applied sciences

MDPI

Article

Concept-Guided Exploration: Building Persistent, Actionable Scene Graphs

Noé José Zapata Cornejo *, Gerardo Pérez, Alejandro Torrejón, Pedro Núñez and Pablo Bustos

RoboLab, Robotics and Artificial Vision, University of Extremadura, 10003 Cáceres, Spain; gperezgonz@unex.es (G.P.); atorrejon@unex.es (A.T.); pnuntru@unex.es (P.N.); pbustos@unex.es (P.B.)
* Correspondence: nzapata@unex.es

Abstract

The perception of 3D space by mobile robots is rapidly moving from flat metric grid representations to hybrid metric-semantic graphs built from human-interpretable concepts. While most approaches first build metric maps and then add semantic layers, we explore an alternative, concept-first architecture in which spatial understanding emerges from asynchronous concept agents that directly instantiate and manage semantic entities. Our robot employs two spatial concepts—room and door—implemented as autonomous processes within a cognitive distributed architecture. These concept agents cooperatively build a shared scene graph representation of indoor layouts through active exploration and incremental validation. The key architectural principle is hierarchical constraint propagation: Room instantiation provides geometric and semantic priors to guide and support door detection within wall boundaries. The resulting structure is maintained by a complementary functional principle based on prediction-matching loops. This approach is designed to yield an actionable, human-interpretable spatial representation without relying on any pre-existing global metric map, supporting scalable operation and persistent, task-relevant understanding in structured indoor environments.

Keywords: 3D perception; semantic modelling; scene graphs; robot perception

Academic Editors: Alessandro Umbrico, Daniel Beßler and Nele Russwinkel

Received: 16 September 2025
Revised: 10 October 2025
Accepted: 11 October 2025
Published: 16 October 2025

Citation: Zapata Cornejo, N.J.; Pérez, G.; Torrejón, A.; Núñez, P.; Bustos, P. Concept-Guided Exploration: Building Persistent and Actionable Scene Graphs. *Appl. Sci.* **2025**, *15*, 11084. https://doi.org/10.3390/app152011084

1. Introduction

Robots operating in human environments benefit from semantically rich, shared representations of their surroundings. In recent years, 3D scene graphs (3DSGs) have been used for this purpose, as complex data structures that represent scene objects as nodes and their various relationships as edges [1]. When used to describe large fragments of space, they are typically organised hierarchically, with higher levels encompassing larger spatial aggregations [2]. The computer vision community has developed many algorithms and DNN models that can be used to detect elements in the scene and their relationships, and even to build the graph directly from complete views of the scene [3–5]. Very complex graphs can also be built in real time using data captured by a human-operated robot, as demonstrated in recent frameworks [6,7]. In most experiments, robots are typically used passively, in a remote-controlled setup, to capture 3D data in large environments.

However, if robots are to use 3DSGs in real-time operations, they must build them incrementally, starting from an empty representation when necessary, and proposing actions that improve the current representation by reducing its uncertainty. These actions must coincide with or compete with other requests from other high-priority tasks, forcing the representational system to be opportunistic at its core. Scene graph construction should

be driven by the robot's need to understand its world and current task requirements. This opportunistic process may leave objects partially defined for later refinement, requiring the architecture to continuously update its world beliefs. This graph-building loop underlies the control architecture, adapting to ongoing tasks while competing for hardware access.

In this paper, we introduce a concept-first architecture where autonomous agents, each representing a spatial concept like room or door, cooperatively and asynchronously build a 3DSG. This approach diverges from the classical metric-first paradigm by allowing the robot to actively choose its next action to refine or expand its representation, building spatial understanding directly from human-interpretable concepts. We model the world as a set of rectangular rooms connected by doors. The robot is given a functional definition of the room and door concepts, and the goal is to incrementally build a scene graph representation of the environment by instantiating these concepts and relating their instances geometrically and semantically. The graph is to be constructed directly from these concepts, avoiding a previous free/occupied grid representation. The robot has a 3D LiDAR that provides a stream of points from the scene. Finally, we use the CORTEX architecture [8,9] as a robust, expandable framework for all algorithms and data structures. Figure 1 depicts a schematic overview.

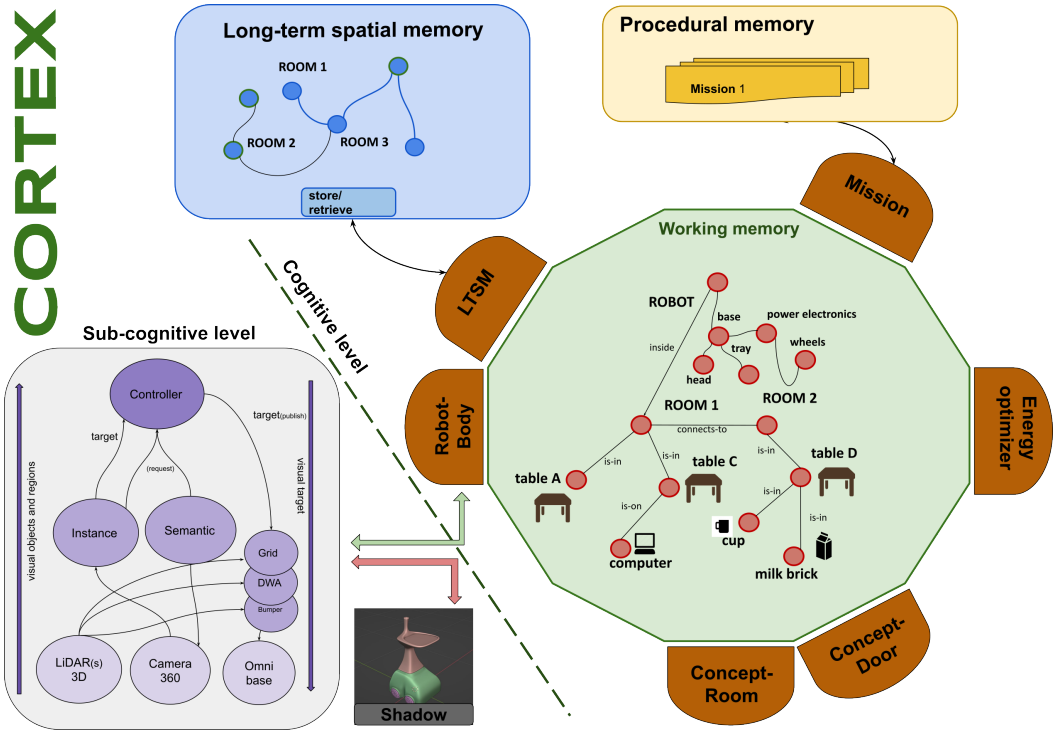


Figure 1. An overview of the CORTEX architecture with the elements used in this work. The sub-cognitive level on the left encompasses low-level perceptive and control components. The working and long-term spatial memories are managed by the six agents depicted as brown shapes.

Our primary contribution is a novel system of asynchronous concept agents. Each agent sustains the life cycle of its specific concept, actively creating, managing, and refining instances to explain and predict real-world objects. Each concept agent in the architecture behaves opportunistically, contributing to the construction of a global, shared representation. In turn, each concept agent accesses this shared representation to acquire contextual information that constrains its creation and update processes according to the hierarchical constraint-propagation principle, in which higher-level concepts guide the detection of lower-level concepts. Our second contribution is a world model composed of connected, local metric frames rather than a single global one. This work is primarily an architectural

exploration that demonstrates how this organisation can build persistent, actionable representations directly from concepts. Our third contribution is to develop an incremental, scene-graph building system designed for real-time operation in a mobile robot's control architecture as an underlying representational dynamics adaptable to other concurrent tasks. Jointly, the concept processes and the other elements in the architecture collaborate to build and maintain a reliable scene graph that different tasks can use.

2. Related Works

2.1. From Geometric SLAM to Semantic and Hybrid SLAM

Classical geometric SLAM has been extensively studied and remains a cornerstone for robot localisation and mapping; however, its lack of high-level semantic content limits human–robot interaction, long-term robustness, and task-oriented reasoning [10,11]. To address these shortcomings, numerous works have proposed augmenting metric maps with semantic labels and object-centric landmarks, a family of methods commonly referred to as semantic or metric–semantic SLAM [12–14]. Other approaches tackle persistent mapping by emphasising the need for robust semantic reasoning in dynamic environments [15,16]. Similarly, CNN-based methods such as SemanticFusion [14] and Mask R-CNN extensions (e.g., PanopticFusion) [17,18] have made dense semantic mapping feasible in (near) real-time, bridging perception and mapping pipelines [19]. Surveys and overviews further summarise trends and techniques in this area, covering pipelines that combine detection, data association, and pose/landmark estimation [20,21]. Open-source libraries such as Kimera demonstrate the practical integration of visual–inertial SLAM, dense reconstruction, and semantic labelling to produce metric–semantic maps in real-time [22]. Although existing methods successfully augment metric maps with object-level semantics, they follow a metric-first paradigm. Our work departs from this by allowing the 3DSG to integrate data based on high-level concepts, which form the primary organisational structure of the representation.

2.2. 3D Scene Graphs and Dynamic Scene Graphs

The idea of representing a richer semantic and topological structure as a graph grounded in 3D geometry was crystallised by Armeni et al., who introduced the 3D scene graph (3DSG) as a unified hierarchical structure linking floors, rooms, objects, and cameras in reconstructed buildings [1]. Subsequent works have extended and operationalised this representation for robotics. Rosinol et al. introduced 3D Dynamic Scene Graphs (DSGs), emphasising actionable perception by adding temporal and agent-centric layers and presenting an automated spatial-perception pipeline to produce DSGs from visual—inertial data [22]. Hydra and related spatial-perception systems build on these ideas and demonstrate real-time 3DSG construction at scale, integrating segmentation, topological reasoning, and loop closure into an online pipeline [2,7]. SceneGraphFusion proposed an incremental, learning-based strategy to predict 3DSG from RGB–D sequences using graph neural networks and attention mechanisms suited to partial observations [3]. In parallel, Bavle et al. explored situational graphs as a lightweight, navigation-focused variant of DSGs, highlighting their role in bridging semantic scene understanding and actionable planning [23]. Most state-of-the-art 3DSG systems, despite their rich hierarchical structures, ground their representation in a global metric frame built from a dense reconstruction. Our architectural contribution is a fundamental departure from this approach, as we construct the metric frame based on instantiated spatial concepts themselves.

2.3. Extracting High-Level Spatial Structure

Within scene graphs, a first level of spatial structuring is provided by the layout of interior buildings, which condense the basic geometric structure of walls, floors, and ceilings. Their integration into a richer topological and semantic representation enables long-term reasoning and planning. More traditional approaches resort to the use of recognised geometric algorithms like RANSAC on the point cloud generated by SLAM to identify walls [24–26], similar to [27], where the Manhattan world assumption is added. Other works formulate layout reconstruction as an optimisation problem [28], resorting to mesh-based normalisation of input point cloud models and then using a constrained optimiser. Their main advantage lies in their independence from large data sets and their geometric accuracy in well-structured environments. However, they are sensitive to hyperparameter tuning, occlusions, orthogonality assumptions, and sensor noise.

A distinct line of research leverages deep neural networks to infer complete room layouts from single panoramic images, with notable examples including LayoutNet [29], FloorNet [30], and HorizonNet [31]. These data-driven models exhibit strong generalisation capabilities and can produce geometrically coherent layouts at a low inference cost. However, from a robotics perspective, their utility is constrained by two primary factors. First, they are dependent on large-scale annotated datasets and often presuppose regular, structured environments. More critically, they operate as 'black-box' systems performing a single-shot inference. This monolithic approach offers no mechanism for incremental refinement or reasoning about partial evidence. Consequently, these models struggle with the ambiguity inherent in a robot's typical starting condition—often a position with limited visibility—a scenario where our concept-driven, hypothesis-validation loop is specifically designed to address. While the choice of a specific detection algorithm can significantly impact performance, our work's primary contribution lies in demonstrating a holistic, functional system. We, therefore, utilized straightforward detection methods to allow the core principles of our architecture to be evaluated independently of their specific perceptual components.

2.4. Incremental, Opportunistic, and Active Perception

While many 3D scene graph (3DSG) systems are validated on offline datasets, the robotics community increasingly demands representations that are built incrementally and refined through active perception for online operation. The principle of closing the planning–perception loop is well established in Active-SLAM, where robot actions are chosen to reduce map uncertainty or gather task-relevant data [32,33]. This is complemented by information-theoretic approaches that provide principled methods for selecting optimal sensing actions under uncertainty [34,35].

More recent work on situational graphs frames scene understanding as the online construction of optimizable, multi-layer representations (metric, topological, and semantic) for navigation [23,36–38]. Collectively, these research efforts motivate architectures where perception is not a passive aggregator of data, but an active process that deliberately seeks information to validate or refute semantic hypotheses—such as the existence of a room or the precise location of a door [33,39].

Our architecture embodies these principles by integrating affordances as context-sensitive controllers within each concept agent. These affordances are pre-activated when their operational preconditions are met. A central *mission_monitoring* agent executes tasks by selectively activating affordances that guide the robot toward its goal, interleaving these actions with others dedicated to constructing and maintaining the internal scene representation. This mechanism provides the foundation for the incremental, opportunistic, and active perception at the core of our system.

2.5. Open-Vocabulary and Language-Grounded Scene Graphs

Recent advances in vision-language models (VLMs) have catalysed a new generation of open-vocabulary 3D Scene Graphs (3DSGs), enabling robots to perceive and reason about an open set of objects and relationships. The dominant pipeline for these systems typically begins with class-agnostic segmentation of RGB-D frames, followed by the extraction and projection of semantic feature vectors into a common 3D frame. The resulting semantically-enriched point cloud is then processed, often with queries to a large language model (LLM), to establish object identities and inter-object relationships.

While these methods demonstrate powerful capabilities for language-grounded querying, our concept-first architecture diverges from this paradigm in several fundamental ways, centred on the principles of incremental construction and the explicit modelling of environmental structure.

A primary distinction is our commitment to incremental, online construction without prior global maps. Systems like HOV-SG [40], for instance, require a complete metric map of the environment as a prerequisite for extracting floors, rooms, and objects. Navigation is then enabled by a permanent Voronoi graph of the total free space. In contrast, our approach builds its understanding dynamically as the robot explores. Navigation is handled locally on demand within the current room's reference frame, while inter-room pathfinding is a simple graph search in the long-term spatial memory.

Furthermore, our architecture prioritises the explicit modelling of spatial structure and connectivity, whereas many open-vocabulary systems are fundamentally object-centric. Works such as Open3DSG [41] and OVSG [42] excel at establishing rich relationships among objects but do not acquire the containing room or the topological connectivity between rooms as first-class entities in the graph. Our work, conversely, treats the "room" concept as the primary semantic and geometric anchor, with the environment's structure serving as the backbone of the scene graph.

This focus enables our core principle of hierarchical constraint propagation, which is architecturally distinct from the bottom-up process common to many VLM-based pipelines. ConceptGraphs [43], for example, incrementally adds objects but must wait until an entire sequence has been processed to generate object captions and relationships. Spatial relations are first estimated by proximity and then refined by querying an LLM, a process that prevents high-level concepts (such as a room's walls) from actively constraining and guiding lower-level perception (such as finding a door). Our top-down approach is designed around this very principle of semantic constraint.

Finally, while the field is advancing towards capturing object affordances and functional relationships, as pioneered by OpenFunGraph [44], our work provides the essential spatial-semantic scaffolding for such reasoning. OpenFunGraph masterfully models interactions between objects and their parts, but does not construct a representation of the surrounding spaces or validate that the graph remains anchored to the world as the robot moves. Our architecture is expressly designed to build and maintain this persistent spatial representation, providing the stable, grounded context in which functional and task-level reasoning can reliably occur.

2.6. Positioning of the Present Work

Most existing 3D scene graph construction approaches follow a metric-first paradigm: They build dense occupancy maps or point clouds, then layer semantic information on top through object detection and classification [3,6,7]. While effective, this approach conceptualises space primarily in terms of free/occupied regions—representations that enable collision-free navigation but provide limited support for high-level reasoning, manipulation planning, or human–robot communication. We explore an alternative concept-first

architecture in which spatial understanding emerges directly from human-interpretable concepts, without requiring prior metric representations. Our approach makes four key departures from existing work. (i) Asynchronous concept agents, rather than sequential metric-then-semantic processing, involves implementing concept classes (room and door) as autonomous processes that run asynchronously within the CORTEX cognitive architecture. Each concept agent manages the complete lifecycle of its instances—from initial detection through validation, maintenance, and execution of affordances—competing opportunistically for robot resources. (ii) Hierarchical constraint propagation is where room instantiation provides geometric and semantic constraints that significantly improve door detection efficiency and robustness. Unlike unconstrained analysis of occupancy grids, doors are searched within the constrained space of validated room walls, leveraging spatial relationships inherent in indoor environments. (iii) World model of local metric frames is where the global representation, managed by long-term spatial memory (LTSM), is a topological graph connecting local, concept-centric metric frames. The robot localises itself relative to the current room's frame, not a global world frame. This architectural pattern is a key distinction from systems like Hydra, which, despite their real-time performance, still ground a hierarchical scene graph in a global metric frame. Our contribution lies in exploring this alternative world model structure, which may offer benefits in scalability and human interpretability. (iv) Architectural vs. algorithmic contribution conveys that this work should be understood as an architectural exploration rather than an algorithmic one. We deliberately use simple, well-understood perception methods (e.g., Hough transforms) to focus attention on the system's organisational principles. The core claim is not that our perception modules are state-of-the-art, but that an architecture of asynchronous, communicating concept agents can effectively build an actionable and persistent scene graph. While we currently focus on predefined concepts, we envision this architecture as a foundation that could be extended to synthesise the code of new concept agents using generative models.

3. Methods

3.1. Architectural Motivation

Before describing the CORTEX implementation, we clarify our motivation for concept-first spatial representation. Traditional robotic mapping conceptualises space using occupancy grids—binary free/occupied classifications that enable path planning but provide limited semantic content. While sufficient for navigation, these representations offer little support for (i) human-robot communication—explaining robot decisions in terms of "occupied cells" rather than "rooms" and "doors"; (ii) high-level task planning requiring reasoning about functional spaces and their relationships; (iii) manipulation constraints—understanding that objects belong to rooms and inherit spatial constraints; and (iv) knowledge transfer—communicating spatial understanding between robots or to humans.

Concept-first representation addresses these limitations by grounding spatial understanding in human-interpretable entities from the outset. Rather than adding semantic layers post hoc, the robot builds its world model directly through concept instantiation, enabling transparent reasoning about spatial relationships and their functional implications. The central insight of our approach is hierarchical constraint propagation: Once a room concept is instantiated, it provides geometric boundaries (walls), functional relationships (containment), and semantic priors (door locations) that constrain and improve subsequent concept detection. This differs fundamentally from unconstrained analysis of metric data, where door detection must consider the entire perceptual field rather than focusing on geometrically and semantically relevant regions. This architectural principle naturally

extends to future object detection—furniture constrained to floor planes, paintings to walls, or items to table surfaces—progressively narrowing search spaces through accumulated spatial context, reducing computational complexity, and improving detection reliability.

At this stage, the potential efficiency and robustness gains should be understood as intended outcomes of the architectural principle rather than as quantitatively demonstrated results. The idea of reducing the search space through hierarchical constraint propagation follows well-known strategies in perception and planning, and our architecture is designed to incorporate this mechanism in a concept-driven manner.

3.2. The CORTEX Architecture

CORTEX is a cognitive robotics architecture initially designed to explore how the robot, the environment, and their interaction can be efficiently represented and anchored. A general scheme is shown in Figure 1. It is organised into two blocks: the cognitive level, a collection of specialised memories interconnected by processes called *agents*, which are responsible for exchanging information among them, and the subcognitive level, which maintains a bidirectional connection with the robot's body. Additional details on the architecture can be found in [8,9,45].

The working memory (WM) depicted as a large green circle in Figure 1 is a distributed graph data structure implemented to provide very low latency and high throughput to a large set of connected processes that can edit it. The content represents the robot and its environment, which is built and maintained by specialised agents. Nodes in the graph represent objects in the world or the robot itself, and edges represent relationships among them: geometric transformations in SE(3) with uncertainty (called RT edges) or logic predicates. Agents run asynchronously, can edit the graph, and react to changes as part of their local goals. Table 1 summarises the types and attributes of elements in the scene graph. In the configuration used in this paper, the *robot_body* agent reacts to specific changes in the graph by controlling the robot's movements. The *mission_monitoring* agent updates the graph to ensure the robot's actions are validly serialised. Finally, the *x_concept* agents are pre-defined concept-aware processes that will attach to compatible objects in the environment. They have access to the subcognitive stream of perceptual data and try to explain out the parts of it that match their concept class (e.g., room or door in this case) by predicting it in the next step.

A key feature of the CORTEX architecture is its management of concurrency. The agents operate asynchronously and communicate entirely through edits to the shared working memory graph. To prevent race conditions, the WM's underlying implementation ensures that all node and edge modifications are atomic operations. This design choice ensures data integrity without requiring complex and potentially high-latency locking mechanisms. Agents function as reactive processes, monitoring the graph for relevant state changes and posting their own updates in turn. This event-driven, pub-sub-like interaction model, built upon a foundation of atomic edits, is fundamental to ensuring a consistent and coherent state across the distributed system.

Table 1. Scene graph schema definition.

Element	Type	Key Attributes	Description
Nodes: Semantic and Geometric Entities			
Node	room	ID, geometry, confidence	Represents the primary spatial and semantic entity. A room is defined as a container constructed from a set of associated wall and corner nodes.

Table 1. *Cont.*

Element	Type	Key Attributes	Description
Node	wall	ID, geometry (plane equation), parent_room_ID	A primary structural component of a room. It acts as a geometric reference and a container for door nodes.
Node	corner	ID, geometry (3D point), confidence	A key geometric feature representing the intersection of walls. Corners serve as stable landmarks for robot localisation within a room's reference frame.
Node	door	ID, geometry (pose, dimensions, anchor_points), confidence, state (open/closed)	Represents an articulation point or gateway between spaces. It is geometrically "hung" from a parent wall node.
Edges: Relationships Between Nodes			
Edge	RT (Rigid Transform)	SE(3) transform, covariance	Represents a geometric parent–child relationship with uncertainty. Establishes the local reference frames (e.g., room → wall, wall → door).
Edge	match	predicate (door1_ID, door2_ID)	A semantic link established when the system identifies that two door nodes, instantiated in different rooms, represent the same physical door, enabling loop closure.
Edge	has_intention	predicate, attributes relating to the status of the mission	A temporary, task-oriented edge originating from the robot agent. It signals a request to perform an action on a target node (e.g., cross a door).

3.3. Agent Interaction and Coordination via the Working Memory

Agent collaboration is achieved implicitly through the shared working memory (WM) graph.

- Communication: Agents communicate by creating, modifying, or deleting nodes and edges. For instance, the $\mathcal{C}_{room}$ agent signals its intent to explore by inserting a has_intention edge into the WM.
- Coordination: Agents are data-driven, reacting to specific patterns in the graph. The $\mathcal{C}_{door}$ agent, for example, remains dormant until a current_room node is present and validated in the WM. This ensures a natural, hierarchical order of operations (find room, then find doors).
- Conflict Resolution: A dedicated mission_monitoring agent acts as a central arbiter. It observes all has_intention proposals from other agents. It prioritises them based on the robot's current high-level task, resolving potential conflicts (e.g., choosing to navigate towards a goal rather than further explore a room).

3.4. Overview of the Spatial Representation Construction Process

The robot's primary goal is to build an actionable scene graph representation of its environment through incremental concept instantiation and validation. This process involves several asynchronous agents, primarily $\mathcal{C}_{room}$ and $\mathcal{C}_{door}$, that cooperatively construct a shared spatial model within the working memory (WM), supported by the *robot_body*, *energy_optimiser*, and *long_term_spatial_memory* agents, and orchestrated by the *mission_monitoring* agent.

The representation construction follows a concept-first paradigm, in which spatial understanding emerges from the instantiation and anchoring of semantic entities rather than from metric grid mapping. The robot begins with no prior spatial knowledge and bootstraps its understanding through active exploration. Upon initialisation or after crossing a door, the $\mathcal{C}_{room}$ agent attempts to instantiate a room concept by detecting corners in the 3D

LiDAR point cloud $P \in R^3$. Successfully instantiated rooms serve as reference frames for subsequent spatial reasoning, with the robot's pose continuously updated relative to the current room via factor graph optimisation.

For this initial exploration, we adopt the Manhattan world assumption—rooms are rectangular with orthogonal walls—which simplifies geometric reasoning while remaining applicable to many real-world indoor environments. Each room maintains a centre-based coordinate frame with child frames for its four walls, forming a kinematic tree that structures spatial relationships. Doors must lie within wall boundaries, with at most one door per wall, and each door connects exactly two rooms. Figure 2 illustrates the reference systems used in the scene and their hierarchy in graph form.

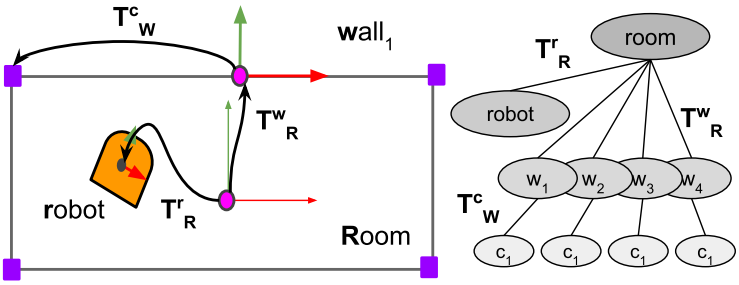


Figure 2. Reference frames used for the elements in the scene graph. On the (**left**), the grey rectangle represents the model room that best fits the corners. $\boldsymbol{T}_R^w$ denotes the transformation from the room to the wall, so a point p in the wall's frame is transformed to the room's frame as $q = \boldsymbol{T}_R^w p$. On the (**right**) is the kinematic tree inserted into the WM , where $\boldsymbol{T}_W^c$ denotes the transformation from the wall to the corresponding right corner in the wall reference system.

The concept lifecycle encompasses detection, initialisation, validation, and maintenance phases. During initialisation, concept agents propose intentional actions (e.g., navigating toward a room's estimated centre or to a vantage position in front of a door) to gather sufficient observational evidence. These actions are mediated by *has_intention* edges in the WM and require approval from the *mission_monitoring* agent before resource allocation. The initialisation process accumulates measurements across multiple robot poses, building temporal pose graphs that resolve geometric ambiguities through optimisation. Once sufficient evidence confirms adherence to concept priors, instances transition from provisional to permanent status in the WM. Figure 3 shows an example of a structural modification in the scene graph. The transition occurs when a room is detected, initialised, and inserted into the graph, causing the robot node to change its parent. The new *RT* edge is now updated by the *energy_optimizer* agent, minimising the differences between measured and nominal features.

Critical to this architecture is the synchronisation achieved through graph-based communication. The $\mathcal{C}_{door}$ agent remains dormant until a current room exists in the WM, ensuring doors are always grounded within validated spatial contexts. This hierarchical dependency—doors require rooms, rooms enable localisation—creates a natural ordering that prevents premature or unconstrained concept instantiation. Similarly, the *energy_optimiser* agent continuously monitors the WM for validated corner and door measurements, incorporating them as landmarks in its localisation factor graph only after concept agents confirm their validity through successful matching against predictions.

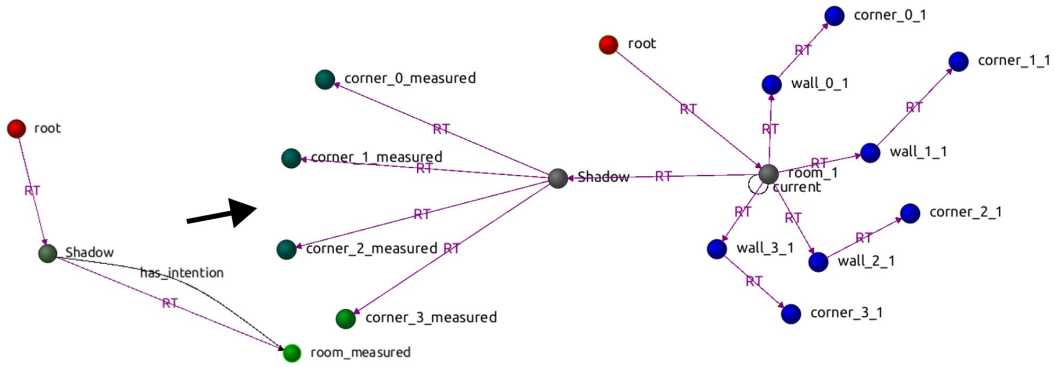


Figure 3. Graph state transition. The robot starts as the graph's initial frame (root is a dummy node) and transforms to hang from the new room when it is inserted in the graph. *corner_x_measured* are nodes that hold the measurements of the corners matched with the corresponding nominal ones on the right.

Through this distributed, asynchronous process, the robot incrementally constructs a scene graph that directly supports navigation, manipulation planning, and human communication without requiring intermediate metric representations. The resulting graph maintains both geometric precision (through continuous localisation) and semantic richness (through concept instantiation), providing an actionable spatial model grounded in human-interpretable entities.

Assumptions and Limitations

The architecture explicitly acknowledges operational boundaries, for example, severely cluttered environments where furniture occludes wall geometry, non-rectangular spaces violate Manhattan assumptions, or dynamic scenes with moving obstacles make successful concept instantiation difficult. Rather than attempting exhaustive coverage of edge cases, the system maintains probabilistic beliefs and can defer instantiation when evidence remains insufficient. Despite this cautious approach, the current implementation faces two critical failure modes: false positives, in which incorrect models satisfy the validation criteria and are accepted as valid representations, and model invalidation, in which initially correct models become inconsistent as additional exploration reveals previously unobserved geometric features. We are developing backtracking mechanisms to revert to previous stable states when model confidence degrades, as well as dynamic restructuring capabilities to adapt existing representations when new evidence contradicts established beliefs. This work is discussed further in the Conclusions and Future Works section. The mission_monitoring agent can interrupt ongoing initialisation processes to pursue higher-priority goals, leaving partial representations for later refinement. This opportunistic approach allows the robot to build useful, if incomplete, spatial models while remaining responsive to task demands.

3.5. Conceptual Agents and Intentional Actions

A conceptual agent $\mathcal{A}_c$ is an autonomous process responsible for the instantiation, maintenance, and refinement of a specific concept class $\mathcal{C}_c \in \mathbb{C}$, where $\mathbb{C}$ denotes the set of all concept classes manageable by the robot (e.g., $\mathcal{C}_{\text{room}}, \mathcal{C}_{\text{door}}$). Each concept class $\mathcal{C}_c$ is defined as a tuple $\mathcal{C}_c = (\mathcal{F}, \Phi, \mathcal{A}_{\text{ff}}, \mathcal{I}, \mathcal{L})$, where

- $\mathcal{F}$ is the set of f_i measurable properties that define the geometry of the concept (e.g., height, width, centre, corners, anchor points, etc.).
- Φ is a set of prior values over the properties that determine when a new instance is created. (e.g., height-to-width ratio).

- $\mathcal{A}_{\text{ff}}$ is the set of affordances associated with the concept. In this case, the affordances are *visit* for the room and *visit* and *cross* for the door. The *visit* intention is notified to the *mission_monitoring* agent with the insertion of a *has_intention* edge in the graph, while the *cross* affordance is notified with the creation of a special node *aff_cross_x* (denoting an affordance to cross door x) hanging from the associated door. If the affordance is accepted by the *mission_monitoring* agent, the *robot_body* agent executes the action until completion.
- $\mathcal{I}$ is the initialisation process for a new instance candidate.
- $\mathcal{L}$ is the life cycle of the concept instances, defined as a behaviour tree that monitors and controls the transitions between internal states.

The lifecycle ($\mathcal{L}$) is implemented as a behaviour tree that continuously orchestrates four primary stages of concept instance management (see Figure 4). First, the agent evaluates whether existing nominal instances in the WM can explain incoming sensor measurements through forward prediction and matching. When sensor data cannot be adequately explained by current instances (matching error exceeds threshold), the initialisation process ($\mathcal{I}$) is triggered, accumulating evidence through intentional actions until sufficient confidence permits instantiation of a new nominal element in the WM. For established instances, the agent maintains their validity by computing predicted feature positions based on current robot pose estimates, enabling these predictions to serve as landmarks for robot localisation through the energy minimisation process. Concurrently, when affordances are active, the agent monitors their execution status through the robot node, tracking action progress until completion or interruption by the mission monitoring agent. This cyclic process ensures that concept instances remain grounded in sensory evidence, provides stable reference frames for navigation, and supports opportunistic exploration through affordance execution.

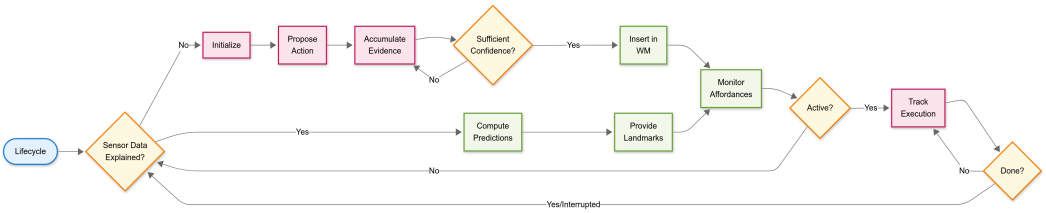


Figure 4. Flow chart showing the life cycle of a concept. The two outgoing lines of the first decision box cover the insert new instance and update existing instances situations.

3.5.1. *Room Concept* Agent

$\mathcal{C}_{room}$ implements the room concept class. The class provides a parameterisation of a room in terms of centre, angle, and dimensions, $\mathcal{F}_{room} = [x_c, y_c, \alpha, w, d, h]$, as well as the equivalent description by its corners $c_i \in C$. $\mathcal{F}_{room}$ components are illustrated in Figure 5. Room instances are actively searched when the robot starts or when it crosses a door. As explained in the next subsection, the *long_term_spatial_memory* agent asynchronously recovers previously known rooms and inserts them into the WM.

Room corners are detected using a simple yet efficient method described in Algorithm 1. This algorithm, along with the door detection method presented later, does not represent state-of-the-art (SOTA) performance but provides a sufficient baseline for demonstrating the proposed architectural principles.

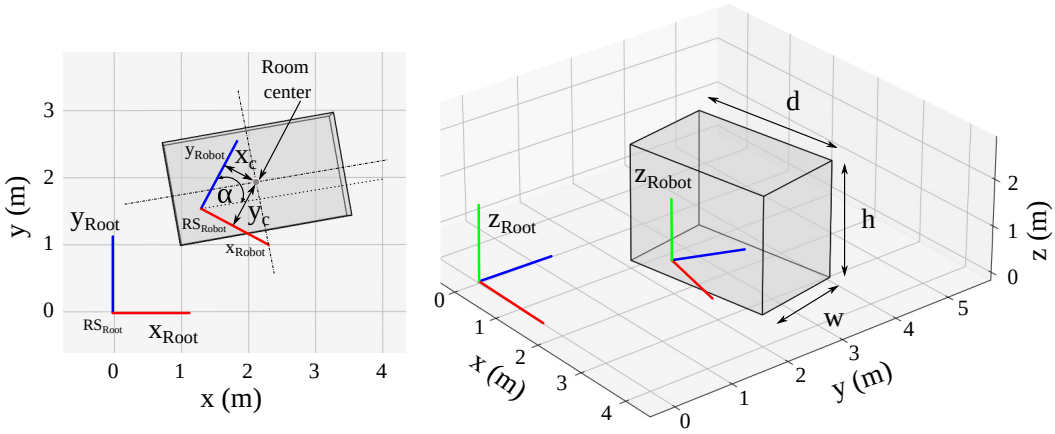


Figure 5. $\mathcal{F}_{room}$ components diagram. Before the room is established, two reference systems are established: the origin reference system and the robot reference system. In the case of the robot, x_c and y_c denote the position of the centre of the room relative to the robot, while α is the angle relative to the longest side. Conversely, w denotes the length of the shorter side, d represents the length of the longer side, and h signifies the height of the room.

Algorithm 1 Corners Detector Algorithm

1: $p_i = P(\rho, \phi, k), z_{min} < p_z < z_{max}$
2: $\mu = \frac{1}{M} \sum p_i$
3: $\Sigma = Cov(p)$
4: $\bar{c} = \mu$
5: $\bar{s} = \lambda_1(\Sigma), \lambda_2(\Sigma)$
6: $l = Hough(p)$
7: $l \in L^* if l > L_{min}$
8: $\hat{c} = \binom{l}{2}$
9: $\hat{c} \in \mathbf{C} \iff l_i \langle l_j = \frac{\pi}{2}$
10: Return **C**

The algorithm begins by (1) filtering the 3D LiDAR field P to retain only those points p_i within a vertical slice defined by $z_{\min} < p_z < z_{\max}$, reducing noise from floor and ceiling artifacts; (2–5) computing the mean and covariance matrix to estimate the room centre $\bar{c}$ and room's dimensions $\bar{s}$; (6) applying a Hough Transform, using OpenCV's point set implementation to directly process the point cloud without rasterization, in order to extract structural lines; (7) retaining candidate walls L^* if their lengths exceed a threshold $L_{\min} = 1m$; and (8,9) checking all pairs of lines (l_i, l_j) for $\pi/2$ angle crossings to select candidate corners $\hat{c}$. The algorithm returns a list of corners.

The $\mathcal{C}_{room}$ agent continuously runs the corner detection algorithm. When a set of detected corners satisfies the instantiation conditions Φ_{room}, and there is no room object in the WM already, the $\mathcal{C}_{room}$ agent inserts a temporary room node into the WM, linked to the robot node via a *has_intention* relation. Upon approval by the *mission_monitoring* agent, computational and physical resources are allocated, and the robot initiates a navigation action toward the estimated centre of the room, computed as the centroid of the currently measured corner points. This action develops locally in the robot's frame.

At the beginning of the action, the robot's position is taken as the initial zero frame. As the action proceeds, new poses are concatenated from the robot's odometry readings and stored in a local buffer. The measured corners are also stored in the current robot frame. When the action is completed, the concept agent evaluates whether the accumulated data is sufficient; if so, it initiates a post-processing phase. Otherwise, sampling continues from the new vantage point. The *mission_monitoring* agent can revoke the affordance at any time.

As no external frame is available, the agent builds a temporal pose graph in the GTSAM (https://github.com/borglab/gtsam, accessed on 16 September 2025) framework using data accumulated during the action. The measured corners serve as landmarks, and because they are observed from multiple robot poses, the graph acquires the necessary redundancy. The optimal graph that minimises the sum of prediction errors provides a set of corrected landmarks and poses. From the set of corrected corner poses, a histogram is built to select the most probable corners that comply with the room constraints. The concept agent applies additional priors at this point to potential candidates, discarding rooms that are too large or too small.

The process ends with inserting the room into WM, along with its walls and corners, and an *RT* edge that anchors the robot to the room. This last step requires some elaboration, since a rectangle lacks an intrinsic orientation. The temporal pose graph was anchored to the action start pose. The selected set of corners has coordinates in that frame, and the robot now occupies the last pose of the graph. Using these data, we want to set the room frame as the initial frame and compute the robot's pose with respect to it. A simple solution is to take the position and size room parameters from the selected corners, and compute the rotation, or equivalently, an enumeration of the corners that assigns index zero to the first corner to the left of the robot heading direction.

3.5.2. *Door Concept* Agent

The $\mathcal{C}_{door}$ agent controls the life cycle of instances belonging to the door concept class. Doors are parametrised in $\mathcal{F}_{door}$ as pairs of anchor points defining the door ends. They have two affordances, $\mathcal{A}_{ff}$: *visit* and *cross*. Φ_{door} is the range of admissible distances between the anchor points, currently set to 0.5–1.3 m.

We propose a simple door detection algorithm, shown in Algorithm 2. It begins by (1) assuming that doors lie within the walls W of the room and filtering out any point p_i whose distance to the walls exceeds a predefined threshold δ = 0.15 m; (2) extracting a polar 2D scan $\rho = p_i(\phi, k = K)$, where ρ is the measured distance for each angular value ϕ at a fixed height index K = 1.7 m, in our experiments; (3) taking the derivative $\dot{\rho} = \frac{\partial \rho}{\partial \phi}$ to identify abrupt changes in the distance dimension; (4) applying the Heaviside step function H to yield a binary peak map where a value of 1 indicates a potential edge and zero otherwise, and a value of λ = 0.5 m represents a distance threshold between continuous points; (5) computing all combinations of pairs of detected points; (6) testing each candidate against the width constraint and the wall constraint that ensures that the line segment, which is defined between detected edge points p and q, lies within the set of wall points; and (7) returning the set D of valid detected doors.

Algorithm 2 Door detector algorithm

1: $p_i = p \in P \wedge \text{dist}(p, W) \leq \delta$
2: $\rho = p_i(\phi, k = K)$
3: $\dot{\rho} = \frac{\partial \rho}{\partial \phi}$
4: $P = H(\dot{\rho} - \lambda) = \begin{cases} 0 & \text{if } \dot{\rho} < \lambda \\ 1 & \text{if } \dot{\rho} \geq \lambda \end{cases}$
5: $\left\{\hat{d}_{i,j}\right\}^N = \binom{\mathbf{P}}{2}$
6: $\hat{d}_{i,j} \in \{D\}$ if $\begin{cases} D_{\min} < \|\hat{d}_{i,j}\| < D_{\max} \\ d_{i,j} = p + t(q - p), \quad 0 \leq t \leq 1, \vec{pq} \in W \end{cases}$
7: Return D

When the current room is detected in the WM, the $\mathcal{C}_{door}$ agent starts running the door detection algorithm. It silently returns if existing nominal doors can explain all the pairs of anchor points detected in the point cloud. When a new door is detected, a process similar to the one described before starts, but with some differences. A provisional door, labelled as *door_x_pre* to signify its preliminary, unvalidated status as a concept instance, is inserted into WM hanging from its containing wall, and a *has_intention* edge is inserted between the robot and the door. Figure 6 shows how these changes occur in the scene graph.

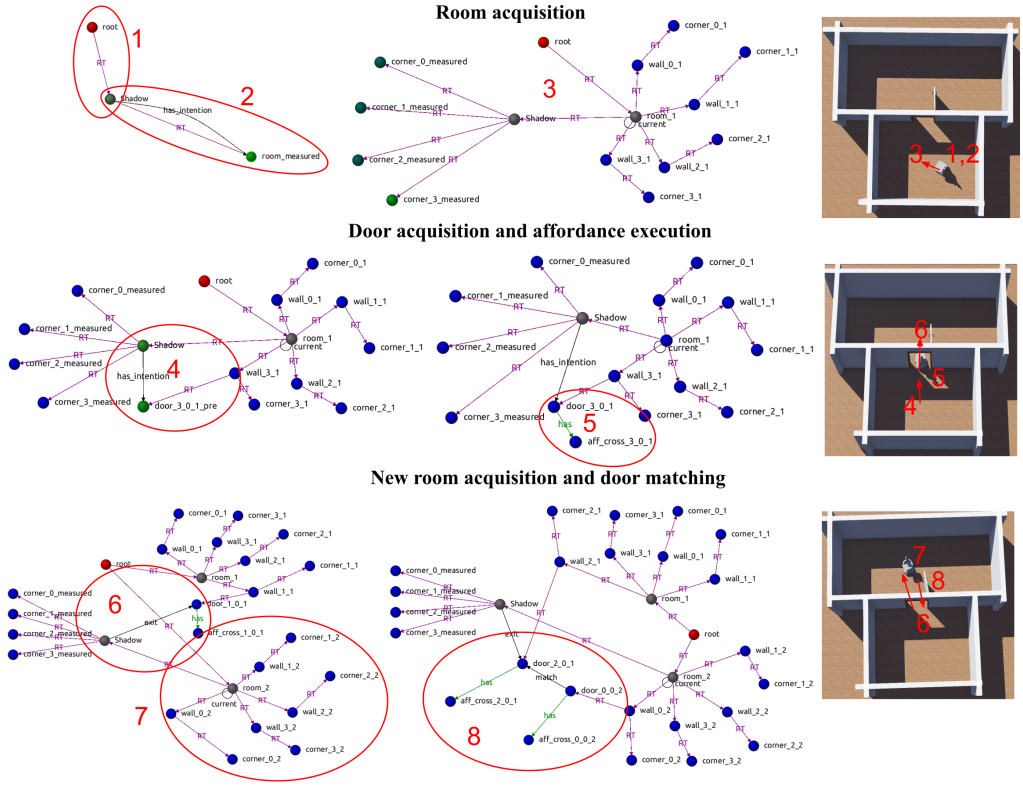


Figure 6. Main stages (sequential steps from 1 to 8) in constructing the scene graph during exploration, reflecting transitions as different concept instances are initialised and affordances are executed. See text for a detailed explanation.

Once the *mission_monitoring* agent assigns resources, the *visit* affordance is executed by the *robot_body* agent. The robot navigates towards a point located 1 m from the door's centre. But now the robot is being localised in the room's frame by the *energy_optimiser* agent running a global optimisation over the scene graph. To benefit from this situation, the $\mathcal{C}_{door}$ agent only updates the matching between the measurements and the provisional anchor points and waits for the end of the action. The *energy_optimiser* agent recognises those objects as landmarks and includes them in the minimisation. When the action finishes, the fine-tuned anchor points are set as permanent, and their nominal coordinates are computed and inserted in the *RT* edge connecting the wall to the new door. The door name is changed to *door_x*, removing the *pre* suffix. The nominal door now provides two additional fixed landmarks for the *energy_optimiser* agent to update the robot's pose in the room. Finally, a node connected to the door with an *has* edge is added to show that a *cross* affordance is available for the *mission_monitor* agent to include in its planning logic.

3.6. Long-Term Spatial Memory

The proposed architecture deliberately sacrifices global metric consistency in favour of local precision and long-term scalability. This is managed by the CORTEX long-term spatial memory (LTSM) agent, whose primary function is to maintain a topological graph

of locally consistent, concept-centric metric frames. The LTSM stores previously visited rooms and their connections via doors. This information is maintained in a persistent graph data structure using the *igraph* library (https://igraph.org/, accessed on 16 September 2025). The agent operates asynchronously over the WM, removing visited rooms when the robot has entered and initialised a new one, and bringing (i.e., predicting) known rooms back when the robot is about to enter a previously explored one. The precise timing of these actions is explained below.

When the robot enters an unfamiliar room, it needs to retain some information about the room it is leaving until a reliable connection can be made between the two. The agent stores doors as dual objects with coordinates in both rooms' frames. This is the minimal information needed to maintain an actionable graph of connected rooms. As the agent observes the crossing, it waits until the new room is initialised following the procedure described before, and only then sets the new room as the current room and removes the old one from the WM. The required synchronisation is done through the edition of node attributes and edges in the WM, since this is the only way for agents to communicate.

The insertion of a known room is simpler since this agent is anticipating what $\mathcal{C}_{room}$ will see. The critical variable is the orientation of the new room relative to the robot, since the success of the corner-matching process, which determines acceptance, depends on it. The information is taken from the door dual coordinates. The new room is oriented by first transforming its corners to the door's coordinate frame. Then, we identify both door frames as being at the same point in space, and use this to transform the corners back to the old room's coordinate frame and, finally, to the robot's frame. In this way, the robot can match the new room's nominal corners with fresh measurements, since both are in the same frame, and confirm the recognition of the new room. Hereafter, the old room is removed, and the robot is changed in the scene graph to hang now from the new room.

Figure 7 shows a metric representation of the rooms in the four-room scenario, with the rooms the robot has previously traversed shown in green, and the room it is currently located in shown in red. Because the robot is currently executing the cross affordance during the capture, a momentary state lag occurs: The system has not yet loaded the map data for the destination room and, therefore, remains localized within the room of origin. Additionally, a green dot indicates the target position the robot is heading towards after crossing the door. The agent can create a metric representation of the graph at any time, as shown in the bottom-left image. The evident alignment errors are due to noise injected into the synthetic LiDAR sensor and the robot's displacements, and could be corrected by constrained optimisation of the rooms' centres, sizes, and orientations, with the doors serving as fixed, shared points between them. It is left for future work, as it does not interfere with the goals of this paper. The red circle on the right shows the result of a loop closure between rooms 1 and 4 through their shared door. Currently, the *LTSM* agent computes a loop closure by projecting the robot's pose into all rooms as it approaches a new one. A more robust method based on visual descriptors and the objects stored in the room is under construction.

Scalability Through Hierarchical Memory Organisation

The LTSM component addresses a fundamental scalability challenge in robotic spatial representation: Maintaining complete world models in active memory becomes computationally prohibitive as environment size increases. The CORTEX architecture's memory organisation provides several mechanisms that enable operation in arbitrarily large environments while preserving real-time performance. First, the architecture employs dynamic memory management based on the principle that robots require detailed spatial representations only for their immediate operational context, thereby avoiding the computational

burden of maintaining and continuously optimising a single, monolithic global map. As the robot navigates multi-room environments, the LTSM dynamically loads rooms into working memory based on current location and planned actions, while unloading distant or irrelevant spaces. This selective memory management maintains bounded computational requirements regardless of total environment size, enabling real-time operation in large-scale facilities such as office buildings, hospitals, or multi-floor residential complexes. Second, the LTSM maintains persistent topological graphs representing complete known environment structures, stored using the *igraph* library. These graphs naturally accommodate arbitrary environment topologies—from linear corridor sequences to complex branching structures with multiple floors, wings, and interconnected spaces. Third, the memory architecture supports hierarchical spatial organisation, with concepts extending beyond individual rooms. Spaces can be organised into floors, buildings, or functional regions (e.g., "residential wing", "laboratory area"). This hierarchical organisation enables efficient navigation planning across multiple scales while maintaining the human-interpretable structure that motivates the concept-first approach. High-level spatial queries can be resolved at appropriate abstraction levels without requiring detailed geometric computation. And fourth, context-dependent predictive loading activates when the robot approaches transition points (doors leading to known rooms) and anticipates spatial context requirements by pre-loading relevant representations. This includes not only the target room's geometry and contained objects, but also connected spaces that might become relevant for navigation alternatives or task planning. This predictive loading strategy ensures smooth spatial transitions while minimising working memory overhead and maintaining responsiveness.

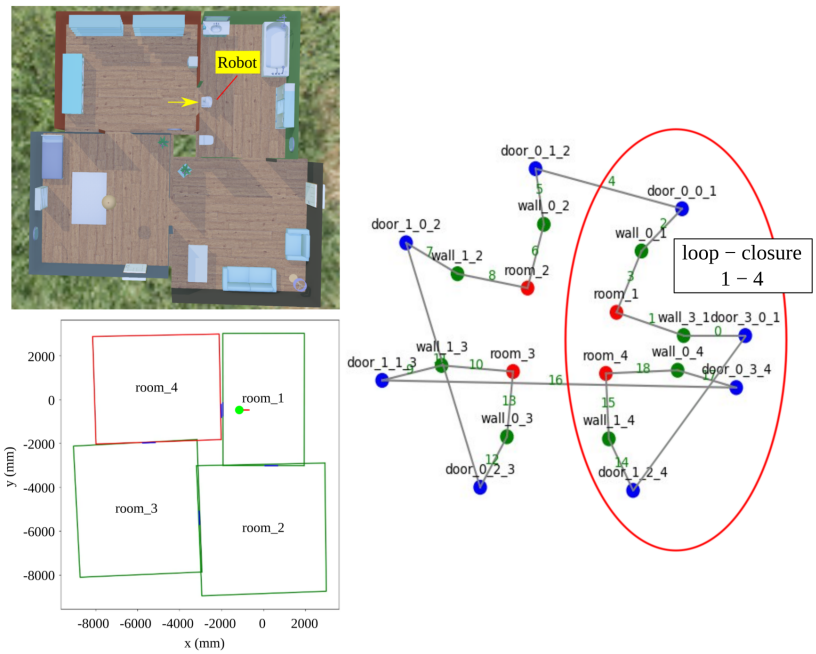


Figure 7. LTSM room representation during a transition from a known room to another known room via an uncrossed door. The scenario used in the Webots simulator for the experiments is shown in the upper-left image. The down-right area shows a metric reconstruction of the graph on the right. The right image shows the network generated and stored by the *LTSM* agent after the robot has toured rooms 1 to 4. The red circle marks the loop closure between rooms 4 and 1 as an edge connecting door_3_0_1 and door_1_2_4.

Taken together, these mechanisms ensure that the system's computational complexity remains bounded and does not grow with the size of the total explored environment. A formal asymptotic analysis is outside the scope of this architectural work; however, a qualitative analysis indicates that high-frequency, computationally intensive operations

(e.g., factor graph optimisation, point cloud processing) are constrained to entities within the currently active room in working memory. By dynamically loading and unloading spatial contexts, the LTSM prevents the unbounded growth of active nodes and factors in real-time processes. Consequently, the architecture's computational load is not a function of the global environment's size but instead depends on the local complexity of the robot's immediate surroundings. This design choice is fundamental to ensuring long-term operational performance and scalability, while preserving the transparency and explainability advantages of the concept-driven approach.

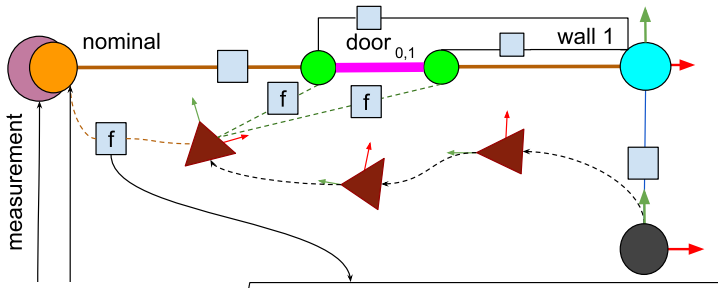
3.7. Navigating Through the Scene Graph

Once a nominal current room is established, the *energy_optimiser* agent updates the robot's position at a fixed frequency of 10 Hz. This agent maintains a GTSAM factor graph. It is initialised with the nominal room corners and the robot's pose in the WM. As fresh odometry data arrive from the robot sensors, the agent inserts new robot poses into the graph up to 30 nodes, applying a FIFO policy. The sliding-window version of iSAM2 in GTSAM computes the optimal robot pose after the graph modifications, accounting for the required changes to the covariance matrix.

At each insertion of a new robot pose, the *energy_optimiser* agent verifies if there are measured corner values in the WM that the corresponding concept agent has validated. This validation results from a matching process between the projection of the nominal corner $\mathbf{y}$ onto the robot's frame and the current measurement $\mathbf{x}$. We use the Mahalanobis distance d_M to perform this critical pre-association check against a statistical threshold, where d_M is calculated as

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\boldsymbol{\mu}_\mathbf{x} - \boldsymbol{\mu}_\mathbf{y})^T \left(\frac{\boldsymbol{\Sigma}_\mathbf{x} + \boldsymbol{\Sigma}_\mathbf{y}}{2} \right)^{-1} (\boldsymbol{\mu}_\mathbf{x} - \boldsymbol{\mu}_\mathbf{y})}$$

If the match is validated, a factor (cost function) is created in the graph. As illustrated in Figure 8, this factor (represented by *f*) formalizes the statistically weighted difference between $\mathbf{x}$ and $\mathbf{y}$ $(\boldsymbol{\mu}_\mathbf{x} - \boldsymbol{\mu}_\mathbf{y})$. In this metric, $\boldsymbol{\Sigma}_\mathbf{x}$ represents the covariance of the measurement, and $\boldsymbol{\Sigma}_\mathbf{y}$ represents the total propagated uncertainty of the predicted feature, combining the uncertainty of the nominal corner's position and the robot's current pose. Validation is determined by inspecting a Boolean attribute stored in the corner measurement node of the WM.



$$d_M(\mathbf{x}, \mathbf{y} = \mathbf{T}_x^{rob}) = \sqrt{(\boldsymbol{\mu}_\mathbf{x} - \boldsymbol{\mu}_\mathbf{y})^T \left(\frac{\boldsymbol{\Sigma}_\mathbf{x} + \boldsymbol{\Sigma}_\mathbf{y}}{2} \right)^{-1} (\boldsymbol{\mu}_\mathbf{x} - \boldsymbol{\mu}_\mathbf{y})}$$

Figure 8. Schematic view of a robot displacement starting at the black circle and advancing through the upper left quadrant. Small grey squares with an *f* are factors, i.e., positive functions measuring the difference between two variables, that link the robot to the objects' parts, corners, and anchor points. Plain grey squares are fixed constraints between elements. The sum of all differences accounts for the system's free energy and is absorbed by the set of past robot poses depicted as red triangles. As more objects are modelled in the WM, more factors are created, making the robot's localisation more robust.

If a corner has been validated, it is incorporated into the factor graph as a measurement associated with its corresponding landmark. This measurement is connected to the pose node whose timestamp most closely matches the corner observation's timestamp. Similarly, if a door has been instantiated in the WM, its nominal position is inserted into the graph as a landmark. Door measurements are handled in the same way as corner observations.

4. Results

We have designed a series of preliminary experiments to evaluate the architecture using the Webots simulator. The scenarios include a digital replica of our Shadow mobile robot [46], with realistic white noise added to the synthetic LiDAR and simulated delays in command execution. All detection parameters and thresholds used in the following experiments, such as those in Algorithms 1 and 2, were tuned empirically to suit this specific robotic setup and the characteristics of our test environments.

It is worth noting that these experimental scenarios are simplified, structured environments primarily intended as a proof-of-concept to demonstrate the architecture's core mechanics. Within these constraints, we included rooms of varying dimensions and with different numbers of doors to explore the system's ability to handle some topological variety, while maintaining controlled conditions for initial validation. Future work will extend these evaluations to more complex layouts, dynamic obstacles, and a broader range of sensor noise levels.

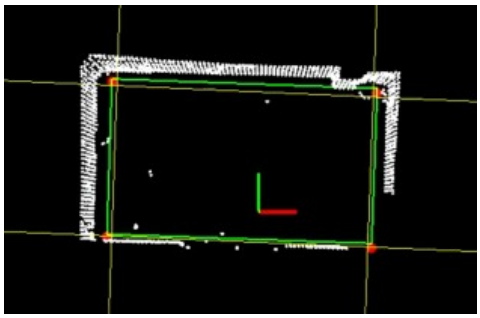
A first experiment, Figure 6, shows the three main stages in the evolution of the scene graph within the WM. Red circles denote interest zones referenced in the text.

- Room acquisition: Following the sequences shown in Figure 6 as red ellipses, in Zone 1, the robot lacks a room representation. The $\mathcal{C}_{room}$ agent is aware of the absence of a room node and starts the initialisation process. This is reflected in the WM by the appearance of the *has_intention* link between the robot *Shadow* and *room_measured*, Zone 2. The $\mathcal{C}_{room}$ waits until the *mission_monitoring* agent authorises the action.
  When the robot reaches the room's centre or sufficient data have been gathered, the $\mathcal{C}_{room}$ agent adds the room to the graph. The coordinate frame changes during this transition, and the robot's pose is expressed relative to the room's reference frame. This relationship is captured by the *RT* link between the room and the robot, as shown in Zone 3. From now on, this link is updated by the *energy_optimiser* agent, which performs continuous optimisation over the room's corner observations.
- Door acquisition and affordance execution: With a current room established in WM, the $\mathcal{C}_{door}$ agent can now proceed with detecting doors. Zone 4 shows a new door proposal *door_3_0_1_pre* and an action proposal in the edge *has_intention*. The robot moves close to the door, and the agent changes the door status from provisional to acquired, removing the *pre* suffix. Additionally, the agent inserts an affordance node *aff_cross_3_0_1* hanging from the new door to notify the *mission_monitoring* of that action's availability. This is shown in Zone 5.
- New room acquisition and door matching: Executing the door-crossing affordance, Zone 6, takes the robot into a new room, triggering a second initialisation process. Upon completion, the $\mathcal{C}_{room}$ agent inserts a new room node, *room_2*, and its constituent elements into Zone 7 of the graph. If the room had been previously known, the *LTSM* agent would instead load the stored room, bypassing the initialisation step. Zone 8 illustrates how doors are associated across rooms immediately before the *LTSM* agent removes the previously visited room and sets the new one as the current. At this point, the door obtains the dual coordinates that place it in both rooms. The *LTSM* agent maintains a local graph of all known spaces and the open transitions between them, while adding and removing rooms as the robot explores its environment.

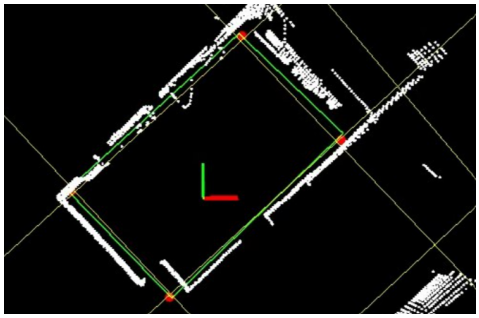
A second experiment demonstrates a more extended activity within a 10-room scenario. Following the steps described in the previous experiment, the robot incrementally builds its local and global representations and, at some point, closes a loop connecting several rooms. A video of the full experiment can be downloaded from the site (https://cloudiepcc.unex.es/index.php/s/Zq2eY9fLeDa7itP, accessed on 16 September 2025). All agents and the components in the sub-cognitive module run in different cores of the onboard computer, an Intel i9 13th generation processor with an NVIDIA RTX-A2000 GPU (NVIDIA Corporation, Santa Clara, CA, USA) , and can sustain the nominal sensor acquisition rates, 20 Hz for the 3D LiDAR and 30 HZ for the 360° camera (Ricoh Company, Ltd., Tokyo, Japan).

The third experiment places the robot Shadow in a real scenario, navigating between two rooms separated by a door. Figure 9 shows the two rooms with their usual furniture and the room layouts extracted from them. Note that processing only the upper part of the 3D LiDAR field, $z > 170$ cm, eliminates most of the non-wall points, and the large structures detected with the Hough lines provide enough information to complete the correct layout. We are exploring ways of removing more of the remaining points that do not belong to the walls, for instance, by filtering the 3D point cloud with a semantic segmentation DNN. The robot has remained located with respect to the rooms' frames for four-hour navigation sessions, with maximum positioning errors of 15 cm.

Room A and B



Room A layout



Room B layout

Figure 9. Real test scenarios with their respective layouts, red corners and green walls detected over the LiDAR profile.

4.1. Measurement Error Analysis

To quantitatively assess the geometric accuracy of the generated floor plans, a metric in both rooms and door widths was also negligible, demonstrating the system's adherence to the simulator. A visual representation of this superposition for one of the test runs can be seen in Figure 10.

The aggregated results from multiple generation runs are summarised in Table 2. The results indicate strong positional and dimensional fidelity. The low mean error in room placement suggests that the generated global map effectively maintains the correct spatial relationships among rooms. The standard deviation for this metric was minimal, suggesting high consistency and low variance across different outputs. Dimensional errors for both rooms and door widths were also negligible, demonstrating the system's ability to adhere to the specified architectural constraints. For the room dimensions, the errors are correlated with the wall thickness of the scenario, which is 200 mm. To prevent the propagation of wall-thickness-related errors, this factor is explicitly accounted for during the generation of the global map. These quantitative findings confirm the qualitative assessment, demonstrating that the generation model produces geometrically sound, accurate floor plans with minimal deviation from the target structure.

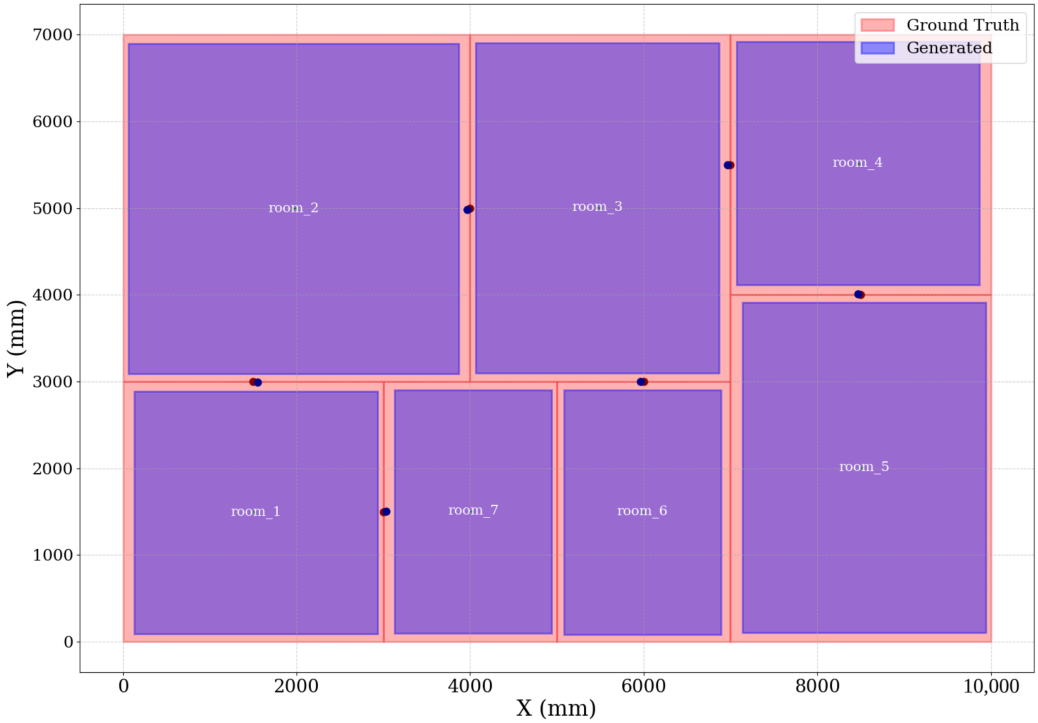


Figure 10. Visual comparison between the ground truth floor plan and a generated result after the automatic alignment process. The ground truth is shown in red, while the generated layout is shown in violet. Rooms are represented as rectangles, and doors are marked with circles.

It is crucial to highlight that while the quantitative analysis reveals marginal geometric deviations, the most relevant outcome of this evaluation is the consistency and correctness in the construction of the topological map. Despite the minor positional and dimensional errors detailed in Table 2, the system successfully identified the correct topological structure, that is, which rooms connect to each other and through which doors, in all of the test cases. This success in inferring connectivity is fundamental to the robot's navigation and planning tasks, validating that the proposed approach generates environment representations that are not only geometrically accurate but also, more importantly, functionally and topologically reliable.

Table 2. Error analysis summary.

Metric	Mean Error (mm)	Standard Deviation (mm)
Room position error	34.1	16.2
Room dimension error	201.5	8.8
Door position error	37.5	13.9
Door width error	17.4	16.7

4.2. Computational Performance Analysis

This section analyzes the computational performance of the architecture, focusing on the CPU and memory consumption of the main processes. The data were recorded during an experimental run that included the construction of a scene graph across several rooms, providing insight into the system's operational efficiency and resource allocation. As shown in Figure 11, the CPU usage for most agents that constitute the architecture remains stable over time. The graph highlights the activation of the door detection process, marked by a notable increase in its CPU consumption, which occurs once a room model has been stabilised. This increased load is attributed to the agent's task of managing and validating multiple door instances within the active room.

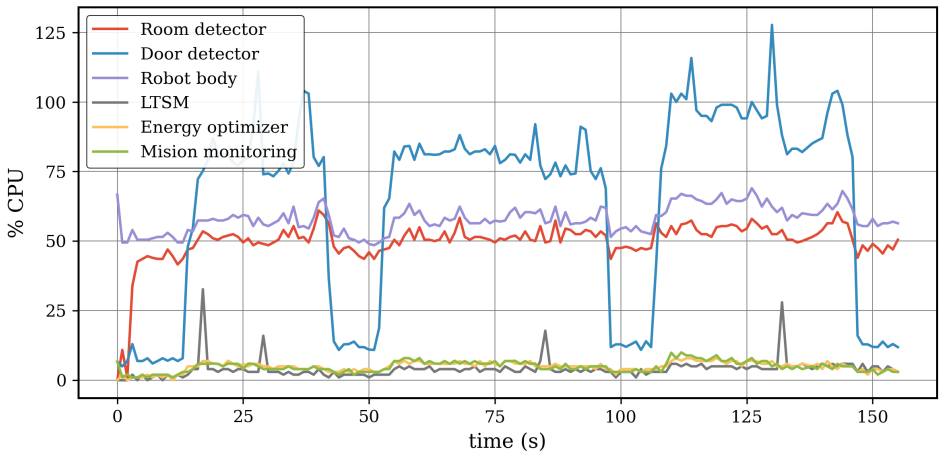


Figure 11. CPU usage over time for the main processes running during the graph construction process.

Table 3 summarizes the memory consumption for the main architectural processes. The results demonstrate the system's high efficiency, with all agents operating with a minimal memory footprint. The mean RAM usage for each process is consistently low, ranging from 0.22% to 0.33%. Furthermore, the low standard deviation across all processes indicates stable and predictable memory consumption, a crucial feature for long-term autonomous operation on resource-constrained robotic platforms.

Table 3. Resource usage summary: memory (total system RAM: 64 GB).

Process	Mean RAM Usage (%)	Mean RAM Usage (MB)	Std. Dev. (%)
Room detector	0.30	197	0.00
Door detector	0.28	184	0.04
Robot body	0.31	203	0.03
LTSM	0.33	216	0.05
Energy optimiser	0.22	144	0.04
Mision monitoring	0.22	144	0.04

5. Noise Sensitivity and Detection Mistakes

The robustness of concept-first spatial representation depends critically on reliable feature detection and consistent model validation. Two primary failure modes affect the current implementation: sensitivity to sensor noise and detection mistakes that manifest either immediately or after additional exploration.

5.1. Noise Sensitivity

The quality of the final spatial representation degrades as sensor and odometry noise increase. While systematic evaluation of noise sensitivity requires extensive parametric studies (planned for future work), preliminary observations from our simulations reveal several noise-related challenges. LiDAR noise affects corner detection reliability, particularly when points near wall intersections are displaced beyond the algorithm's clustering threshold. Similarly, odometry drift compounds during the initialisation phase, potentially causing misalignment between the selected corner set and the actual room geometry. The GTSAM optimisation partially mitigates these effects by distributing error across the pose graph, but at high noise levels, incorrect room models can be accepted as valid.

Future work will quantify these relationships through controlled experiments varying (i) LiDAR point cloud noise ($\sigma = 0.01$ m to 0.1 m), (ii) odometry drift rates (0.5% to 5% of distance travelled), and (iii) angular measurement uncertainty (0.5% to 5%). The expected outcome is a noise tolerance envelope within which the concept-first approach maintains acceptable spatial accuracy.

5.2. Detection Mistakes and Model Revision

The current lifecycle implementation assumes that once validated, concept instances remain fixed. This rigid approach fails to handle two critical scenarios that occur in practice.

Early Model Mistakes: Despite validation, incorrect models may still satisfy the instantiation criteria due to partial occlusions, symmetric ambiguities, or sensor limitations. For example, a large L-shaped space might initially be interpreted as a rectangular room when only one branch is visible. The current architecture lacks mechanisms to revise these early commitments when contradictory evidence emerges.

Temporal Model Invalidation: Initially correct models may become inconsistent as exploration reveals previously unobserved features. A room model validated from limited viewpoints might prove incompatible with newly discovered walls or openings. Without dynamic revision capabilities, these inconsistencies accumulate, degrading the overall representation quality.

Addressing these limitations requires extending the concept lifecycle with continuous model fitness evaluation and revision mechanisms. This would provide the corrective capabilities that the current system lacks. We envision a more flexible lifecycle incorporating the following.

1. Continuous Fitness Monitoring: Each concept instance maintains a running fitness metric quantifying the agreement between predicted and observed features. A persistent low fitness score would serve as the trigger to invalidate the model and its child concepts, forcing a re-evaluation. This metric would track both recent observations (for rapid response) and historical consistency (for stability).
2. Graduated Response Strategy: When fitness degrades below threshold, the system would engage a hierarchical response: (i) local parameter adjustment for minor discrepancies, (ii) structural revision for significant geometric changes, and (iii) complete model replacement when the current instance becomes untenable.
3. Backtracking and Alternative Hypotheses: The architecture would maintain alternative concept instantiations as latent hypotheses, enabling rapid switching when the

primary model fails. This requires extending the concept agents to support probabilistic beliefs over multiple competing interpretations. A particle filter would be a good starting point.

4. Graceful Degradation: When no satisfactory model exists, the system should maintain partial representations rather than forcing incorrect instantiations. This might involve temporary metric patches or undefined regions marked for future exploration.

These extensions would transform the current deterministic lifecycle into a probabilistic framework where concept instances compete, evolve, and occasionally fail. While computationally more demanding, this flexibility is essential for robust operation in real-world environments where perfect sensing and complete observability cannot be guaranteed.

6. Discussion

The experimental results confirm the viability of our concept-first architecture for building actionable scene graphs. Due to the fundamental architectural differences, a direct quantitative comparison with existing metric-first systems is not our primary focus. Instead, this section contextualises our contributions and limitations by qualitatively comparing our approach against leading alternative frameworks.

Our architecture's primary advantage is its independence from a pre-existing global metric map, which contrasts with frameworks like Hydra that ground their 3D scene graph (3DSG) in a dense mesh. By building a world model from connected local reference frames, our system can offer greater scalability while avoiding the need to maintain a single, monolithic metric representation. However, a key trade-off of this design is that our sparse, concept-based representation lacks the dense geometric fidelity of metric-first systems. While this can be a disadvantage for low-level tasks such as navigation and collision avoidance in cluttered, unstructured environments, it is important to note that a metric grid can be created for each room when necessary to support these functions.

Our work prioritises explicit modelling of environmental structures—such as rooms and walls—as first-class entities in the graph. This enables our core principle of hierarchical constraint propagation—using the geometric context of a room to guide the search for a door—a top-down mechanism distinct from the primarily bottom-up, object-centric approach of many open-vocabulary systems. The clear drawback is our reliance on predefined concepts, which limits our semantic flexibility compared to open-vocabulary systems that can perceive and ground an open set of objects and relationships. However, this intentional design choice ensures that our architecture remains deterministic and interpretable. In contrast to the "black-box" nature of deep learning-based detectors used in many open-vocabulary systems, our approach provides clear, explicit mechanisms for reasoning about the environment's structure.

7. Conclusions and Future Works

This work presents an architectural exploration of concept-first spatial representation for mobile robots. By implementing room and door concepts as asynchronous agents within the CORTEX cognitive architecture, we demonstrate that semantic scene graphs can be built incrementally without requiring prior metric maps. The key innovation is hierarchical constraint propagation, where room instantiation provides geometric and semantic priors that improve door detection efficiency and robustness compared to unconstrained metric analysis. While our current implementation is limited to rectangular rooms in structured environments, the architectural principles—such as asynchronous concept agents, direct scene graph construction, and hierarchical constraint propagation—extend naturally to more complex spatial concepts and irregular geometries. We intentionally employ simple detection algorithms (e.g., Hough transforms and gap analysis) to emphasise architectural

rather than algorithmic contributions. More sophisticated perception methods would enhance system performance while preserving the core organisational principles.

An important avenue for future research is to extend the architecture to multi-robot or multi-agent scenarios. While our current work is scoped to a single robot, a shared WM and LTSM across multiple agents could, in principle, enhance global graph consistency by exploiting inter-robot detections as additional constraints. This would help mitigate aggregated sensor noise, but also raises significant challenges in data association (e.g., ensuring that one robot's '*room*_5' is correctly matched with another's '*room*_2'), communication latency, and distributed optimisation under heterogeneous sensing conditions. Addressing these challenges will be key to making the architecture scalable to cooperative multi-robot exploration.

This work leaves many open research topics, most of which are already under way: (a) extending the system to handle non-rectangular rooms; (b) replacing room and door detection algorithms with more robust data-driven learnt functions; (c) using differentiable programming in the characterisation of object geometries so they can be adjusted online through the minimisation of a cost function; (d) incorporating additional concept classes (e.g., furniture, household objects) into the scene graph; and (e) and implementing a more robust loop closure mechanism based on visual descriptors and objects stored within rooms.

The final goal is to demonstrate that concept-first architectures can provide transparent, human-interpretable spatial representations that support both autonomous robot operation and effective human-robot collaboration in real-world environments. This exploration represents an initial step toward cognitive architectures that reason about space using human-meaningful concepts rather than metric primitives, potentially enabling more natural and explainable robotic spatial intelligence.

Author Contributions: Conceptualization, P.B., N.J.Z.C., and G.P.; methodology, N.J.Z.C. and G.P.; software, G.P., A.T., and N.J.Z.C.; validation, G.P., N.J.Z.C., and P.B.; formal analysis, N.J.Z.C. and P.N.; investigation, N.J.Z.C.; resources, N.J.Z.C.; data curation, N.J.Z.C. and G.P.; writing—original draft preparation, P.B., A.T., N.J.Z.C. and G.P.; writing—review and editing, P.N. and P.B.; visualization, A.T.; supervision, P.N. and P.B.; project administration, P.N.; funding acquisition, P.B. and P.N. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially funded by the FEDER Project 0124_EUROAGE_MAS_4_E (2021–2027 POCTEP Program) and by the Spanish Ministry of Science and Innovation (PDC2022-133597-C41). It has also been co-financed (85%) by the European Union through the European Regional Development Fund and the Regional Government of Extremadura. Managing Authority: Spanish Ministry of Finance. Grant reference: GR24169.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy and ethical restrictions.

Acknowledgments: We want to express our gratitude to Lucas Bonilla for his outstanding work in the implementation of the system.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Armeni, I.; He, Z.Y.; Gwak, J.; Zamir, A.R.; Fischer, M.; Malik, J.; Savarese, S. 3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera. 2019. Available online: http://arxiv.org/abs/1910.02527 (accessed on 16 September 2025).
2. Hughes, N.; Chang, Y.; Hu, S.; Talak, R.; Abdulhai, R.; Strader, J.; Carlone, L. Foundations of Spatial Perception for Robotics: Hierarchical Representations and Real-time Systems. *Int. J. Robot. Res.* **2024**, *43*, 1457–1505. [CrossRef]
3. Wu, S.C.; Wald, J.; Tateno, K.; Navab, N.; Tombari, F. SceneGraphFusion: Incremental 3D Scene Graph Prediction from RGB-D Sequences. 2021. Available online: https://arxiv.org/abs/2103.14898 (accessed on 16 September 2025).
4. Huang, S.; Qi, S.; Zhu, Y.; Xiao, Y.; Xu, Y.; Zhu, S.C. Holistic 3D Scene Parsing and Reconstruction from a Single RGB Image. 2018. Available online: http://arxiv.org/abs/1808.02201 (accessed on 16 September 2025).
5. Liu, X.; Zhao, Y.; Zhu, S.C. Single-View 3D Scene Reconstruction and Parsing by Attribute Grammar. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 710–725. [CrossRef] [PubMed]
6. Rosinol, A.; Violette, A.; Abate, M.; Hughes, N.; Chang, Y.; Shi, J.; Gupta, A.; Carlone, L. Kimera: From SLAM to Spatial Perception with 3D Dynamic Scene Graphs. 2021. Available online: http://arxiv.org/abs/2101.06894 (accessed on 16 September 2025).
7. Hughes, N.; Chang, Y.; Carlone, L. Hydra: A Real-time Spatial Perception System for 3D Scene Graph Construction and Optimization. *arXiv* **2022**, arXiv:2201.13360. [CrossRef]
8. Bustos, P.; Manso, L.; Bandera, A.; Bandera, J.; García-Varea, I.; Martínez-Gómez, J. The CORTEX cognitive robotics architecture: Use cases. *Cogn. Syst. Res.* **2019**, *55*, 107–123. [CrossRef]
9. Bustos García, P.; García, J.C.; Cintas Peña, R.; Martinena Guerrero, E.; Bachiller Burgos, P.; Núñez Trujillo, P.; Bandera, A. DSRd: A Proposal for a Low-Latency, Distributed Working Memory for CORTEX. In Proceedings of the Advances in Physical Agents II, Madrid, Spain, 19–20 November 2020; Springer Nature: Cham, Switzerland, 2021; pp. 109–122.
10. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [CrossRef]
11. Mur-Artal, R.; Montiel, J.M.M.; Tardós, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
12. Salas-Moreno, R.F.; Newcombe, R.A.; Strasdat, H.; Kelly, P.H.; Davison, A.J. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 1352–1359. [CrossRef]
13. Bowman, S.L.; Atanasov, N.; Daniilidis, K.; Pappas, G.J. Probabilistic data association for semantic SLAM. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1722–1729. [CrossRef]
14. McCormac, J.; Handa, A.; Davison, A.J.; Leutenegger, S. SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks. In Proceedings of the 2017 IEEE International Conference on Robotics and automation (ICRA), Singapore, 29 May–3 June 2017.
15. Narayana, M.; Kolling, A.; Nardelli, L.; Fong, P. Lifelong update of semantic maps in dynamic environments. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021.
16. Santos, J.M.; Krajník, T.; Duckett, T. Spatio-temporal exploration strategies for long-term autonomy of mobile robots. *Robot. Auton. Syst.* **2017**, *88*, 116–126. [CrossRef]
17. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
18. Narita, G.; Seno, T.; Ishikawa, T.; Kaji, Y. PanopticFusion: Online Volumetric Semantic Mapping at the Level of Stuff and Things. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019.
19. Bharati, P.; Pramanik, A. Deep Learning Techniques—R-CNN to Mask R-CNN: A Survey. In *Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2019*; Das, A.K., Nayak, J., Naik, B., Pati, S.K., Pelusi, D., Eds.; Springer Nature: Singapore, 2020; pp. 657–668.
20. Chen, K.; Zhang, J.; Liu, J.; Tong, Q.; Liu, R.; Chen, S.; Chen, K.; Xiao, J.; Liu, J.; Tong, Q.; et al. Semantic Visual Simultaneous Localization and Mapping: A Survey. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 7426–7449. [CrossRef]
21. Xia, L.; Cui, J.; Shen, R.; Xu, X.; Gao, Y.; Li, X. A survey of image semantics-based visual simultaneous localization and mapping: Application-oriented solutions to autonomous navigation of mobile robots. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1729881420919185. [CrossRef]
22. Rosinol, A.; Abate, M.; Chang, Y.; Carlone, L. Kimera: An Open-Source Library for Real-Time Metric-Semantic Localization and Mapping. 2020. Available online: https://arxiv.org/abs/1910.02490 (accessed on 16 September 2025).

23. Bavle, H.; Sanchez-Lopez, J.L.; Shaheer, M.; Civera, J.; Voos, H. Situational graphs for robot navigation in structured indoor environments. *IEEE Robot. Autom. Lett.* **2022**, *7*, 9107–9114. [CrossRef]
24. Hossein Pouraghdam, M.; Saadatseresht, M.; Rastiveis, H.; Abzal, A.; Hasanlou, M. Building floor plan reconstruction from slam-based point cloud using ransac algorithm. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2019**, *42*, 483–488. [CrossRef]
25. Murali, S.; Speciale, P.; Oswald, M.R.; Pollefeys, M. Indoor Scan2BIM: Building information models of house interiors. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 6126–6133. [CrossRef]
26. Han, J.; Rong, M.; Jiang, H.; Liu, H.; Shen, S. Vectorized indoor surface reconstruction from 3D point cloud with multistep 2D optimization. *ISPRS J. Photogramm. Remote Sens.* **2021**, *177*, 57–74. [CrossRef]
27. Khanal, B.; Rijal, S.; Awale, M.; Ojha, V. Structure-Preserving Planar Simplification for Indoor Environments. 2024. Available online: http://arxiv.org/abs/2408.06814 (accessed on 16 September 2025).
28. Wang, Q.; Zhu, Z.; Chen, R.; Xia, W.; Yan, C. Building Floorplan Reconstruction Based on Integer Linear Programming. *Remote Sensing* **2022**, *14*, 4675. [CrossRef]
29. Zou, C.; Colburn, A.; Shan, Q.; Hoiem, D. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2051–2059.
30. Liu, C.; Wu, J.; Furukawa, Y. Floornet: A unified framework for floorplan reconstruction from 3d scans. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 201–217.
31. Sun, C.; Hsiao, C.W.; Sun, M.; Chen, H.T. HorizonNet: Learning Room Layout with 1D Representation and Pano Stretch Data Augmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Beach, CA, USA, 15–20 June 2019.
32. Burgard, W.; Fox, D.; Thrun, S. Active mobile robot localization. In Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI 97), Nagoya, Japan, 23–29 August 1997; pp. 1346–1352.
33. Placed, J.A.; Strader, J.; Carrillo, H.; Atanasov, N.; Indelman, V.; Carlone, L.; Castellanos, J.A. A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers. *IEEE Trans. Robot.* **2023**, *39*, 1686–1705. [CrossRef]
34. Bourgault, F.; Makarenko, A.; Williams, S.; Grocholsky, B.; Durrant-Whyte, H. Information based adaptive robotic exploration. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 30 September–4 October 2002; Volume 1, pp. 540–545. [CrossRef]
35. Lluvia, I.; Lazkano, E.; Ansuategi, A. Active mapping and robot exploration: A survey. *Sensors* **2021**, *21*, 2445. [CrossRef]
36. Shaheer, M.; Millan-Romera, J.A.; Bavle, H.; Sanchez-Lopez, J.L.; Civera, J.; Voos, H. Graph-based global robot localization informing situational graphs with architectural graphs. In Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Detroit, MI, USA, 1–5 October 2023; pp. 9155–9162.
37. Bavle, H.; Sanchez-Lopez, J.L.; Cimarelli, C.; Tourani, A.; Voos, H. From SLAM to Situational Awareness: Challenges and Survey. *Sensors* **2023**, *23*, 4849. [CrossRef] [PubMed]
38. Millán Romera, J.A.; Bavle, H.; Shaheer, M.; Oswald, M.R.; Voos, H.; Sánchez López, J.L. Better Situational Graphs by Inferring High-level Semantic-Relational Concepts. *arXiv* **2023**, arXiv:2310.00401.
39. Bajcsy, R.; Aloimonos, Y.; Tsotsos, J.K. Revisiting Active Perception. *arXiv* **2016**. Available online: http://arxiv.org/abs/1603.02729 (accessed on 16 September 2025).
40. Werby, A.; Huang, C.; Büchner, M.; Valada, A.; Burgard, W. Hierarchical Open-Vocabulary 3D Scene Graphs for Language-Grounded Robot Navigation. In Proceedings of the Robotics: Science and Systems XX. Robotics: Science and Systems Foundation, RSS2024, online, 15–19 July 2024. [CrossRef]
41. Koch, S.; Vaskevicius, N.; Colosi, M.; Hermosilla, P.; Ropinski, T. Open3DSG: Open-Vocabulary 3D Scene Graphs from Point Clouds with Queryable Objects and Open-Set Relationships. 2024. Available online: http://arxiv.org/abs/2402.12259 (accessed on 16 September 2025).
42. Chang, H.; Boyalakuntla, K.; Lu, S.; Cai, S.; Jing, E.; Keskar, S.; Geng, S.; Abbas, A.; Zhou, L.; Bekris, K.; et al. Context-Aware Entity Grounding with Open-Vocabulary 3D Scene Graphs. 2023. Available online: http://arxiv.org/abs/2309.15940 (accessed on 16 September 2025).
43. Gu, Q.; Kuwajerwala, A.; Morin, S.; Jatavallabhula, K.M.; Sen, B.; Agarwal, A.; Rivera, C.; Paul, W.; Ellis, K.; Chellappa, R.; et al. ConceptGraphs: Open-Vocabulary 3D Scene Graphs for Perception and Planning. 2023. Available online: http://arxiv.org/abs/2309.16650 (accessed on 16 September 2025).
44. Zhang, C.; Delitzas, A.; Wang, F.; Zhang, R.; Ji, X.; Pollefeys, M.; Engelmann, F. Open-Vocabulary Functional 3D Scene Graphs for Real-World Indoor Spaces. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 11–15 June 2025; pp. 19401–19413.

45. García, J.C.; Núñez, P.; Bachiller, P.; Bustos, P. Towards the design of efficient and versatile cognitive robotic architecture based on distributed, low-latency working memory. In Proceedings of the ICARSC, Santa Maria da Feira, Portugal, 29–30 April 2022.
46. Torrejón, A.; Zapata, N.; Bonilla, L.; Bustos, P.; Núñez, P. Design and Development of Shadow: A Cost-Effective Mobile Social Robot for Human-Following Applications. *Electronics* **2024**, *13*, 3444. [CrossRef]