RoboBIM: Scaling Robot Semantic Foundations through Agentic Extraction of BIM Priors

Ashwin Nedungadi*, Bharathi Kannan Nithyanantham*, and Stefan Luedtke
Institute for Visual & Analytical Computing, Universität Rostock
{ashwin.nedungadi, bharathikannan.nithyanantham, stefan.luedtke}@uni-rostock.de

***Abstract*— Building Information Models (BIM) stored in the Industry Foundation Classes (IFC) format contain rich priors for indoor robot autonomy. However, most pipelines use static one-pass conversions that produce occupancy maps or simulation assets while leaving much of the semantic information unused. We present RoboBIM, an agentic pipeline that converts IFC files into metric-semantic navigation contracts. RoboBIM separates structural extraction from semantic enrichment. First, a deterministic backbone creates a reliable geometric baseline, including occupancy maps, waypoints, and room topology. Then, an LLM agent infers missing semantic information, such as room types, hazards, and traversability, through sandboxed code execution. We also introduce an online query agent that retrieves these priors during robot operation. We evaluate RoboBIM on four public IFC models. The results show that RoboBIM helps close the semantic gap by increasing room-type coverage by about 74.5% and filling previously empty semantic fields. The enriched graph natively exports to ROS 2 formats and natural-language summaries to ground scalable autonomy.**

I. Introduction

Mobile robots deployed in structured indoor environments often build metric-semantic maps using onboard perception pipelines, such as Kimera [1] and Hydra [2], even when high-fidelity digital twins already exist as design-time Industry Foundation Classes (IFC) models [3]–[5]. This repeated mapping process ignores valuable architectural priors and forces the robot to spend time rediscovering known structures. However, raw IFC files cannot be used directly as navigation maps. IFC is the standard data schema for building models, but its semantic information is often incomplete or not organized for robotic use. Although IFC models can provide precise geometry, they often lack the functional information needed for safe autonomy. For example, room functions may be missing, and floor material properties may be attached to structural elements, such as floor slabs, rather than to the navigable spaces themselves. Since these attributes are often omitted or distributed across different levels of the IFC hierarchy, deterministic parsers struggle to extract a unified, robot-ready prior.

The IFC semantic gap. Existing IFC-to-graph pipelines usually follow a fixed process: they extract `IfcSpace` entities, segment space with a generalized Voronoi diagram (GVD), add door waypoints, and export the result to ROS 2 formats [6]–[10]. These pipelines can reliably produce occupancy grids, room-transition graphs, and waypoint maps. However, their semantic output is limited by the information that is explicitly available in the IFC schema.

Our contribution. RoboBIM addresses this gap with a deterministic backbone and a staged LLM layer. The backbone first extracts the structural graph $\mathcal{G}_{\text{static}} = (\mathcal{O}, \mathcal{W}, \mathcal{T})$. Then, a seven-stage agentic layer enriches this graph with semantic fields that the deterministic backbone cannot infer. The pipeline produces a versioned navigation contract that integrates occupancy maps, waypoint graphs, topology, semantic annotations, and per-field confidence scores.

Fig. 1 shows the full RoboBIM vision. The top part shows the offline agentic IFC-to-prior compiler. The bottom part shows a future online dual-agent navigation loop. This paper focuses on the offline compiler. The online loop is discussed briefly in Sec. V and left for future work.

Concretely, we make the following contributions:

1) We introduce a three-layer navigation graph extracted from IFC in a top-down manner. The graph is designed for robot use and can be exported directly to Nav2, Open-RMF, GraphML, and GeoJSON.
2) We propose a staged LLM agent that tunes backbone hyperparameters within fixed ranges, infers missing semantic fields, and produces a deterministic natural-language briefing for an onboard robot LLM.
3) We provide a preliminary evaluation on four public IFC models. RoboBIM increases room-type coverage, fills hazard, material, traversability, and policy fields that are null in the baseline.

II. Related Work

Deterministic IFC-to-robot priors. Existing work has shown how IFC models can be converted into robot-usable geometric and topological priors. Hamieh et al. [3], [11] convert IFC models into weighted hypergraphs for cascaded A* planning. Follini et al. [4] define an IFC-to-costmap pipeline, while later work [5], [10], [12] automates waypoint extraction for construction robots. The IFC-graph literature [6]–[9] further formalizes how topological graphs can be generated from IFC. These methods provide reliable structural priors, but they have a shared limitation: they can only expose semantic information that is explicitly available in the IFC schema. RoboBIM builds on this deterministic line of work. Its main contribution is an agentic enrichment layer that adds missing robot-relevant semantics on top of the deterministic backbone.

*Equal Contribution.

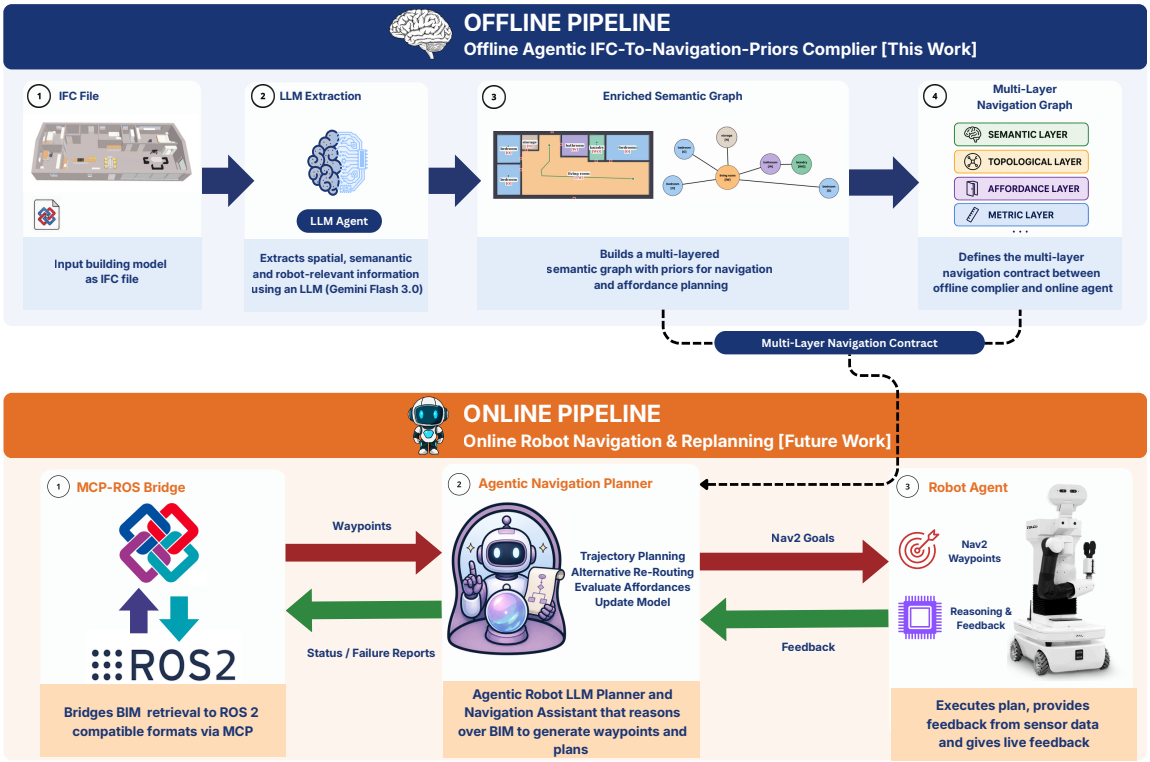


Fig. 1: RoboBIM system overview. **Top (this paper):** The IFC model is compiled into a multi-layer navigation contract using a staged LLM agent (Gemini 3.0 Flash). This stage combines a deterministic structural backbone with a semantic enrichment layer. **Bottom (future extension, Sec. V):** The navigation contract serves as input to an online agentic planner, which communicates with a ROS 2 robot agent through an MCP bridge for failure-driven replanning. This paper focuses on the offline agentic prior.

3D scene graphs and perception-based semantic priors. Recent systems such as SayPlan [13], EmbodiedRAG [14], ContextMatters [15], and Taskography [16] build hierarchical 3D scene graphs from onboard sensing and use LLMs for high-level task planning. Ray et al. [17] also define structured interfaces for scene-graph reasoning. These methods address a problem that is complementary to ours. They recover rich semantic information from perception when no prior map is available. RoboBIM addresses the reverse problem: it starts from an existing design-time IFC model and enriches it into a robot-ready semantic prior, without requiring perception at compile time. Pix2G [18] connects these directions by combining IFC with online scene-graph construction. RoboBIM targets the upstream step: making the IFC model itself semantically rich enough to support such downstream systems.

LLM-BIM integration for design. Recent work shows that LLMs can interact with IFC models. MCP4IFC [19], ifcMCP [20], modular MCP-BIM architectures [21], and Text2BIM [22] use LLMs for BIM querying, editing, or generative authoring. BIM2RDT [23] applies agentic reasoning to point-cloud-to-BIM alignment. Kim et al. [24] use LLMs for construction-robot task planning based on BIM data. These works demonstrate the value of LLMs for BIM reasoning, but they mainly focus on design, authoring, alignment, or downstream task planning. They do not provide a safety-constrained pipeline for compiling IFC into a robot navigation prior. RoboBIM addresses this gap by using LLMs to enrich deterministic IFC-to-robot priors while preserving the structural baseline.

LLMs for affordance- and capability-aware planning. Systems such as SayCan [25], AutoGPT+P [26], VAMOS [27], TIC-VLA [28], AINav [29], BrainBody-LLM [30], and IndoorR2X [31] ground LLM plans in robot affordances, capabilities, and adaptive replanning. These methods focus on how a robot should act once a semantic prior is available. RoboBIM addresses the earlier step: it creates the navigation contract that can serve as input to such planners.

Positioning. Table I summarizes this research landscape. RoboBIM differs from prior work in three main ways. First, it uses staged LLM inference instead of a single monolithic LLM call. Second, it includes an explicit topology regression gate to protect the deterministic structural baseline. Third, it produces a multi-consumer navigation contract rather than a single task-specific output format.

III. Methodology

The RoboBIM system has two main parts: a deterministic geometric backbone and an LLM agent wrapper, as shown in Fig. 2. The backbone takes one IFC file as input and produces the static graph $\mathcal{G}_{\text{static}} = (\mathcal{O}, \mathcal{W}, \mathcal{T})$. This graph contains per-storey occupancy grids $\mathcal{O}$, a clearance-aware waypoint graph $\mathcal{W}$, and a room-transition topology $\mathcal{T}$. The LLM agent wrapper runs seven stages on top of the same backbone. It inspects the IFC file through a sandboxed Python interpreter,

TABLE I: **Positioning.** RoboBIM focuses on agentic IFC-to-prior compilation. This direction is complementary to deterministic IFC pipelines and perception-based semantic-prior methods.

Capability	Det. IFC	3DSG + LLM	RoboBIM
IFC as prior source	✓	✗	✓
Rich semantic annotation	✗	✓	✓
No perception required	✓	✗	✓
Staged LLM with bounded roles	✗	partial	✓
Topology regression gate	✗	✗	✓
Multi-consumer contract	partial	✗	✓
Per-robot traversability	✗	partial	✓

and adds semantic fields Σ to the graph. These fields include room types, hazards, materials, traversability, and building-level policies. The final output is the enriched graph $\mathcal{G}_{\text{agent}} = (\mathcal{O}, \mathcal{W}, \mathcal{T}, \Sigma)$. RoboBIM exports this graph to Nav2, OpenRMF, GraphML, and GeoJSON. It also produces a deterministic natural-language briefing that can ground onboard robot LLMs. Appendix provides the stage specifications, prompts, clamp ranges, and confidence-threshold tables.

A. Deterministic backbone

The backbone is a five-module pipeline. First, the loader extracts storeys, doors, and `IfcSpace` entities from the IFC file. Second, the occupancy module rasterizes walls and columns into a 5 cm grid for each storey. Third, the segmentation module computes a Euclidean signed distance field, extracts a generalized Voronoi diagram (GVD) skeleton, and labels rooms using door-aware connected components. If no `IfcSpace` entities are available, it falls back to a purely geometric segmentation. Fourth, the waypoint module traces the skeleton into a sparse graph. It also injects door-crossing waypoints so that the graph follows the building's access topology. Finally, the topology module builds the room-transition graph, attaches material information from property sets, and checks cross-storey consistency. This check flags stair or elevator transitions whose endpoints do not align across floors. The resulting $\mathcal{G}_{\text{static}}$ is produced without LLM intervention and serves as the safety baseline.

B. Agentic enrichment layer

The agent wrapper adds seven stages. Each stage reads from and writes to a shared Pipeline Context. Stage 0 explores the IFC file through a sandboxed Python tool backed by `IfcOpenShell` [32]. The tool uses a hard timeout, blocks file writes, and disallows unsafe imports. Stage 1 produces extraction hyperparameters, including grid resolution, dilation radius, minimum room area, door search radius, and the include-openings flag for the deterministic backbone. These parameters are controlled by a policy in $\{\texttt{apply}, \texttt{suggest}, \texttt{fixed}\}$, with `fixed` as the default. Stage 2 infers room types. A regex and property-set prefilter first resolves deterministic cases. The remaining rooms on each storey are then sent to the LLM in a single batched prompt, which gives the model cross-room context. For example, an unlabelled space with multiple door transitions and adjacent office-labelled rooms can be inferred as a corridor. This type of regularity is difficult to encode with fixed rules but is easier to infer in context. Stage 2b reviews the segmentation and may issue merge directives. Merge directives with confidence ≥ 0.70 are applied, while split suggestions are recorded but not applied. Stage 3 validates connectivity and proposes add, remove, or reclassify edits. A topology regression gate then audits the connected-transition count and rolls back any edit that reduces it below the deterministic baseline. Stage 4 writes hazards (controlled vocabulary {wet area, narrow passage, fall risk, high traffic, obstacle dense, restricted access}), floor materials, and a traversability score in the range $[0, 1]$. Stage 5 produces a natural-language audit and briefing. This briefing is designed to be copied directly into the system prompt of an onboard robot LLM, so the robot does not need runtime access to the IFC file.

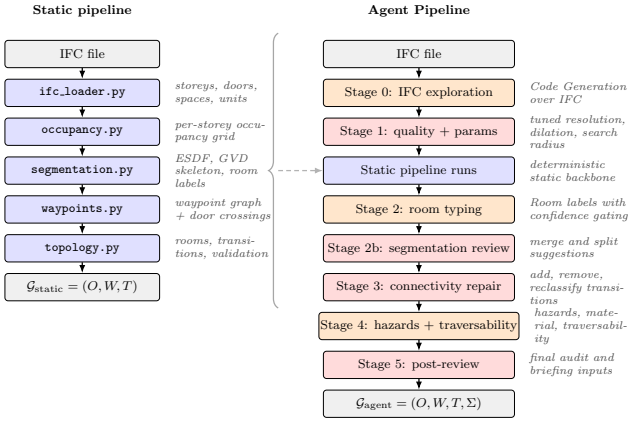


Fig. 2: Static pipeline (left) vs RoboBIM agent pipeline (right). The static pipeline is five deterministic Python modules that produce $\mathcal{G}_{\text{static}}$. The agent pipeline wraps the same backbone with seven LLM stages and a topology regression gate, producing $\mathcal{G}_{\text{agent}} = (\mathcal{O}, \mathcal{W}, \mathcal{T}, \Sigma)$.

C. Output graph

The enriched graph $\mathcal{G}_{\text{agent}}$ has three linked layers. Appendix Fig. 3 shows these layers on `BasicHouse`. Layer 1 is a per-storey occupancy grid with 5 cm resolution. Each cell is labelled as free, occupied, or unknown. Layer 2 is a per-storey waypoint graph derived from the GVD skeleton. Each edge stores a clearance value $c \in \mathbb{R}_{\geq 0}$, which determines whether a robot with inflated radius r can traverse the edge, i.e., $c \geq r$. Layer 3 is a topological graph of rooms and transitions, including doors, wall openings, and stair or elevator landings. The layers are connected through explicit cross-layer links. Each room is linked to its waypoint cluster, and each transition is linked to its crossing waypoint. These links support hierarchical planning, from a room-level topological plan to a metric trajectory. The deterministic baseline reserves per-room semantic fields but leaves them empty. These fields include the room type label with confidence and source, hazard list, floor-material string, and traversability score in $[0, 1]$. The agentic layer populates these fields. It also derives building-level policies, such as the tightest door-

width invariant and non-navigable storeys. These policies are attached to the graph rather than to a single room.

IV. Experiments

A. Setup

For the preliminary results, we evaluate RoboBIM on four public IFC models from IFC Bench [33]: `4351`, `BasicHouse`, `digital_hub`, and `samuel_macalister_sample_house`. These models cover residential, mixed-use, and office layouts, with 14-64 rooms across 2-6 storeys. `BasicHouse` contains no `IfcSpace` entities and is therefore used to test the geometric fallback. Appendix G reports the licenses, and raw element counts for all models. We run both the deterministic backbone and the full agent pipeline using Gemini 3.0 Flash on each building. Each reported value is based on a single run, and we discuss this limitation in Sec. V. We compare the two settings along three axes: the geometric graph, the populated semantic fields, and the wall-clock and token cost of the agent pass.

B. Results

Table II summarizes the results for each IFC model. Across the full corpus, room-type coverage increases from 51 to 89 out of 131 rooms (an increase of +74.5%). The deterministic baseline leaves hazard, traversability, material, and policy fields empty, while the agentic output populates these fields. On `BasicHouse`, which has no `IfcSpace` entities, the agent labels all 7 navigable rooms using only geometry and adjacency. It also infers hazards for kitchen, bathroom, and laundry areas. On `digital_hub`, the agent closes the 38-of-64 gap left by the regex prefilter and reaches full room-type coverage. No agent-proposed edit reduced the connected-transition count below the deterministic baseline in any of the four runs.

Agent wall-clock time ranges from 294 to 489 s per IFC model, with a mean of 403 s. Input token usage ranges from 200 k to 348 k, and output token usage ranges from 24 k to 41 k. Per-storey batching keeps the number of LLM calls between 7 and 15 per run. Stage 0 and Stage 2 account for most of the runtime. Stage 2b is the main cost on `BasicHouse`, where the geometric fallback is used. Stage 3 is the main cost on `digital_hub`, which has the densest connectivity graph. Appendix G reports the per-stage breakdowns in Fig. 5 and Tables V.

C. Qualitative analysis and online query agent (motivation)

A qualitative comparison on Floor 0 of `BasicHouse` is shown in Appendix Fig. 4. The static pipeline outputs grey rooms with only geometric indices. In contrast, the agent output colours rooms by inferred type, including `living_room`, `bedroom`, `bathroom`, `laundry`, and `storage`. It also shows hazard codes: W for wet area, O for obstacle-dense area, T for high-traffic area, and N for narrow passage. Listing 1 in the appendix shows a four-room excerpt from the generated briefing. The excerpt includes room type, hazards, material, traversability, and derived navigation policies, such as the tightest door width of 0.91 m, a robot width limit of 0.81 m, caution rooms, and non-navigable storeys. These fields are not populated by the deterministic baseline. The exports and briefing can be provided to a robot-side LLM through the Model Context Protocol [34]. For example, when the local planner reports a blocked door or occluded corridor, an onboard agent can query alternatives that satisfy width, hazard, and traversability constraints. This allows the robot to replan using the prior instead of re-exploring the full global state space. We leave the empirical evaluation of this online loop to future work.

V. Conclusion and Future Work

We presented RoboBIM, an IFC-to-navigation-graph pipeline that enriches a deterministic geometric backbone with robot-relevant semantics. On four public buildings, the agent increases room-type coverage by 74.5%, fills hazard, material, traversability, and policy fields that are empty in the baseline, and preserves connected topology in every run. The current study has three main limitations: the results are based on single-run measurements, the topology gate checks only connected-transition count, and the confidence thresholds are chosen heuristically. Future work will address these limitations in three directions. First, we will study the gap between as-designed and as-built environments by fusing perception with the prior. In this setting, $\mathcal{G}_{\text{agent}}$ can serve as a stable reference target for Hydra-class systems [2], with updates supported by DAAAM-style captions [35]. Second, we will stress-test the regression gate with synthetic bad-merge prompts. Third, we will evaluate the online query agent from Fig. 1 in end-to-end ROS 2 deployments.

References

[1] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: From SLAM to spatial perception with 3D dynamic scene graphs," in *International Journal of Robotics Research*, 2021.

[2] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," in *RSS*, 2024.

[3] A. Hamieh, A. A.-R. Moughlbay, B. Marche, and C. Francois, "Semantic navigation using building information on construction sites," *arXiv preprint arXiv:2104.10296*, 2021.

[4] C. Follini, M. Terzer, C. Marcher, A. Giusti, and D. T. Matt, "BIM-integrated collaborative robotics for application in building construction," in *Proceedings of the International Symposium on Automation and Robotics in Construction*, 2020.

[5] K. Kim and M. Peavy, "BIM-based semantic building world modeling for robot task planning," *Automation in Construction*, vol. 138, 2022.

[6] T.-A. Teo and K.-H. Cho, "i-GIT: Intelligent generation of indoor topology from IFC models," *ScienceDirect*, 2018.

[7] Y.-H. Lin *et al.*, "Generating straight skeleton-based navigation networks with IFC," *ScienceDirect*, 2020.

[8] J. Zhu *et al.*, "A new graph-based approach for building information querying," in *CIB W78 Conference*, 2024.

[9] "Semantics-based connectivity graph for indoor pathfinding powered by IFC-Graph," *ScienceDirect*, 2025.

[10] Construction Robotics Authors, "Improving autonomous robotic navigation using IFC files," *Construction Robotics*, 2023.

[11] A. Hamieh, A. A.-R. Moughlbay, and B. Marche, "Intuitive BIM-aided robotic navigation and assets localization," *Frontiers in Robotics and AI*, 2025.

[12] S. Kim, H. A. Pham, J. Jang, and G. Lee, "Development of BIM-integrated construction robot task planning system," *Automation in Construction*, 2021.

TABLE II: Per-building results on four IFC models. The Geometry columns are identical for both pipelines (the gate forbids regression) and listed once. Semantic enrichment columns are written static → agent.

	Geometry (preserved)				Semantic enrichment (static → agent)					Agent cost	
Model	storeys	rooms	conn. trans.	waypoints	type	w/ hazard	traversability	material	policy	time (s)	tok. in/out
4351	5	22	10	90	$1 \rightarrow 1$	$0 \rightarrow 3$	$0 \rightarrow 22$	$0 \rightarrow 5$	$0 \rightarrow 5$	294	285k/32k
BasicHouse	2	14	6	32	$0 \rightarrow 7$	$0 \rightarrow 7$	$0 \rightarrow 14$	$0 \rightarrow 7$	$0 \rightarrow 6$	489	200k/24k
digital_hub	3	64	60	304	$38 \rightarrow 64$	$0 \rightarrow 0$	$0 \rightarrow 64$	$0 \rightarrow 0$	$0 \rightarrow 1$	423	343k/32k
macalister	6	31	9	392	$12 \rightarrow 17$	$0 \rightarrow 9$	$0 \rightarrow 31$	$0 \rightarrow 17$	$0 \rightarrow 6$	407	348k/41k
total	**16**	**131**	**85**	**818**	$\mathbf{51 \rightarrow 89}$	$\mathbf{0 \rightarrow 19}$	$\mathbf{0 \rightarrow 131}$	$\mathbf{0 \rightarrow 29}$	$\mathbf{0 \rightarrow 18}$	1612	1176k/129k

[13] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "SayPlan: Grounding large language models using 3D scene graphs for scalable robot task planning," in *CoRL*, PMLR 229, 2023.

[14] M. Booker, C. Agia, P. Sermanet, and A. Z. Ren, "EmbodiedRAG: Dynamic 3D scene graph retrieval for efficient robot task planning," *arXiv preprint arXiv:2410.23968*, 2024.

[15] "ContextMatters: Relaxing goals with LLMs for feasible 3D scene planning," *arXiv*, 2026.

[16] C. Agia *et al.*, "Taskography: Evaluating robot task planning over large 3D scene graphs," in *CoRL*, PMLR, 2022.

[17] A. Ray *et al.*, "Structured interfaces for automated reasoning with 3D scene graphs," *arXiv preprint arXiv:2510.16643*, 2025.

[18] A. Longo *et al.*, "Pixels-to-graph: Real-time integration of BIM and scene graphs," *arXiv preprint arXiv:2506.22593*, 2025.

[19] B. K. Nithyanantham *et al.*, "MCP4IFC: IFC-based building design using LLMs," *arXiv preprint arXiv:2511.05533*, 2025.

[20] J.-R. Lin and P. Pan, "ifcMCP: Enabling LLM agents to talk with IFC files," 2025.

[21] "A modular reference architecture for MCP-servers for agentic BIM interaction," *arXiv preprint arXiv:2601.00809*, 2025.

[22] C. Du *et al.*, "Text2BIM: Generating building models using a LLM-based multi-agent framework," *arXiv preprint arXiv:2408.08054*, 2024.

[23] "BIM2RDT: Building information models to robot-ready site digital twins," *arXiv preprint arXiv:2509.20705*, 2025.

[24] K. Kim *et al.*, "Framework for LLM-enabled construction robot task planning," *Robotics*, vol. 14, no. 9, p. 117, 2025.

[25] M. Ahn, A. Brohan, N. Brown, *et al.*, "Do as I can, not as I say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[26] T. Birr *et al.*, "AutoGPT+P: Affordance-based task planning using LLMs," in *RSS*, 2024.

[27] "VAMOS: A hierarchical VLA model for capability-modulated navigation," *arXiv preprint arXiv:2510.20818*, 2025.

[28] Z. Huang *et al.*, "TIC-VLA: A think-in-control VLA model for robot navigation," *arXiv preprint arXiv:2602.02459*, 2026.

[29] "AINav: Interactive navigation via LLM-based skill tree and adaptive replanning," *arXiv preprint arXiv:2503.22942*, 2025.

[30] "BrainBody-LLM: Grounding LLMs for robot task planning with closed-loop feedback," *Advanced Robotics Research*, 2025.

[31] "IndoorR2X: Indoor robot-to-everything coordination with LLM-driven planning," *arXiv preprint arXiv:2603.20182*, 2026.

[32] IfcOpenShell Contributors, "IfcOpenShell: An open-source toolkit for IFC model parsing and manipulation." `https://ifcopenshell.org`, 2024.

[33] S. Hellin, S. Nousias, and A. Borrmann, "Natural Language Information Retrieval from BIM Models: An LLM-Based Agentic Workflow Approach," in *EC3*, 2025.

[34] Anthropic, "Model context protocol (MCP) specification." `https://modelcontextprotocol.io`, 2024.

[35] N. Gorlo *et al.*, "DAAAM: Dense annotated affordance maps for robot navigation via vision-language models," *arXiv preprint*, 2025.

APPENDIX

This appendix expands the seven agent stages introduced in Section III. For each stage we describe the inputs it reads, the output it produces, and the conditions under which its proposals are written back into the graph.

A. Stage 0: building exploration

The agent receives a compact summary of the input building file that lists entity counts, storeys, doors, and representative property sets. It is also given a sandboxed Python runner with read-only access to the loaded building model through IfcOpenShell [32]. The runner enforces a hard timeout and blocks imports of non-standard libraries and file writes. The agent responds with free-form findings, which are parsed into evidence items and stored in the Pipeline Context.

B. Stage 1: quality analysis and parameter tuning

The agent reads the building summary together with the Stage 0 findings. It returns a structured object with a completeness score, a recommended strategy, and five numeric parameters: grid resolution, dilation radius, minimum room area, door search radius, and an include-openings flag. All numeric parameters are clamped to safe ranges. A parameter policy controls the write-back behaviour. In the apply mode, the parameters are used directly. In the suggest mode, they are used only if they lie inside the clamp range. In the fixed mode, which is the default, they are logged but not used.

C. Stage 2: room-type inference

A regex prefilter types rooms with obvious names, such as corridor, CX, or WC. An element-set prefilter handles rooms with matching content, a room with two or more sanitary terminals is treated as a bathroom. Rooms resolved by either prefilter skip the LLM. The remaining rooms on a given storey are sent to the LLM in a single batched prompt. A newly assigned type requires a confidence of at least 0.80, and overriding an existing type requires a confidence of at least 0.75.

D. Stage 2b: segmentation review

The agent reads the room list and the transitions, then proposes merge or split directives. Merges with a confidence of at least 0.70 are applied at the topology layer: the bounding boxes are combined, the transitions are reassigned, and the absorbed room is removed. Split directives are recorded only, because they require label-map access that is not preserved after extraction.

E. Stage 3: connectivity validation and repair

The agent produces a list of issues, such as isolated rooms or suspicious connections. It then outputs a repair plan with three directive kinds: `add_connection`,

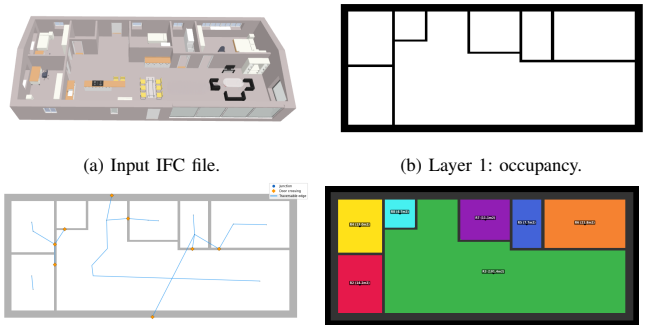
remove_connection, and reclassify. Directives with a confidence of at least 0.70 are applied. The topology regression gate then checks the resulting connected-transition count, and any edit that reduces it is rolled back.

F. Stage 4: hazards and traversability

For each room, the agent writes a list of hazard flags from a controlled vocabulary, a floor-material string, a traversability score in $[0, 1]$, a short robot-obstacles string, and a prose note. The stage reuses the prefilter and the per-storey batching introduced in Stage 2.

G. Stage 5: post-extraction review

The agent reads the final enriched graph and returns a short audit that contains an overall quality score, a completeness score, a gaps list, and improvement suggestions. The audit is stored in the Pipeline Context and is used as input to the natural-language briefing.



(a) Input IFC file. (b) Layer 1: occupancy. (c) Layer 2: waypoints. (d) Layer 3: rooms.

Fig. 3: Input IFC file and the three linked layers on BasicHouse, ground floor. Top-left: input IFC rendering. Top-right: Layer 1, 5 cm occupancy grid. Bottom-left: Layer 2, GVD-derived waypoint graph with clearance-annotated edges. Bottom-right: Layer 3, room-and-transition topology. Cross-layer links (not drawn) connect each room to its waypoint cluster and each door to its crossing waypoint.

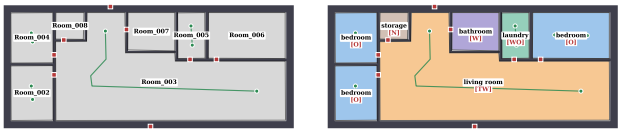


Fig. 4: Floor 0 in BasicHouse. Left: static output. Right: agent output. Rooms coloured by inferred type with hazard codes in brackets (W=wet, O=obstacle-dense, T=high traffic, N=narrow).

The four building models used in the main evaluation are all public and redistributable, and each is taken directly from its original repository. 4351[1] and samuel_macalister_sample_house[2] come from the BIMserver TestFiles collection. digital_hub[3] comes from the RWTH E3D DigitalHub release. BasicHouse[4] comes from the BIM-Whale samples repository. Together, these sources span several design disciplines and both major revisions of the IFC file format.

Table III lists the upstream origin, the license, and the intended building use for each model. Table IV lists raw statistics extracted from each file: the schema version, the on-disk file size, and the total counts of storeys, doors, rooms, and walls. These counts are independent of our pipeline and reflect the authoring state of each file.

TABLE III: Provenance of each building model in the corpus.

Model	Upstream source	License	Use
4351	BIMserver TestFiles	GPL v3	office
BasicHouse	BIM-Whale Samples	MIT	residential
digital_hub	RWTH E3D DigitalHub	MIT	mixed-use
macalister	BIMserver TestFiles	GPL v3	residential

TABLE IV: Raw building-file statistics extracted from each model. The rooms column shows how many rooms are explicitly annotated by the author; a zero means the pipeline must recover rooms geometrically.

Model	Schema	Size (MB)	Storeys	Doors	Rooms	Walls
4351	IFC2X3	7.3	5	17	6	43
BasicHouse	IFC2X3	51.7	2	8	0	13
digital_hub	IFC4	9.0	3	64	64	178
macalister	IFC4	25.4	6	16	14	48

digital_hub is the densest building in the corpus, with 64 annotated rooms and 64 doors on three storeys, and tests the pipeline on a richly annotated model. 4351 is a 5-storey office with only 6 rooms annotated, so the agent must infer room types on a partially annotated file. BasicHouse has no room annotations at all, so segmentation falls through to the geometric GVD pass described in Section III; it is also the largest file on disk, because it carries detailed geometry for a small floor plan. samuel_macalister_sample_house is a 6-storey residential model with mechanical and plumbing content, and stresses the cross-storey consistency check.

Fig. 5 stacks the seven stage timings per building and overlays the input-token count on a secondary axis. Stage 0 is always the first LLM stage; its time scales with the complexity of the input file, because the agent controls how many code-execution snippets to issue. Stage 2 (room-type inference) scales with the storey count, because it is batched per storey. Stage 2b (segmentation review) is the largest single cost on BasicHouse, where the lack of annotated rooms forces a richer geometric review. Stage 3 is the largest cost on digital_hub, which has the densest connectivity graph in the corpus.

Table V reports the LLM usage per building. Both the call count and the token usage are dominated by Stage 2 and Stage 4 on the smaller buildings. On BasicHouse, Stage 0 dominates, because the agent chose to issue more exploration snippets than on the other three buildings.

This appendix collects additional per-storey visualisations produced by the agent pipeline on two of the four buildings. Each panel is a composite view written by the pipeline that

[1] https://github.com/opensourceBIM/TestFiles
[2] https://github.com/opensourceBIM/TestFiles
[3] https://github.com/RWTH-E3D/DigitalHub
[4] https://github.com/andrewisen/bim-whale-ifc-samples/tree/master/BasicHouse

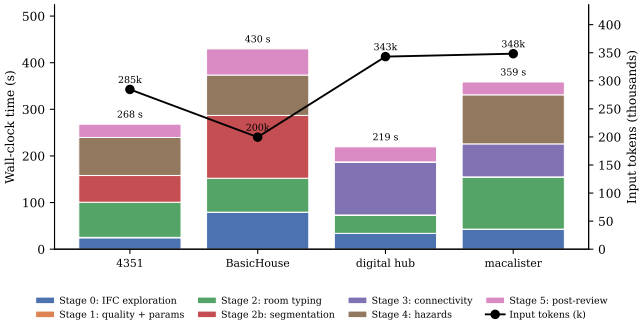


Fig. 5: Per-IFC agent wall-clock time stacked by stage (bars, left axis), with the input-token count overlaid as a black line (right axis). Stage 0, Stage 2, Stage 2b, and Stage 4 dominate the time budget. Confidence-gated per-storey batching keeps the total LLM call count between 7 and 15 per run.

TABLE V: LLM usage per building.

Model	LLM calls	Code execs	Tokens in (k)	Tokens out (k)
4351	13	28	285	32
BasicHouse	9	15	200	24
digital_hub	7	16	343	32
macalister	15	27	348	41
total	44	86	1176	129

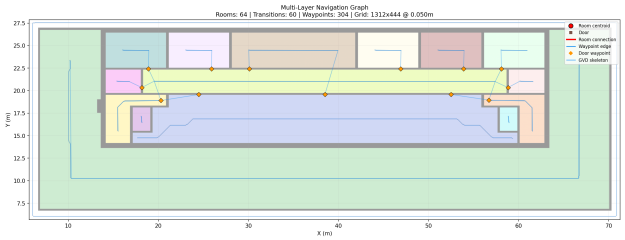
Listing 1: Short summary of the BasicHouse briefing derived from the agent trace. The full agent trace is larger and includes stage-by-stage details.

```
Room_003: living_room, concrete, trav=0.80
  hazards: [high_traffic, wet_area]
  notes:   circulation hub, 8 door transitions

Room_005: laundry, concrete, trav=0.50
  hazards: [wet_area, obstacle_dense]
  notes:   narrow channels around appliances

Room_007: bathroom, concrete, trav=0.60
  hazards: [wet_area]
  notes:   constrained layout, plumbing fixtures

Room_008: storage, concrete, trav=0.90
  hazards: [narrow_passage]
  notes:   small walk-in closet

derived_policies:
  tightest_door_m:       0.91
  width_limit_m:         0.81
  caution_rooms:         [Room_003, Room_005, Room_007]
  non_navigable_storeys: [Floor_1]
```

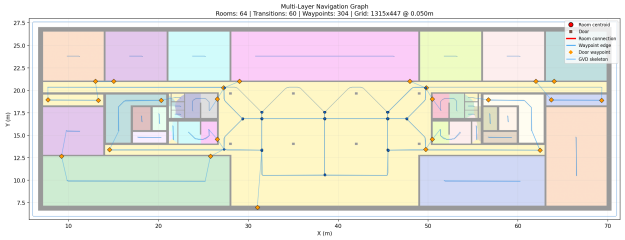
overlays the occupancy grid, the waypoint graph, and the room segmentation on a single storey.

Fig. 6 shows the three storeys of `digital_hub`. The building has annotated rooms on every floor, so the segmentation follows the authored partitions and the waypoint graph densifies cleanly around the shared corridor on each level.

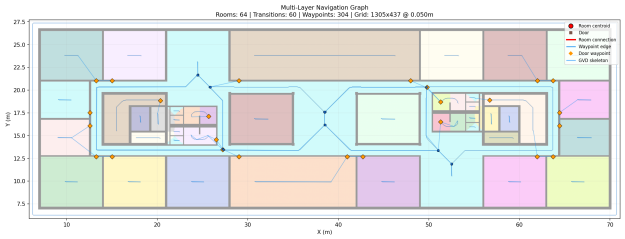
Fig. 7 shows two of the six storeys in `samuel_macalister_sample_house`. The dwelling has a residential layout that varies across floors, and the waypoint graph contracts around the stair and landing regions on each level.



(a) Basement (B01_OKRD).

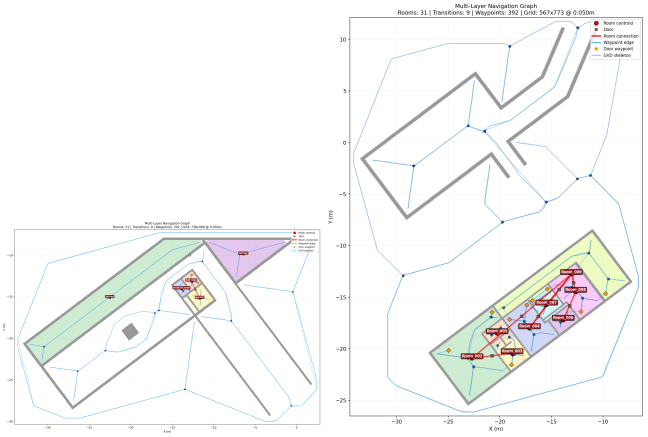


(b) Ground floor (E00_OKRD).



(c) First floor (E01_OKRD).

Fig. 6: Per-storey composite for `digital_hub`. Each panel overlays the occupancy grid (grey), the waypoint graph (blue nodes and edges), and the room segmentation (coloured regions) on one storey of the building.



(a) Level 1. (b) Level 2.

Fig. 7: Composite view on two storeys of `samuel_macalister_sample_house`. The pipeline produces a consistent waypoint graph across levels, with room boundaries drawn from the authored room partitions where available.