UniSG^GA: A 3D scenegraph powered by Geometric Algebra unifying geometry, behavior and GNNs towards generative AI

Manos Kamarianakis
kamarianakis@uoc.gr
FORTH - ICS, University of Crete, ORamaVR
Heraklion, Greece

Antonis Protopsaltis
aprotopsaltis@uowm.gr
University of Western Macedonia, ORamaVR
Kozani, Greece

Dimitris Angelis
dimitris.aggelis@oramavr.com
FORTH - ICS, University of Crete, ORamaVR
Heraklion, Greece

Paul Zikas
paul@oramavr.com
University of Geneva, ORamaVR
Geneva, Switzerland

Mike Kentros
mike@oramavr.com
FORTH - ICS, University of Crete, ORamaVR
Heraklion, Greece

George Papagiannakis
papagian@ics.forth.gr
FORTH - ICS, University of Crete, ORamaVR
Heraklion, Greece

ABSTRACT

This work presents the introduction of UniSG^GA, a novel integrated scenegraph structure, that to incorporates behavior and geometry data on a 3D scene. It is specifically designed to seamlessly integrate Graph Neural Networks (GNNs) and address the challenges associated with transforming a 3D scenegraph (3D-SG) during generative tasks. To effectively capture and preserve the topological relationships between objects in a simplified way, within the graph representation, we propose UniSG^GA, that seamlessly integrates Geometric Algebra (GA) forms. This novel approach enhances the overall performance and capability of GNNs in handling generative and predictive tasks, opening up new possibilities and aiming to lay the foundation for further exploration and development of graph-based generative AI models that can effectively incorporate behavior data for enhanced scene generation and synthesis.

KEYWORDS

Geometric Algebra, Generative AI, Graph neural networks, 3D scenegraph, Entity Component System, Behavior embedding

1 INTRODUCTION

The recent success of pre-trained foundation models, such as GPT (Generative Pre-trained Transformer), has paved the way for evolution in geometric deep learning [2] and GNNs [6]. Such advancements have greatly improved the generation of static 3D scenes [3] by incorporating relational patterns within the graph topology as node or link features. Typically, these scenes rely on well-defined 3D-SGs. The creation of immersive VR experiences require the incorporation of behavioral information and interactions, that are specified with the adoption of the graph structure Lessons-Stages-Actions (LSA) [19].

Nevertheless, the efficient input of all encapsulated data to GNNs poses a challenge, as it requires managing three distinct graph structures (see Figure 1), namely 3D geometry, interactive event-based animations encapsulated as behaviours (LSAs) and GNNs. This introduces a great complexity in maintaining transformations between these graphs that may lead to a potential bottleneck. To address these limitations, we propose the Universal Scenegraph (UniSG), a novel data structure aimed at providing a no-code approach featuring GNNs, that generate new nodes, edges, and features, reflecting the creation of 3D models, scenes, and behavioral steps. UniSG paves the way towards generative AI techniques, by integrating Entities-Components-Systems (ECS), 3D-SGs, and LSAs with GNNs, simplifying the creation of 3D scenes with embedded behavior, and mitigating existing process bottlenecks.

UniSG leverages a representation form that is able to capture and preserve relative topological information between parent and child entities. Rather than relying on conventional Euclidean-based matrix form or Euler angles or dual/single quaternions, commonly employed in 3D scenes, we utilize Geometric Algebra (GA) based forms, such as multivectors; the resulting model is denoted as UniSG^GA. GA-based representations enable the encapsulation of diverse transformation data in a unified format, facilitating [14] deeper geometric connections, thereby influencing the performance of GNNs across various tasks (see Section 1.2).

1.1 The importance of GNNs for Generative AI

GNNs have gained significant attention in recent years due to their effectiveness in handling graphs of varying types, sizes, structures, connectivity patterns and data with complex relational structures, due to the high flexibility and adaptability of their architectures. Their design makes them particularly well suited for generative and predictive AI tasks that involve graph-structured data, like complex 3D scenes, where nodes represent objects and edges encode relationships or connections between them, as they are able to capture spatial relationships, model dependencies and extract meaningful representations. Specifically for an entity-component-systems (ECS) in a scenegraph CG framework [13, 17], a GNN involves heterogeneous nodes, representing entities and diverse components, containing object-related data (transform, mesh, image texture data, etc.).

GNN aggregation allows the capture of the graph's local dependencies, while its propagation through the graph allows the capture of global dependencies. In this context, complex interactions between nodes may also be captured by iterative node representation refinement, using message-passing mechanisms. Such rich information about the nodes and their spatial relationships, learned

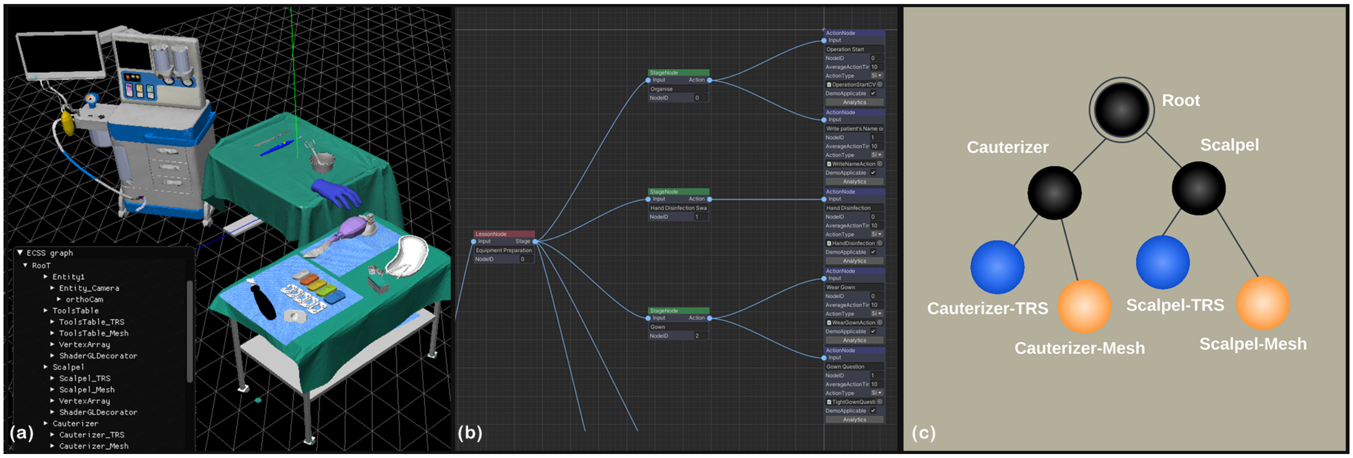

Figure 1: The UniSG^GA unifies the three diverse graphs that must be maintained for a 3D scene that includes behaviour, digestible by a GNN: (a) the 3D scenegraph with Entity-Component-Systems, (b) the behavioral LSA graph and (c) the deriving GNN graph. The components describing the parent-child relative topology are expressed in GA-based forms, for increased performance on predictive and/or generative tasks.

from the training data, may be encoded in meaningful and low-dimensional embeddings, that involve fixed-length vectors or a continuous feature space. The GNN model may be trained in a) supervised manner, involving annotated 3D-SGs, aiming to predict missing elements or labels, and b) unsupervised manner involving graph similarity or reconstruction losses, aiming to optimize the generative model.

1.2 GA and GNNs

The combination of GA with GNNs offers several benefits across different domains and tasks[1]. GA-based approaches have demonstrated superior information (inherent structures and correlations among multiple dimensions) preservation, as multi-dimensional data are represented through multivectors. This leads to improved performance, compared to traditional techniques, in tasks including as time series processing, hyperspectral image analysis, and traffic prediction [9–12, 16]. They also exhibit reduced overfitting risks, compared to real-valued counterparts, making them more effective in capturing complex features while maintaining the multi-dimensionality of the data.

GA is particularly advantageous in handling rotational data, making it valuable for computer vision tasks, like pose estimation or protein prediction [14, 15]. GA-based formulations enable better regression on rotations and can reduce errors in high-noise datasets while learning fewer parameters. Additionally, GA-based graph feature embedding enhances the quality and presentation of graph features in GNNs. By leveraging the high algebraic dimensions of GA, feature information distortion across hidden layers can be minimized, resulting in improved performance in graph-related tasks. Furthermore, GA-based approaches can reduce computational complexity by utilizing appropriate multivector representations and exploiting the algebraic properties of GA. This reduction in complexity enables more efficient data processing and analysis, with fewer parameters to be learned without compromising performance.

In summary, the integration of GA with Neural Networks offers benefits, such as enhanced representation of multi-dimensional data, improved information preservation, effective handling of rotational data, better graph feature embedding, robustness to poor network conditions, and reduction of computational complexity. These advantages make GA a valuable framework for various scientific domains and tasks, facilitating more accurate and efficient data processing and analysis.

Paper Overview. In Section 2 we introduce the UniSG model, whereas in Section 3 we propose the enhanced UniSG^GA model that exploits GA-based representation forms. These models are implemented and available to use within the Elements project, which now includes enhanced GA-functionalities, as described in Section 4. Results obtained for our models performance are presented in Section 5, followed by Conclusions, Future Work and Acknowledgments.

2 UNISG: A UNIVERSAL SCENEGRAPH

The UniSG system, introduced in a concise manner in [17], exhibits a heterogeneous graph structure built upon the Entity Component System in a Scenegraph (ECSS) model, such as the one proposed in [13]. This graph encompasses diverse component types capable of storing both geometric and behavioral information relevant to interaction with the 3D scene and events triggered by specific conditions. Specifically, the UniSG graph incorporates three types of components: `info`, `TRS`, and `mesh`. The `info` components maintain a count of node types among their children, while the `TRS` components store a 16-dimensional vector obtained by flattening the corresponding transformation matrix. The `mesh` components house a feature vector of size 1024, representing the mesh using a suitable encoder such as the AtlasNetEncoder [4] combined with a Poisson sampling process. This encoding methodology ensures a fixed-size representation regardless of the complexity of the original mesh.

Subsequently, the resulting vector can be decoded using the AtlasNetDecoder to generate a point cloud, which can then be further reconstructed into a triangulated mesh.

To incorporate behavioral functionality, the UniSG system introduces a forth `ActionData` component that stores data pertaining to desired behavioral characteristics, accompanied by appropriate `Action` systems responsible for processing this data. These ECS components and systems effectively represent user actions required within a training scenario, akin to those stored in the Lesson-Stages-Actions (LSA) data structure [18]. The `ActionData` nodes adhere to a standardized structure for all actions and store action-specific data and conditions in vector form. The diverse `Action` systems continuously traverse the graph or its designated sections to validate whether the specified conditions are met.

The architectural elements of the ECS framework are depicted in Figure 2 as follows. The black nodes represent entities, while the blue nodes represent components, which encapsulate various data such as transformations, meshes, and actions. Systems, represented by red lines, process the data contained in components and perform specific tasks while traversing the graph. Graph features, highlighted in yellow, are represented in vector form, enabling their utilization by GNNs for further analysis and processing.

Figure 2 also exemplifies the implementation of an "Insert" action within the UniSG system. In this specific scenario, the `InsertAction` system is responsible for verifying whether the placement of the scalpel on the knee adheres to the specified spatial boundaries. This check is performed when the system visits the `ActionData` component.

To consolidate disparate data types into a unified format, various file formats commonly employed have been merged into a single master file. Pixar's Universal Scene Description (USD) (http://graphics.pixar.com/usd/) future-proof format has been selected for its exceptional versatility, enabling the inclusion of more advanced features such as VR-Recording [7].

3 UNISG^GA: EMPOWERING UNISG WITH GEOMETRIC ALGEBRA

The original UniSG model employed a `TRS` component, which stored the topological relationship between an entity and its parent as a 16-dimensional array vector. This vector was obtained by flattening a 4x4 transformation matrix, resulting from the multiplication of Translation, Rotation, and Scaling matrices.

In this paper, we propose the UniSG^GA model, which overcomes the limitation of relying solely on matrix-derived vectors. The UniSG^GA model suggests the utilization of alternative forms of transformation data, allowing for a more diverse range of representations. Particularly, we advocate for the adoption of GA to express data that represents geometrical relationships. The integration of GA is not merely intended to promote its acceptance, but rather to demonstrate its potential to yield improved outcomes in various scientific domains, particularly those involving predictive and generative tasks, with a special focus on GNNs.

4 UNISG^GA WITHIN THE ELEMENTS PROJECT

The proposed UniSG^GA structure is already implemented within the Elements project, introduced in [13], similar to its predecessor UniSG. Elements, presents a pioneering open-source pythonic framework based on entity-component-systems (ECS) implemented within a scenegraph architecture. It is explicitly tailored to address the demands of scientific, visual, and neural computing applications. Comprised of three vital Python components—pyECSS, pyGLV, and pyEEL—the Elements package offers a foundational implementation of the ECS paradigm, accompanied by practical examples that proficiently familiarize even inexperienced computer graphics programmers with fundamental principles and methodologies. Notwithstanding its straightforwardness, Elements retains a transparent nature, affording users the ability to scrutinize and manipulate each stage of the graphics pipeline. Leveraging Python's inherent advantages in rapid prototyping and development, users can augment Elements' capabilities by introducing novel components and systems or refining existing ones.

The collection of jupyter notebooks within the pyEEL repository serves as a demonstrative repository for showcasing the influence of Elements' present and future features across diverse scientific domains and packages, thereby establishing a valuable pedagogical resource for both novice and intermediate developers. To facilitate the transition to GA forms, pyEEL now incorporates a series of Jupyter notebooks that serve three purposes: (a) introducing basic GA concepts to users unfamiliar with GA, (b) demonstrating the equivalence between different representation forms in a digestible manner for intermediate GA users, and (c) presenting more advanced applications of these principles, such as model animation using GA, for experienced GA users.

4.1 Geometric Algebra powered 3D scenegraph

Currently, matrix representations dominate the field due to their ease of implementation and compatibility with GPU shader-level operations. Although quaternions have mitigated issues such as gimbal lock and interpolation artifacts when evaluating rotation matrices, GA introduces a further advancement in representation forms. By utilizing translators, rotors, and dilators as GA-based counterparts for translation, rotation, and dilation, respectively, we can achieve improved results both quantitatively (reducing the number of keyframes required for interpolation) and visually [8].

Complex operations, such as extracting geometric information from motors (i.e., geometric products of a translator and a rotor), are now performed with ease, by leveraging the capabilities of the well-maintained Clifford Python package [5], facilitating efficient transmutation between different forms.

Specifically, let M be a 4x4 matrix representing a rotation followed by translation. It is well known that the top left 3x3 submatrix is a rotation matrix and the 3 first elements of the last column is the translation vector t. From R matrix we can extract the angle/axis, and therefore determine the equivalent unit quaternion q that expresses the same rotation. Finally, having the quaternion and the translation vector you can easily concatenate them to obtain the respective dual-quaternion dq. The following is summarized in (1), where rotational data are represented in cyan, translational in blue

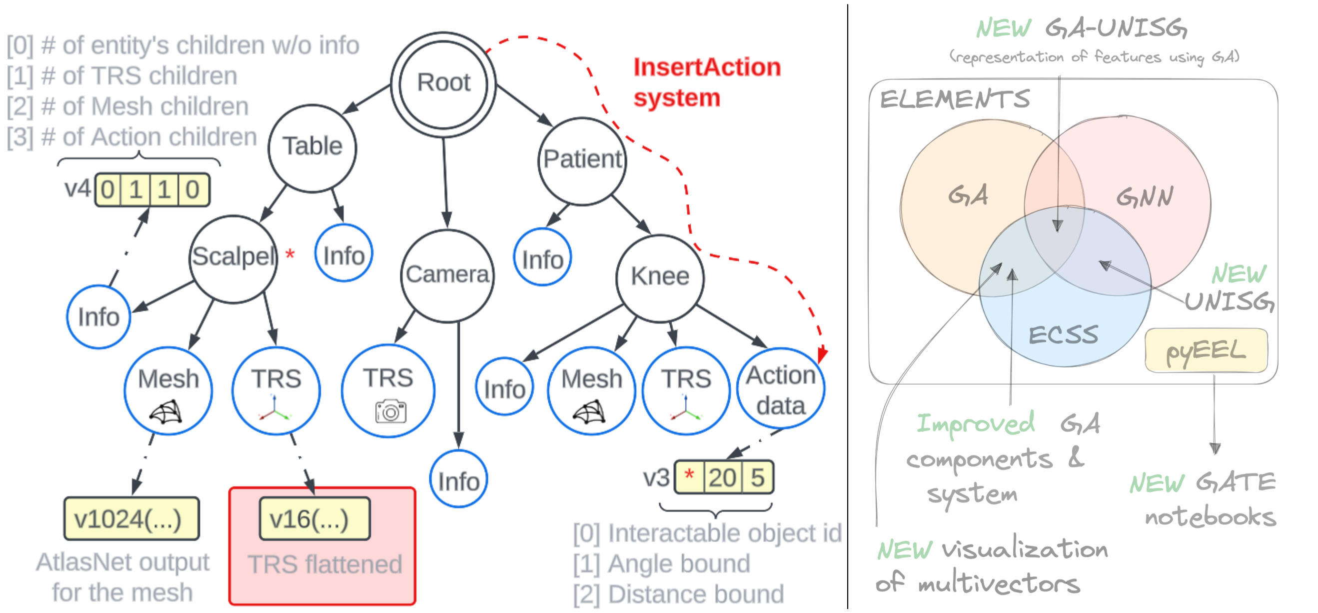

Figure 2: (Left) As opposed to the UniSG model [17], the proposed UniSG^GA model suggests using any GA-based representation form for the TRS component (red box), instead of the original 16-dimensional array vector, deriving from the flattening of a transformation matrix. (Right) A diagram denoting the contributions presented in this paper, with respect to state-of-the-art.

and mixed data in purple.

$$M = \begin{bmatrix} m_1 & m_2 & m_3 & t_1 \\ m_4 & m_5 & m_6 & t_2 \\ m_7 & m_8 & m_9 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Leftrightarrow R = \begin{bmatrix} m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 \\ m_7 & m_8 & m_9 \end{bmatrix} \;\&\; t = (t_1, t_2, t_3)$$

$$\Leftrightarrow \text{(Angle, Axis)} \;\&\; t \Leftrightarrow \text{Quaternion } q \;\&\; t \Leftrightarrow \text{Dual-Quaternion } dq. \tag{1}$$

From the translation vector t, we can easily determine the corresponding translator T_{PGA} in 3D PGA as follows:

$$T_{PGA} = 1 - 0.5e'_0(t_1 e'_1 + t_2 e'_2 + t_3 e'_3), \tag{2}$$

where e'_0, e'_1, e'_2 and e'_3 are basis vectors of 3D PGA. Similarly, we can derive the corresponding translator T_{CGA} in 3D CGA as :

$$T_{CGA} = 1 - 0.5e_0(t_1 e_1 + t_2 e_2 + t_3 e_3)(e_4 + e_5), \tag{3}$$

where e_0, e_1, e_2, e_3, e_4 and e_5 are basis vectors of 3D CGA. Extraction of the vector t from both T_{PGA} and T_{PGA} is apparent as long as the multivectors are normalized; otherwise, a division by the scalar part is initially required.

Given a unit quaternion $q = q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k}$, we can easily determine the respective rotor R_{PGA} in 3D PGA and R_{CGA} in 3D CGA (see [8, Section 2.4]) as

$$R_{PGA} = q_0 - q_3 e'_{12} + q_2 e'_{13} - q_1 e'_{23}, \text{ and} \tag{4}$$

$$R_{CGA} = q_0 - q_3 e_{12} + q_2 e_{13} - q_1 e_{23}, \tag{5}$$

where $\{e'_{12}, e'_{13}, e'_{23}\}$ and $\{e_{12}, e_{13}, e_{23}\}$ are respectively PGA and CGA basis vectors. In conclusion, the following equivalencies holds:

$$T_{PGA} \Leftrightarrow t \Leftrightarrow T_{CGA}, \text{ and } \; R_{PGA} \Leftrightarrow q \Leftrightarrow R_{CGA}. \tag{6}$$

Lastly, in [8, Section 2.4], it is shown that given a PGA motor M_{PGA} resulting from the geometric product of T_{PGA} and R_{PGA}, one may extract the latter two. The same holds for a CGA motor M_{CGA} resulting from the geometric product of the translator T_{CGA} and the rotor R_{CGA}, yielding:

$$M_{PGA} \Leftrightarrow R_{PGA} \;\&\; T_{PGA},, \text{ and } \; M_{CGA} \Leftrightarrow R_{CGA} \;\&\; T_{CGA}. \tag{7}$$

Using all equivalencies described above we can now extend (1) to the complete equivalency list of representation forms; all equivalencies can occur using functions implemented within the Elements framework:

$$\text{Transformation Matrix } M \Leftrightarrow \text{Rotation Matrix } R \;\&\; \text{vector } t \Leftrightarrow$$
$$\text{(Angle, Axis)} \;\&\; t \Leftrightarrow \text{Quaternion } q \;\&\; t \Leftrightarrow \text{Dual-Quaternion } dq.$$
$$\Leftrightarrow M_{PGA} \Leftrightarrow R_{PGA} \;\&\; T_{PGA} \Leftrightarrow M_{CGA} \Leftrightarrow R_{CGA} \;\&\; T_{CGA}. \tag{8}$$

5 RESULTS

To validate the effectiveness of our proposed approach, we conducted three experimentation tasks in the domains of classification, generative modeling, and topology prediction for 3D scenegraphs. In Figures 3 and 5, we present the obtained results using different representation forms for the TRS component of the UniSG^GA model. Specifically, we compare the use of a) flatten matrices (representing the original UniSG), b) CGA and c)PGA multivectors, d) a vector for translation combined with an angle and an axis for rotation , as well as a e) dual- quaternion representation.

Each task is accompanied by a comparison graph, demonstrating the performance of the GA-based representations in relation to the conventional Euclidean-oriented formats. The results consistently show that the utilization of GA-based representation forms,

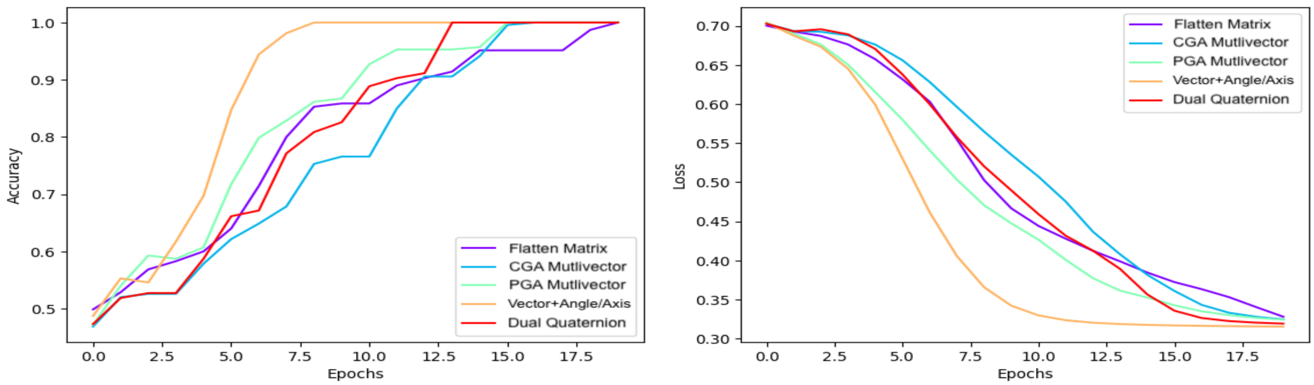

Figure 3: Train accuracy (Left) and Loss (Right), for the classification task described in Section 5.1. These are mean values after running the experiment 10 times.

such as CGA/PGA multivectors and dual-quaternions, either outperforms or performs on par with the traditional flatten matrices representation.

5.1 Classification

Our methodology was evaluated through a classification task involving a neural network architecture composed of two Convolutional layers. The GraphSAGE convolution operation was applied to the input graph within this framework. To assess the performance of our approach, we curated a dataset comprising of 100 3D scenes. These scenes were generated using a random noise- based data augmentation technique, which involved perturbing the components of two behaviorally rich 3D scenes modeled using both the UniSG and UniSG^GA system. The scenes selected for augmentation were a surgical operating room (OR) and a living room. The dataset was split into training and testing sets, with a ratio of 70% for training and 30% for testing. The neural network model was trained for 20 epochs, and the GNN attention mechanism was employed. In the experimentation phase of our approach, we performed 10 runs for each experiment, which, remarkably, achieved a 100% accuracy on both the training and testing splits, demonstrating its effectiveness.

In the experimentation results of the classification task, depicted in Fig 3, we notice a low initial mean accuracy on all methods, indicating a possible need for longer training or model adjustments. Accuracy improves consistently over epochs, exhibiting a few fluctuations in CGA and PGA. The steepness of the Vector+Angle/Axis curve indicates that the model learns quickly as its accuracy get 100% after 7.5 epochs. All curves seem to be converging to 100% accuracy after 17 epochs, a clear sign that it is performing well on the training data. We also notice a low initial loss on all curves, with vector+Angle/Axis curve to be minimizing faster, after 10 epochs, than the others. All loss curves seem to converge after 18 epochs, indicating a well performing model.

5.2 Generative AI using UniSG^GA

Our approach was further tested on a generative task. For this purpose, we generated a dataset of 1000 unique scenes with meaningful layouts, specifically representing a surgical operating room (OR). These scenes were then utilized to train a Conditional Graph Variational AutoEncoder (CGVAE). The primary objective of the CGVAE is to enable the addition of objects to an existing or empty scene based on their category, either sequentially or in bulk. Ultimately, since the utilized UniSG^GA structure includes behavior components, for all object entities, and the respective systems, we aim to train our autoencoder with scene objects that incorporate behavior and provide a complete generative AI solution (currently only topology generation is evaluated).

To achieve this, each entity node within the UniSG^GA was labeled with its corresponding category, e.g., "Scalpel". During the training process, the Encoder module, which encompasses a GNN with Graph Convolutional layers, encodes the N nodes of the graph using their inherent F features and their associated category embeddings. For each of the nodes a vector E is produced, by passing the labels through the embeddings, resulting in a NxE matrix. The resulting encodings/latent space representation for each node, $\mathcal{Z}$, are concatenated with their respective category embeddings, by concatenating the input graph node matrix, of size NxF, with the embeddings, resulting in a Nx$(F+E)$ matrix. This concatenated representation, denoted as $\mathcal{Z}$, is subsequently fed into the Decoder module, which consists of two Multilayer Perceptrons (MLPs): one for decoding the node features from $\hat{\mathcal{Z}}$ and one for decoding the adjacency matrix from $\hat{\mathcal{Z}}$ (see Figure 4).

Our training procedure incorporates several loss functions. Specifically, we employ mean squared error (MSE) loss for node feature reconstruction, binary cross-entropy (BCE) loss for adjacency matrix reconstruction, and Kullback-Leibler (KL) divergence loss to encourage diversity in scene generation. As the model is conditionally trained using these categories, a conditional sampling of the generated scenes is possible, based on specific object categories. This allows the generation of scenes that are greatly influenced by the categories of existing or newly introduced nodes.

The experimentation results of the generative task, we see the loss in Fig 5 (Left), that depicts the discrepancy between the generated and the target output. In this regard we notice that all mean losses are initially relatively low, with PGA and CGA significantly lower, meaning that all models produce high-quality outputs from the start. All loss reductions are minimized rapidly consistently below 1.0. Although all methods seem to converge very early, CGA

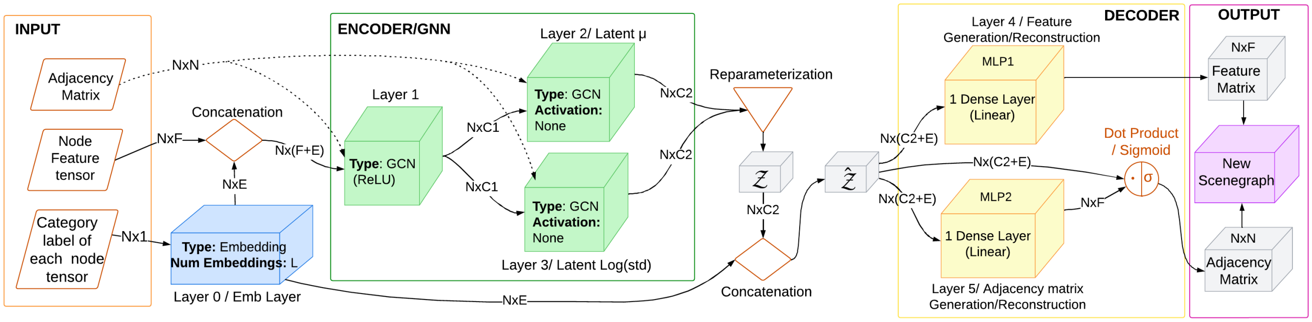

Figure 4: Diagram describing the generative task of Section 5.2.

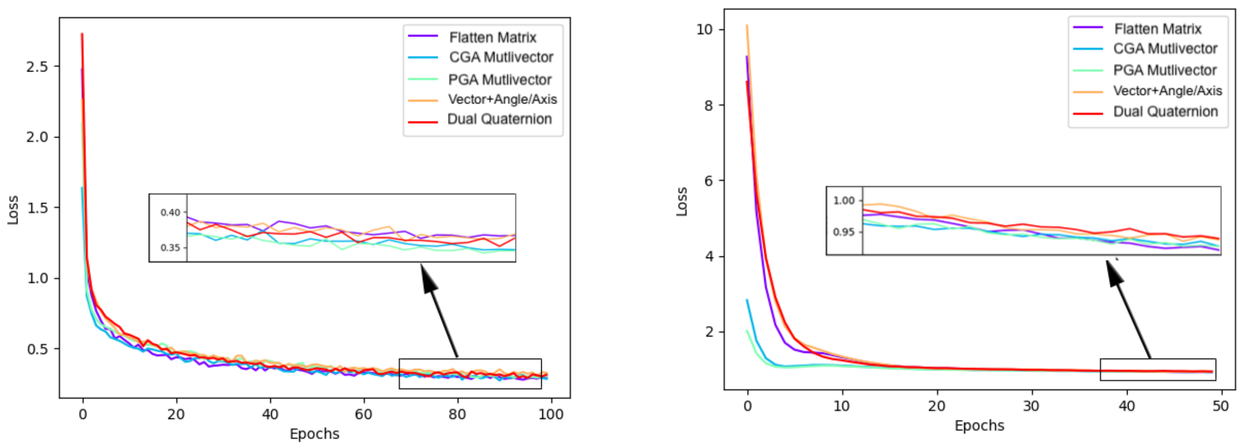

Figure 5: Loss of epochs 0-100, regarding the generative task described in Section 5.2 (Left) and the topology (edge) prediction task described in Section 5.3 (Right). These are mean values after running the experiment 10 times.

and PGA mean loss curves are always below the others; which is indicative that the model is well-performing and that it has learned the generative task.

5.3 Topology prediction

Finally, a topology prediction task was utilized to further evaluate the differences between UniSG and the GA-empowered UniSG^GA. In such tasks, it is common to seek accurate predictions regarding the spatial relationships between objects, including relationships such as "above", "below", "right-of", as well as higher-level relationships like "part-of" or "connected-to". Our approach was specifically evaluated on a topology prediction task involving the identification of the "on-top-of" relationship between two objects. To address this prediction task, we made modifications to our previous model by transforming the Graph Variational AutoEncoder into a simplified Graph AutoEncoder that focused on adjacency matrix reconstruction for predicting the desired topology link based on the graph structure. It is worth noting that while our modified model proves effective for certain topology prediction tasks, it may not capture the complexity of relationships or high-level semantics within the UniSG.

The experimentation results of the topology prediction task, depicted in Fig 5 (Right), show that mean loss (on 10 runs) with CGA and PGA are initialy low and are minimized rapidly compared to other methods. Although all methods seem to converge early, after 15 epochs, CGA and PGA mean loss curves are always below the others, indicating a well-performing model. The loss curves do not show any signs of overfitting which is a direct consequence of the performed data augmentation, increasing diversity and quantity, of the training samples. For each of the 10 runs, a single random scene was generated with 10000 cubes, and link prediction was performed on each run on a single scene.

6 CONCLUSIONS AND FUTURE WORK

In this work, we introduced UniSG^GA, an integrated graph structure designed to be seamlessly compatible with Graph Neural Networks (GNNs) while incorporating behavior data. A key contribution of UniSG^GA is its ability to overcome the challenges associated with transforming a 3D scenegraph (3D-SG) when conducting generative tasks. By leveraging GA forms, UniSG^GA effectively captures and stores the topological relations between objects within the graph, while enhancing the performance and capability of GNNs when handling predictive and generative tasks. This advancement paves the way for more efficient and intuitive approaches in generating complex 3D scenes with embedded behavior.

As a future endeavor, our plan is to train the GNN architecture of UniSG^GA using an extensive corpus of 3D scenes encompassing both content and behavior. This training dataset will consist of various types of scenes, including models and even segments of educational curricula. Through this training process, we aim to evaluate the performance of UniSG on intricate generative AI tasks, with the ultimate objective of enabling the generation of behavior-embedded 3D scenes in a streamlined manner, towards a no-code authoring pipeline.

ACKNOWLEDGMENTS

The project was partially funded by the National Recovery and Resilience Plan "Greece 2.0" - NextGenerationEU, under grant agreement No ΤΑΣΦΡ-06378 (REVIRES-Med), and Innovation project Swiss Accelerator under grant agreement 2155012933 (OMEN-E), supported by Innosuisse.

REFERENCES

[1] E.J. Bayro-Corrochano. 2001. Geometric Neural Computing. *IEEE Transactions on Neural Networks* 12, 5 (Sept. 2001), 968–986.

[2] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

[3] Helisa Dhamo, Fabian Manhardt, Nassir Navab, and Federico Tombari. 2021. Graph-to-3d: End-to-end generation and manipulation of 3d scenes using scene graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 16352–16361.

[4] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 2018. A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[5] Hugo Hadfield, Eric Wieser, Alex Arsenovic, Robert Kern, and The Pygae Team. 2021. *pygae/clifford: v1.4.0.* https://doi.org/10.5281/zenodo.5116900

[6] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, et al. 2020. Strategies for Pre-training Graph Neural Networks. In *International Conference on Learning Representations*.

[7] Manos Kamarianakis, Ilias Chrysovergis, Mike Kentros, and George Papagiannakis. 2022. Recording and Replaying Psychomotor User Actions in VR. In *ACM SIGGRAPH 2022 Posters*. Association for Computing Machinery, New York, NY, USA, Article 30, 2 pages.

[8] Manos Kamarianakis, Ilias Chrysovergis, Nick Lydatakis, Mike Kentros, and George Papagiannakis. 2023. Less Is More: Efficient Networked VR Transformation Handling Using Geometric Algebra. *Advances in Applied Clifford Algebras* 33, 1 (Feb. 2023), 6.

[9] Yanshan Li, Rongjie Xia, Xing Liu, and Qinghua Huang. 2019. Learning Shape-Motion Representations from Geometric Algebra Spatio-Temporal Model for Skeleton-Based Action Recognition. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*. 1066–1071.

[10] Qifan Liu and Wenming Cao. 2022. Geometric algebra graph neural network for cross-domain few-shot classification. *Applied Intelligence* 52, 11 (2022), 12422–12435.

[11] Chenglin Miao, Wen Su, Yanqing Fu, Xihao Chen, and Di Zang. 2022. Long-Term Traffic Speed Prediction Based on Geometric Algebra ConvLSTM and Graph Attention. *2022 IEEE International Conference on Smart Internet of Things* 00 (2022), 108–115.

[12] Yigit Oktar and Mehmet Turkan. 2022. Preserving Spatio-Temporal Information in Machine Learning: A Shift-Invariant k-Means Perspective. *Journal of Signal Processing Systems* 94, 12 (2022), 1471–1483.

[13] George Papagiannakis, Manos Kamarianakis, Antonis Protopsaltis, Dimitris Angelis, and Paul Zikas. 2023. Project Elements: A computational entity-component-system in a scene-graph pythonic framework, for a neural, geometric computer graphics curriculum. In *Eurographics 2023 - Education Papers*. The Eurographics Association.

[14] Alberto Pepe and Joan Lasenby. 2023. *Modeling Orientational Features via Geometric Algebra for 3D Protein Coordinates Prediction*. Preprint. Authorea.

[15] Alberto Pepe, Joan Lasenby, and Pablo Chacón. 2022. Learning rotations. *Mathematical Methods in the Applied Sciences* (2022).

[16] Di Zang, Xihao Chen, Juntao Lei, Zengqiang Wang, Junqi Zhang, Jiujun Cheng, and Keshuang Tang. 2022. A multi-channel geometric algebra residual network for traffic data prediction. *IET Intelligent Transport Systems* 16, 11 (2022), 1549–1560.

[17] Paul Zikas, Mike Kentros, Dimitris Angelis, Antonis Protopsaltis, Manos Kamarianakis, and George Papagiannakis. 2023. *UniSG: Unifying Entity-Component-Systems, 3D & Learning Scenegraphs with GNNs for Generative AI*. Preprint. Authorea.

[18] Paul Zikas, George Papagiannakis, Nick Lydatakis, Steve Kateros, Stavroula Ntoa, Ilia Adami, and Constantine Stephanidis. 2020. Immersive visual scripting based on VR software design patterns for experiential training. *The Visual Computer* 36 (2020), 1965–1977.

[19] Paul Zikas, Antonis Protopsaltis, Nick Lydatakis, Mike Kentros, et al. 2023. MAGES 4.0: Accelerating the world's transition to VR training and democratizing the authoring of the medical metaverse. *IEEE Computer Graphics and Applications* 43, 2 (2023), 43–56.