query is constructed and performed.*

Compared to the original shape of the problem, this novel version must deal with the preprocessing step of the 3DOnt framework - which takes in input a classified point cloud and generates a 3D graph- and to give up to the directedness of the above-mentioned approaches. Given the fully automatic nature of the preprocess performed by 3DOnt and given the advantages in accounting for the needs (ii)-(v), we consider this drawback acceptable.

2.3. Related works in NL to SPARQL translation

Now that the problem is recast, we will explore some proposed solutions for nl-to-SPARQL translation tasks, to find the best candidate for 3DOnt implementation. We identified two main research branches.

The first one regards *template-based approaches.* The works of Athreya et al [ABN*21] and Litvin et al [LVK21] can both be used as representative of this branch. Even if by mean of different methods, the core idea of both approaches is to extract the entities concerned in the query, to match the nl query with the right predetermined SPARQL template (chosen amongst a set) and to fill such template with the concerned entities. These methods, if implemented in 3DOnt, will only preserve the accounting of (v), while partially compromising all the others. Both the possibility to account for (ii), (iii) and (iv) would heavily suffer the rigidity imposed by the templates, while the possibility to account for (iv) would also be specifically compromised by the labor-intensive job which would be required every time the user modifies something in the ontology structure. Since the SPARQL queries must strictly match the graph structure, every major modification in the structure would involve a modification in the templates.

The second main research branch concerns *Seq2Seq-based approaches*, of which the work from Panchbhai et al [PSM20] has been chosen as a representative. In this approach a Neural Machine Translation process is directly applied to convert the natural language query in a SPARQL query functioning on the graph on which the network has been trained. Here (ii) and (iii) are fully preserved, due to the model effectiveness in handling complex syntaxis, which in turn is needed to fully extract the complex semantic encoded in 3DOnt graphs. However, this approach completely neglects (iv), since the model is heavily dependent on training, and as such can work only on a certain, very specific, graph structure, and (v), given the opaque nature of the neural processes involved. Implementing this proposal to solve the NL-to-SPARQL problem in 3DOnt framework would result in a loss of the 3DOnt *virtually open vocabulary* feature, which is the possibility of the domain ontology to be adapted by the user in the graph structure in every desired way *before* importing the cloud. Once the cloud is imported the vocabulary is fixed, but the user has the possibility to decide freely the structure of the incoming graph. Giving up on this possibility by implementing a NL-based query system which requires graph immutability does not seem a reasonable step to be taken.

3. NL-2-SPARQL: The LLM + GEQB approach

3.1. Introduction to the approach

Due to the problems affecting the approaches reviewed in the previous paragraph, we developed an ad-hoc approach meant to preserve all needs amongst (ii), (iii), (iv), (v). We argue that, to do so, an approach needs to be (a) not graph-specific, (b) graph exploring and (c) not template-based. (a) grants the approach to be able to operate on different graphs, preserving the possibility for the user to modify the 3DOnt ontologies according to his needs, while (b) helps both in achieving (a) itself and to strengthen the accounting for (i). A graph-exploration guided nl-to-SPARQL tool can lift from the user both the technical burden of knowing

FEATURES CURRENTLY SUPPORTED BY THE LLM	FEATURES NOT FULLY SUPPORTED YET BY THE LLM
Select queries: *Standard SPARQL queries with a SELECT construct. they return a numerical value or some other tabular result.*	Superlatives: *ORDER BY ASC, ORDER BY DESC, and associated LIMIT constructs in SPARQL.*
Show queries: *queries which output is compatible with the viewer, specific for 3DOnt implementation.*	Implicit relationships: *object relationships expressed with "with", "of", "'s" in the natural language query.*
Instance count: *filtering on the number of objects which are in a certain relation with a subject.*	High syntactic complexity: *very complex queries are still not fully supported.*
Grouping functions: *AVG, SUM, HAVING and COUNT constructs in SPARQL (both global and grouped.*	Sameness/difference between individuals: *Queries when it is semantically relevant to distinguish or identify two variables of the same class.*
Inner OR: *OR applied to different values of the same data property, also with nested conjunctions.*	Outer XOR: *XOR applied to instances of different data properties or object relationships.*
Outer OR: *OR applied to instances of different data properties or object relationships.*	
Inner XOR: *XOR applied to different values of the same data property.*	
Logical operators nesting: *Nesting of one or more logical operators within the scope of another.*	
Simple data property filters: *Filters on data properties.*	
Data property filters with inner operations: *Filters on data property values which contain inner operations between variables.*	
Explicit string filters: *string filters on data properties which range over strings, comprised within double quotes in the NL query.*	

Table 1*: GEQB features currently supported or not by the LLM. Such distinction is defined on the basis of the evaluation process presented in Table 2, which revealed two main distribution groups.*

SPARQL language and the one of being aware of the graph structure itself, which constitutes a lighter form of technical barrier. The proposed approach (NL-2-SPARQL) leverages both an LLM and a GEQB (Graph Explorer and Query Builder) to translate natural language queries into SPARQL queries operating on a real-time explored 3DOnt graph. The basic pipeline is shown in Figure 1, while a detail explanation is provided by Figure 2 and an implementation example is presented in Figure 3.

3.2. The LLM

The first step of NL-2-SPARQL pipeline is performed by a fine-tuned GPT-4o-mini. Details regarding the training dataset and procedure are presented in Section 4. The fine-tuned LLM takes in input a natural language query and returns in output a structured parsing of it. Such parsing is meant to represent in a format compatible with the GEQB the syntactic structure and the semantic content of the user's natural language query. Is worth noting that this process is not graph-specific. The LLM is only trained to parse natural language queries following a schema which is not dependent on particular graph structures. While a more exhaustive evaluation is demanded to section 4, we briefly present in Table 1 the query features which are already supported after the first fine-tuning and the ones which instead would require a new training to be properly managed. Before being fed into the GEQB, the LLM's output gets preprocessed by a dedicated tool, which helps handling cases of missing commas or quotes. Specifically, the structured parsing consists in five python lists (Figure 3):

- L1 the vocabulary used and broadly associates each term with its semantically closest counterparts among the others;
- L2 the specific instances corresponding to each word identified in the first list, distinguishing between different occurrences and, for words recognized as properties or relationships, linking them to their respective subjects and objects;
- L3 logical structures, representing logical operators along with the elements under their scopes - nested logical expressions are also included here;
- L4 filters related to numerical values or strings, detailing any constraints applied in the query on property values, string matches, or, in the case of relationships, on the number of instances (it also covers SPARQL constructs like SUM, AVG, or COUNT when they apply to elements in the WHERE clause rather than the SELECT clause);
- L5 the query head. i.e. the content of the SELECT clause, indicating what the query is intended to retrieve. If the query involves aggregation functions like sum, average, or count, these are noted accordingly with SUM, AVG, or COUNT.

3.3. The GEQB

The second and wider step is performed by the Graph Explorer and Query Builder (GEQB). In the first place, a mapping occurs between the words employed in the natural language query and the vocabulary defined within the ontology schema to which the graph refers. An exact match is tried, and in case of failure a normalized average between a jaro-winkler and an edit distance score is calculated. If it does not trespass the established threshold, all the process is repeated with the wordnet synonyms of the concerned word. If this also fails, the user is asked for a manual mapping. At the end of the mapping process, the user is asked for a confirmation to enhance control. After the mapping a continuous subgraph is created with the concerned terms and consequently enriched with the class instance variables present in the parsed query - such subgraph comprises all the nodes which separate the concerned nodes, so that the user is not required to know the inner structure of the graph (e.g., if *column* and *porosity* are two related terms in

the query, and if in the graph a *material* class plays the role of the middle term between the *Column* class and the *Porosity* data property in a way that *Porosity* does not directly contain *Column* in its domain, the subgraph will still correctly include the *Material* node). Then, using the parsed query for guidance, a set of paths is generated on such graph to link couples of semantically related words. These paths can be bonded to pass through a certain node, if a relationship is specified, so that their semantic accuracy is preserved. For each pair of terms, multiple paths may be identified. Before selecting the shortest one, a filtering step is applied to remove candidate paths that exhibit a dome-like structure. We define a dome-like structure as any path that ascends and descends the class hierarchy without traversing a *rdfs:domain* or *rdfs:range* edge. This filtering is intended to eliminate false inheritance and prevent the inclusion of semantically meaningless paths. After filtering, each path is converted into a SPARQL snippet. In this transformation, any *rdfs:subClassOf* chain of classes between a *rdfs:range* and an *rdfs:domain* edge is replaced by a class variable, which is linked to the rest of the snippet through the properties or relationships associated with the enclosing *rdfs:domain* and *rdfs:range*. Once this is done, filters are considered and stored in a format ready to be inserted in the final SPARQL query. At this point logical operators are analyzed, ordered by their scope and utilized to combine all the SPARQL snippets in the right logical shape. Finally, the query head is generated and the full, working SPARQL query is assembled. At the present time, the GEQB supports all the features considered in Table 1, also the ones not yet supported by the fine-tuned LLM. Furthermore, the GEQB manages to understand as an implicit string filter words which are directly semantically related to the concerned data property, and which are not recognized during the mapping process. However, further work on the GEQB algorithm is still needed in order to also support: [a] queries explicitly featuring resources defined within RDF, RDFS and OWL ontologies, [b] queries which need subqueries, such as queries with multiple groupings needed or queries where the result of a grouping function falls inside the scope of a logical operator, [c] ASK queries, [d] queries for the insertion of data, [e] queries where a string value is neither explicitly marked by double quotes nor directly assigned to the data property which is relative to, [f] graph structures relying on multiple namespaces, [g] Queries featuring data property filters in which both terms of the comparison are compositional (i.e. feature inner operations).

We want to stress that the presence of the GEQB is crucial for two main reasons. The first one is that it ensures robustness to structural changes in the ontologies. SPARQL queries are specific for a certain graph structure, so that purely LLM-based approaches which directly translates into SPARQL need to be trained on a *specific graph structure*. Not only there is no possible generalization to other ontologies, but even minor modifications in the ontology structure would result in a need for retraining. Such a solution would completely fail in addressing need (iv). The second reason is that it prevents the user to be required to know the inner structure of the graph. Beside highly specific cases (Section 6, RDF reification), the user is neither required to know the vocabulary nor the structure employed within the graph. Mapping and graph exploration processes will suffice for this lack of isomorphism between the NL and the graph structure, producing a SPARQL query which reflects the former and works over the latter.

Figure 3: *NL-2-SPARQL implementation within the 3DOnt framework, presenting both a particular case of a "show query" resulting in a visual plotting and the general shape of the structured parsing. The five lists (L1-L5) are here represented.*

	ALREADY SUPPORTED FEATURES	NOT SUPPORTED YET FEATURES
SEMANTIC COMPLETENESS: *Does the parsing encode for all semantic features of the NL query?*	(93/100). Usually every entity, property and relationship have its token in the output, only in rare cases the parsing leaves behind semantic content. Of the 7 wrong outputs, only 2 were heavily wrong.	(70/100). Semantic content is recognized decently. identity or difference between individuals and implicit relationships are particularly problematic.
LOGICAL CORRECTNESS: *Does the parsing correctly encode for the logic implied by the NL query?*	(79/100) Logical structures are often correctly mapped, even for nested logical operators.	(55/100). Highly problematic when logical operators nesting reaches the third grade (which is actually a rare case in real-scenario natural language queries). Outer XOR structure is also very problematic.
FORMAL CORRECTNESS: *Is the parsing python readable and compatible with the GEQB?*	(68/100) Both the python readability and the GEQB compatibility have been achieved except for 32 cases, 27 of which have been correctly managed by the preprocessing tool.	(57/100). The main issue here is provided by the superlative featuring queries. This feature is parsed in a python readable but not GEQB compatible way. 31/43 cases have been handled by the preprocessing tool, while 8 out of the remaining 12 were the ones featuring superlatives.

Table 2: *Evaluation of the fine-tuned GPT-4o-mini.*

3.3. LLM+GEQB application to 3DOnt graphs

In implementing NL-2-SPARQL tool into the 3DOnt framework, we had to consider a specific aspect. While normal queries (i.e., queries which output a tabular result containing the concerned variables) are still useful when querying 3D data ("Which is the average height of the columns?"), there is a kind of query which only arises in this specific context, the *show query* (Figure 3). Such a query, disregarding the directly concerned entities, implicitly aims at their spatial coordinates, in a format compatible with the viewer. For instance, the query "Show me every column" would implicitly be understood as "Select the x, y, z values of the points which constitute the columns". Even if *show queries* are not a SPARQL primitive, they are hereby worth to be distinguished as a particular kind. This is due to the 3D orientation of the specific implementation and to technical details of the viewer, and ultimately aims at providing the most intuitive access possible to the user. To manage this specificity, we dedicated a portion of the training dataset (50%) to natural language queries with a prompt word related to visual plotting, training the model to automatically consider the associated implicit select query. The dedicated portion is this large in percentage since this first fine-tuning process is aimed at 3DOnt implementation (same holds for the prevalence of urban and heritage-related queries, described in the next section. The aim was to make the LLM familiarize with niche terms.).

4. LLM training specifics

4.1. Training dataset

The training dataset has been created ad hoc, given the custom nature of the structure of the parsing. It comprises 2000 rows, which are divided along three axes:

Prompt word:
[a] Standard SELECT queries (35%)
[b] COUNT queries (15%)
[c] SHOW queries (50%)

Semantic domain:
[a] Heritage related queries (40%)
[b] Urban related queries (40%)
[c] Random semantic domain (20%)

Formal structure features
[a] Logical operators at various levels of nesting (30%)
[b] Filters or operations on data properties (30%)
[c] Filters on object properties (20%)
[d] Explicit string filters (5%)
[e] Identity/difference filters between individuals (5%)
[f] Queries mixing the previous categories at high level of syntactic complexity (10%)

4.2. Model evaluation

In evaluating the fine-tuned LLM we had to face two main issues. In the first place, the model is dedicated to the very specific task of parsing a natural language query according to a structured output format which is non-standard, and from which we expect both semantic coherence and purely formal correctness. Metrics like BLEU [PRW*02] or ROUGE [Lin04] would correctly address the first aspect, but would miss the second, while metrics available for structured outputs are dedicated to standard formats like JSON or XML [LLW*24]. Secondly, the model has only undergone an initial fine-tuning with a limited dataset. It should not be regarded as a final version which would require a proper benchmarking process - but rather as a provisional iteration. The primary objective at this stage is not to achieve definitive performance but to obtain results that are sufficiently promising to validate the approach as a viable proof of concept. Developing a dedicated formal metric would be a reasonable approach only after conducting a more extensive fine-tuning with a larger dataset, ensuring a robust foundation for evaluation. However, generating such a dataset is highly time-intensive and demands substantial dedicated effort. Consequently, we have deferred its creation to a subsequent phase of our research. Nevertheless, given that the current results are sufficiently robust to demonstrate the validity of our approach, we opted for a provisional human-based quantitative and qualitative evaluation of the model. With reference to Table 1, we can distinguish between two main categories of query features,

the already supported and the not supported yet ones. We will divide the evaluation for these two categories, since the answers for the particular features will mainly be possible to group according to the same distinction. We carry on such evaluation in Table 2, considering the results for 200 (100 for each main group) test natural language query. Is important to notice that the success rates of the three considered parameters are mainly independent, so that the overall success rate does not result from a function of them. Although the model requires additional fine-tuning to achieve the level of reliability necessary for practical application, we overall regard the performance as highly promising, suggesting that the primary challenges can be mitigated by increasing the size of the fine-tuning dataset.

a) "Which is the average height of the columns?"	7.41200531
b) "Provide me with the sum of the surface values of all the floorings"	906.43143306
c) "Provide me with the number of columns"	62
d) "For each column, which is the average height of the objects which are part of the capital? (such capital should be contained in the column)"	[0.47315507, 0.93455377, 1.52868512, 0.92443973, 1.56493735, …, 0.9219001]

Table 3: *Examples of tabular results related to: Numerical query featuring a global average (a); Numerical query featuring a global sum (b); Numerical query featuring a count (c); Numerical query featuring a grouped average (d).*

5. Results

Before moving to the presentation of the results, we here stress out how the above-conducted evaluation, while it helps in assessing the concrete possibilities of NL-2-SPARQL in performing a specific step - the structured parsing of the natural language query - of a comprehensive pipeline which NL-2-SPARQL itself introduced, it does not provide meaningful insights about the comparison between NL-2-SPARQL and the other SotA solutions of the nl-to-SPARQL problem. This is because what really makes NL-2-SPARQL standing out is not single-handedly the fine-tuned LLM, but rather the LLM+GEQB architecture as a whole, which allows NL-2-SPARQL not only to solve the nl-to-SPARQL problem, but to do so in an explainable, graph-wise, user-friendly and generalizable way. After the discussion of the specific visual results, we are going to present all the features which make NL-2-SPARQL an *unicum* in the SotA. Along with Figure 3, Figure 4 and Table 3 show other sets of natural language queries (and associated visual or tabular results) performed over the 3DOnt graph representation of our test 3D dataset. Such dataset represents the Temple of Neptune in Paestum, Italy, [GR19] and comprises a classified (10 classes) terrestrial and UAV photogrammetry point cloud of the abovementioned temple. Figure 4a Presents the original cloud, while Figure 4b, 4c present respectively its semantic and instance segmentation, to provide some ground truth for the following images. Figure 4d-4i show examples of visual queries featuring relationships between individuals, simple and compositional filters on data properties and some logical operators, while Table 3 presents a set of numerical queries featuring global and grouped AVG, SUM and COUNT SPARQL constructs. Notably, we only leveraged geometrical and semantic features – the only contained in the employed dataset, but with additional data available, leveraging 3DOnt's data integration capabilities, more advanced queries could have been performed. In the case of visual queries, the natural language queries have been prompted within NL-2-SPARQL-implemented 3DOnt user interface, then converted by our fine-tuned GPT4o-mini into the structured parsing compatible with the GEQB and finally converted by it into a SPARQL query operating on 3DOnt's Heritage Ontology Schema.

Lastly, through a SPARQL endpoint, the query has been executed on a Virtuoso local indexed quadstore containing the 3DOnt graph representation of the Neptune Temple dataset. The visual results have been then plotted on CloudCompare [Clo25], according to the 3DOnt pipeline. Numerical queries have been processed and executed in the same way, except for being directly shown within 3DOnt user interface. Errors like the number of columns (which are in fact more than 62) are to be imputed not to NL-2-SPARQL, which just consults an already structured knowledge graph, but rather to 3DOnt. In the case of the number of columns, the error lies in how its instance segmentation algorithm handles cases of very near objects of the same class (Figure 4c). Besides these examples, we here underline the core strengths showed by our proposed solution which we hereby call NL-2-SPARQL:

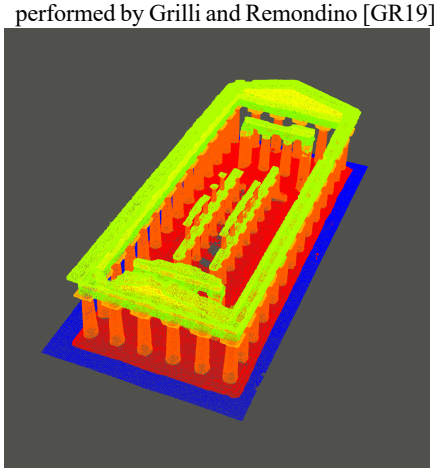
Lowering the technical barrier: NL-2-SPARQL addresses (i) on three different levels:

a) **the user is not required to know SPARQL**, since NL-2-SPARQL tool directly manages the translation between natural language queries into SPARQL queries operating on the concerned particular graph;
b) **the user is not required to know the exact vocabulary employed in the graph**, due to the GEQB tool automatically performing a mapping of the words employed into the natural language query into the vocabulary of the ontology schema and user manual confirmation is employed to augment control;
c) **the user is not required to fully know the inner structure of the graph**, since the GEQB autonomously explores the graph structure and finds effective paths between the nodes highlighted in the parsing. In this regard, NL-2-SPARQL achieves the same usage-ease of the template-based solution, without giving up to the flexibility which characterizes neural-based approaches and the robustness of queries which are built graph-wisely.

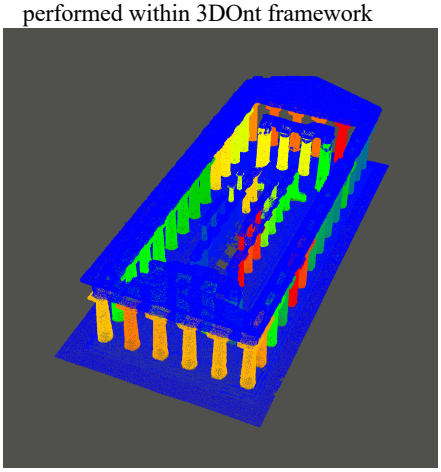
Generalizability: need (iv) is completely fulfilled. Due to the role of the GEQB, NL-2-SPARQL can adapt to every graph structure, both allowing the 3DOnt user to modify the structure of the employed domain ontology – and as such allowing him to model its 3D point cloud with different semantic structures and features and to still be able to query over them and thus to practically exploit them – and being prone to use-cases which go beyond querying over 3D point clouds, towards a broader applicability to knowledge graphs in general.

a) The unclassified RGB point cloud employed for test

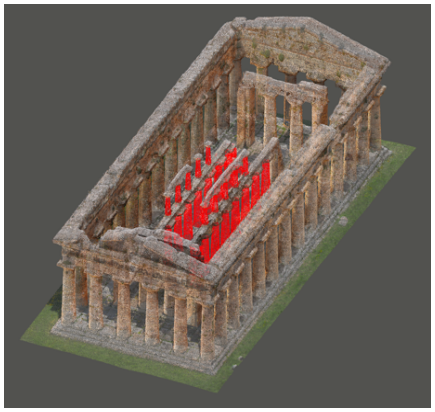
b) Semantic segmentation of the cloud performed by Grilli and Remondino [GR19]



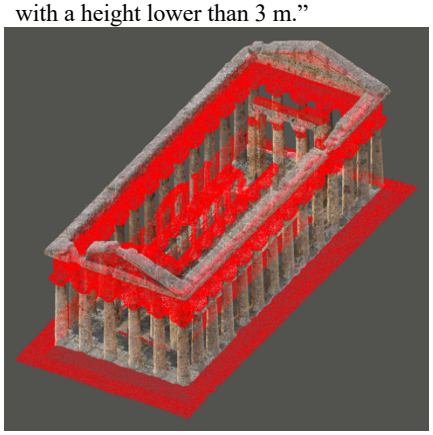
c) Instance segmentation of the cloud performed within 3DOnt framework



d) “Show me all the shafts belonging to columns and which are not higher than 7 m.”

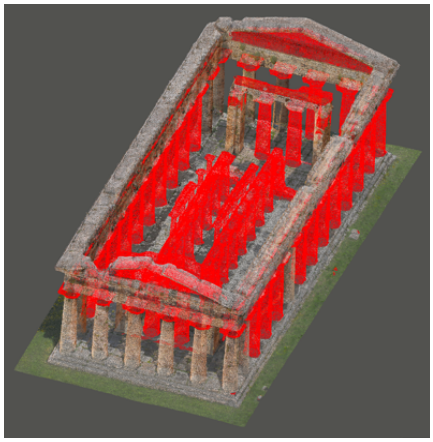


e) “Present me visually with all the objects with a height lower than 3 m.”

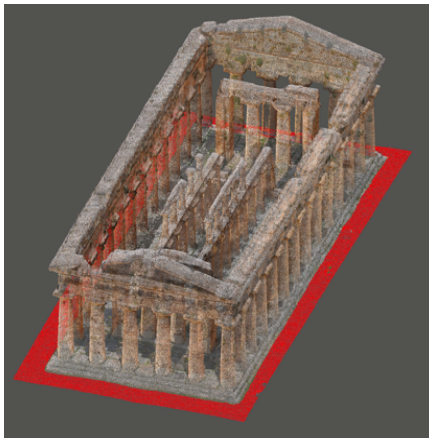


f) “I want to see all the capitals, but only if between the echinus or the abacus they contain there is a surface value between 1 m and 2 m.”

g) “Would you show me all the entities which are part of the building and for which the product between the height and the surface value is lower than 180 m^3?”



h) “Plot on the 3D model every environmental element.”



i) “Highlight all the objects which have a max Z lower than 4 m or a min Z higher than 7 m (max Z should be parsed as a unique expression).”

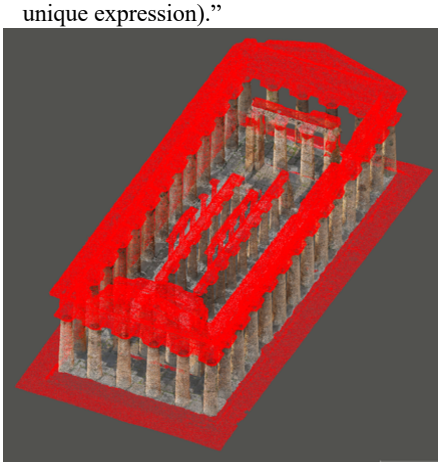


Figure 4: *Ground truth and Some visual results: Original point cloud (a); Semantic segmentation of the cloud (b); Instance segmentation of the cloud (c); Visual query featuring relationships and NOT operator (d); Visual query featuring a simple data property value (e); Visual query featuring an outer OR operator, relationships and filters on data properties (f); Visual query featuring relationships and compositional filters on data properties (g); Visual query featuring ontology’s classes not defined within the original las file classification (h); Visual query featuring an outer OR operator, filters on data properties and an explicit tokenizing prompt (i).*

Considering the nl-to-SPARQL SotA, we can highlight how the approaches which achieve some generalizability always do so at the cost of sacrificing their graph-wiseness, while none of them manages, like NL-2-SPARQL does, to feature both. Moreover, direct approaches to natural language querying over 3D point clouds, even if zero-shot generalizable, always achieve such a result by giving up to both syntactic and semantic complexity - (ii) and (iii) - and explainability (v).

Syntactic and semantic complexity: Even after this first provisional fine-tuning, NL-2-SPARQL is already able to manage syntactical and semantic complexities which other methods cannot handle in the same priceless way. This allows the user to fully extract structured knowledge contained in the 3D point cloud modeled in the 3DOnt framework as a 3D Graph, allowing NL-2-SPARQL to completely fulfill needs (i) and (ii), outclassing other methods. This not only holds for nl-to-SPARQL solutions, but also for direct approaches for natural language querying over 3D point clouds. Amongst all the presented methods from both the SotAs, the generalizable ones can at maximum extract simple semantic relationships holding between entities, while the Seq-2-Seq approaches, the only ones which achieve such complexities, are able to do so only at the cost of sacrificing their generalizability in favor of a rigid graph-specificity.

Explainability: NL-2-SPARQL, as featuring a composite pipeline, only employs an opaque neural-based process in its very first step - namely, the natural language query structured parsing. Everything after is fully algorithmic and, as such, explainable. The structured parsing output is the first and main *explainable checkpoint*, so that the user not only can know which query is actually executed on the 3D Graph – notice that this explainability level is achieved by all the reviewed nl-to-SPARQL methods – but can also understand how the parsed query has been processed and elaborated in its way to become a proper SPARQL query. Also under this regard, NL-2-SPARQL shows itself as out-performing the SotA.

To sum up, NL-2-SPARQL meets (i) in a way in which it effectively preserves 3DOnt native capacity to account for (ii), (iii), (iv) and (v). Besides the LLM fine-tuning evaluation considerations - which are indeed promising enough to allow us to present NL-2-SPARQL as a concretely viable approach - none of the other presented methods, both in natural language querying over 3D point clouds and in nl-to-SPARQL translation, manages to fulfill all of them without sacrificing one or more. Apart from NL-2-SPARQL validity as an independent tool for natural language-to-SPARQL translation, these results allow us to consider NL-2-SPARQL's implementation into 3DOnt framework an effective solution for the nl-to-Point Clouds issue, and as such a viable way to lower the technical barrier addressed in the introduction, providing easier access to fundamental data assets to experts engaged with challenges in non-engineering sectors, such as cultural heritage monitoring and preservation.

6. Conclusions and further research

The actual NL-2-SPARQL's implementation into 3DOnt framework is a promising approach for natural language querying over point clouds. Moreover, beside 3DOnt implementation, NL-2-SPARQL has a wider range of possible applications as an autonomous solution to the NL-to-SPARQL issue. Such an application domain concerns every data which is structured into a knowledge graph. It was presented how NL-2-SPARQL manages to perform nl-to-SPARQL while fulfilling more demands than the other SotA solutions, by featuring a novel approach which combines an LLM and an algorithmic tool for graph exploration and query construction (GEQB).

Further research will address some limitations which, at the present stage of development, affect NL-2-SPARQL's usage:

- In case of RDF reification, if the user's query regards multiple properties of the same reified entity, it currently has to be isomorphic to the graph structure under this regard. (i.e. not "give me the average of the temperatures dated 2024" but rather "give me the average of the temperatures of the acquisitions dated 2024", in a scenario where both "Temperature" and "Date" are properties of "Acquisition"). This is a quite specific case – only occurring in 3DOnt while querying for imported additional raster-encoded properties – but, given its possibility, it will be addressed in further developments by allowing for a more intelligent exploration in other scenarios.
- currently, NL-2-SPARQL considers each token in the natural language query as corresponding to a single node in the graph. Terms with an implicit semantic would not be "unpacked" (i.e., the word "part", which implicitly mean "an object which is part of". In cases like this the user should not formulate his query as "select every part of the column" but rather as "select every object which is part of the column"). For each term employed in the query, in cases of implicit semantic content, the user currently must make this semantic content explicit. This limitation will be addressed by providing to the LLM, as context, the vocabulary of the ontology schema, where each word is annotated with its type. In such a way the LLM could try to find within the list one or more nodes with, together and in a certain relationship, could suffice for the expression of the full implicit semantic content.
- to apply NL-2-SPARQL over point clouds these must have been already imported into 3DOnt framework, which currently requires them to be already classified. Nonetheless, we are already working on an implementation of an open-vocabulary internal classification tool within the 3DOnt framework, so that, as a whole, the 3DOnt framework enriched with both such tool and NL-2-SPARQL could provide the possibility to perform natural language querying on originally unclassified point clouds.

Acknowledgements

The work is partially funded by the 3D-4CH - Online Competence Centre in 3D for Cultural Heritage – EU project (GA 101195149). The training dataset has been prepared by Margherita Bianchetti, who also cured paragraph 4.1.

References

[ABN*21] ATHREYA, R. G., BANSAL, S., NGONGA NGOMO, A.-C., USBECK, R.: Template-Based Question Answering Using Recursive Neural Networks. *Proc. 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021)* (2021), pp. 1234-1245.

[AG08] ANGLES, R., GUTIERREZ, C.: C.: The Expressive Power of SPARQL. *The Semantic Web – ISWC 2008. Lecture Notes in Computer Science*, (2008), 5318.

[AR21] ADAMOPOULOS, E., RINAUDO, F.: Close-range

Sensing and Data Fusion for Built Heritage Inspection and Monitoring. *Remote Sensing, 13* (2021), 3936.

[AR24] ALAMI, A., REMONDINO, F.: Querying 3D Point Clouds Exploiting Open-Vocabulary Semantic Segmentation of Images. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLVIII-2/W8* (2024), pp. 1-7.

[Clo25] CLOUDCOMPARE. https://www.danielgm.net/cc/, 2025.

[CBR24] CODIGLIONE, M., BEBER, R., REMONDINO, F.: Leveraging Ontology for Enhanced Queries and Analyses of Urban Point Clouds. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., 48(2/W8)* (2024), pp. 101-108.

[CMR24] CODIGLIONE, M., MAZZACCA, G., REMONDINO, F.: Dimensional discoveries: Unveiling the potential of 3D heritage point clouds with a robust ontology framework. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., 48* (2024), pp. 119-125.

[GDB*23] GRILLI, E., DANIELE, A., BASSIER, M., REMONDINO. F., SERAFINI. L.: Knowledge Enhanced Neural Networks for Point Cloud Semantic Segmentation. *Remote Sensing,* 15.10. (2023), 2590.

[GLU12] GEIGER, A., LENZ, P., URTASUN, R.: Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. *Proc. CVPR*. (2012).

[GR19] GRILLI, E., REMONDINO, F.: Classification of 3D Digital Heritage. *Remote Sensing,* 11.7. (2019), 847.

[HPZ*23] HUANG, R., PAN, X., ZHENG, H., JIANG, H., XIE, Z., SONG, S., HUANG, G.: Joint Representation Learning for Text and 3D Point Cloud. *arXiv preprint arXiv:2301.07584* (2023).

[HPZ*23] HUANG, R., PAN, X., ZHENG, H., JIANG, H., XIE, Z., SONG, S., HUANG, G.: Joint Representation Learning for Text and 3D Point Cloud. *Pattern Recognition* 137 (2024), 110086.

[HVP*23] HEGDE, D., VALANARASU, J. M. J., PATEL, V. M.: CLIP Goes 3D: Leveraging Prompt Tuning for Language-Grounded 3D Recognition. *Proc. ICCVW* (2023).

[KVCN24] KOCH, S., VASKEVICIUS, N., COLOSI, M., NAVAB, N.: Open3DSG: Open-Vocabulary 3D Scene Graphs from Point Clouds. *arXiv preprint arXiv:2401.09413* (2024).

[Lin04] LIN, C.: ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, (2004), pp 74-81.

[LJT19] LIN, D., JARZABEK-RYCHARD, M., TONG, X., MAAS, H-G.: Fusion of thermal imagery with point clouds for building façade thermal attribute mapping. *ISPRS Journal of Photogrammetry and Remote Sensing,* 151 (2019), pp. 162-175.

[LLW*24] LIU, Y., LI, D., WANG, K., XIONG, Z., SHI, F., WANG, J., LI, B., HANG, B.: Are LLMs good at structured outputs? A benchmark for evaluating structured output capabilities in LLMs. *Information Processing & Management*, 61, 5 (2024).

[LVK21] LITVIN, A. A., VELYCHKO, V. Y., and KAVERYSNKYI, V. V. Tree-Based Semantic Analysis for Natural Language to SPARQL Conversion. *Cybernetics & Systems Analysis* 57.6 (2021), pp. 897–905.

[NSR*24] NEX, F., STATHOPOULOU, E. K., REMONDINO, F., YANG, M. Y., MADHUANAND, L., YOGENDER, Y., ALSADIK, B., WEINMANN, M., JUTZI, B., QIN, R.: UseGeo - A UAV-based multi-sensor dataset for geospatial research. *ISPRS Open Journal of Photogrammetry and Remote Sensing* 13 (2024), 100070.

[PGJ*23] PENG, S., GENOVA, K., JIANG, C., TAGLIASACCHI, A., POLLEFEYS, M.: OpenScene: 3D Scene Understanding with Open Vocabularies. *Proc. CVPR* (2023).

[PGR*22] PATRUCCO, G., GÓMEZ, A., ADINEH, A., ADINEH, A., LERMA, J. L.: 3D Data Fusion for Historical Analyses of Heritage Buildings Using Thermal Images: The Palacio de Colomina as a Case Study. *Remote Sensing* 14, (2022) 5699.

[PRW*02] PAPINENI, K., ROUKOS, S., WARD, T., ZHU, W.: BLEU: a Method for Automatic Evaluation of Machine Translation. *Proc. 40th ACL*, (2002), pp. 311-318.

[PSM20] PANCHBHAI, A., SORU, T., MARX, E.: Exploring Sequence-to-Sequence Models for SPARQL Pattern Composition. *Proc. ISWC*, (2020), pp. 234-249.

[RC14] REMONDINO, F., CAMPANA, S.: 3D Recording and Modelling in Archaeology and Cultural Heritage: Theory and Best Practices. *Bar Publishing*, (2014). ISBN 978-140731-230-9.

[RR15] RAMOS, M. M., REMONDINO, F.: Data fusion in cultural heritage – A review. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.,* 40 (2015), 359-363.

[S25] SPARQL. https://www.w3.org/TR/rdf-sparql-query/, 2025.

[SR16] STYLIANIDIS, E., REMONDINO, F.: 3D Recording, Documentation and Management of Cultural Heritage. *Whittles Publishing*, (2016). ISBN 978-184995-168-5.

[Spa25] SPARQL, 2025, https://www.w3.org/TR/sparql11-query/

[XTZ20] XIE, Y., TIAN, J., ZHU, X. X.: Linking points with labels in 3D: A review of point cloud semantic segmentation. *IEEE Geoscience and Remote Sensing Magazine* 8 (2020), pp. 38-59.

[ZZCL19] ZHANG, J., ZHAO, X., CHEN, X., LU, Z.: A Review of Deep Learning-Based Semantic Segmentation for Point Cloud. *IEEE Access* 7 (2019), pp. 179118–179133.