A Topological Graph Framework for Enhanced Scene Detection and Understanding in Robotic Navigation

Yicheng Duan, Duo Chen
Computer and Data Sciences, School of Engineering
Case Western Reserve University
Cleveland, Ohio, 44106, USA
yxd245@case.edu dxc830@case.edu

Abstract

Recent advances in Embodied AI have increasingly incorporated multi-modal language models (MLLMs), particularly vision-language models (VLMs), to enable instruction-following agents in complex environments. However, such models often struggle with spatial understanding due to the lack of explicit topological reasoning. Graph-based navigation frameworks, which showed strong performance prior to the emergence of large language models, offer a promising solution by encoding structured spatial semantics. In this work, we propose a topological graph framework to support semantic reasoning for MLLMs in navigation tasks. The environment is represented as a graph, where each node is attributed with a semantically meaningful region label (e.g., bedroom, kitchen), and a node classification model predicts these region types. This semantic graph enrichment equips VLM-based agents with structured spatial priors and enforces strict spatial constraints, leading to more accurate and interpretable navigation in unfamiliar environments.

1. Introduction

Embodied AI aims to build intelligent agents that can perceive, reason, and act within physical environments. Traditionally rooted in reinforcement learning, recent advances have increasingly leveraged multimodal models that integrate visual, auditory, and textual inputs. Notable developments include OpenVLA [10], a Vision-Language-Action model, and Google's Gemini Robotics [15], which enables robots to follow spoken instructions in real-world settings.

Vision-and-Language Navigation (VLN) is a core task in Embodied AI, requiring agents to navigate through complex spaces by interpreting natural language instructions grounded in visual observations. Effective VLN models must combine semantic understanding with spatial reasoning. Benchmarks such as Room-to-Room (R2R) [2] have catalyzed progress in this area, while recent efforts [17] explore the use of large-scale foundation models to improve generalization and performance in diverse environments.

Graph-based approaches [1, 16, 17] have been widely adopted in VLN to represent environments as topological graphs, where nodes correspond to physical locations or panoramic observations, and edges capture navigable connectivity. These graphs enable efficient path planning and multimodal reasoning. However, existing methods often lack the ability to project semantic understanding back into the graph, limiting the agent's ability to ground language in the environment at a fine-grained level.

To address this limitation, we propose a topological graph framework that integrates spatial structure with semantic reasoning. Our method constructs a graph from the environment and applies community detection to identify pseudo-clustered sub-scenes, which serve as part of the node features. These enriched node representations are then classified using Graph Convolutional Networks (GCNs) to infer semantic region labels (e.g., *bedroom*, *kitchen*). The resulting graph provides structured spatial priors that enhance instruction-following in VLN tasks. We evaluate our approach on the Matterport3D dataset [4]—the basis of the Room-to-Room benchmark—measuring semantic labeling accuracy.

2. Related work

Graph Convolutional Networks (GCNs), introduced by Kipf and Welling [11], extended convolution operations to non-Euclidean data, enabling effective information propagation across graph nodes. Building on this foundation, Zhao *et al.* [18] proposed Semantic Graph Convolutional Networks (SemGCNs) for 3D human pose estimation, leveraging semantic edge types to model fine-grained dependencies between joints. These results underscore the utility of semantically enriched graphs for capturing structural and contextual relationships.

Recent studies have demonstrated that converting visual data into graph-based representations can significantly enhance navigation performance in Embodied AI. Dong *et al.* [1] proposed ETPNav, which projects image features onto a topological graph to facilitate efficient navigation planning. Hong *et al.* [7] introduced language-visual entity graphs that jointly model linguistic and visual information, while Chen *et al.* [5] designed a dual-scale graph transformer to capture both global and local contextual dependencies. GeoVLN [9] further advanced this direction by incorporating geometry-aware visual features via slot attention mechanisms. Building on these insights, our framework aims to construct graph representations that utilize raw color RGB images, resulting in structures that are both topologically coherent and semantically enriched.

Our task is highly related to the iterative topology learning framework in [16], which generates graph-based floorplans from room-level attributes using link prediction. Together, these advances highlight the versatility of GCNs in modeling complex semantic and spatial dependencies across diverse domains.

3. Methods

3.1. Raw Dataset

Our framework is evaluated using the Matterport3D (MP3D) dataset, a large-scale RGB-D dataset captured from real-world indoor environments. MP3D provides high-resolution RGB images, depth maps, and semantically aligned regions across a diverse range of residential and commercial scenes. Each environment is densely scanned and globally registered, enabling precise spatial reasoning and navigation. In our pipeline, MP3D serves as the foundation for extracting topological graphs, where each node corresponds to a semantically meaningful region sampled from the navigable space. By leveraging panoramic RGB-D imagery and associated 3D geometry, we construct topological graphs enriched with object-level semantic information, which serve as structured inputs for building semantically-aware navigation frameworks [4].

Simulator We utilize the Habitat-Sim simulator, which provides prebuilt MP3D meshes along with semantically aligned regions. This integration enables efficient extraction of node features and corresponding semantic training labels, streamlining the data processing pipeline for graph construction and learning [12–14].

3.2. Framework

Our framework comprises two sequential stages:

1. **Topological Graph Construction** — We first create a topological representation of the environment, yielding the *final topological graph*. This graph captures both geometric connectivity and semantic relationships among sampled regions (see Section 3.2.1).
2. **Node Classification (Leiden + GCNs)** — The graph is then processed by the Leiden community detection algorithm to obtain *pseudo* community labels. For each node we concatenate: (i) its 3-D position, (ii) the pseudo-label, and (iii) object-feature embeddings extracted from the node's field of view. Edge weights are set to the 3-D geodesic distance between connected nodes. These node and edge features are fed into a Graph Convolutional Network (GCN) that refines the labels and produces the final node classifications (see Section 3.2.2).

An overview of the entire pipeline is shown in Figure 1. For a detailed implementation of the framework, please refer to our GitHub repository in 5.

3.2.1 Topological Graph Construction and Method

The first task is to map the scene—initially without any semantic information—into a topological graph. We aim for the graph not only to capture navigable connectivity but also to reflect the underlying structure of the environment, allowing features such as rooms to emerge naturally as clusters or groups within the graph. Therefore, to construct a topological graph from a 3D scene, we must carefully define three key components: nodes,edges and the attributes.

Nodes. In our framework, nodes are defined as walkable sampled points extracted from the navigable regions of the 3D scene. To ensure that the sampling is both representative and spatially balanced, we employ an adaptive Poisson disk sampling strategy, assigning a radius to each sampled point to prevent them from being placed too close to one another. The radius is determined by the openness of the environment: larger radii are used in wide, open areas to avoid excessive density, while smaller radii are applied in confined spaces, such as narrow rooms or hallways, to capture finer details. This adaptive strategy ensures that the topological graph accurately reflects the spatial structure and navigability of the scene. The formula for the sampling radius is given by:

$$r_p = clip(\alpha d(p), r_{min}, r_{max})$$

where r_p denotes the radius of sampled point p, α is a scaling factor that controls the overall sparsity of the sampling, and $d(p)$ represents the estimated openness or clearance at point p. The function clip(x, a, b) constrains the value of x to lie within the range $[a, b]$, ensuring the radius stays within predefined bounds r_{min} and r_{max}.

In our implementation, the number of sampled nodes is not fixed in advance but is implicitly controlled by the scaling factor α, which adjusts the sampling radius at each point

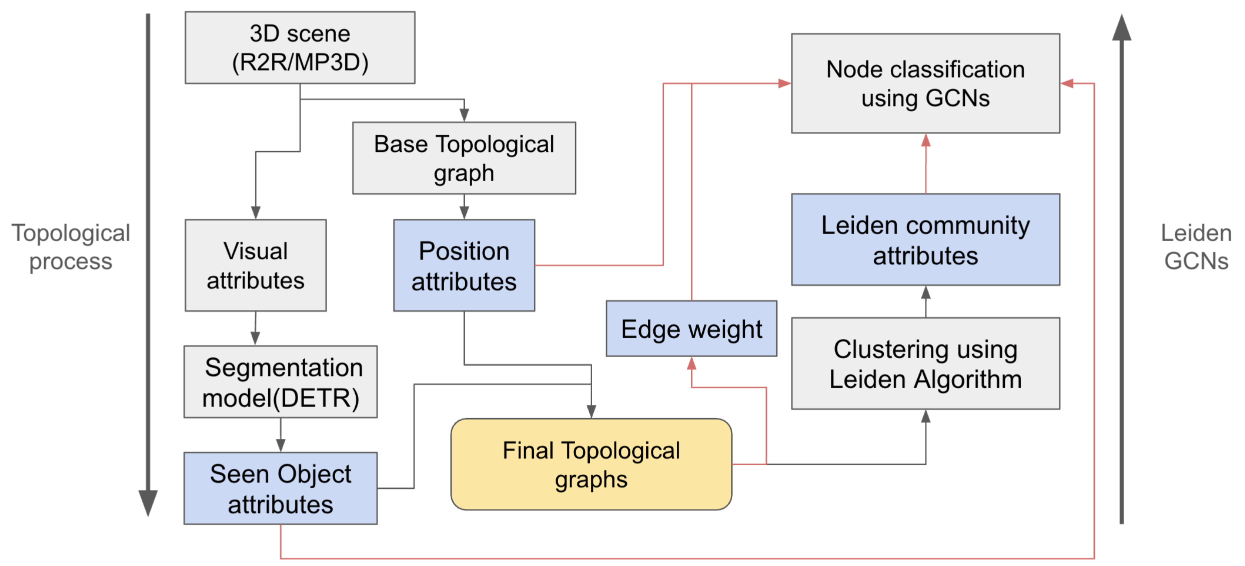


Figure 1. Cross-column overview of the proposed framework. **Stage 1**: Construction of the topological graph. **Stage 2**: Leiden community detection generates pseudo labels that, together with node and edge features from the *final topological graph*, are fed into a GCN for node classifications.

based on local clearance. To estimate the expected number of nodes, we approximate it using the formula:

$$N \approx \frac{A}{\pi R^2}$$

where A is the total walkable area and R is the expected average radius determined by α. This approximation assumes uniform disk-based coverage and provides an upper bound on node count under the adaptive Poisson disk sampling strategy.

To allow each sampled point to estimate the openness of its surrounding area without relying on any semantic information, we propose a method called the ray-clearance algorithm. As discussed earlier, our goal is to achieve denser sampling in smaller or more confined spaces, which requires assigning smaller radii in such regions. The ray-clearance algorithm approximates local openness by casting rays uniformly in multiple directions from a candidate point and measuring their intersection distances with nearby obstacles. The mean distance of these rays, denoted as $d(p)$, serves as an indicator of how open the area is. A lower mean clearance implies a tighter space, prompting a smaller sampling radius, while higher clearance corresponds to open areas and allows for larger radius. However, certain edge cases can still arise—for example, when a point is sampled near a corner. Although the point is technically within a large room, the surrounding geometry may yield a low mean clearance, mistakenly indicating a confined space. To mitigate this issue, we apply a simple outlier filtering strategy: before computing the average clearance, we discard the maximum and minimum ray distances. This helps reduce the influence of extreme values and leads to a more reliable estimation of the local openness. The main process of the algorithm are shown as below:

Algorithm 1: Ray-Clearance Algorithm

```
Input: Point $p$, number of rays $N$, max ray length $L$
Output: Mean clearance value $d(p)$
Initialize total distance $D \leftarrow 0$;
for $i \leftarrow 1$ to $N$ do
    Sample direction $\theta_i$ uniformly from $[0, 2\pi)$;
    Cast ray from $p$ in direction $\theta_i$ up to length $L$;
    Record distance $d_i$ to first obstacle (or $L$ if no hit);
    $D \leftarrow D + d_i$;
Remove maximum and minimum values from $\{d_1, \dots, d_N\}$;
Recompute total distance $D'$ from remaining rays;
Compute average clearance: $d(p) \leftarrow \frac{D'}{N-2}$;
return $d(p)$
```

By using clearance-aware sampling, the nodes not only capture the geometric layout of the environment but also reflect its underlying functional structure. In particular, nodes sampled within the same room—especially in smaller, confined spaces—tend to form dense clusters. These clusters implicitly highlight spatial boundaries and localized regions within the scene, such as individual rooms or corridors. As a result, the topological graph provides not only navigational connectivity but also structural cues that can support higher-level tasks like scene segmentation, region labeling, or semantic reasoning.

Edges In a topological graph, edges represent valid navigable connections between pairs of nodes, indicating that an agent can travel directly between them without encountering obstacles. In our framework, edge connectivity is determined by a one-hop geodesic threshold τ, which defines the maximum allowable path length between two nodes for an edge to be established.

Rather than using a fixed threshold, we define the one-hop connectivity threshold τ adaptively based on each node's sampling radius—itself derived from local clearance. This allows the agent to take longer steps in open areas and enforces shorter connections in confined regions, thus aligning the graph's connectivity with the environment's geometric scale. Concretely, for two nodes p_i and p_j with sampling radii r_i and r_j, we set

$$\tau_{r_i,r_f} = \beta(r_i + r_j)$$

and we add an edge (p_i, p_j) if and only if

$$GeoDist(p_i, p_j) \leq \tau$$

Here, $\mathrm{GeoDist}(\cdot,\cdot)$ is the geodesic distance along the navigable surface. This one-hop constraint preserves spatial locality and adaptiveness: only nodes within locally reachable distance are directly connected. The scaling factor β governs the trade-off between sparsity and connectivity and can be tuned to match environment size or the agent's movement capabilities.

Graph Attributes Each node in the graph is enriched with multiple attributes, including its 3D position (x,y,z), a semantic region label (e.g., living room, bathroom), and a visual observation representing what an agent would perceive at that location. These features provide spatial, semantic, and perceptual context to each node. For each edge, we assign a weight corresponding to the geodesic distance between the connected nodes, calculated over the navigable mesh. This attribute-enhanced graph structure supports more informed spatial reasoning, semantic understanding, and efficient path planning for downstream embodied tasks.

After obtaining the visual observation at each node, we further process the image to extract object-level semantic attributes. Specifically, we utilize the DETR-ResNet-50 model [3] to perform object detection. Only objects with confidence scores exceeding a predefined threshold (90% in our configuration) are retained, resulting in a filtered list of detected objects per node.

To standardize the representation, we define a set of 80 commonly encountered indoor object categories and encode each detected object list as a one-hot vector over this predefined vocabulary. Consequently, each node in the graph is characterized by three attributes: (1) its 3D position (x, y, z), (2) its semantic region label (e.g., *kitchen*, *bathroom*), and (3) a one-hot vector indicating the presence of observed objects. This enriched attribute set enables the graph to capture geometric, semantic, and perceptual context, facilitating downstream embodied AI tasks such as navigation and scene understanding. The parameters of topological graph construction is shown in tabel 3.

3.2.2 Node Classification using Leiden + GCNs

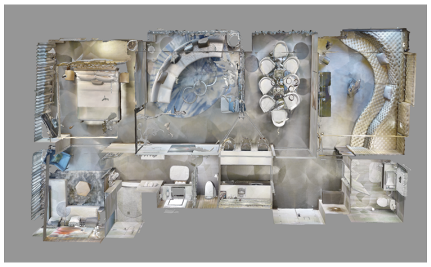
Why still use Leiden? Although ground-truth semantic labels are available during training, a robot exploring an unfamiliar building typically perceives only partial and disconnected fragments, such as a short corridor or a single room, at a time. Relying solely on local visual cues under these conditions is often unreliable.

To provide a broader geometric context, we can apply the Leiden community detection algorithm to *each* intermediate topological graph snapshot in a real-world deployment. This produces geometry-driven *pseudo* labels that reflect the underlying spatial structure. These local cluster IDs are *not* used as supervision targets; rather, they are appended to each node's feature vector via one-hot encoding. This design brings two key advantages:

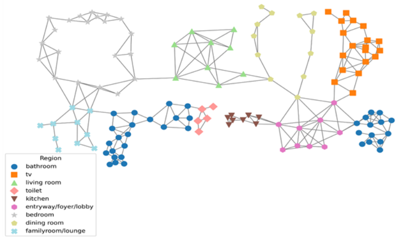
- **Topological prior** — Nodes that are geodesically close and densely connected are assigned the same cluster ID, encouraging the GCN to produce spatially consistent semantic predictions.
- **Rapid adaptation** — Leiden clustering can be recomputed efficiently in near-linear time as new scans are collected, enabling the GCN to continually integrate updated topological context without the need for retraining.

The Leiden community-detection algorithm uses a tunable `resolution` parameter to control the granularity of clustering: values greater than 1 yield finer partitions with smaller, more numerous communities, while values below 1 produce coarser groupings. In our setup, we set the Leiden resolution parameter to 0.8, encouraging moderately coarse clusters that correspond to meaningful spatial substructures (e.g., rooms or corridors) (report in Table 2).

Feature construction. Each node in the graph is represented by a rich feature vector composed of three components: (i) its 3-D spatial position (i.e., X, Y, Z), (ii) a one-hot encoded pseudo label generated by Leiden clustering, and (iii) an object feature vector extracted from the node's 360-degree visual observations. These object features are obtained by aggregating the observed object's semantic label frequency encoded from four color RGB images, each with a 90-degree field of view, effectively covering all directions (see Section 3.2.1). This design enables each node



(a) Top–down view of the reconstructed indoor scene.



(b) Resulting topological graph built over the scene.

Figure 2. (a) A 3D reconstruction rendered in a top–down projection, showing room layouts and furnishings; (b) the corresponding topological graph, where nodes represent sampled vantage points and edges denote navigable connections.

to encode both its local visual appearance and broader topological context.

Edge weights are defined by the actual 3-D geodesic distance between connected nodes, computed from their spatial positions. This accurately preserves spatial relationships across the graph. The resulting combination of geometric, local community, and semantic object information forms a robust input for the GCN, enabling effective and context-aware node classification.

Our training labels for each node are collected using the semantic map annotations provided by the Matterport3D dataset through Habitat-Sim (see Section 3.2.1). The full structure of the constructed PyTorch Geometric dataset is summarized in Table 1.

Table 1. Summary of PyTorch Geometric dataset structure.

Component	**Description**
`data.x`	Node features matrix: $[N \times F]$ Includes: 3D position (float), comm pseudo label (one-hot), object label (frequency).
`data.edge_attr`	Edge weights (geodesic distance): $[E \times 1]$
`data.y`	Ground-truth semantic labels per node: $[N]$

GCN-Based Node Classification As the final stage of our framework, we employ three representative Graph Neural Network (GNN) architectures—vanilla Graph Convolution Network (GCN), GINE, and GraphSAGE—to evaluate the effectiveness of our method in the node Classification task. Each model operates on the topological graph constructed in Section 3.2.1 and takes the enriched node and edge features as input. While all three architectures use structured message-passing layers, they differ in how edge features—specifically, 3D geodesic distances—are incorporated into the aggregation mechanism:

- **Vanilla GCN** [11] Implements first-order spectral convolution by aggregating features from neighboring nodes using a normalized adjacency matrix. In our setup, the edge weights—computed as 3D geodesic distances—are directly embedded into the graph's adjacency matrix and used as weighted connections in the convolution kernel.
- **GINE** (PyTorch modified from "Strategies for Pretraining Graph Neural Networks") [8] Extends GIN by incorporating edge attributes into the message-passing function. In our setup, GINE uses a two-layer MLP to transform the geodesic edge weights into a learned embedding, enabling more expressive modeling of spatial relationships between nodes.
- **GraphSAGE** [6] Applies an inductive mean-aggregation strategy based on neighborhood sampling. In our setup, GraphSAGE does not utilize edge weights during message passing, relying purely on node features and graph connectivity.

All three models are configured with a fixed depth of three layers and a hidden dimension of 64 to ensure a fair comparison. Additional training details are summarized in Table 2. The complete architectural configurations are illustrated in Figure 3. This module completes our framework by refining semantic region predictions for each node.

4. Experiments

4.1. Experimental Setup

We evaluate the three graph convolutional backbones introduced in Section 3.2.2 on a node-classification task that predicts the ground-truth semantic label of every node. The training settings (learning rate, batch size, etc.) are summarised in Table 2. All training and evaluation runs are executed on a single Apple *MPS*-enabled workstation.

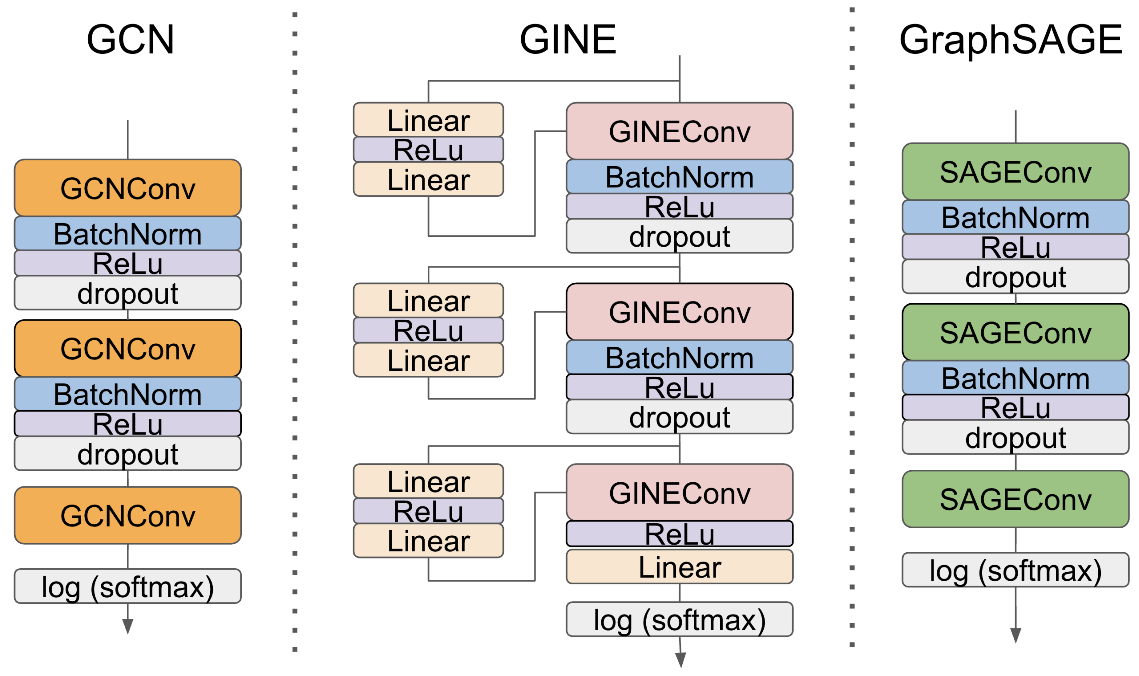


Figure 3. Architectural overview of the three GCN variants used in our framework. Edge features are handled differently across models: vanilla GCN uses weighted adjacency from geodesic distances; GINE uses a 2-layer MLP to embed edge weights; GraphSAGE ignores edge weights entirely and aggregates using mean pooling.

Table 2. Training configuration used across all GCN-based models.

Parameter	**Value**	**Notes**
Resolution	0.8	Leiden community
Batch Size	1	Full-graph training
Learning Rate	0.01	Fixed across all models
Weight Decay	5×10^{-4}	Regularization
Epochs	100	
Hidden Dimension	64	
Dropout Rate	0.5	
Random Seed	12345	For reproducibility
Optimizer	Adam	

Table 3. Configuration for constructing the topological graph

Parameter	**Value**	**Notes**
α (clearance weight)	0.35	Ray-clearance
Number of rays N	8	Ray-clearance
max ray length L	20	Max ray distance
Expected radius r_{exp}	0.35	Number of nodes
Minimal radius r_{min}	0.05	Poisson sampling
Maximal radius r_{max}	2.5	Poisson sampling
Scaling factor β	1.25	Edge construction
Confidence threshold	0.9	Object detection

We construct **187** topological graphs from **90** Matterport3D scenes (Section 3.1); multi-level scenes are treated level-by-level, yielding one graph per floor. Each graph's visualizations are reported on our GitHub in 5. One example of our constructed graph is shown in Figure 2. Each graph is converted into a PyTorch Geometric data object via the feature construction pipeline described in Paragraph 3.2.2. The resulting dataset is split *at the scene level* into disjoint training, validation, and test sets using an 80% / 10% / 10% ratio.

Graph-scale characteristics. As shown in Figure 4, the vast majority of our 187 topological graphs contain fewer than ~ 600 nodes and 1500 edges. Only a handful of outliers reach up to ~ 1200 nodes and 3500 edges—still well within the memory budget of modern GNN accelerators. This mid-scale regime is expressive enough to capture indoor spatial complexity while remaining lightweight for the efficient training of our GCN variants. The global transitivity (clustering coefficient) is concentrated in the range 0.5–0.6, revealing tightly knit local communities connected by long-range links. This pattern mirrors the way rooms and corridors cluster in real buildings, providing rich relational cues for node classification. These statistics confirm that

Table 4. Test accuracy (%) on our node-classification task. The last column cites the LP-Baseline's published accuracy on its own link-prediction task on spatial.

Model	ACC (ours)	ACC (orig. task)
LP-Baseline [16]	—	81.6
Vanilla GCN	14.0	—
GINE	17.7	—
GraphSAGE	**25.3**	—

our topological-graph construction is both *computationally tractable* and *semantically faithful*, forming a solid foundation for the node-classification experiments that follow.

Evaluation Metrics During training, we track three metrics:

(a) **Training loss** – the per-epoch cross-entropy $\mathcal{L}_{\text{CE}}$ averaged over all nodes in the current mini-batch.

(b) **Validation accuracy** (Val-ACC) – the proportion of correctly classified nodes on the validation set, monitored after every epoch.

(c) **Test accuracy** (ACC) – the same metric applied to the unseen test split for final comparison.

Formally, node-classification accuracy is defined as

$$\text{ACC} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[\hat{y}_i = y_i],$$

where N is the number of evaluated nodes, y_i the ground-truth label, and $\hat{y}_i$ the model prediction.

4.2. Baselines

We compare our framework with the state-of-the-art link-prediction model proposed by Yin *et al.*, *Generating Topological Structure of Floorplans from Room Attributes* [16], which treats rooms as nodes and learns latent embeddings to infer adjacency relations; although that work focuses on predicting room–room links rather than node semantics, it leverages the same room-level attributes and thus serves as a strong baseline for our setting where ground-truth room labels are scarce.

4.3. Main Results

Table 4 compares the test accuracy (ACC) of our three GCN variants with the link-prediction baseline proposed by Yin *et al.* [16]. Because the LP-Baseline is tailored to link prediction, it does not yield a meaningful node-classification score on our dataset; we therefore mark that entry with "—". For context, the right-most column reproduces the accuracy the authors reported for their original task of spatial (room classification).

Discussion. The link-prediction baseline of Yin *et al.* remains highly effective for its original spatial-relation task (81.6 %), but, as expected, does not transfer directly to our finer-grained node-classification setting. Among our backbones, GraphSAGE achieves the highest accuracy (25.3 %), followed by GINE and vanilla GCN. A detailed breakdown of how individual components contribute to these results and more findings are presented in the ablation study that follows; we note that the Yin baseline provides a strong point of reference for topology-centric approaches.

4.4. Ablation Study

Configurations. We evaluate 10 variants to quantify the contribution of each component:

(a) **Full** – all components enabled (our complete framework).

(b) **Removal variants**
- (b)1. **–Pos** – without 3-D positional encoding
- (b)2. **–Comm** – without community pseudo-label
- (b)3. **–Obj** – without object semantics
- (b)4. **–EdgeW** – with uniform edge weights

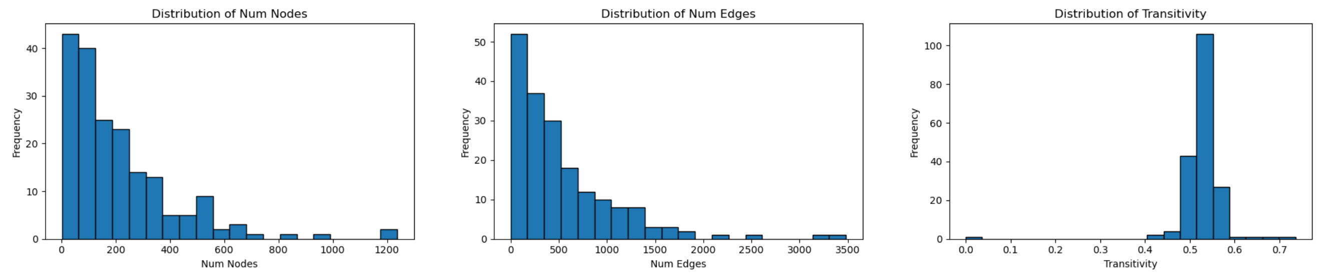
(c) **Isolation variants**
- (c)1. **Pos-Only** – only 3-D positional encoding
- (c)2. **Comm-Only** – only community pseudo-label
- (c)3. **Obj-Only** – only object semantics
- (c)4. **EdgeW-Only** – only edge weights

(d) **Null baseline** – uniform node features *and* uniform edge weights.

The 10 ablation variants are summarized in Table 5, where each component—3-D positional encoding (P), community pseudo-label (C), object semantics (O), and edge weight learning (E)—is toggled on (**1**) or off (**0**) following the label pattern `P#_C#_O#_E#`.

Results and Discussion. Following the experimental protocol in Paragraph 4.1 and the metrics defined in Paragraph 4.1, we train and evaluate all **10** ablation configurations on *each* of the three backbones introduced earlier. Figure 5 plots the *training loss* and *validation accuracy* curves versus epochs, one combined panel per backbone, while Figure 6 summarises the final *test accuracy* for every configuration. Full numerical values are provided in our GitHub repository in 5.

Across all three GNN backbones, the best-performing configuration is `P0_C1_O1_E1`, which incorporates community structure, object semantics, and learned edge weights—while explicitly excluding positional encoding. This configuration achieves test accuracies of 34.1% for GINE, 32.8% for GCN, and 25.3% for GraphSAGE. Notably, removing any single component—community signal (`C0`), object semantics (`O0`), or edge weights (`E0`)—leads to a significant drop in performance (3–10 percentage points),



(a) Node count distribution (b) Edge count distribution (c) Transitivity distribution

Figure 4. Statistics over the 187 generated topological graphs: (a) number of nodes, (b) number of edges, and (c) graph transitivity (global clustering coefficient).

Table 5. Ablation configurations expressed with binary flags (**1**=enabled, **0**=disabled). The label `P#_C#_O#_E#` is the identifier used in our result graphs.

Variant	**Label**	**3-D Pos Encoding**	**Community Pseudo-label**	**Object Semantics**	**Edge Weight**[†]
Full	P1_C1_O1_E1	1	1	1	1
–Pos	P0_C1_O1_E1	0	1	1	1
–Comm	P1_C0_O1_E1	1	0	1	1
–Obj	P1_C1_O0_E1	1	1	0	1
–EdgeW	P1_C1_O1_E0	1	1	1	0
Pos-Only	P1_C0_O0_E0	1	0	0	0
Comm-Only	P0_C1_O0_E0	0	1	0	0
Obj-Only	P0_C0_O1_E0	0	0	1	0
EdgeW-Only	P0_C0_O0_E1	0	0	0	1
Uniform[*]	P0_C0_O0_E0	0	0	0	0

[†] 1 = Geodesic distance edge weights; 0 = uniform edge weight (value 1).
[*] Every node feature is the constant value 1; every edge weight is 1.

underscoring their complementary contributions. In particular, the absence of learned edge weights erases approximately half of the total gain in each model. Additionally, the `P0_C1_O1_E1` configuration demonstrates favorable convergence behavior across all three backbones, as reflected in smooth training loss decay and steadily increasing validation accuracy.

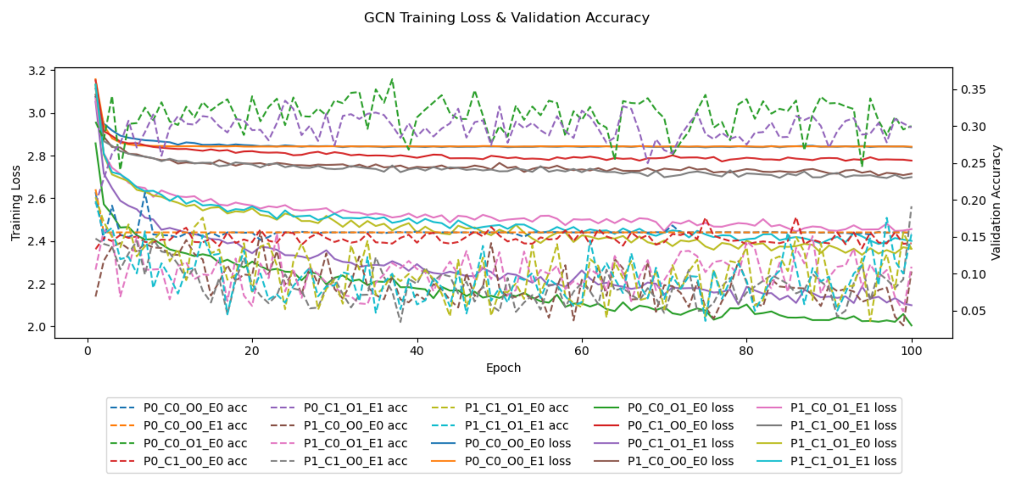
In contrast, introducing 3-D positional encoding (`P1`) consistently degrades performance. The `Pos-Only` variant (`P1_C0_O0_E0`) underperforms the uniform baseline (`P0_C0_O0_E0`) by 2–5 percentage points across all backbones. Furthermore, augmenting the strong `C1_O1_E1` configuration with positional features (`P1_C1_O1_E1`) results in consistent performance deterioration, confirming the detrimental effect of positional offsets in this task. For vanilla GCN and GraphSAGE in particular, adding positional encoding often leads to poor convergence or training instability, suggesting that such features may interfere with the learning dynamics in these architectures.

Among all isolated features, object semantics is the **most impactful**. Enabling object features alone (`P0_C0_O1_E0`) yields a +11–14 pp improvement over the Null uniform baseline in all models, demonstrating that even coarse bag-of-objects descriptors are highly discriminative for semantic label prediction.

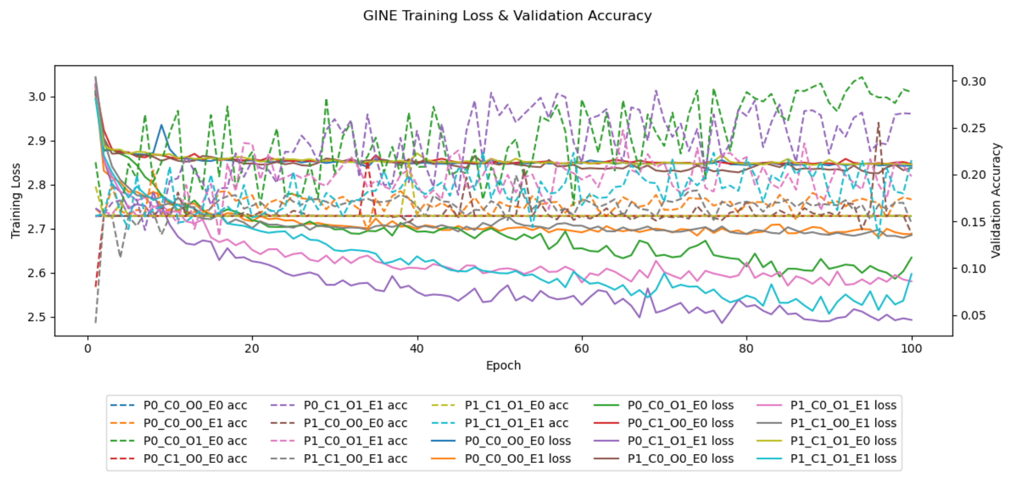
Edge weights, when used in isolation (`P0_C0_O0_E1`), do not provide meaningful improvement for GCN or GraphSAGE, but significantly benefit GINE (+5 pp), whose message-passing mechanism explicitly incorporates edge attributes. This suggests that the utility of edge features is architecture-dependent.

In summary: **Positional encoding hurt**, **community structure helps**, **object semantics helps even more**, and **learned edge weights amplify both**, particularly when paired with edge-aware aggregators such as GINE. The cooperation among these three components (`C1_O1_E1`) explains the peak performance of the `P0_C1_O1_E1` configuration, while the universal drop in any configuration involving positional encoding explains the dip in the full model.

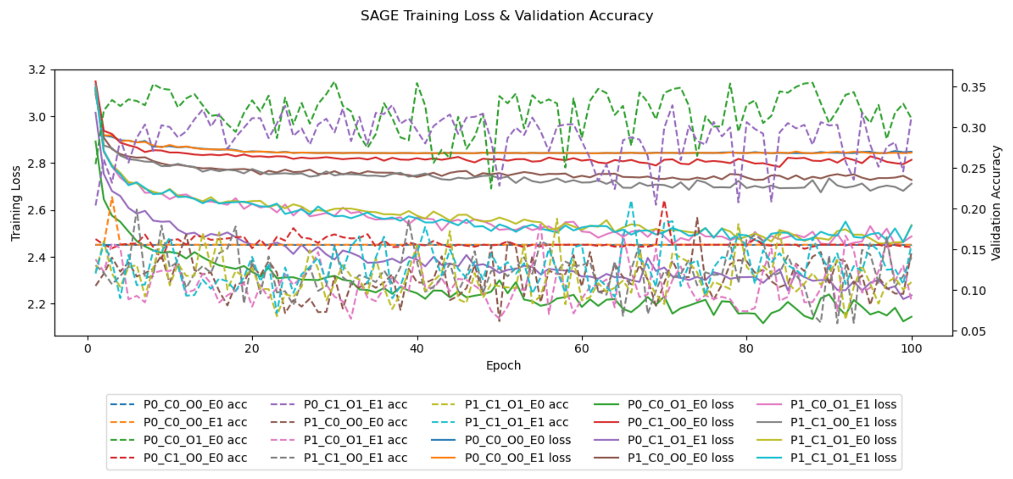
Future works. In this work, we treat the topological graph and node classification as a static and discrete formulation. A promising direction for future work is to explore



(a) Vanilla GCN



(b) GINE



(c) GraphSAGE

Figure 5. Training loss (solid) and validation accuracy (dashed) versus epochs for the 10 ablation configurations on each backbone.

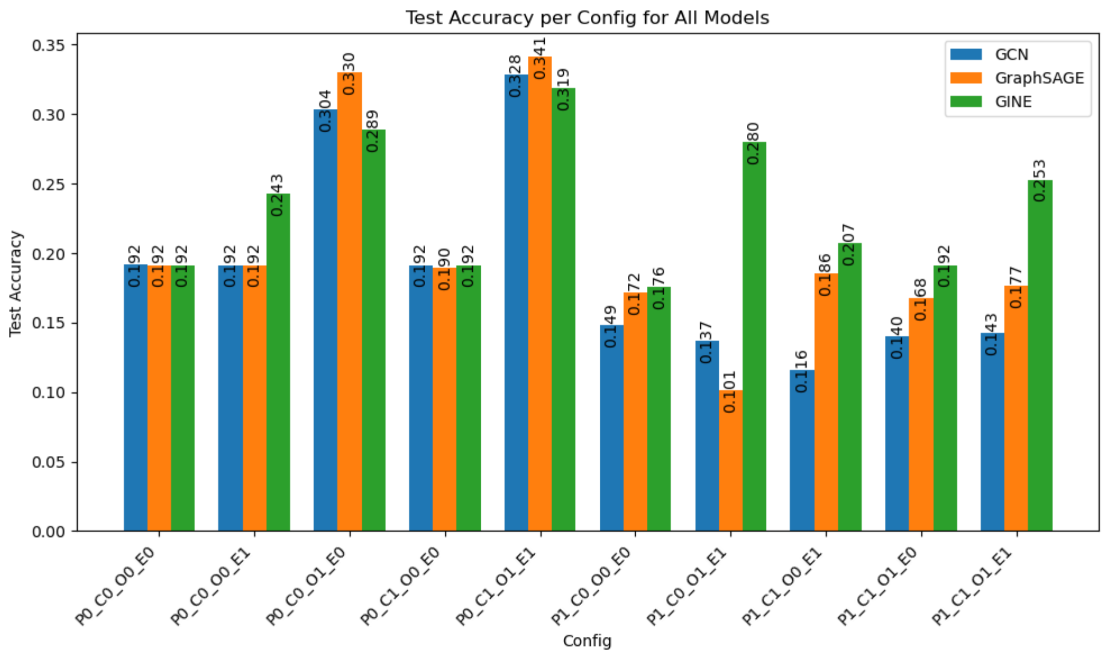


Figure 6. Test accuracy of the 10 configurations across the three backbones.

time-varying or adaptive topological structures, where the graph evolves based on environmental dynamics or agent interactions. Such formulations could better capture temporal patterns and enhance generalization in dynamic navigation scenarios.

5. Conclusion

In this work, we introduced a topological graph framework for enhanced scene understanding and semantic reasoning in robotic navigation. Our method integrates 3-D positional information, Leiden-based community structure, object-level semantics, and geodesic distance edge weights. Extensive ablation studies highlight the contribution of each component, with object semantics emerging as the most influential signal for accurate scene classification.

Interestingly, we found that incorporating 3-D positional encodings consistently degrades model performance across all backbones, emphasizing the importance of selective and task-appropriate feature design in graph-based representations.

Overall, our results demonstrate that a graph combination of community structural, object semantic, and geometric relational forms a robust foundation for semantic-aware navigation in complex indoor environments.

Code Availability

The code and resources for this work are publicly available at https://github.com/YichengDuan/topog.

References

[1] Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. Etpnav: Evolving topological planning for vision-language navigation in continuous environments, 2024. 1, 2

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments, 2018. 1

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020. 4

[4] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 1, 2

[5] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Think global, act local: Dual-scale graph transformer for vision-and-language navigation, 2022. 2

[6] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017. 5

[7] Yicong Hong, Cristian Rodriguez-Opazo, Yuankai Qi, Qi

Wu, and Stephen Gould. Language and visual entity relationship graph for agent navigation, 2020. 2

[8] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Pre-training graph neural networks. *CoRR*, abs/1905.12265, 2019. 5

[9] Jingyang Huo, Qiang Sun, Boyan Jiang, Haitao Lin, and Yanwei Fu. Geovln: Learning geometry-enhanced visual representation with slot attention for vision-and-language navigation, 2023. 2

[10] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024. 1

[11] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. 1, 5

[12] Xavi Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Ruslan Partsey, Jimmy Yang, Ruta Desai, Alexander William Clegg, Michal Hlavac, Tiffany Min, Theo Gervet, Vladimir Vondrus, Vincent-Pierre Berges, John Turner, Oleksandr Maksymets, Zsolt Kira, Mrinal Kalakrishnan, Jitendra Malik, Devendra Singh Chaplot, Unnat Jain, Dhruv Batra, Akshara Rai, and Roozbeh Mottaghi. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023. 2

[13] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[14] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2

[15] Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, Steven Bohez, Konstantinos Bousmalis, Anthony Brohan, Thomas Buschmann, Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Oscar Chang, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, David D'Ambrosio, Sudeep Dasari, Todor Davchev, Coline Devin, Norman Di Palo, Tianli Ding, Adil Dostmohamed, Danny Driess, Yilun Du, Debidatta Dwibedi, Michael Elabd, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Leonard Hasenclever, Nicolas Heess, Brandon Hernaez, Alexander Herzog, R. Alex Hofer, Jan Humplik, Atil Iscen, Mithun George Jacob, Deepali Jain, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Chase Kew, Jerad Kirkland, Sean Kirmani, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Jacky Liang, Yixin Lin, Sharath Maddineni, Anirudha Majumdar, Assaf Hurwitz Michaely, Robert Moreno, Michael Neunert, Francesco Nori, Carolina Parada, Emilio Parisotto, Peter Pastor, Acorn Pooley, Kanishka Rao, Krista Reymann, Dorsa Sadigh, Stefano Saliceti, Pannag Sanketi, Pierre Sermanet, Dhruv Shah, Mohit Sharma, Kathryn Shea, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Jost Tobias Springenberg, Rachel Sterneck, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Vincent Vanhoucke, Jake Varley, Grace Vesom, Giulia Vezzani, Oriol Vinyals, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Yuxiang Yang, Rui Yao, Sergey Yaroshenko, Wenhao Yu, Wentao Yuan, Jingwei Zhang, Tingnan Zhang, Allan Zhou, and Yuxiang Zhou. Gemini robotics: Bringing ai into the physical world, 2025. 1

[16] Yu Yin, Will Hutchcroft, Naji Khosravan, Ivaylo Boyadzhiev, Yun Fu, and Sing Bing Kang. Generating topological structure of floorplans from room attributes. In *Proceedings of the 2022 International Conference on Multimedia Retrieval*, page 295–303, New York, NY, USA, 2022. Association for Computing Machinery. 1, 2, 7

[17] Yue Zhang, Ziqiao Ma, Jialu Li, Yanyuan Qiao, Zun Wang, Joyce Chai, Qi Wu, Mohit Bansal, and Parisa Kordjamshidi. Vision-and-language navigation today and tomorrow: A survey in the era of foundation models, 2024. 1

[18] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N. Metaxas. Semantic graph convolutional networks for 3d human pose regression. *CoRR*, abs/1904.03345, 2019. 1