AffordGraph: Affordance-Guided Relation Filtering for Functional 3D Scene Graph Generation

Eunjae Shin[1], Seok-Young Kim[1] and Woontack Woo[1,2,*]

[1]KAIST UVR Lab, 291 Daehak-ro, Yuseong-gu, 34141, Daejeon, Republic of Korea
[2]KAIST KI-ITC Augmented Reality Research Center, 291 Daehak-ro, Yuseong-gu, 34141, Daejeon, Republic of Korea

Abstract

Functional 3D scene graphs capture objects, interactive elements, and their functional relationships in indoor environments, allowing AR agents to understand the scene functionality. However, existing methods rely on expensive foundation models, making scene graph construction prohibitively slow for practical use. We propose AffordGraph, an efficient pipeline whose primary contribution is affordance-guided relation filtering. Each node is assigned a supply or demand role drawn from a physically grounded taxonomy of six interaction categories; a context-aware reclassification mechanism then refines these assignments by disambiguating them using spatially adjacent objects. Only pairs with compatible supply–demand categories are subsequently forwarded to relationship inference, reducing the dominant LLM/VLM calls. On FunGraph3D, AffordGraph prunes 83.7% of candidate pairs and, by also replacing per-node VLM captioning with a CLIP affordance embedding, cuts directly measured per-scene construction time from about 36 minutes to roughly 35 seconds, an approximately 60× speedup that brings functional scene graph construction within the latency and power budget of AR agents. This speed comes at a moderate, controllable cost in recall.

Keywords

Functional Scene Graph, Affordance, 3D Scene Understanding, Augmented Reality

1. Introduction

Understanding the functional structure of indoor environments is a fundamental challenge for AR agents. Perceiving the world egocentrically through posed RGB-D sensing, the agent is expected to assist the user as they move through spaces; to do so, it must understand not only objects and their spatial relationships but also *how* objects interact: which handle opens which cabinet, which switch controls which ceiling light, or which outlet powers which appliance. This level of functional understanding is what enables the most compelling AR agent experiences: understanding actionable elements in the agent's field of view using 3D scene graphs [1, 2, 3]. Because the user continuously changes viewpoint and expects timely, in-situ responses, the agent must acquire such functional understanding with minimal latency.

Functional 3D scene graphs [1] encode this knowledge as directed graphs where nodes represent objects and interactive elements (e.g., handles, knobs, switches, buttons), and edges encode functional relationships (e.g., *opens*, *controls*, *powers*). The pioneering work OpenFunGraph [1] demonstrated that such graphs can be automatically constructed from posed RGB-D image sequences using foundation models in an open-vocabulary manner, without task-specific training. Its successor [4] further enriched this representation with hierarchical object-part structures and broader coverage of tabletop manipulable objects.

Despite these advances, a critical and largely unaddressed bottleneck remains: **computational efficiency**. The OpenFunGraph pipeline involves two particularly expensive stages. The first is *node description generation*: for every detected object and interactive element, a VLM (specifically LLaVA [5]) is invoked to produce a natural language description of the node (e.g., "a silver knob on the left side of the cabinet door"). These descriptions are later used as input for LLM-based relationship inference. For

APMAR'26: The 18th Asia-Pacific Workshop on Mixed and Augmented Reality, Aug. 3-5, 2026, Taipei, Taiwan
*Corresponding author.
ejshin@kaist.ac.kr (E. Shin); seokyoung@kaist.ac.kr (S. Kim); wwoo@kaist.ac.kr (W. Woo)
0009-0005-9297-3487 (E. Shin); 0009-0008-7336-5699 (S. Kim); 0000-0002-5501-4421 (W. Woo)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-09-17

CEUR-WS.org/Vol-4267/paper18.pdf

a scene with N objects and M interactive elements, this requires $O(N + M)$ VLM calls. The second bottleneck is *remote relationship inference*: for interactive elements not locally connected to any object, the pipeline queries an LLM for every candidate element-object pair to assess functional feasibility, resulting in up to $O(N \times M)$ model calls. Such pipelines are fundamentally misaligned with AR agents, where perception must run under stringent latency, memory, and energy budgets, whether on-device or through constrained edge offload, and remain responsive as the user explores. Making functional scene graph construction efficient is therefore not a convenience but a prerequisite for deploying it on AR platforms.

We identify two key observations that motivate our approach. First, **node descriptions are a costly intermediate representation that can be replaced**. OpenFunGraph generates natural language descriptions to give the LLM a textual understanding of each node. However, the information critical for functional relationship inference is not the visual appearance of a node, but rather its *functional role*: is this object something that controls another object (a *supply*), or something that is controlled (a *demand*)? This supply-demand role can be determined efficiently using CLIP text embedding similarity without invoking a VLM. Second, **the supply-demand structure of functional relationships provides a strong prior for pruning candidates**. A switch (supply) can only be functionally related to objects that can be controlled (demand), not to other supply objects or unrelated demand objects. Exploiting this structure allows us to eliminate the majority of candidate pairs before any LLM/VLM inference.

Based on these observations, we propose **AffordGraph**, an efficient pipeline for functional 3D scene graph construction. Our contributions are:

- **Affordance Role Assignment**: We classify each node into one of twelve supply/demand affordance categories via cosine similarity between the node's semantic label and CLIP affordance-description embeddings. This lightweight assignment provides the supply–demand roles that drive our filtering stage.
- **Context-Aware Affordance Reclassification**: Many interactive elements are inherently ambiguous in isolation: a knob could open a cabinet or adjust a radiator. We propose a context-aware reclassification mechanism that examines the 3D spatial neighborhood of each ambiguous node and uses the affordance categories of neighboring objects to resolve the ambiguity, improving the accuracy of downstream filtering.
- **Affordance-Guided Pair Filtering**: We introduce a supply-demand compatibility filter that prunes candidate pairs before relationship inference. Only pairs where the interactive element holds a supply role and the object holds a compatible demand role are forwarded to the LLM/VLM inference stage. On FunGraph3D this reduces candidate pairs by 83.7%, cutting the dominant $O(N \times M)$ relationship-inference calls to $O(K)$ with $K \ll N \times M$. Together with the CLIP-based replacement of node captioning, this lowers directly measured per-scene construction time from $\sim$36 min to $\sim$35 s (about $60\times$), which is the central result of this work.

2. Related Work

2.1. Functional Scene Understanding

The concept of affordance, introduced by Gibson [6], describes the action possibilities that objects offer to agents in their environment. In computer vision and robotics, affordance understanding has evolved from early pixel-level segmentation approaches [7] to language-guided methods [8, 9] and most recently to sequential affordance reasoning [10]. These works primarily focus on affordances of individual objects in isolation.

SceneFun3D [3] established the first benchmark for fine-grained functionality understanding in 3D scenes, introducing annotations of interactive elements such as handles, knobs, and buttons within real-world indoor environments. While SceneFun3D provides rich affordance-level annotations, it does not model the functional relationships between interactive elements and the objects they control.

OpenFunGraph [1] addressed this gap by introducing functional 3D scene graphs that jointly model objects, interactive elements, and their functional relationships. The method leverages RAM++ [11] and GroundingDINO [12] for node detection, LLaVA [5] for node description generation, and GPT-4 for relationship inference. OpenFunGraph distinguishes between *local* relationships (where the interactive element is physically part of the object, e.g., a door handle) and *remote* relationships (where the interactive element controls the object from a distance, e.g., a wall switch controlling a ceiling light). [4] built upon OpenFunGraph by introducing hierarchical structures that model intermediate functional carriers (e.g., drawers) between objects and interactive units, as well as expanding coverage to tabletop manipulable objects. Both works heavily rely on VLM/LLM calls throughout the pipeline, and our work directly addresses this inefficiency.

2.2. Affordance Detection and Open-Vocabulary Grounding

OpenAD [9] pioneered open-vocabulary affordance detection in 3D point clouds by jointly learning text-point correlations using CLIP [13], enabling zero-shot detection of previously unseen affordances without requiring annotated examples. 3D-AffordanceLLM [14] reformulated affordance detection as an instruction reasoning task, introducing the Instruction Reasoning Affordance Segmentation (IRAS) paradigm and leveraging LLMs with a specialized `<AFF>` token for context-aware affordance segmentation. TASA [15] introduced task-aware affordance detection that first extracts affordance concepts from task descriptions, then uses these concepts to guide CLIP-based frame selection and subsequent 2D-to-3D affordance grounding.

Our work is inspired by these approaches but differs fundamentally in purpose: rather than performing fine-grained affordance *segmentation*, we use affordance categories as a lightweight categorical signal to determine functional *compatibility* between pairs of scene elements, enabling efficient pruning without invoking VLMs or LLMs.

2.3. Scene Graph Construction

ConceptGraph [2] proposed an open-vocabulary 3D scene graph construction approach using object-level CLIP features and direct LLM prompting for relationship inference. Open3DSG [16] leveraged pre-trained vision-language models for open-vocabulary 3D scene graph prediction from point clouds. More recently, DeWorldSG [17] improved the robustness of lifting-based 3D scene graph generation by modeling objects as probabilistic 3D Gaussians and refining ambiguous relations with priors from a video world model, while SceneLinker [18] focused on generating layout-consistent 3D scenes from predicted scene graphs via a joint shape-layout VAE. However, these methods, along with ConceptGraph and Open3DSG, focus primarily on *spatial* relationships between objects and do not address functional relationships involving part-level interactive elements.

MomaGraph [19] proposed a unified scene graph representation that jointly encodes spatial and functional relationships with state annotations. Importantly, MomaGraph defines six functional relationship types derived from over 350 household scenes, providing empirical validation that functional relationships in indoor environments can be meaningfully categorized into a small set of interaction types. Our affordance taxonomy is directly grounded in this insight: if functional relationships can be categorized, then the objects participating in those relationships can be pre-assigned to compatible supply-demand roles, enabling filtering before inference.

3. Method

3.1. Problem Formulation

Given a sequence of posed RGB-D frames $\{(\mathcal{I}_i, \mathcal{D}_i)\}_{i=1}^{n}$ of an indoor environment, our goal is to construct a functional 3D scene graph $\mathcal{G} = (\mathcal{O}, \mathcal{I}_e, \mathcal{R})$, where $\mathcal{O}$ denotes the set of object nodes, $\mathcal{I}_e$ denotes the set of interactive element nodes, and $\mathcal{R}$ denotes the set of directed functional relationship

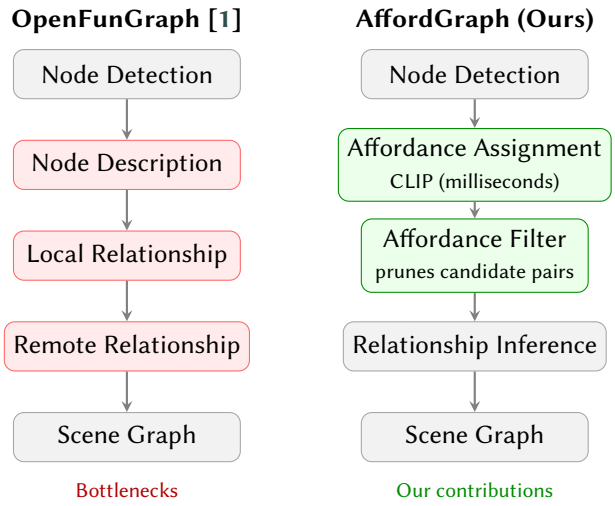

Figure 1: Pipeline comparison. Red boxes in OpenFunGraph indicate computationally expensive stages. Our AffordGraph replaces the node description stage with fast CLIP-based affordance assignment and inserts an affordance-guided filter before relationship inference (applied to both local and remote candidate pairs).

edges pointing from interactive elements to the objects they control. Each node $v \in \mathcal{O} \cup \mathcal{I}_e$ is associated with a 3D point cloud $\mathcal{P}_v$, a 3D bounding box $\mathcal{B}_v$, a semantic label l_v, and a centroid $c_v \in \mathbb{R}^3$.

Our objective is to drastically reduce the total number of VLM and LLM calls required for construction, keeping graph quality as close to OpenFunGraph [1] as the filtering permits, so that construction becomes practical for real-world AR deployment.

3.2. Pipeline Overview

Figure 1 illustrates our AffordGraph pipeline in comparison with OpenFunGraph. Our pipeline consists of five stages: (1) node detection, (2) affordance assignment with context-aware reclassification, (3) affordance-guided pair filtering, (4) relationship inference, and (5) graph construction. The key modifications relative to OpenFunGraph are the replacement of Stage 2 (node description generation via LLaVA) with our affordance assignment stage, and the insertion of Stage 3 (affordance-guided filtering) before relationship inference, applied uniformly to all candidate pairs regardless of local or remote relationship type.

3.3. Stage 1: Node Detection

Following OpenFunGraph [1], we perform node detection in two sub-steps. First, RAM++ [11] is applied to each input frame to generate a set of semantic object tags (e.g., “cabinet”, “door”, “knob”). These tags serve as prompts for GroundingDINO [12], which detects 2D bounding boxes and generates segmentation masks for each tag. For interactive elements, we follow the strategy of OpenFunGraph: the LLM is queried once (not per pair) to enumerate the types of interactive elements associated with each detected object type, and these element types are used as additional prompts for GroundingDINO.

The 2D detection results from multiple frames are fused in 3D using the associated depth maps and camera projection matrices, following the multi-view fusion approach of ConceptGraph [2]. Each fused 3D node candidate stores its associated point cloud, 3D bounding box, centroid, and semantic label. This stage is shared with OpenFunGraph and is not the focus of our contribution.

3.4. Stage 2: Affordance Assignment

Taxonomy Design. We define a supply-demand taxonomy consisting of six interaction categories, each with a supply role (the interactive element that initiates the interaction) and a demand role (the object that receives or responds to the interaction). The taxonomy is grounded in the functional relationship

types validated by prior large-scale household scene datasets: MomaGraph [19] defines six functional relationship types across scenes. Our categories are designed to cover the intersection of these validated types:

1. **Open/Close**: Physical opening and closing of containers and passages.
   Supply: handle, knob, lever, latch. *Demand*: door, drawer, cabinet, lid.
2. **Device Control**: Switching devices on, off, or between modes.
   Supply: switch, remote control, switch panel. *Demand*: ceiling light, TV, fan, appliance.
3. **Water Flow**: Controlling the flow of water or liquids.
   Supply: faucet, tap, valve, showerhead. *Demand*: sink, bathtub, drain, toilet bowl.
4. **Power Supply**: Providing or receiving electrical power.
   Supply: outlet, power strip, socket. *Demand*: kettle, lamp, hair dryer, charger.
5. **Parameter Adjustment**: Adjusting a continuous parameter such as temperature or intensity.
   Supply: dial, knob, slider, thermostat. *Demand*: radiator, oven, air conditioner, stove.
6. **Special Function**: Triggering specialized one-shot functions.
   Supply: flush button, dispenser button. *Demand*: toilet, soap dispenser, trashcan.

This results in twelve labels in total: six supply labels and six demand labels. Each label is associated with a natural language description designed to be discriminative in CLIP's text embedding space, as listed in Table 1.

CLIP-Based Assignment. For each detected node with semantic label l_v, we assign an affordance category using cosine similarity between CLIP text embeddings. Let $\mathcal{A} = \{a_1, \ldots, a_{12}\}$ denote the set of affordance labels and d_a the natural language description of affordance a. The affordance assignment is:

$$a_v^* = \arg\max_{a \in \mathcal{A}} \cos\big(\phi(l_v),\ \phi(d_a)\big) \tag{1}$$

where $\phi(\cdot)$ denotes the CLIP ViT-B/32 text encoder and $\cos(\cdot, \cdot)$ denotes cosine similarity. All label embeddings $\{\phi(d_a)\}_{a \in \mathcal{A}}$ are pre-computed once and cached; runtime assignment requires only $|\mathcal{A}| = 12$ dot products per node. This reduces the affordance assignment stage from $O(N + M)$ LLaVA calls to a single batch text encoding operation completing in under one second for any realistic scene.

Although the assignment is a purely text-to-text comparison, we deliberately use CLIP's text encoder rather than a dedicated sentence encoder. Table 2 compares the resulting affordance-filter quality across encoders on the FunGraph3D scenes, holding the filtering logic fixed. CLIP attains the highest recall and F1 at a comparable pruning rate, outperforming even all-mpnet-base-v2, the strongest general-purpose sentence-embedding model. We attribute this to CLIP's contrastive image-text pre-training: the similarity between an object label and an affordance description reflects shared *physical and visual* semantics, how objects look and how they are manipulated, rather than the generic sentence similarity optimized by text-only encoders [20]. This is precisely the signal needed for affordance role assignment. Prior work has shown that CLIP's text encoder can outperform dedicated language models on vision-centric phrase understanding tasks when paired with appropriate prompts [21], consistent with our findings.

Context-Aware Reclassification. A fundamental limitation of label-based CLIP assignment is that many interactive elements are lexically ambiguous across affordance categories. The label "knob" is associated with both *open_close_supply* (a cabinet knob) and *parameter_adjustment_supply* (a radiator knob) in CLIP's embedding space. In isolation, CLIP often defaults to the most statistically common association, leading to systematic misclassification.

To address this, we propose a context-aware reclassification mechanism that uses the 3D spatial neighborhood of each ambiguous supply node to refine its affordance assignment. The key insight is that in real indoor environments, functional supply-demand pairs are almost always spatially co-located for local relationships (e.g., a radiator knob is adjacent to a radiator). Thus, if an ambiguous supply node has a nearby demand node whose affordance category differs from the initially assigned supply

Table 1
Complete taxonomy with CLIP prompt descriptions. Descriptions are designed to be semantically discriminative in CLIP's text embedding space.

Affordance Label	CLIP Prompt Description
`open_close_supply`	an object used to pull, push, or press to open or close something, like a handle or door knob
`open_close_demand`	an object that can be opened or closed by pulling, pushing, or pressing, like a door, drawer, or cabinet
`device_control_supply`	an object used to control, switch on, or switch off a device, like a remote control or wall switch
`device_control_demand`	an electrical device that can be controlled, switched on, or switched off, like a television or ceiling light
`water_flow_supply`	an object used to control the flow of water, like a faucet, tap, or valve
`water_flow_demand`	an object that receives or collects water flow, like a sink, bathtub, or toilet bowl
`power_supply_supply`	an object that provides electrical power to other devices, like a wall outlet, power strip, or socket
`power_supply_demand`	an object that requires electrical power to operate, like a kettle, lamp, or hair dryer
`parameter_adjustment_supply`	an object used to rotate or slide to adjust a temperature, speed, or setting, like a dial or adjustment knob
`parameter_adjustment_demand`	an object whose temperature, speed, or operating parameter is adjusted by rotation or sliding, like a radiator or oven
`special_function_supply`	an object pressed or activated to trigger a specific one-shot function, like a flush button or soap dispenser button
`special_function_demand`	an object that performs a specific function when its trigger is activated, like a toilet or soap dispenser

Table 2
Affordance-filter quality with different text encoders for the role-assignment step (14 FunGraph3D scenes, filtering logic held fixed). CLIP's text encoder yields the best recall and F1 at a comparable pruning rate, despite the comparison being purely text-to-text.

Text Encoder	Reduct.	Recall	Prec.	F1	FN
CLIP ViT-B/32	83.7%	**0.765**	**0.548**	**0.639**	**28**
SBERT all-mpnet-base-v2	84.3%	0.689	0.512	0.588	37
SBERT all-MiniLM-L6-v2	86.1%	0.529	0.444	0.483	56
BERT-base (mean-pool)	82.8%	0.454	0.307	0.366	65

category, and if reclassifying the supply node to be compatible with the nearby demand is physically plausible, we perform the reclassification.

Formally, for an ambiguous supply node v with centroid c_v and initial assignment a_v^*, we search for spatially adjacent demand nodes:

$$\mathcal{N}(v, \delta) = \{u \in \mathcal{V} \mid \|c_v - c_u\|_2 \leq \delta,\ \text{role}(a_u) = \text{demand}\} \tag{2}$$

Algorithm 1 Context-Aware Affordance Assignment

Require: Node set $\mathcal{V}$, centroids $\{c_v\}$, label embeddings $\{\phi(d_a)\}$, distance threshold δ
Ensure: Affordance assignments $\{a_v\}_{v \in \mathcal{V}}$

Pre-compute $\phi(d_a)$ for all $a \in \mathcal{A}$ ▷ One-time
for each node $v \in \mathcal{V}$ **do**
  $a_v \leftarrow \arg\max_a \cos(\phi(l_v), \phi(d_a))$ ▷ CLIP assignment
end for
for each supply node v with $\Phi_v \neq \emptyset$ **do** ▷ Ambiguous supply
  $\mathcal{N} \leftarrow \{u : \|c_v - c_u\| \leq \delta,\ \text{role}(a_u) = \text{demand}\}$
  if $\mathcal{N} \neq \emptyset$ **then**
    $u^* \leftarrow \arg\min_{u \in \mathcal{N}} \|c_v - c_u\|$
    if $\sigma(a_{u^*}) \in \Phi_v$ **and** $\sigma(a_{u^*}) \neq a_v$ **then**
      $a_v \leftarrow \sigma(a_{u^*})$ ▷ Reclassify
    end if
  end if
end for
return $\{a_v\}$

For each neighbor $u \in \mathcal{N}(v, \delta)$, we compute the corresponding supply affordance via a deterministic mapping $\sigma : \mathcal{A}_{\text{demand}} \to \mathcal{A}_{\text{supply}}$ (e.g., $\sigma($*parameter_adjustment_demand*$) =$ *parameter_adjustment_supply*). If $\sigma(a_u) \in \Phi_v$ (the set of physically allowable supply categories for object type v) and $\sigma(a_u) \neq a_v^*$, we reclassify:

$$a_v \leftarrow \sigma(a_{u^*}), \quad u^* = \underset{u \in \mathcal{N}(v,\delta)}{\arg\min} \|c_v - c_u\|_2 \tag{3}$$

where u^* is the nearest valid neighbor. The allowable reclassification set Φ_v encodes physical constraints. This prevents physically impossible reclassifications (e.g., reclassifying a handle as *power_supply_supply*). We set $\delta = 1.0$m empirically, which covers typical local element-object distances in indoor scenes. Algorithm 1 summarizes the complete affordance assignment procedure.

3.5. Stage 3: Affordance-Guided Pair Filtering

Given affordance assignments for all nodes, we apply a compatibility filter to prune candidate interactive element-object pairs before relationship inference. The filter is based on the supply-demand structure of our taxonomy: a valid functional relationship requires one supply node and one compatible demand node.

Filtering Rule. For a candidate pair (v_i, v_j) where $v_i \in \mathcal{I}_e$ is an interactive element and $v_j \in \mathcal{O}$ is an object, the pair passes the filter if:

$$\text{keep}(v_i, v_j) = \begin{cases} \text{True} & \text{if cat}(a_{v_i}) = \text{cat}(a_{v_j}) \wedge \text{role}(a_{v_i}) \neq \text{role}(a_{v_j}) \\ \text{True} & \text{if cross-category power-device exception} \\ \text{False} & \text{otherwise} \end{cases} \tag{4}$$

where $\text{cat}(\cdot)$ extracts the interaction category (e.g., *open_close*) and $\text{role}(\cdot) \in \{\text{supply}, \text{demand}\}$. The first condition passes same-category supply-demand pairs. The cross-category exception handles the physically valid case where an electrical outlet (*power_supply_supply*) powers electrically-controlled devices (*device_control_demand*), as these objects share a power delivery relationship despite belonging to different primary categories.

Filtering Rate. Let $P = |\mathcal{I}_e| \times |\mathcal{O}|$ be the total number of candidate pairs before filtering and K be the number of pairs passing the filter. The filtering rate $r = 1 - K/P$ measures the fraction of pairs eliminated. On FunGraph3D, we observe $r = 83.7\%$ across all 14 scenes (166 of 1,021 candidate pairs

Table 3
Node association, edge prediction, and overall triplet evaluation on FunGraph3D. Node Association is the object-recall term of the triplet evaluation (n_{na}/n_{tr}), so that Overall Triplets = Node Association × Edge Prediction and all columns are directly comparable to OpenFunGraph. Results marked with * are from OpenFunGraph [1].

	Node Assoc.		Edge Pred.		Overall Triplets	
Method	R@5	R@10	R@5	R@10	R@5	R@10
OpenFunGraph*	45.8	49.3	65.1	91.4	29.8	45.0
AffordGraph (Ours)	27.3	33.9	60.6	**92.7**	16.5	31.4

retained; see Section 4), meaning that only 16.3% of candidate pairs are forwarded to the relationship inference stage. Since each forwarded pair requires at least one VLM or LLM call, this translates directly to an 83.7% reduction in the dominant computational bottleneck of the pipeline.

3.6. Stage 4: Relationship Inference

For pairs passing the affordance filter, we follow OpenFunGraph's sequential reasoning strategy to infer functional relationships. We distinguish between two types of relationships based on spatial proximity.
Local Relationships. Interactive elements that are spatially co-located with an object (i.e., their bounding boxes overlap or are in close proximity) are candidates for local functional relationships (e.g., a cabinet handle is part of the cabinet). For each such pair passing the affordance filter, the LLM is queried with the node labels and bounding box information to assess whether a local functional connection is plausible, and if so, to generate a description of the relationship.
Remote Relationships. Interactive elements that are not locally attached to any object may still control objects from a distance (e.g., a wall switch controlling a ceiling light). For each candidate remote pair passing the affordance filter, we follow OpenFunGraph's confidence-aware reasoning: the VLM is queried with top-view images of both elements to assess visual feasibility, and a confidence score is assigned. The LLM then takes a global view of all proposed remote connections to assign final confidence scores and resolve conflicts.

4. Experiments

4.1. Experimental Setup

Dataset. We evaluate on FunGraph3D [1], a dataset of 14 real-world indoor scenes comprising 6 kitchens, 2 living rooms, 3 bedrooms, and 3 bathrooms. Scenes are captured with high-resolution Faro laser scans and accompanying iPad video sequences. The dataset contains 228 annotated functional relationships across 201 interactive elements and 146 objects of interest.
Evaluation Metrics. Following OpenFunGraph [1], we report Recall@K scores decomposed into three components. *Node Association* (R@5, R@10) measures whether the correct object-element pair is detected. *Edge Prediction* (R@5, R@10) measures whether the functional relationship type is correctly inferred given a correct node association. *Overall Triplets* (R@5, R@10) requires all three components: object node, interactive element node, and relationship to be correctly retrieved. In addition, we report two efficiency metrics: (1) the number of VLM/LLM calls required per scene, and (2) wall-clock construction time per scene (measured on a machine with an NVIDIA A40 GPU using GPT-4o as the LLM and LLaVA-1.6 as the VLM). Our primary comparison is against OpenFunGraph [1], the closest work in terms of task setup and dataset.

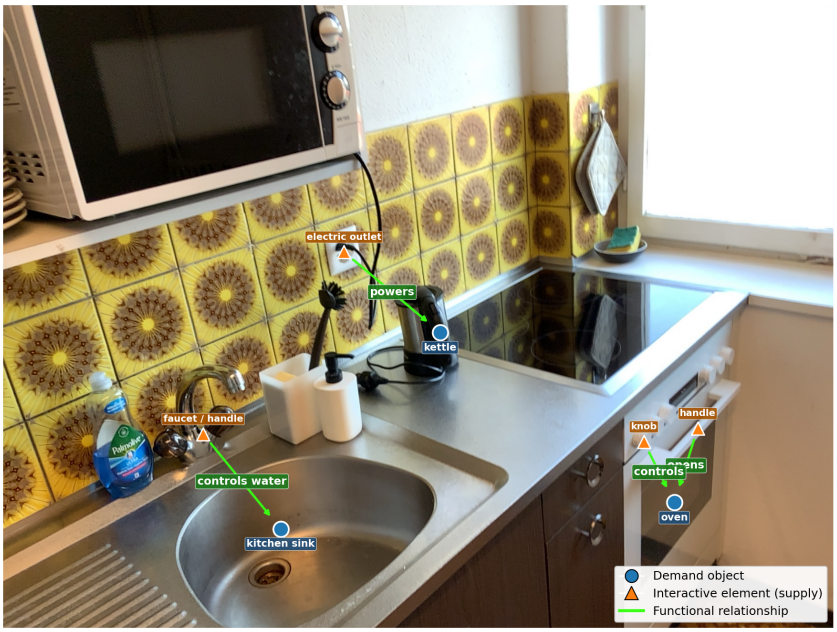

Figure 2: Qualitative functional scene graph predicted by AffordGraph, overlaid on a representative RGB frame. Orange triangles mark interactive elements (supply role), blue circles mark the objects they act on (demand role), and green arrows denote the inferred functional relationship. The displayed relationships are functional pairs correctly retained by the affordance filter.

4.2. Scene Graph Quality

Table 3 reports recall under OpenFunGraph's official evaluator. AffordGraph deliberately trades recall for speed: pruning 83.7% of candidate pairs lowers node-association recall and overall triplet recall, while edge prediction stays comparable. This recall is recoverable by forwarding all $O(N \times M)$ pairs, but only at an order-of-magnitude higher inference cost; AffordGraph instead optimizes for speed, which is where its contribution lies and which we quantify next (Table 4).

4.3. Qualitative Results

Figure 2 shows the functional scene graph produced by AffordGraph for a representative kitchen scene, overlaid on a dataset RGB frame. The supply–demand filter correctly links each interactive element to the object it acts on: a faucet controlling the sink, a handle opening the oven, an outlet powering the kettle, and a knob on the oven spanning the *controls water*, *opens*, *powers*, and *controls* relationship types.

4.4. Efficiency Analysis

Efficiency is AffordGraph's central contribution, and Table 4 quantifies it with directly measured per-stage wall-clock times, averaged over all 14 FunGraph3D scenes. The dominant cost of OpenFunGraph is node captioning: producing LLaVA descriptions for every detected object and interactive element takes on average 2080 s (∼35 min) per scene. AffordGraph eliminates this stage entirely, replacing it with CLIP-based affordance assignment that completes in 0.13s, more than four orders of magnitude faster. Affordance filtering further reduces relationship inference by forwarding only 166 of 1,021 candidate pairs (an 83.7% reduction across the 14 scenes), cutting that stage from 79 s to 35 s. In total, per-scene construction time drops from ∼36 min to ∼35s, roughly 60× speedup on the stages that differ between the two pipelines (node detection, a shared upstream stage, is excluded from both). Because LLaVA node captioning is the single dominant cost and AffordGraph removes it outright, the efficiency gain is decisive rather than marginal.

Table 4
End-to-end construction time per scene, directly measured and averaged over all 14 FunGraph3D scenes (NVIDIA A40, GPT-4o + LLaVA-1.6). Node detection is an identical shared upstream stage and is excluded from both columns. The dominant cost of OpenFunGraph is LLaVA node captioning, which AffordGraph replaces with CLIP-based affordance assignment; affordance filtering additionally reduces relationship inference by forwarding only 166 of 1,021 candidate pairs (an 83.7% reduction across the 14 scenes).

Stage (per scene)	OpenFunGraph	AffordGraph (Ours)
Node Description (LLaVA)	2080 s	—
Affordance Assignment (CLIP)	—	0.13 s
Relationship Inference	79 s	35 s
Total	2159 s (∼36 min)	35 s

5. Discussion

5.1. Effect of Context-Aware Reclassification

Context-aware reclassification accounts for a substantial part of AffordGraph's filtering quality, reducing false negatives and recovering node associations that CLIP-only assignment would miss. The benefit comes from disambiguating interactive elements whose semantic label is dominated by a single sense in CLIP's embedding space: a "knob," for instance, defaults to *open_close_supply*, yet many knobs in FunGraph3D instead control radiators or ovens (parameter adjustment) or sinks (water flow). By examining the affordance categories of spatially adjacent demand objects, the reclassification step reassigns such ambiguous elements to the correct supply category, allowing the corresponding supply–demand pairs to survive the filter. As the false-negative analysis below shows, knob ambiguity is the single largest source of filtering errors, which is precisely what this step targets.

5.2. Analysis of False Negative Patterns

Because the affordance filter operates over distinct semantic-label pairs (affordance roles are assigned per node label), we evaluate it on the 119 unique supply–demand label-pairs that carry at least one ground-truth functional relationship; the 228 individual annotations collapse to these 119 pairs once multiple same-label element–object instances within a scene are merged. Of these, the full AffordGraph pipeline produces 28 false negatives (a 23.5% false-negative rate), which provide important insights into the current limitations of the approach. Table 5 summarizes these cases by error pattern. We categorize them into four patterns:

Table 5
False negative cases in affordance-guided filtering on FunGraph3D, categorized by error pattern.

Error Pattern	Count	Percentage
Knob misclassified to wrong supply category	17	60.7%
Handle to non-door demand	5	17.9%
Button / switch misclassification	4	14.3%
Cross-category remote (power → device)	2	7.1%
Total	28	100%

Pattern 1: Knob Misclassification (17 cases, 60.7%). The label "knob" is strongly associated with *open_close_supply* in CLIP's embedding space due to the prevalence of door and cabinet knobs in training data. Knobs that instead control radiators or ovens (parameter adjustment) or sinks (water flow) are systematically assigned the wrong supply category and filtered out. This is by far the dominant error mode; the context-aware reclassification step discussed above recovers many such cases by leveraging

the proximity of the controlled demand object, but requires that object to lie within the 1.0m distance threshold and to itself be correctly classified.

Pattern 2: Handle-to-Non-Door Relationships (5 cases, 17.9%). Handles classified as *open_close_supply* are correctly retained for doors and cabinets, but handles that also connect to toilets, microwave ovens, coffee makers, or trashcans (*special_function* or *device_control* demands) fall outside the same-category rule and are filtered out. This reflects a genuine limitation of single-category supply assignment for multi-functional handles.

Pattern 3: Button/Switch Misclassification (4 cases, 14.3%). Buttons and switches are assigned a single role, but in practice trigger heterogeneous functions: a button that actually operates an oven or exhaust hood, or a switch mislabeled as a demand object. These mismatches cause the corresponding pairs to be pruned.

Pattern 4: Cross-Category Remote Relationships (2 cases, 7.1%). A small number of relationships span affordance categories, most commonly electrical outlets or power strips (*power_supply_supply*) connected to appliances. Our cross-category exception covers the common power → device case, leaving only a few residual cross-category pairs; extending the exception set could recover them.

5.3. Implications for AR agents

The efficiency gains reported in Section 4 are precisely what make functional scene graphs practical for AR agents. Reducing per-scene construction from roughly half an hour to tens of seconds (Table 4) brings the pipeline within the latency a user tolerates while a device maps an unfamiliar space, and the order-of-magnitude reduction in LLM/VLM calls directly lowers the energy and bandwidth cost of the dominant compute stage, a decisive factor for battery-powered glasses and for offloading to a constrained edge server. Equally important, replacing per-node VLM captioning with a single CLIP affordance-role embedding converts the costly stage into a lightweight text-encoding pass that is cheap enough to sit in the always-on perception loop of a device. The resulting functional graph is what AR agents need: it allows the system to identify the functional relations in the user's view. AffordGraph therefore contributes not only an efficiency technique but a deployable scene-understanding primitive for AR agents, narrowing the gap between functional 3D scene understanding and its use in real AR systems.

6. Conclusion

We presented AffordGraph, an efficient approach for constructing functional 3D scene graphs that targets the dominant computational bottleneck of OpenFunGraph. By replacing per-node VLM captioning with a CLIP affordance-role embedding and pruning 83.7% of candidate pairs before relationship inference, AffordGraph reduces directly measured per-scene construction time from $\sim$36 min to $\sim$35 s, an approximately $60\times$ speedup. This moves automatic functional scene graph construction out of a server-bound, offline regime and into the latency and power budget of AR agents, where devices must build and query such representations responsively, which we regard as the central contribution of this work. This speedup comes at a moderate, deliberate cost in recall.

Our core insight that the supply-demand structure of functional relationships provides a strong and computationally cheap prior for pruning irrelevant candidates is grounded in the observation that functional relationship types in indoor environments can be meaningfully categorized into a small set of physically motivated interaction categories. This insight, supported by prior work in functional scene understanding, enables a principled and generalizable filtering mechanism that does not require additional training data or fine-tuning.

Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning

& Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2026-25523396, Core Technology Development for Immersive Content) and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2019-II191270, WISE AR UI/UX Platform Development for Smartglasses).

Declaration on Generative AI

During the preparation of this work, the author used GPT-5 to verify grammar. The author reviewed and edited the content and assumes full responsibility for the content.

References

[1] C. Zhang, A. Delitzas, F. Wang, R. Zhang, X. Ji, M. Pollefeys, F. Engelmann, Open-vocabulary functional 3D scene graphs for real-world indoor spaces, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2025.

[2] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, et al., ConceptGraphs: Open-vocabulary 3D scene graphs for perception and planning, in: International Conference on Robotics and Automation, 2024.

[3] A. Delitzas, A. Takmaz, F. Tombari, R. Sumner, M. Pollefeys, F. Engelmann, SceneFun3D: Fine-grained functionality and affordance understanding in 3D scenes, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024.

[4] X. Hu, C. Zhang, A. Delitzas, X. Zhang, M. Pollefeys, F. Engelmann, X. Ji, Hierarchical and holistic open-vocabulary functional 3D scene graphs for indoor spaces, arXiv preprint arXiv:2605.15753 (2026).

[5] H. Liu, C. Li, Q. Wu, Y. J. Lee, Visual instruction tuning, in: Advances in Neural Information Processing Systems, 2024.

[6] J. J. Gibson, The senses considered as perceptual systems (1966).

[7] S. Deng, X. Xu, C. Wu, K. Chen, K. Jia, 3D AffordanceNet: A benchmark for visual object affordance understanding, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021.

[8] Y. Li, N. Zhao, J. Xiao, C. Feng, X. Wang, T.-s. Chua, LASO: Language-guided affordance segmentation on 3D object, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024.

[9] T. Nguyen, M. N. Vu, A. Vuong, D. Nguyen, T. Vo, N. Le, A. Nguyen, Open-vocabulary affordance detection in 3D point clouds, in: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023.

[10] C. Yu, H. Wang, Y. Shi, H. Luo, S. Yang, J. Yu, J. Wang, SeqAfford: Sequential 3D affordance reasoning via multimodal large language model, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2025.

[11] Y. Zhang, X. Huang, J. Ma, Z. Li, Z. Luo, Y. Xie, Y. Qin, T. Luo, Y. Li, S. Liu, et al., Recognize anything: A strong image tagging model, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024.

[12] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al., Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection, in: European Conference on Computer Vision, 2024.

[13] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al., Learning transferable visual models from natural language supervision, in: International Conference on Machine Learning, 2021.

[14] H. Chu, X. Deng, Q. Lv, X. Chen, Y. Li, J. Hao, L. Nie, 3D-AffordanceLLM: Harnessing large language models for open-vocabulary affordance detection in 3D worlds, in: International Conference on Learning Representations, 2025.

[15] L. He, M. Liu, Q. Ye, Y. Zhou, X. Deng, G. Ding, Task-aware 3D affordance segmentation via 2D guidance and geometric refinement, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2026.

[16] S. Koch, N. Vaskevicius, M. Colosi, P. Hermosilla, T. Ropinski, Open3DSG: Open-vocabulary 3D scene graphs from point clouds with queryable objects and open-set relationships, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024.

[17] S.-Y. Kim, A. Elskhawy, T. Ha, D. Kim, E. Shin, B. Busam, W. Woo, DeWorldSG: Depth-aware 3D semantic scene graph generation via world-model priors, arXiv preprint arXiv:2607.00889 (2026).

[18] S.-Y. Kim, D. Kim, W. Cho, H. Song, S. Kang, W. Woo, SceneLinker: Compositional 3D scene generation via semantic scene graph from RGB sequences, arXiv preprint arXiv:2602.02974 (2026).

[19] Y. Ju, Y. Liang, Y.-J. Wang, N. Gireesh, Y. Ju, S. Lee, Q. Gu, E. Hsieh, F. Huang, K. Sreenath, MomaGraph: State-aware unified scene graphs with vision-language model for embodied task planning, in: International Conference on Learning Representations, 2026.

[20] Z. Chen, G. H. Chen, S. Diao, X. Wan, B. Wang, On the difference of BERT-style and CLIP-style text encoders, in: Findings of the Association for Computational Linguistics: ACL 2023, 2023, pp. 13710–13721.

[21] A. Yan, J. Li, W. Zhu, Y. Lu, W. Y. Wang, J. McAuley, CLIP also understands text: Prompting CLIP for phrase understanding, arXiv preprint arXiv:2210.05836 (2022).