Carving Time and Space

A mutual stimulation of IT and Archaeology to craft multidimensional VR data-inspection

B. Fanini, E. Demetrescu

Keywords: graph databases; real-time visualization; immersive VR; 3D-UI design; graph theory

Abstract
Interactive inspection of semantically-enriched Immersive Virtual Environments (IVEs) is designed on top of complex hierarchies combining both semantic and rendering aspects. Within Cultural Heritage, multi-dimensional IVEs represent a common solution in order to understand, query and inspect virtual reconstructions across different time-spans. The contribution presents innovative experiments about how the digital heritage record is organized and represented. Such approaches fit several scientific requirements within the Cultural Heritage domain as the annotation of the sources employed and the reasoning that are behind a reconstructive hypothesis. The methodological implications on the use of IT approaches can improve both the quality of the user fruition and the scientific content, offering, at the same time, formalisms and tools to boost the scientific research with real-time immersive representation of complex CH record. Graph-databases are already employed in such contexts since they represent one of the best solutions to address complex and dynamic relationships in highly connected datasets, also in terms of performance and scalability. A set of formalisms and replicable models for immersive inspection will be presented and discussed, addressing their interplay with a graph-based formalism specifically designed for 3D hypothesis creation and visualization in Cultural Heritage (CH) domain, targeting multitemporal scenarios - namely the Extended Matrix (EM).

Introduction

This article presents a new way of managing and inspecting semantically-enriched immersive virtual environments (IVEs). To explain the approach we will use the metaphors of the Mayan Veil (Schopenhauer) and the Time Machine. We can describe the reality perceived in the IVEs as "a veil of Maya" represented by the 3D models in the various epochs while the effect of vitality that these virtual worlds transmit to the user passes through some original tools of interaction (which will be discussed in the article). These tools also allow us to have access to what is behind the “Veil of Maya”: a Matrix of information organized according to a natural language without fixed patterns (node database) and according to a specific formalism: the Extended Matrix [] []. This language allows to collect and organize information on a timeline and to express also the lifespan of CH actors within the virtual world. The other aspect is the

Time Machine: the user's experience is to traverse time-periods in an immersive virtual environment (IVE) thanks to the visual formalisms and inspection tools described in this paper. One complex aspect of IVEs is the maintenance of a stable connection between the scientific information that is the basis of the three-dimensional model (for instance the scientific hypothesis behind a 3D virtual reconstruction) and the virtual experience session.
The method can be applied to all those data structures and all those virtual worlds that have behind them data structures organized on a temporal basis as it happens for the simulations, the virtual reconstructions or different 3D surveys made of a site after years.

Our contributions focus on:

- A set of reusable formalisms (blueprints) for immersive inspection of multi-temporal IVEs driven by graph databases, specifically the Extended Matrix[1]
- Replicable and efficient techniques targeting real-time applications and immersive fruition (through consumer-level HMDs) and its demands
- Multi-temporal scene-graph design to minimize memory footprint and maximize caching within WebVR/XR implementations
- Interaction models and best practices for immersive validation of Extended Matrices
- A prototype inspection tool crafted on top of such blueprints, called *EMviq*[2]

Related Work

In the last years there is and increased adoption of graph databases, especially in scenarios where the connections between the information is a valuable aspect. The visualization of data through graph-based visual structures is the main approach used in data visualization, but has been scarcely involved in the field of cultural heritage. Apparently in this domain the elements have a better and more compact representation in forms and tables. When it comes to representing strongly interconnected information (linked data), such as in the case of virtual reconstructions, visual graph databases allow for better adherence to the scientific record, better visual appeal, improved effectiveness (for the aesthetic principles for information visualization), and reduced complexity.

A graph database (GraphDB) is a database that exploits graph structures for semantic queries with nodes, edges and properties to represent and store data. At the core of the system is the *graph* directly relating data items: such relationships allow objects to be linked together directly and they are equally as important as the objects themselves. Graph databases are indeed based on graph theory (nodes, edges and properties) directly storing the relationships between records.

Such definition presents huge differences compared to relational databases that - through relational database management systems - allow manipulation of the data without imposing implementation aspects like physical record chains. For instance, links between data are stored in the DB itself at the logical level, and relational operations (e.g. join) can be used to manipulate and return related data in the relevant logical format. Relational queries can be

[1] http://osiris.itabc.cnr.it/extendedmatrix/
[2] http://osiris.itabc.cnr.it/scenebaker/index.php/projects/emviq/

performed through the database management systems at the physical level (e.g. using indexes), allowing to boost performance without modifying the logical structure of the database. Graph databases offer simple and fast retrieval of complex hierarchical structures that can be difficult to model in relational systems.
In order to retrieve data from a GraphDB, a query language other than SQL is required, which was designed for the manipulation of data in a relational system - thus not suitable to handle graph traversals. As of today, no single graph query language has been universally adopted, and most systems are closely tied to specific products. Some efforts to create a standard did lead to multi-vendor query languages like *Gremlin*, *SPARQL* [], and *Cypher* [].

Graph drawing tools, and other tools dealing with relational data, have to store graphs and related data. Despite the previous attempts to create a standard, there is still lack of a format that is widely accepted and several tools support only a limited number of custom formats typically restricted in their expressibility and specific for a given application field. The Demand for interoperability fueled the research and motivated the definition of an XML-based format. An informal task group was in fact created to propose a modern graph exchange format suitable for data transfer between graph drawing tools and other applications: the GraphML format [].

Interactive inspection of semantically-enriched Virtual Environments (VEs) is designed on top of complex scene hierarchies and combines both semantic and rendering aspects, while maintaining several aspects separated []. Within Cultural Heritage, multi-dimensional VEs represent a common solution in order to understand, query and inspect virtual reconstructions across different time-spans.

Within such context the Extended Matrix offers a Schema-Less Database Approach. The Extended Matrix is a formal language with which to keep track of virtual reconstruction processes. It is intended to be used by archaeologists and heritage specialists to document in a robust way their scientific hypothesis. It organizes 3D archaeological record so that the 3D modeling steps are smoother, transparent and scientifically complete. The EM offers a standardized workflow and visual tools for analysis, synthesis, data visualization, and publication. Starting from a stratigraphic reading of masonry (Building Archeology), all the sources used in the reconstruction are provided along (and integrated) with the 3D model. In other words, the Extended Matrix is a semantic graph that leads to a schema-less data model: the reconstructed objects and their descriptive elements are heterogeneously fitted into space and time, in a way that better suits the incompleteness of the historical record. The descriptive elements are used as a modular grammar to compose the final description of the reconstruction process (data-driven re-construction).

Within immersive virtual environments (IVEs) consumed through common consumer-level HMDs (Oculus Rift, HTC Vive, etc...) additional challenges arise when interactive inspection of Graph databases is performed. First of all, interactive immersive VR alone, presents several *performance* challenges: interactive rendering of a complex 3D scene (e.g. multi-resolution dataset) presents demanding requirements due to several factors, including stereoscopic rendering, larger FOVs and display resolution []. One of the very first ingredients for a smooth experience is in fact to maintain high frame rates (around 90 fps) and low latency using recent HMDs. The second macro-challenge for semantic inspection is at *presentational* level: how to extract and represent complex relationships at runtime in a suitable manner for immersive VR? What kind of layouts should we use? Past and recent literature [], [] already investigated information visualization within immersive fruition and best practices using consumer-level HMDs. Furthermore with the rise of WebVR/XR as a standard [] already employed by major commercial products such as SketchFab[3], additional challenges arise on data transmission and how to properly handle multiple temporal representations of the scene: what kind of solutions can we adopt to maximize streaming efficiency? What kind of multi-temporal scene-graph design can we exploit?

Case Study

The *EMviq* inspection tool (resulting from the formalisms described in the next section) has been employed on a case study drawn from the Building Archaeology domain: the ancient Roman town Colonia Dacia Sarmizegetusa Ulpia Traiana (a temple and a Bath building built in the Second century AD). These examples show the use of EMviq within two projects of virtual reconstruction of Roman contexts. The hypotheses of virtual reconstruction of the Great Temple and of the Baths (at Sarmizegetusa) have been developed starting from a photogrammetric survey by drone, from a bibliographic study, from an analysis of the architectural elements found in the site and from a comparative study with other similar contexts. Starting from all this information, a virtual reconstruction of the contexts was made. The steps of the reconstruction were annotated using the language of the Extended Matrix and the software tools made available by the EMF (Extended Matrix Framework), namely the EMTools[4], a commercial freeware node editor (yEd) and finally, the EMviq.

Semantic Inspection for Immersive VR

This section describes and formalize *blueprints* that aim to create a replicable and reusable set of models for different semantic VR inspection contexts. The main goal is to address the interplay of such set with a graph-based formalism - namely the Extended Matrix - specifically designed for 3D hypothesis creation and visualization, specifically targeting multi-temporal scenarios. We define a set of operators also implemented in a VR prototype, called “EMviq” that’s also been applied to different case studies. At first, we may indeed observe within such framework that we have to deal with multiple 3D representations: an object, a context or a large area during different time periods. This leads to the definition of a *collection* of scene-graphs, that should be properly mapped into specific temporal spans, given a specific Extended Matrix (EM).

Extraction routines at runtime

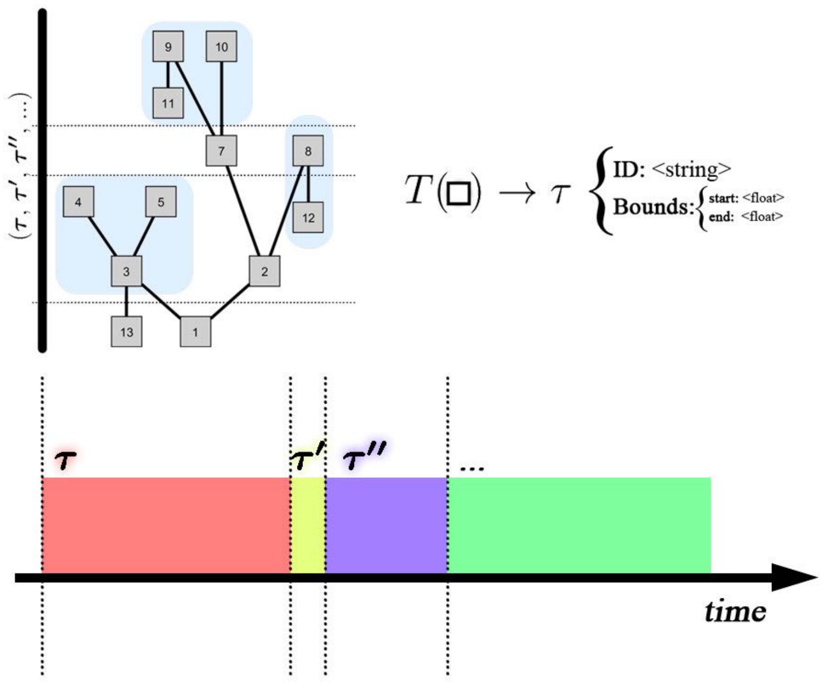
In this section we describe routines to be implemented in order to extract runtime data from a single EM (GraphDB). Such procedures have to be designed to create intermediate data structures for fast access by a real-time immersive application. Within XML-based input formats - for instance GraphML - this is achieved by means of fast parsing procedures traversing the file and producing intermediate data structures (runtime graphs). As previously

[3] https://sketchfab.com
[4] https://github.com/zalmoxes-laran/EMBlenderTools

described in a previous research [] we define and formalize three different extraction steps applied to Extended Matrix formalism:

- Timeline extraction
- ProxyGraph extraction
- SourceGraph extraction

The above computational steps need to be performed only when involved GraphDB is modified: more precisely, it has to be performed only on the modified sub-graphs (localized updates). Within immersive VR contexts, intermediate runtime data generated by such approach has the objective of providing high framerates and low latency during query and inspection.



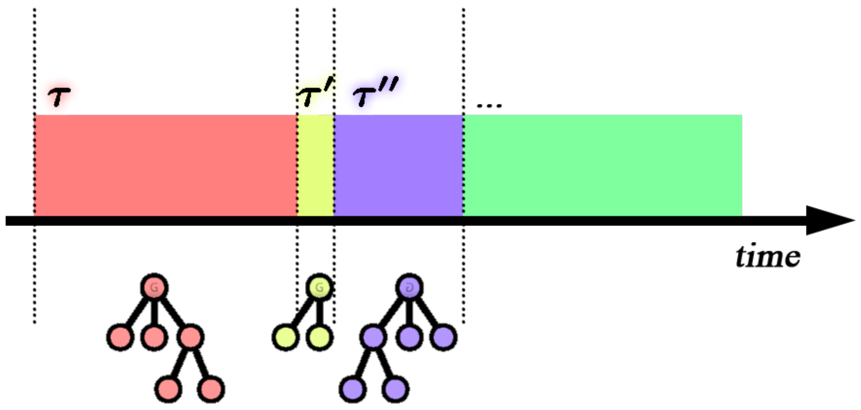
By EM definition, each node □ in the GraphDB has temporal property, so for timeline extraction we define an operator T such as:

$$\mathbf{T}(\square) \rightarrow \tau$$

Where τ is a specific period. We can use T to map to map each node in the GraphDB to a collection of time-periods (τ, τ', τ'' ….). Each period has unique ID and <start, end> pair, that also defines its duration (centuries, years, days, seconds…). We define also a selector S, such that:

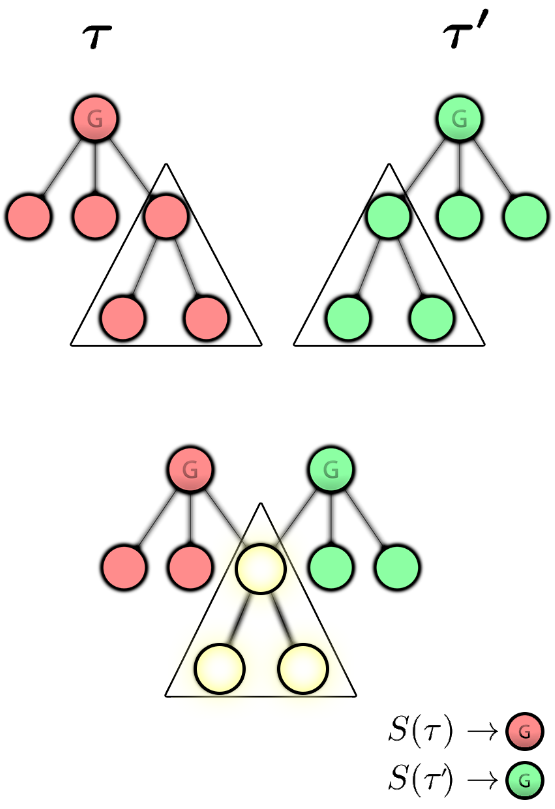
$$\mathbf{S}(\tau) \rightarrow G$$

Where G is a scene-graph associated with input time-period τ. S can be employed at runtime by VR application to switch between different periods, by mean of user input (e.g. VR controller). Using a *naive* approach, we could simply map each time-period τ to a specific scene-graph representation and switching sub-graph during VR session depending on user input. Although a given context may present areas or portions having different temporal pacing: for instance, a part of the 3D scene did not evolve during multiple periods. Such approach may indeed result in a waste of resources and poor optimization from a memory footprint perspective.



Temporal Instancing

It’s a common scenario that a single scene-graph G may include a sub-graph that is shared with another time-period (thus a scene portion re-used by another graph G’). With temporal instancing, we describe the collection as multi-root DAG: each root acts as entry-point for selector S: this approach allows different time-periods to refer to the same scene-graph G or a part of it.

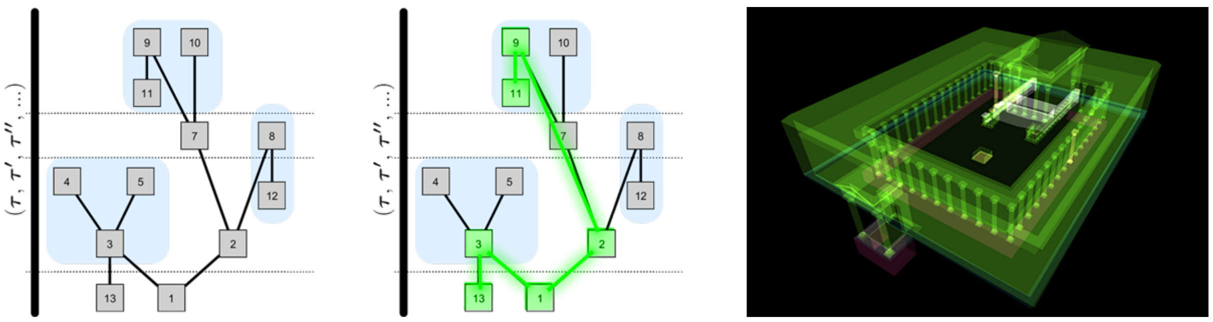


In fig. Xxx a sample temporal instancing between two scene-graphs is shown: note the selector S always returns coherent scene representation with the two graphs sharing a sub-graph that spans across τ and τ'. Such cross-temporal organization allows elegant and compact overall scene design, and offers following major advantages due to re-use of scene portions:

- Compact memory footprint at runtime during VR inspection
- Caching for WebVR applications (sub-graphs re-use) thus providing online efficiency

VR Query operator

In order to offer smooth, consistent and efficient 3D queries for immersive VR applications, the application layer must provide routines to extract and automatically build from the GraphDB a hierarchy of semantic 3D descriptors, namely the Proxy-Graph.



$(\tau, \tau', \tau'', \dots)$
9
10
11
7
8
12
4
5
3
2
13
1
$(\tau, \tau', \tau'', \dots)$
9
10
11
7
8
12
4
5
3
2
13
1

Similarly to collision routines employed in modern game engines and frameworks, efficient ray-casting procedures are performed on simplified geometries, in this case proxy-nodes. The automated realization of such runtime data structure is defined by the procedure P:

$$\mathbf{P}(G_{db}, f) \rightarrow G_p$$

Where G_{db} is the GraphDB (a single Extended Matrix), f is an optional filtering function to traverse only specific edges of G_{db} and G_p is the realized Proxy-Graph for interactive queries.

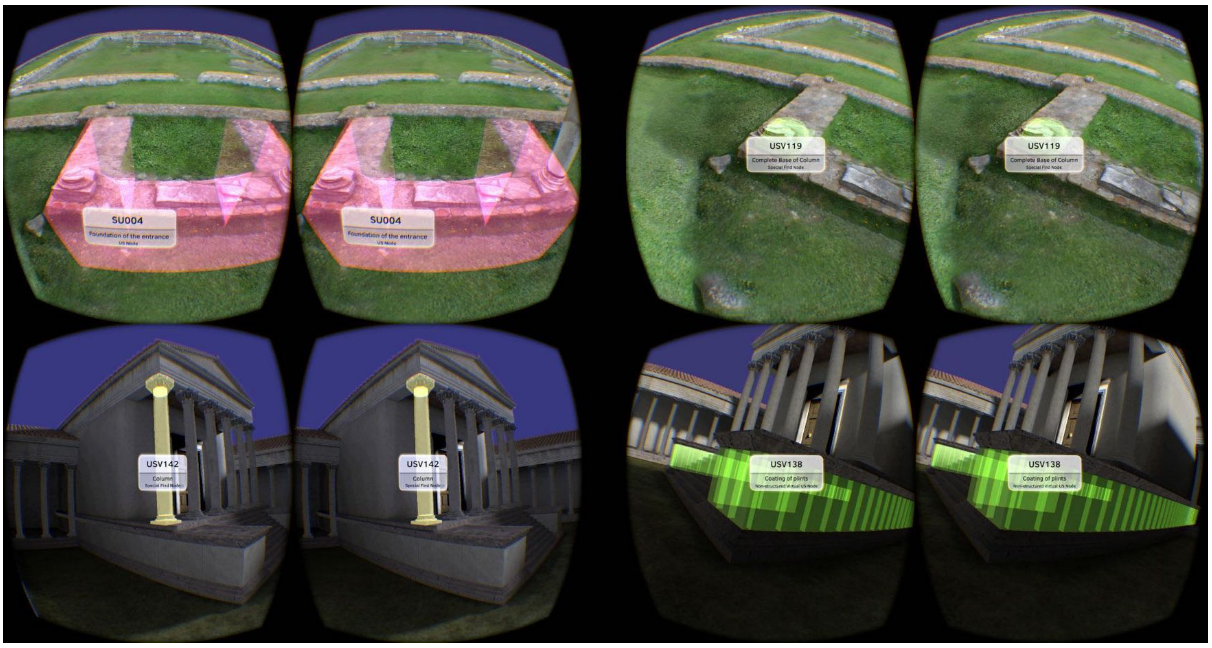
Runtime efficiency in VR is guaranteed by performing 3D queries using common segment intersectors, offered by most modern frameworks and game engines (e.g. *IntersectionVisitor* in OpenSceneGraph, *LineTraceByChannel* in Unreal Engine 4, etc...). The VR query operator Q can thus be defined as:

$$\mathbf{Q}(s, e) \rightarrow p \qquad\qquad p \in G_p$$

Where $s,e \in R^3$ represent start and end points of the segment, while p is the returned proxy-node (the 3D semantic descriptor). Notice the definition allows different VR interaction models for semantically enriched IVEs, the following are commonly used:

- s is coincident with current head location in virtual space, and e is defined by current HMD orientation (depending on a given maximum distance - e.g.: 100 m)
- s is attached to a VR controller and e defined by its current orientation

The Q operator allows of course additional interaction models, although the above are generally sufficient to cover common scenarios (HMD alone and HMD + VR controllers).



Furthermore, Q operator - together with scene collision geometries (commonly used to simulate physics) - can be employed to implement a *proxy-driven* locomotion, using common teleport techniques that already proven minimal motion sickness []. User input can trigger artificial locomotion on hovered proxy-node to a new computed location depending on surrounding proxy-nodes, physical tracked area of HMD, and physical constraints (scene colliders). Within immersive VR inspection, such approach offers interesting *semantic locomotion* models while inspecting multi-temporal IVE driven by an Extended Matrix.

Peel operator

Previous definition for query operator Q still suffer from a common issue that may occur in semantically complex IVEs (complex Proxy-Graphs): occlusion. A proxy-node can be in fact unreachable by ray-based queries (nested proxy-nodes, etc…) thus making the user incapable of inspecting certain spots of the 3D space. The peel operator acts as spherical subtractor given a *center* and a *radius*, thus allowing to *carve* semantic descriptors (and/or visible scene-graphs). The spherical carving can be also localised to specific time-periods or operate on the entire timeline, thus offering great flexibility in terms of VR fruition.

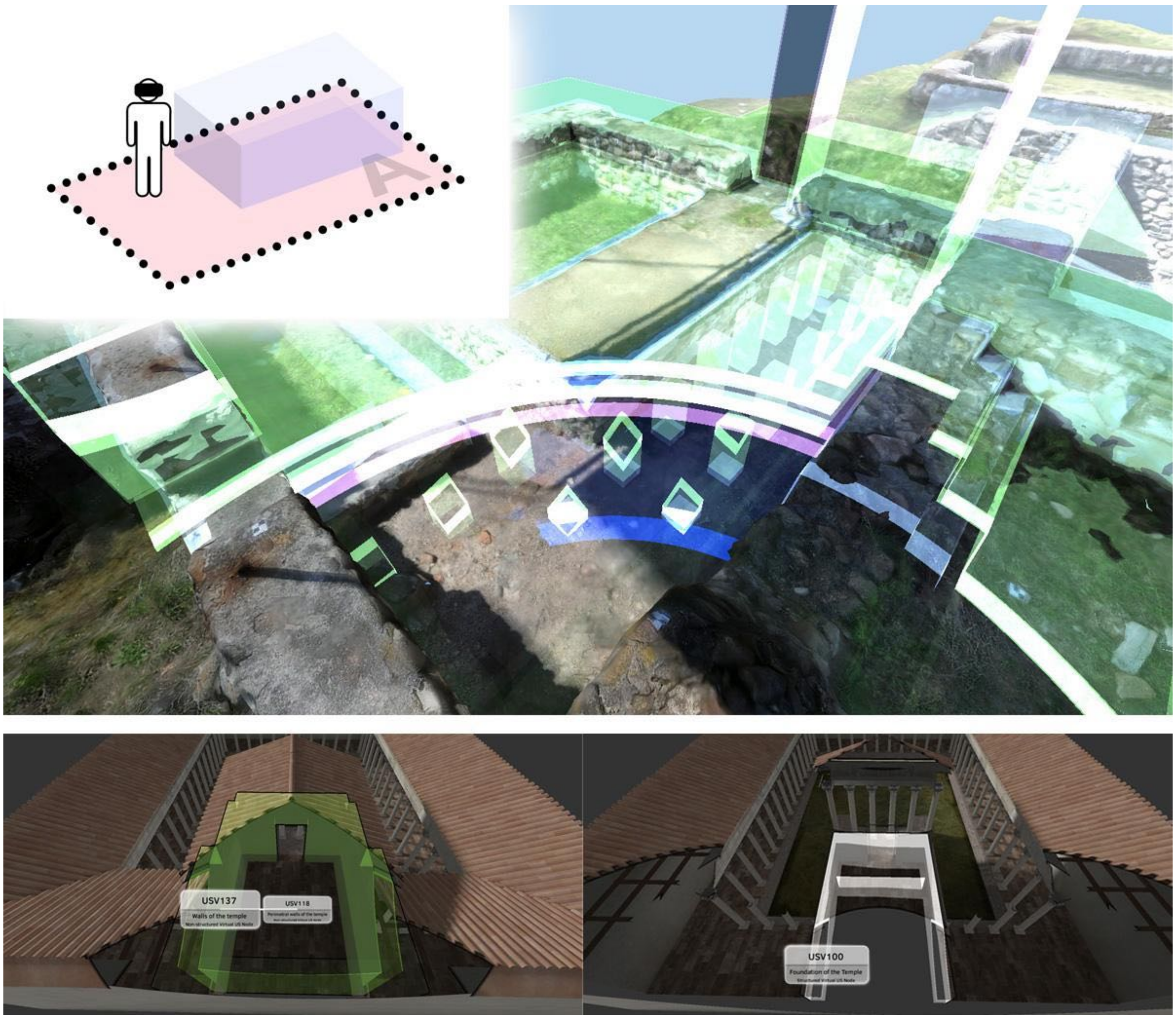


Fig. AA

The peel operator is also particularly useful in combination with positional tracking, including both outside-in (commonly shipped in consumer-level HMDs like Oculus Rift, HTC Vive, etc...) and inside-out tracking approaches. The operator can be in fact attached to user location or VR controllers, also allowing modification of peel radius at runtime. Such interactive approach provides the user with maximum flexibility inside the physical area A (see figure AA) for localized inspection during immersive sessions.

Source-Graphs Presentation

Within the Extended Matrix framework, *source-graphs* represent internal runtime structures of sources relationships (paradata). A single Source-Graph is extracted from a given proxy-node $p \in G_p$, as shown in the example in fig. YY.

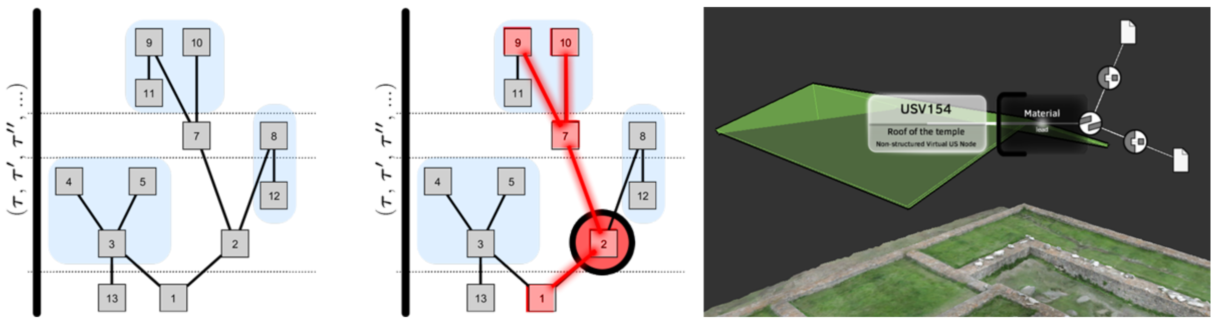


Fig. YY

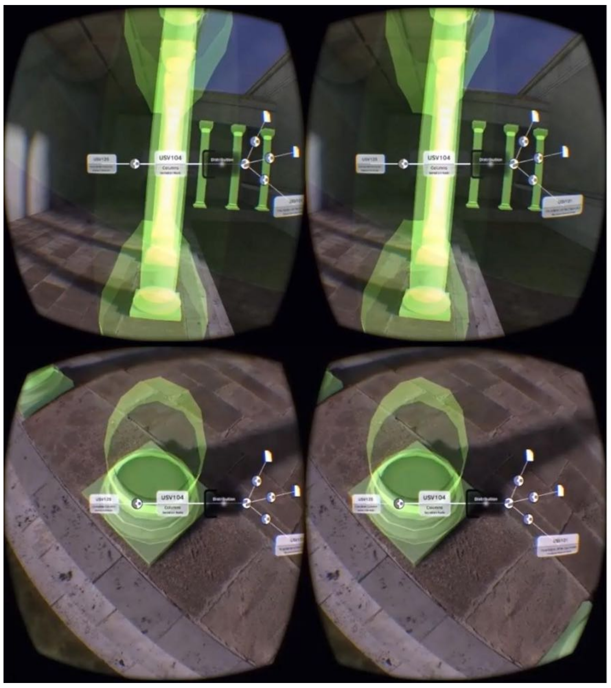
For a given EM, this leads to a set of Source-Graphs:

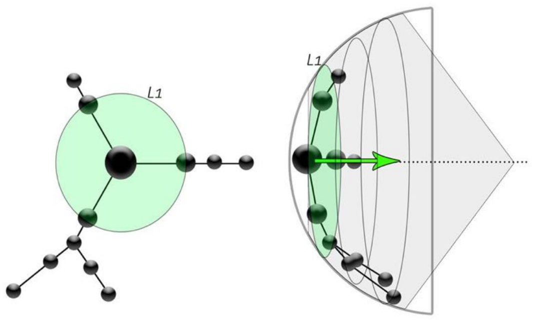
$$\{ G_u^1, G_u^2, \dots G_u^k \}$$

each associated with a specific proxy-node. Since the extraction routine from the GraphDB may be computationally expensive in some cases (thus not suitable for VR interaction), the application typically pre-computes all the Source-Graphs and indexes them. Using such approach, the immersive application may safely and quickly access a Source-Graph upon querying a specific proxy-node *p*. The Source-Graph extraction can thus be defined as:

$$\mathbf{U}(G_{db}, p, f) \rightarrow G_u$$

Where *f* is the usual edge filtering function and G_u is the returned Source-Graph. Once we accessed the graph, how can we represent such relationships in VR? In our framework and for the EMviq prototype we adopted a tree analogy by deploying growing 3D layouts. The 3D structure can be spawn in a given location (application point) in the virtual space, typically the queried proxy-node. Runtime generation of such 3D layouts leverages on algorithms for immersive graph visualization [] exploiting the effectiveness of stereoscopic perception. Furthermore aesthetic aspects (balance, proportion, etc…) should also be taken into account [] for layout generation, while maintaining robust performances.





View-dependent and distance-based techniques may offer good and usable layout presentations taking into account also past literature and best practices for presentation of information in VR (also including specific fonts for readability). The 3D paraboloid layout here proposed takes inspiration from 2D parent-centered layouts []: it allows dynamic growth (including fold/unfold of local branches) of active G_u on application point. Specifically, the 3D layout automatically provides to:

- Scale the overall graph depending on distance to intersection location and 3D graph extents (bounding box of G_u)
- Orient the growth axis depending on look direction
- "Embrace" the user by using incremental offsets for each graph level (L_1, L_2, etc…)

The paraboloid 3D layout can be pre-computed during extraction routines from current EM and dynamically placed by means of a parent transform. This also allows the G_u (and related interfaces) to be attached on 6-DOF controllers or virtual spots depending on application requirements. Additional amplification techniques (responsive to head orientation) may be employed to boost Source-Graph inspection and - more importantly - support *immersive validation* of current Extended Matrix.

Cloud-based Session and Workflow

Extraction routines, the graph-based approaches and presented formalizations also allows crafted VR applications to exploit cloud-based scenarios. For instance the developed EMviq VR prototype (based on OpenSceneGraph framework and ownCloud) offers inspection of hybrid local/remote scene-graphs and EMs: this allows professionals to design their EM while remote users inspect or validate it without taking off their HMDs, providing a smooth immersive workflow.

Conclusions and Current Directions

In this paper we proposed and discussed a set of reusable formalisms (blueprints) for immersive inspection of multi-temporal IVEs driven by graph databases - in particular the Extended Matrix. The work investigated replicable and performance-oriented techniques aiming at real-time, immersive fruition by means of consumer-level HMDs. The routines are also discussed in terms of computational perspective, thus separating those functionalities that require intermediate data structures for efficient access at runtime by the immersive inspection tool. The paper discussed also multi-temporal design for scene-graphs in order to minimize memory footprint and maximize caching within WebVR/XR implementations. Interaction models and resulting best practices for query and inspection were described for different fruition scenarios, including for instance HMD alone and HMD equipped with 6-DOF controllers. A prototype inspection tool (EMviq) was crafted and developed on top of such blueprints, including collaborative perspective by means of cloud-based workflow. This led to novel approaches for Extended Matrices validation through immersive fruition also between remote professionals.

Regarding current directions, the development of the EM and its related EMF will result in new versions with the addition of both methodological and technical improvements. The next steps will focus on the support for different, self excluding reconstruction hypotheses (as a 5th dimension for the EM). In some cases there are more than one possible reconstructive solution that have to be stored and organized accordingly in the EM. Another future step, already in progress, involves the creation of WebVR/XR component based on described formalisms for the new upcoming version of ATON 2.0. Such process will enable on-line 3D fruition of Extended Matrices on all major browsers without installing any additional plug-in or software, including specific touch interfaces targeting mobile deployment. Described formalisms will be also serve to create a drag&drop plugin component for Unreal Engine 4 thus targeting modern game engines and desktop-based systems.