Project Group/Seminar/Lab

WT/ST 2025

Language-Grounded Dynamic Scene Graphs for Interactive Object Search with Mobile Manipulation

(Critical Analysis Report on MoMa-LLM)

Pratik Adhikari

Matriculation No: 9046973

Supervisor: Rohit Menon

December 26, 2025

Abstract

Large language models (LLMs) have recently been adopted as high-level planners for robotic agents. However, most existing work focuses either on navigation or manipulation in isolation and assumes a fully explored environment. The paper *Language-Grounded Dynamic Scene Graphs for Interactive Object Search with Mobile Manipulation* by Honerkamp *et al.* introduces MoMa-LLM, an approach that uses a dynamically updated scene graph to ground an LLM in partially observable household environments. The proposed system formulates interactive object search as a POMDP and constructs a two-level scene graph consisting of rooms and objects. It computes a navigational graph from a BEV occupancy map using a generalized Voronoi diagram, segments it into rooms via a door-based kernel density estimate and assigns objects to rooms through a distance-weighted path formulation. High-level actions such as *navigate*, *open* and *explore* are mapped to low-level policies and embedded in a structured language prompt that guides the LLM. To evaluate performance, the authors introduce a search efficiency curve and its area under the curve (AUC-E) to capture success as a function of interaction cost. This seminar report critically analyses the problem formulation, algorithmic design and experimental results of MoMa-LLM and discusses its contributions and limitations in the context of mobile manipulation.

Contents

1 Introduction

1.1 Intention and how to read this document

This report is a *technical* and *verification-friendly* walkthrough of the MoMa-LLM paper by Honerkamp et al. It is written with two simultaneous goals:

- **Engineering comprehension:** explain what the system does (representations, modules, interfaces, failure modes) well enough that a robotics engineer could re-implement a simplified version.
- **Research positioning:** state precisely what MoMa-LLM contributes relative to work *before* it, and how later papers extend or deviate from the same design space.

To force traceability, we attach a *highlighted verification box* to each citation. The box states the **page**, an approximate **paragraph index**, and a short **anchor phrase** that should be visible on that page. Example: [p. 2, ¶1; "POMDP"].

The main body follows the seminar template: *Introduction*, *Related Work*, *Paper Summary*, *Discussion*, *Conclusion*. A large appendix provides a *self-contained math derivation* for the core constructs used in MoMa-LLM (POMDP, distance fields, Voronoi graphs, KDE for door inference, and evaluation metrics). [1, p. 1] [p. 1, Abstract, ¶1; "Language Grounded Dynamic Scene Graphs"]

1.2 Problem setting: interactive object search for mobile manipulation

Interactive object search differs from "go-to-a-visible-goal" navigation: the robot must *actively gather information* (open doors, move to viewpoints) while simultaneously planning navigation and manipulation. Formally, the target object location is initially uncertain and becomes known only through exploration and interaction. MoMa-LLM explicitly frames the task as a *partially observable* sequential decision problem. [1, p. 2] [p. 2, Sec. III, ¶1; "POMDP"]

In household environments, the search space is structured by *rooms* and *connectivity through doors*; many objects are *occluded* or *inside containers*. This pushes the robot beyond purely geometric mapping toward representations that connect geometry (free space, traversability) and semantics (rooms, objects, affordances). [1, p. 1] [p. 1, Abstract, ¶1; "Interactive object search"]

1.3 Why MoMa-LLM matters

Before MoMa-LLM, two lines of work were developing rapidly: (i) LLM-centric planning (language-to-actions, tool selection, affordance grounding), and (ii) geometric/semantic mapping for navigation and manipulation.

LLM-based robotics systems such as SayCan emphasize *grounding* language in feasible action choices (rather than treating the LLM as an oracle). [2, p. 1] [p. 1, Title/Abstract, ¶1; "Gr In manipulation, VoxPoser demonstrates that language can be tied to *composable value maps* in 3D. [3, p. 1] [p. 1, Title, ¶1; "Composable 3D Value Maps"]

In mapping, systems like Hydra focus on building and optimizing *3D scene graphs* in real time, enabling downstream reasoning. [4, p. 1] [p. 1, Title, ¶1; "3D Scene Graph Constructio MoMa-LLM is interesting because it explicitly tries to *close the loop* for object search: it maintains a dynamic scene graph, uses an LLM to propose high-level actions conditioned on that structured state, and executes low-level navigation/manipulation policies to gather new evidence and update the graph. [1, p. 1] [p. 1, Abstract, ¶1; "constructs a two-level policy"]

1.4 What you should be able to answer after reading

If you cannot answer these, you did not read carefully (and you should not pretend you did):

1. What are the *state variables* MoMa-LLM maintains, and what is observed vs. latent?
2. What is the exact role of the *Voronoi graph* in the system (why not a standard nav graph)?
3. How are *rooms* inferred from geometry (doors), and how are objects assigned to rooms?
4. What is the MoMa-LLM *high-level action space*, and how does it connect to low-level skills?
5. How is success measured? What does *AUC-E* capture that SPL does not?

Each of these is answered in Sec. 3 and derived in Sec. A. [1, p. 6] [p. 6, Sec. V, ¶2; "AUC-E"]

2 Related Work

This section is deliberately organized by *design choice*, not by citation-count. The goal is to locate MoMa-LLM in a space of alternatives: representation (maps vs. graphs), planning (reactive vs. deliberative), and grounding (text-only vs. structured state).

2.1 LLM-grounded robotics: from language to feasible actions

A recurring failure mode of naive "LLM plans" is hallucinated actions that are physically impossible or unsafe. SayCan is an influential response: it combines language models with *affordance* or feasibility scores so that language suggestions are filtered by what the robot can actually do. [2, p. 1] [p. 1, Abstract, ¶1; "Do As I Can"]

VoxPoser extends grounding to continuous 3D manipulation by representing language-conditioned goals as *value maps* that can be composed for more complex tasks. [3, p. 1] [p. 1, Abstract, ¶1; "Composable 3D Value Maps"]

MoMa-LLM adopts the same philosophical stance (LLM as a high-level proposer, not a low-level controller), but differs in what it supplies to the LLM: a *dynamic scene graph + geometric connectivity* rather than raw text. [1, p. 3] [p. 3, Sec. III-B, ¶1; "scene gra

2.2 Scene graphs and semantic mapping for mobile manipulation

The mapping community has a long history of using layered representations: metric maps for navigation and symbolic/semantic structures for reasoning. Hydra is a prominent real-time system for constructing and optimizing 3D scene graphs, serving as infrastructure for downstream tasks. [4, p. 1] [p. 1, Title, ¶1; "Real-time Spatial Perception System"]

MoMa-LLM can be interpreted as a task-driven specialization: it builds a scene graph that is *just rich enough* for interactive object search (rooms, doors, object candidates), while maintaining a navigation graph derived from distance-field structure (Voronoi). [1, p. 4] [p. 4, Sec. III-C, ¶2; "Voronoi"]

2.3 Interactive object search and hierarchical policies

Learning-based interactive search for mobile manipulation often uses hierarchical structures: a high-level selector chooses *which room* or *which container* to inspect; a low-level policy executes navigation and manipulation.

For example, HIMOS studies *hierarchical* interactive multi-object search for mobile manipulation and emphasizes the need to reason about interactions (e.g., opening) rather than only traversing free space. [5, p. 1] [p. 1, Title/Abstract, ¶1; "Interactive Multi-Object

MoMa-LLM shares the hierarchical flavor, but the high-level decision is produced by an LLM conditioned on the current graph state, not by a learned policy

trained end-to-end for the specific environment. [1, p. 3] [p. 3, Sec. III-A, ¶2; "two-level policy"]

2.4 Post-MoMa (and parallel) directions: LLM navigation and object search

Several recent papers attack similar problems (language-conditioned navigation/search), often with different representations or assumptions:

Zero-shot object navigation with LLM priors. VoroNav proposes Voronoi-based zero-shot object navigation assisted by an LLM, emphasizing geometry-aware planning combined with language priors. [6, p. 1] [p. 1, Title/Abstract, ¶1; "Voronoi-based Zero-shot

Language-guided navigation with state summaries. SayNav focuses on language-instructed navigation using LLM-driven reasoning over summarized state. [7, p. 1] [p. 1, Title, ¶1; "SayNav"]

Online scene-graph updates for navigation. OrionNav highlights online updates of a scene representation during navigation and uses this structure for reasoning about goals. [8, p. 1] [p. 1, Title/Abstract, ¶1; "OrionNav"]

Memory and retrieval mechanisms. MORE emphasizes memory (storage and retrieval) as a key ingredient for long-horizon robot behavior. [9, p. 1] [p. 1, Title/Abstract, ¶1; "Mem

Semantic reasoning for concealed spaces. StretchAI targets search and retrieval in *dynamic and concealed spaces*, explicitly confronting the "behind/inside" problem that dominates household search. [10, p. 1] [p. 1, Title/Abstract, ¶1; "dynamic and concealed space

These works provide a useful lens for discussion: if MoMa-LLM's main idea is *structured state + LLM high-level action selection*, then later work can be categorized by what they strengthen: (i) better state estimation/memory, (ii) better exploration policies, (iii) better handling of dynamics and concealment, and (iv) more robust grounding. [1, p. 11] [p. 11, Conclusion, ¶1; "future work"]

2.5 Comparison table

Table 1 intentionally uses *binary* columns to force clarity. If a paper only hints at a feature but does not operationalize it, it is marked as "partial".

Work	LLM used	Structured state	Interactive (open-loop)	Zero-shot	Primary task
SayCan [2, p. 1] [p. 1, Title, ¶1; “Do As I Can”]	yes	partial	yes	partial	language-to-skills
VoxPoser [3, p. 1] [p. 1, Title, ¶1; “Composable 3D Value Maps”]	yes	yes (3D values)	yes	partial	manipulation
HIMOS [5, p. 1] [p. 1, Title, ¶1; “Interactive Multi-Object Search”]	no	partial	yes	n/a	interactive search
MoMa-LLM [1, p. 1] [p. 1, Abstract, ¶1; “MoMa-LLM”]	yes	yes (DSG+Voronoi)	yes	partial	interactive search
VoroNav [6, p. 1] [p. 1, Title, ¶1; “VoroNav”]	yes	yes (Voronoi)	no/partial	partial	object nav-igation
SayNav [7, p. 1] [p. 1, Title, ¶1; “SayNav”]	yes	partial	no	partial	navigation
OrionNav [8, p. 1] [p. 1, Title, ¶1; “OrionNav”]	yes	yes (online graph)	partial	partial	navigation
MORE [9, p. 1] [p. 1, Title, ¶1; “Memory-Augmented”]	yes	yes (memory)	partial	partial	long-horizon tasks
StretchAI [10, p. 1] [p. 1, Title, ¶1; “dynamic and concealed spaces”]	yes	yes (semantics)	yes	partial	search & re-trieval

Table 1: High-level comparison of MoMa-LLM and adjacent work.

3 Paper Summary

This chapter summarizes the MoMa-LLM pipeline in a *systems* style: what state is maintained, how it is updated, what the LLM sees, what actions it can choose, and how success is measured.

3.1 Problem formulation

MoMa-LLM models interactive object search as a POMDP. The robot maintains an internal belief over object locations while receiving partial observations as it navigates and interacts (e.g., opening doors). [1, p. 2] [p. 2, Sec. III, ¶1; “POMDP”]

Key implication. Because the state is partially observed, the planner must trade off *information gathering* and *goal reaching*. In MoMa-LLM, this trade is largely made at the high level via LLM-selected actions such as "search room A" or "open door D", while low-level controllers provide navigation/manipulation execution. [1, p. 3] [p. 3, Sec. III-A, ¶2; "high-level action"]

3.2 State representation: maps + dynamic scene graph

MoMa-LLM maintains two coupled representations:

- A **metric map** (voxel/occupancy) used to compute traversability, distance fields, and a navigation graph.
- A **dynamic scene graph (DSG)** storing rooms, doors, and object hypotheses with semantic labels and spatial relations.

The paper's emphasis is that *structured state* reduces hallucination: the LLM is constrained to propose actions that reference entities present in the graph (known rooms/doors/objects), rather than inventing arbitrary options. [1, p. 3] [p. 3, Sec. III-B, ¶1; "dynamic sc

3.3 From geometry to connectivity: ESDF, GVD, and Voronoi graph

A nontrivial and easy-to-miss part of the pipeline is how the navigation graph is built. Instead of only using an occupancy grid + A* over cells, MoMa-LLM computes a Euclidean Signed Distance Field (ESDF), then a Generalized Voronoi Diagram (GVD), then extracts a graph of topological connectivity. [1, p. 4] [p. 4, Sec. III-C, ¶2; "Voronoi gra

Why this matters. A Voronoi-derived graph tends to favor paths that maximize clearance from obstacles (useful for mobile manipulators in clutter) and yields a compact topological skeleton for reasoning about room-to-room connectivity. The math of ESDF/GVD/Voronoi extraction is derived in Sec. B. [1, p. 4] [p. 4, Sec. III-C, ¶2; "distance field"]

3.4 Room segmentation and door inference

MoMa-LLM needs to reason at the level of *rooms*. The paper treats doors as key separators and estimates door locations using observations and a kernel density estimation (KDE) procedure. [1, p. 5] [p. 5, Sec. III-D, ¶1; "door locations"]

Outcome. The DSG includes room nodes and door nodes. Room adjacency is derived from which rooms are connected via inferred doors. This adjacency is what the LLM later uses when it proposes which room to search next. [1, p. 5] [p. 5, Sec. III-D, ¶2; "connected rooms"]

3.5 Assigning objects to rooms

Given candidate object detections in the map, MoMa-LLM assigns each object to the most plausible room by minimizing a cost that combines geometric distance-to-room via the connectivity graph. The paper provides an explicit cost definition (see Sec. D). [1, p. 5] [p. 5, Sec. III-E, ¶1; "distance-weighted cost"]

3.6 High-level action space and low-level skills

The system defines a discrete high-level action set operating over graph entities, such as:

- Navigate to a frontier / viewpoint
- Open a door
- Search a particular room
- Attempt grasp / retrieval of a detected target

The LLM selects among these actions using a prompt that includes the current DSG state, the room connectivity, and the task goal. The selected action is then executed by conventional navigation and manipulation modules. [1, p. 3] [p. 3, Sec. III-A, ¶2; "two-level policy"]

3.7 LLM prompting: what information is exposed

A critical engineering detail is the *interface* between the structured state and the LLM. The paper describes how it converts the DSG (rooms, doors, object candidates) into a text representation and asks the LLM to output an action selection. The prompt is therefore a *serialization* of the current belief/state, not raw sensor data. [1, p. 6] [p. 6, Sec. IV, ¶1; "prompt"]

3.8 Belief, observations, and DSG updates (practical view)

A POMDP formulation is only meaningful if you can point to what the system treats as *belief*. In MoMa-LLM, the belief is distributed across: (i) occupancy/free-space estimates in the map, (ii) probabilistic door hypotheses from KDE peaks, (iii) object hypotheses and their room assignments, and (iv) the current "search status" history encoded in the prompt (what has been tried, what failed). [1, p. 2] [p. 2, Sec. III, ¶1; "POMDP"]

Update loop. After executing an action (navigate, open, observe), new sensor data update the metric map. Perception modules propose objects/doors; these are integrated into the DSG (new nodes, updated attributes). Then the high-level policy is queried again with the updated serialized state. This loop is the essence of "dynamic" in dynamic scene graph. [1, p. 3] [p. 3, Sec. III-B, ¶1; "dynamic scene graph"]

3.9 Action selection constraints and parsing

MoMa-LLM does not ask the LLM to output free-form plans. Instead, it provides a *closed* action space (parameterized by graph entities) and asks the LLM to select one. This is a crucial safety/robustness trick: it bounds what the LLM can command. [1, p. 3] [p. 3, Sec. III-A, ¶2; "action space"]

Practical implementation note. In an implementation, you typically parse the LLM output into a structured command (e.g., JSON or a fixed grammar). If parsing fails or the LLM proposes an invalid entity, you fall back to a safe default (e.g., explore frontier). The paper's description implies this style of constrained selection even if some parsing details are abstracted. [1, p. 6] [p. 6, Sec. IV, ¶1; "prompt"]

3.10 Low-level navigation execution on the Voronoi graph

Once a high-level goal is chosen (e.g., "go to door D"), the system needs a concrete path. The Voronoi graph provides a compact set of waypoints; shortest paths on this graph serve as the navigation backbone. This is conceptually different from grid A*: the graph already encodes free-space topology and clearance. [1, p. 4] [p. 4, Sec. III-C, ¶2; "Voronoi graph"]

3.11 Baselines and ablations (what must be compared)

When reading the results, you should ask: *which component caused the gain?* The paper evaluates MoMa-LLM against baselines and discusses the contribution of structural components like room reasoning and the metric/graph backbone. Even if you disagree with the baselines, the comparison strategy (remove/replace one component at a time) is the right instinct. [1, p. 6] [p. 6, Sec. V, ¶1; "baselines"]

3.12 Evaluation: success, efficiency, and AUC-E

MoMa-LLM evaluates both *success* and *efficiency*. Beyond standard metrics like success rate (SR) and SPL, it introduces an "efficiency curve" and computes the area under that curve (AUC-E). [1, p. 6] [p. 6, Sec. V, ¶2; "AUC-E"]

Intuitively, AUC-E rewards methods that find the target *quickly* (few steps, few interactions), not only eventually. The appendix derives AUC-E with a toy example so you can compute it by hand and stop hand-waving. [1, p. 6] [p. 6, Sec. V, ¶2; "efficiency curve"]

3.13 Results and reported takeaways

The paper reports gains over baselines in both simulated and real environments, and highlights that the combination of DSG structure and LLM planning helps reduce failure modes where the robot repeatedly searches irrelevant areas. [1, p. 7] [p. 7, Sec. V-C, ¶1; "real-world experiments"]

What to be skeptical about. As with many LLM-in-the-loop systems, performance depends on the quality of perception (object detections, door observations), the prompt format, and the chosen action set. These dependencies are not "details"; they are the system. Limitations are discussed in Sec. 4 with specific citations. [1, p. 11] [p. 11, Conclusion, ¶1; "limitations"]

4 Discussion

This discussion is intentionally *comparative*. If you only list strengths and weaknesses in isolation, you learn nothing.

4.1 What MoMa-LLM contributes (bullet summary)

The core contributions of MoMa-LLM can be stated crisply as:

1. **A hybrid representation for search:** dynamic scene graph for semantics + Voronoi-based navigation graph for connectivity and safe traversal. [1, p. 4] [p. 4, Sec. III-C, ¶2; "Voronoi graph"]
2. **LLM as a high-level policy:** the LLM operates on a serialized structured state and selects among a predefined action set. [1, p. 3] [p. 3, Sec. III-A, ¶2; "two-level policy"]
3. **Room-centric reasoning:** explicit inference of doors/rooms and assignment of objects to rooms, so search can be planned over rooms rather than raw coordinates. [1, p. 5] [p. 5, Sec. III-D, ¶1; "door"]
4. **An efficiency-oriented metric (AUC-E):** evaluates not just success but how rapidly success is achieved over a budget. [1, p. 6] [p. 6, Sec. V, ¶2; "AUC-E"]

4.2 Why these were needed (comparison to pre-MoMa work)

Before MoMa-LLM, LLM-grounded robotics often suffered from *underspecified state*. SayCan mitigates hallucination by filtering with affordance scores, but does not by itself provide a rich spatial-semantic map for object search. [2, p. 1] [p. 1, Abstract, ¶1; "affordances"]

Similarly, hierarchical object search work (e.g., HIMOS) emphasizes interaction and hierarchical decision making, but is not designed around using an LLM as the high-level selector over a scene graph representation. [5, p. 1] [p. 1, Abstract, ¶1; "hierarchical"]

MoMa-LLM's design is therefore a specific synthesis: it takes the *grounded decision philosophy* of LLM robotics and plugs it into a *graph-structured world model* specialized for interactive object search. [1, p. 3] [p. 3, Sec. III-B, ¶1; "world model"]

4.3 How later papers extend the same design space

After MoMa-LLM (and in parallel), many systems push on the same weak points:

Better geometry/graph backbones. VoroNav keeps Voronoi structure but targets *zero-shot object navigation*, suggesting that Voronoi graphs are a reusable backbone for LLM-assisted spatial decision making. [6, p. 1] [p. 1, Title/Abstract, ¶1; "Voronoi-based"]

Online updates and richer state summaries. OrionNav emphasizes online updates of scene structure, which is essential when the environment changes during execution. [8, p. 1] [p. 1, Abstract, ¶1; "online"]

Memory as a first-class component. MORE argues that long-horizon autonomy needs explicit memory mechanisms; this is directly relevant to MoMa-LLM because object search often spans multiple rooms and revisits. [9, p. 1] [p. 1, Title/Abstract, ¶1; "Memory-A

Concealed spaces and interaction-heavy retrieval. StretchAI explicitly targets dynamic and concealed spaces, sharpening the claim that search success depends on reasoning about *inside/behind* relations and interactions, not only room-level navigation. [10, p. 1] [p. 1, Title, ¶1; "dynamic and concealed"]

4.4 Limitations (with citations that you can verify)

The paper itself notes several limitations, and additional ones follow logically from the design.

- **Perception dependency.** If door detections or object proposals are wrong, the DSG becomes a confident-but-wrong belief state. This is not an edge case; it is the dominant failure channel in cluttered homes. [1, p. 11] [p. 11, Conclusion, ¶1; "percept
- **Prompt and action-space brittleness.** The LLM can only select among the provided actions and only reason over the serialized state; redesigning either can change behavior substantially, making reproducibility tricky. [1, p. 6] [p. 6, Sec. IV, ¶1; "prompt"]
- **Room abstraction limits.** Assigning objects to rooms simplifies planning but can hide important sub-structure (containers, shelves) that drives real retrieval difficulty. [1, p. 5] [p. 5, Sec. III-E, ¶1; "assign"]
- **Dynamics and partial observability.** The DSG is dynamic, but the system does not fully solve non-stationarity (humans moving objects, doors closing). Later work increasingly treats dynamics and memory as explicit problems. [9, p. 1] [p. 1, Abstract, ¶1; "memory"]

4.5 A ruthless sanity check: what would you implement first?

If you want to *learn* from MoMa-LLM, do not start by calling an LLM API. Start by implementing:

1. ESDF → GVD → Voronoi graph extraction on a simple occupancy grid.
2. Door KDE from a set of noisy "door observations" in 2D.

3. Room graph + object-to-room cost assignment.

4. AUC-E computation from a small set of episode traces.

The appendix gives minimal derivations and toy examples for each, so you can validate your implementation against hand calculations. [1, p. 4] [p. 4, Sec. III-C, ¶2; “ESDF”]

5 Conclusion

MoMa-LLM is best understood as a *world-model interface* between robotics and LLM reasoning: it constructs a task-relevant structured state (dynamic scene graph + Voronoi connectivity) and uses an LLM to choose high-level actions that drive navigation and interaction during object search. [1, p. 1] [p. 1, Abstract, ¶1; "two-level policy"]

The main intellectual move is not "LLMs are powerful"; it is that *structured state* enables controllable, verifiable decision making, and that interactive object search benefits from room- and connectivity-level abstraction. [1, p. 5] [p. 5, Sec. III-D, ¶1; "rooms"]

Future work, as evidenced by adjacent literature, pushes toward richer memory, stronger handling of dynamics and concealed spaces, and more robust online map/graph updates. [10, p. 1] [p. 1, Abstract, ¶1; "dynamic and concealed"]

The appendix is not optional: if you cannot derive the core math objects used in the paper, you are not understanding the method—you are memorizing words. [1, p. 2] [p. 2, Sec. III, ¶1; "POMDP"]

A Appendix: Mathematical Walkthrough (Newbie-Friendly)

This appendix is intentionally *self-contained*. It explains the core mathematical objects used in MoMa-LLM and its supplementary material, with small toy examples so that the symbols do not feel like "magic". The goal is not to re-derive SLAM or perception pipelines, but to make the paper's *evaluation* and *graph-based reasoning* readable if you are rusty with math.

A.1 Minimal notation you must not be sloppy about

We will use the following conventions:

- A **set** is written as $S = \{a, b, c\}$. Its size is $|S|$.
- A **sequence** indexed by time is written $(x_t)_{t=1}^{T}$ or simply $x_1, \dots, x_T$.
- A **graph** is $G = (V, E)$ where V is the set of nodes and E the set of edges.
- "Estimated" quantities get subscript e and "ground truth" quantities get subscript g.

If you do not distinguish sets vs sequences, and estimates vs ground truth, you will misread almost every metric in the supplementary.

A.2 What is a (dynamic) scene graph, mathematically?

At any time step t, the robot maintains a scene graph

$$G_t = (V_t, E_t).$$

The node set V_t typically includes multiple *types* of nodes: rooms, objects, and sometimes containers or articulated parts (e.g., a drawer). Each node $v \in V_t$ has attributes such as:

$$\text{label}(v) \in \mathcal{L}, \qquad \text{pose}(v) \in SE(3), \qquad \text{state}(v) \in \{\text{open}, \text{closed}, \text{unknown}\}.$$

Edges E_t encode relations, e.g. "object o is in room r" or spatial proximity.

Why "dynamic" matters. The subscript t is not decoration. As the robot explores, it discovers new rooms/objects, corrects earlier beliefs, and updates relations:

$$V_{t+1} \neq V_t, \quad E_{t+1} \neq E_t.$$

This is why evaluation often averages quantities over time (see Sec. A.4).

A.3 From continuous distances to discrete language tokens

Many systems avoid putting raw floating point distances into an LLM prompt. Instead, a measured distance $d \in \mathbb{R}_{\geq 0}$ is *binned* into a small number of language tokens.

One typical mapping (reported in the MoMa-LLM supplementary) is:

Table 2: Example: mapping a continuous distance into discrete language tokens.

Distance threshold ($\leq$)	Token
3.0	`veryclose`
10.0	`near`
20.0	`far`
∞	`distant`

Toy example. If $d = 7.2$ meters, then $3.0 < 7.2 \leq 10.0$, so the token is `near`. If $d = 25$ meters, it becomes `distant`. This discretization makes the prompt compact and reduces sensitivity to noisy metric estimates.

A.4 Room-segmentation precision and recall (supplementary Eqs. (4)–(7))

The paper evaluates how well an estimated room partition matches the ground-truth partition. Let

$$R_g = \{r_g^{(1)}, \ldots, r_g^{(m)}\}, \qquad R_e = \{r_e^{(1)}, \ldots, r_e^{(n)}\}$$

be the sets of ground-truth and estimated rooms, represented as sets of occupied grid cells.

The supplementary defines precision and recall as:

$$P = \frac{1}{|R_e|} \sum_{r_e \in R_e} \max_{r_g \in R_g} \frac{|r_g \cap r_e|}{|r_e|}, \tag{1}$$

$$R = \frac{1}{|R_g|} \sum_{r_g \in R_g} \max_{r_e \in R_e} \frac{|r_e \cap r_g|}{|r_g|}. \tag{2}$$

Interpretation (don't guess—read it).

- The inner fraction in Eq. (1) is *"how much of the estimated room is correct"*. You match each estimated room to the best ground-truth room.
- The inner fraction in Eq. (2) is *"how much of the ground-truth room is recovered"*. You match each ground-truth room to the best estimated room.

Tiny numeric example. Suppose there are two ground-truth rooms $R_g = \{A, B\}$ and two estimated rooms $R_e = \{\hat{A}, \hat{B}\}$. Assume each room is a set of 100 grid cells. If $\hat{A}$ overlaps A on 90 cells and overlaps B on 10 cells, then:

$$\max_{r_g \in R_g} \frac{|r_g \cap \hat{A}|}{|\hat{A}|} = \max\left(\frac{90}{100}, \frac{10}{100}\right) = 0.9.$$

Do this for $\hat{B}$, average over estimated rooms, and you get P. For R, you instead normalize by $|A|$ and $|B|$ (ground-truth room sizes).

Why average over time. Because G_t changes over exploration, the supplementary reports mean and standard deviation over $t = 1, \ldots, T$:

$$\bar{P} = \frac{1}{T}\sum_{t=1}^{T} P_t, \qquad \sigma_P = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(P_t - \bar{P})^2}, \tag{3}$$

$$\bar{R} = \frac{1}{T}\sum_{t=1}^{T} R_t, \qquad \sigma_R = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(R_t - \bar{R})^2}. \tag{4}$$

A.5 Interactive search: success, cost, and the efficiency curve

MoMa-LLM evaluates object search with a metric that explicitly trades off *success* against *effort*. Let c denote a cost budget (e.g., total traveled distance plus interaction cost). Define a success indicator:

$$\mathbf{1}[\text{success at cost } c] = \begin{cases} 1 & \text{if target found within budget } c, \\ 0 & \text{otherwise.} \end{cases}$$

Across multiple episodes, the **success rate at budget** c is the average of this indicator.

If you plot success rate as a function of budget, you obtain an **efficiency curve**:

$$S(c) \in [0, 1], \quad \text{for } c \in [0, c_{\max}].$$

A method that succeeds *only if you allow huge budgets* will look worse than a method that succeeds early.

A.6 AUC-E: area under the efficiency curve (with a toy trapezoid calculation)

AUC-E compresses the entire curve $S(c)$ into one number:

$$\text{AUC-E} = \frac{1}{c_{\max}} \int_0^{c_{\max}} S(c)\, dc.$$

In practice, you only have discrete samples $S(c_0), S(c_1), \ldots, S(c_K)$. A standard numerical approximation is the trapezoidal rule:

$$\int_0^{c_{\max}} S(c)\, dc \approx \sum_{k=0}^{K-1} \frac{S(c_k) + S(c_{k+1})}{2} (c_{k+1} - c_k).$$

Toy example. Suppose we evaluate at budgets $c = \{0, 1, 2, 3\}$ and observe success rates

$$S(0) = 0, \quad S(1) = 0.4, \quad S(2) = 0.7, \quad S(3) = 0.8.$$

Then the trapezoid areas are:

$$A_0 = \tfrac{0+0.4}{2}(1-0) = 0.2,\ A_1 = \tfrac{0.4+0.7}{2}(2-1) = 0.55,\ A_2 = \tfrac{0.7+0.8}{2}(3-2) = 0.75.$$

Total area ≈ 1.5. If $c_{\max} = 3$, then AUC-E $\approx 1.5/3 = 0.5$.

What you should learn from this. AUC-E rewards methods that reach high success *quickly*. It is harder to "game" than reporting success at one arbitrary budget.

A.7 High-level planning as a constrained decision process

It helps to view the system as a decision process:

$$s_t \xrightarrow{a_t} s_{t+1}.$$

Here the **state** s_t contains the current graph G_t plus robot pose and memory variables. The **action** a_t is chosen from a small, allowed set $\mathcal{A}$ (go-to, search-room, open, inspect, ...). The key idea is *constraint*: the LLM is not asked to output arbitrary robot commands; it must output an action from $\mathcal{A}$ with valid arguments.

Why this reduces hallucination. If the LLM can only choose actions that exist in the system and reference nodes that exist in G_t, you eliminate an entire class of invalid plans (e.g., "go to the imaginary kitchen" when no kitchen has been observed).

A.8 How prompt "serialization" relates to math

Even though the LLM consumes text, the text is a serialization of structured state. Conceptually, the prompt is a function:

$$p_t = f(G_t, \text{robot state}, \text{memory}).$$

Good serialization design is equivalent to good feature design: you choose what information is exposed to the planner, in what format, and at what granularity (e.g., distance tokens from Tab. 2).

A.9 Implementation note: low-level execution and failure interpretation

MoMa-LLM plans at the high level, but execution has continuous failure modes: collisions, unreachable handles, segmentation errors, etc. In practice, each action a_t has an execution policy π_{a_t} that runs until success/failure. When you see an "LLM failure" in results, ask first:

> Was it really reasoning, or did the low-level policy fail due to geometry/perception?

This distinction matters for future work: fixing perception is not the same as improving high-level reasoning.

A.10 Bonus math you will meet everywhere in robotics: $SE(2)$, $SE(3)$, and homogeneous transforms

Even if MoMa-LLM is "LLM-heavy", the underlying robot state still lives in rigid-body geometry. A robot pose in 3D is an element of the **special Euclidean group**:

$$SE(3) = \left\{ \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \,\middle|\, R \in SO(3),\ t \in \mathbb{R}^3 \right\}.$$

Here R is a 3×3 rotation matrix and t is a translation vector.

Composition. If frame A to frame B is T_{AB} and B to C is T_{BC}, then:

$$T_{AC} = T_{AB} T_{BC}.$$

This is *exactly* what you use when you transform an object pose observed in the camera frame into the map frame.

Toy example in 2D (easier to visualize). In $SE(2)$, a pose is a rotation by θ and translation (x, y):

$$T(\theta, x, y) = \begin{bmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{bmatrix}.$$

Take $\theta = 90^\circ$ $(\pi/2)$, $x = 1$, $y = 0$:

$$T = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Apply it to a point $p = (2, 0)$ in homogeneous coordinates $\tilde{p} = (2, 0, 1)^\top$:

$$\tilde{p}' = T\tilde{p} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \Rightarrow p' = (1, 2).$$

So the point was rotated left by 90 degrees and shifted right by 1.

A.11 The "Voronoi graph" idea in one page

MoMa-LLM (and related work) often plans navigation on a sparse graph rather than on every grid cell. The intuition is simple: you want a graph that stays in the *middle of free space*.

Voronoi diagram. Given obstacle points $\mathcal{O}$ in a plane, the Voronoi diagram partitions space into regions closest to each obstacle. The **Voronoi edges** are points that are equally far from (at least) two obstacles. Those edges form a graph-like skeleton: the *generalized Voronoi graph* (GVG).

Why it helps. If you drive along the GVG, you maximize clearance from obstacles, which is useful for long-range indoor navigation. Planning reduces to shortest path on a graph:

$$\text{path}^\star = \arg\min_{\text{path}} \sum_{(i,j)\in\text{path}} w_{ij},$$

where w_{ij} is edge length (or any weighted cost).

Toy example. Imagine a corridor: obstacles are the two walls. The Voronoi edge is the centerline of the corridor. The graph collapses a 2D free space into a 1D curve plus junctions.

A.12 Shortest path on a graph (Dijkstra in equations, not code)

Suppose you have a graph $G = (V, E)$ with nonnegative edge weights $w_{ij} \geq 0$. Dijkstra's algorithm computes the minimum-cost distance from a start node s to every node.

Define the optimal cost-to-come:

$$d(v) = \min_{\text{paths } s \to v} \sum_{(i,j) \in \text{path}} w_{ij}.$$

The algorithm maintains a set of "settled" nodes whose $d(\cdot)$ is final. The key update (relaxation) step is:

$$d(j) \leftarrow \min\big(d(j),\ d(i) + w_{ij}\big),$$

for every edge (i, j) out of the current node i.

Why you care. Every time MoMa-LLM chooses "go to room X", the low-level planner is essentially solving a shortest-path problem (maybe with additional constraints).

A.13 Search as expected cost minimization (a clean mental model)

A clean mathematical picture of object search is:

> Pick the next place to look to minimize expected remaining cost.

Let the object be in one of m locations $\ell \in \{\ell_1, \ldots, \ell_m\}$ (rooms, containers, etc.). Let $p_t(\ell)$ be your belief at time t. If you choose to check location ℓ, you pay a cost $c(\ell)$ (travel + interaction), and you succeed with probability $p_t(\ell)$.

A one-step greedy expected cost criterion can be written as:

$$\text{score}(\ell) = \frac{p_t(\ell)}{c(\ell)} \quad \Rightarrow \quad \ell^\star = \arg\max_{\ell} \text{score}(\ell).$$

This is not what MoMa-LLM explicitly optimizes, but it explains why the graph context and distance tokens are valuable: they help the planner implicitly trade off probability vs effort.

Tiny example. Two candidate drawers: A has $p = 0.6$ and cost $c = 30$, B has $p = 0.4$ and cost $c = 10$. Then p/c gives $A : 0.02$ vs $B : 0.04$, so you check B first even though A is more likely, because it is much cheaper.

A.14 Belief update when you fail to find the object

If you check a location ℓ and do *not* find the object, you should reduce its probability and renormalize:

$$p_{t+1}(\ell) = 0, \qquad p_{t+1}(\ell') = \frac{p_t(\ell')}{1 - p_t(\ell)} \quad \text{for } \ell' \neq \ell.$$

This is the simplest possible update rule and already explains why "memory" (objects already checked) matters.

Toy example. If $p_t(A) = 0.3$, $p_t(B) = 0.7$, you check A and fail, then $p_{t+1}(B) = 0.7/(1 - 0.3) = 1.0$. So after exhausting A, the belief collapses onto B.

A.15 How this connects back to MoMa-LLM

MoMa-LLM does not run explicit Bayesian filtering over container priors in the paper. Instead, it uses a structured prompt (graph + memory) so that the LLM can apply human priors and update plans after new observations. But if you keep the simple belief-update equations above in mind, you will read the qualitative behaviours in the results section with much more clarity.

A.16 Optional deepening: MDP view and Bellman equations (so RL papers stop scaring you)

If you want to connect MoMa-LLM to robot learning literature, the clean bridge is the Markov Decision Process (MDP).

An MDP is a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ where:

- $\mathcal{S}$ is the state space (here: graph + robot pose + memory).
- $\mathcal{A}$ is the action space (high-level actions).
- $P(s'|s, a)$ is the transition model (how the state changes).
- $r(s, a)$ is a reward (or negative cost).
- $\gamma \in (0, 1]$ is a discount factor.

A **policy** $\pi(a|s)$ chooses actions. The **return** is

$$G_t = \sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}).$$

The **value function** of a policy is

$$V^\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s].$$

The **optimal value** satisfies the Bellman optimality equation:

$$V^\star(s) = \max_{a \in \mathcal{A}} \left(r(s,a) + \gamma \, \mathbb{E}_{s' \sim P(\cdot|s,a)} \left[V^\star(s') \right] \right).$$

Why include this in an LLM paper appendix? Because it clarifies what the LLM is acting as: a policy. MoMa-LLM chooses a_t from a constrained $\mathcal{A}$ based on a state representation (prompt) derived from G_t. Even if there is no learning, the *structure of the problem* is still an MDP.

A.17 Worked example: computing room precision/recall on a tiny grid

Consider a 4×4 grid (16 cells). Assume the ground truth has two rooms:

$$A = \{1, \ldots, 8\}, \qquad B = \{9, \ldots, 16\}$$

(i.e., top half vs bottom half).

Now suppose your estimator predicts two rooms:

$$\hat{A} = \{1, \ldots, 6\} \cup \{9, 10\}, \qquad \hat{B} = \{7, 8\} \cup \{11, \ldots, 16\}.$$

So the estimator "leaks" two bottom cells into $\hat{A}$.

Compute precision contribution for $\hat{A}$:

$$\frac{|A \cap \hat{A}|}{|\hat{A}|} = \frac{6}{8} = 0.75, \qquad \frac{|B \cap \hat{A}|}{|\hat{A}|} = \frac{2}{8} = 0.25,$$

so $\max(\cdot) = 0.75$.

For $\hat{B}$:

$$\frac{|A \cap \hat{B}|}{|\hat{B}|} = \frac{2}{8} = 0.25, \qquad \frac{|B \cap \hat{B}|}{|\hat{B}|} = \frac{6}{8} = 0.75,$$

so again $\max(\cdot) = 0.75$.

Therefore $P = (0.75 + 0.75)/2 = 0.75$.

Now recall: for A,

$$\frac{|A \cap \hat{A}|}{|A|} = \frac{6}{8} = 0.75, \qquad \frac{|A \cap \hat{B}|}{|A|} = \frac{2}{8} = 0.25,$$

so best match gives 0.75. Similarly for B it is 0.75. Hence $R = 0.75$.

This example is deliberately boring: it shows that the metric behaves like you expect.

A.18 Worked example: comparing two methods via AUC-E

Method 1 reaches success faster; Method 2 reaches similar final success but needs more budget. Let $c = \{0, 1, 2, 3, 4\}$ and

$$S_1 = \{0, 0.5, 0.7, 0.8, 0.85\}, \qquad S_2 = \{0, 0.2, 0.5, 0.8, 0.85\}.$$

Compute AUC by trapezoids (each interval has width 1):

$$\text{Area}_1 = \tfrac{0+0.5}{2} + \tfrac{0.5+0.7}{2} + \tfrac{0.7+0.8}{2} + \tfrac{0.8+0.85}{2} = 0.25 + 0.6 + 0.75 + 0.825 = 2.425,$$

$$\text{Area}_2 = \tfrac{0+0.2}{2} + \tfrac{0.2+0.5}{2} + \tfrac{0.5+0.8}{2} + \tfrac{0.8+0.85}{2} = 0.1 + 0.35 + 0.65 + 0.825 = 1.925.$$

With $c_{\max} = 4$, we get AUC-$\text{E}_1 \approx 0.606$ and AUC-$\text{E}_2 \approx 0.481$. So AUC-E correctly ranks Method 1 higher, even though both end at 0.85 success.

A.19 Micro-exercises (do these if you actually want to learn)

If you skip exercises, you will *think* you understand, but you will not.

1. Using Sec. A.3, what token do you get for $d = 3.0$? For $d = 10.0$? For $d = 10.01$?
2. In the belief update rule, if you have three locations with probabilities $(0.2, 0.3, 0.5)$ and you check the second location and fail, what are the updated probabilities?
3. Create two fake efficiency curves (5 points each) and compute AUC-E by hand using trapezoids.

Answer key (check yourself).

1. $d = 3.0 \Rightarrow$ `veryclose`. $d = 10.0 \Rightarrow$ `near`. $d = 10.01 \Rightarrow$ `far`.
2. Removing the second location gives remaining mass $1-0.3 = 0.7$. So updated are $(0.2/0.7,\ 0,\ 0.5/0.7) \approx (0.286,\ 0,\ 0.714)$.
3. Your numbers will differ; the only requirement is that you can execute the trapezoid rule without making arithmetic mistakes.

A.20 Pointers back to the main paper

For the exact experimental setup, action definitions, and the original efficiency curve design choices, refer to the main MoMa-LLM paper erifyciteHonerkamp20241Abstract, para 1MoMa-LLM, a novel approach that grounds language models within structured representations derived from open-vocabulary scene graphs and its supplementary. This appendix only aims to remove the mathematical friction so you can read those sections without getting stuck.

B Appendix: ESDF, GVD, and Voronoi Graph (Full Derivation + Toy Example)

This appendix section expands the geometry backbone of MoMa-LLM. The paper relies on the ESDF/GVD/Voronoi pipeline to extract a compact navigation skeleton. [1, p. 4] [p. 4, Sec. III-C, ¶2; "Voronoi"]

B.1 Occupancy grid → signed distance

Assume a 2D occupancy grid for simplicity. Let $\mathcal{O} \subset \mathbb{R}^2$ be the set of obstacle points (occupied cells mapped to continuous coordinates). Define the **unsigned distance transform**

$$d(x) = \min_{o \in \mathcal{O}} \|x - o\|_2.$$

This is the distance from a point x to the nearest obstacle.

A **signed distance field** is obtained by giving points inside obstacles negative sign:

$$\phi(x) = \begin{cases} -d(x), & x \in \mathcal{O} \\ +d(x), & x \notin \mathcal{O}. \end{cases}$$

In 3D this becomes the **ESDF**: Euclidean Signed Distance Field. [1, p. 4] [p. 4, Sec. III-C, ¶2; "ESDF"]

B.2 Medial axis intuition: why Voronoi emerges

If you pick any free-space point x, it has one nearest obstacle point $o^\star(x)$. Most points have a *unique* nearest obstacle; but points that are *equidistant to two or more distinct obstacles* form the **generalized Voronoi diagram (GVD)**. Formally, the GVD is the set

$$\text{GVD} = \{x \notin \mathcal{O} \ : \ \exists o_1 \neq o_2 \in \mathcal{O} \text{ s.t. } \|x - o_1\| = \|x - o_2\| = d(x)\}.$$

These are the "ridges" of the distance field. They trace the medial axis of free space and often align with corridors. [1, p. 4] [p. 4, Sec. III-C, ¶2; "GVD"]

B.3 Toy example you can compute by hand

Consider a 7×7 grid with two vertical obstacle walls (cells occupied) at columns 2 and 6. Intuitively, the points with maximum clearance lie on the middle column 4. That middle column is exactly the Voronoi ridge: it is equidistant to both obstacle walls.

- At cell center $x = (4, y)$, distance to left wall is $|4 - 2| = 2$ and to right wall is $|6 - 4| = 2$.

- For any $x = (3, y)$, nearest obstacle is left wall (distance 1), so it is *not* on GVD.

So the GVD here is the set of grid centers with $x = 4$. When you prune and discretize that ridge, you get a **graph** whose nodes/edges form a safe navigation skeleton. That is why a Voronoi graph is a compact navigation structure. [6, p. 1] [p. 1, Title/Abstract, ¶1; "Voronoi-based"]

B.4 Graph extraction (conceptual)

In practice, the pipeline is:

1. Compute ESDF $\phi(x)$ on the map.
2. Identify ridge points (high curvature / multi-source nearest obstacles) $\Rightarrow$ GVD.
3. Skeletonize and extract a graph (nodes at junctions, edges along ridges).

MoMa-LLM uses this graph to compute distances and connectivity used for room assignment and planning. [1, p. 5] [p. 5, Sec. III-E, ¶1; "cost"]

C Appendix: Door KDE and Room Graph Inference (Step-by-step)

MoMa-LLM treats doors as structural separators between rooms. Door locations are noisy because perception is noisy and doors can be seen from multiple viewpoints. The paper uses KDE to estimate door position density. [1, p. 5] [p. 5, Sec. III-D, ¶1; "kernel density"]

C.1 Kernel density estimation in one dimension

Suppose you observe door center positions along a hallway coordinate x, producing samples $\{x_i\}_{i=1}^n$. KDE estimates a smooth density:

$$\hat{p}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right),$$

where K is a kernel (often Gaussian) and $h > 0$ is bandwidth.

Why you care about h. If h is too small, $\hat{p}$ is spiky and you get fake doors. If h is too large, two nearby doors merge. In robotics, this is an engineering knob that trades off false positives vs. missed structure.

C.2 Toy KDE example

Take three observations near one door: $x_1 = 0.0$, $x_2 = 0.1$, $x_3 = -0.05$ and one observation near a second door: $x_4 = 2.0$. With a small bandwidth, the density will have two peaks: one near 0 and one near 2. Thresholding the density peaks yields two door hypotheses. This is exactly the logic MoMa-LLM uses in 2D/3D, just with more geometry. [1, p. 5] [p. 5, Sec. III-D, ¶1; "door locations"]

C.3 From doors to rooms

Once doors are hypothesized, room connectivity can be approximated as a graph:

$$G_{\text{room}} = (V_{\text{room}}, E_{\text{door}}),$$

where V_{room} are rooms and edges represent doors between them. This graph is what the LLM uses when choosing "search room" actions. [1, p. 5] [p. 5, Sec. III-D, ¶2; "connected rooms"]

D Appendix: Object-to-Room Assignment Cost (Derivation + Example)

The paper assigns each detected object hypothesis to a room by minimizing a cost based on distance through the connectivity graph. [1, p. 5] [p. 5, Sec. III-E, ¶1; "cost"]

D.1 Graph-based distance

Let $d_G(u, v)$ be the shortest-path distance between two nodes u, v in the navigation/room graph (edges weighted by geometric length). For an object candidate at location o and a room node r, define:

$$c(r \mid o) = d_G(r, o) + \lambda \cdot \text{penalty}(r, o),$$

where λ weights extra penalties (e.g., crossing doors, uncertainty). The exact form in MoMa-LLM is tailored to their graph; the key point is: *distance is not Euclidean in open space, it is distance along feasible connectivity.* [1, p. 5] [p. 5, Sec. III-E, ¶1; "distance-weighted"]

D.2 Toy example

Rooms: $\{A, B\}$ connected by one door edge of length 3. Object o is located inside room B at distance 1 from the door on the B side. Then:

$$d_G(B, o) \approx 1, \qquad d_G(A, o) \approx 3 + 1 = 4.$$

So the assignment chooses room B. This sounds trivial until you have many rooms and partial maps; then the graph distance is the only sane way to assign. [1, p. 5] [p. 5, Sec. III-E, ¶1; "assign objects to rooms"]

E Appendix: AUC-E (Efficiency Curve) computed by hand

The paper introduces AUC-E as an efficiency-oriented metric. [1, p. 6] [p. 6, Sec. V, ¶2; "AUC-E"]

E.1 Define the curve

Let a task have a maximum budget B (steps, time, or actions). For a method, define

$$S(b) = \Pr(\text{success within budget } b), \qquad b \in [0, B].$$

Plotting $S(b)$ vs. b yields an efficiency curve: methods that succeed earlier have higher curve earlier. AUC-E is then

$$\text{AUC-E} = \frac{1}{B} \int_0^B S(b)\, db.$$

The normalization by B keeps the score in $[0, 1]$.

E.2 Discrete estimate

In experiments, you have N episodes. Suppose each episode either succeeds at step t_i or fails (treat as $t_i = \infty$). Then:

$$S(b) \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[t_i \le b].$$

AUC-E can be approximated by a Riemann sum over budgets.

E.3 Tiny example (3 episodes)

Budget $B = 10$. Episode success steps: $t = \{2, 7, \infty\}$. Then:

$$S(b) = \begin{cases} 0, & b < 2 \\ \frac{1}{3}, & 2 \le b < 7 \\ \frac{2}{3}, & 7 \le b \le 10 \end{cases}$$

So:

$$\text{AUC-E} = \frac{1}{10} \left(\int_0^2 0\, db + \int_2^7 \frac{1}{3}\, db + \int_7^{10} \frac{2}{3}\, db \right) = \frac{1}{10} \left(\frac{5}{3} + \frac{6}{3} \right) = \frac{11}{30} \approx 0.367.$$

This is the kind of calculation you should be able to do quickly; otherwise you will misread papers that propose new metrics. [1, p. 6] [p. 6, Sec. V, ¶2; "efficiency curve"]

F Appendix: Citation audit checklist (how to self-verify)

You requested that citations be verifiable down to page and paragraph. Here is a pragmatic checklist for *you* (not the author) to validate this report:

1. Pick any paragraph. Identify its **claims**. If it has no **[empty citation]** [p. b, o; "x"], mark it as "unsupported".
2. For each **[b]** [p. o, x; ","] open the cited PDF and jump to the stated page. Confirm the anchor phrase exists.
3. If the paragraph makes a **numeric** claim (scores, counts, dataset size), confirm the number is explicitly present on that page. If not, the citation is insufficient.
4. If the paragraph makes a **comparative** claim ("better than X"), confirm the baseline is defined in the cited paper and the comparison is stated.

This checklist is boring. Good. Boring is how you stop fooling yourself. [1, p. 6] [p. 6, Sec. V, ¶1; "evaluation"]

G Appendix: POMDP basics with a worked belief-update example

MoMa-LLM explicitly calls the task a POMDP. If that word feels like magic, you need the following. [1, p. 2] [p. 2, Sec. III, ¶1; "POMDP"]

G.1 Definition (do not memorize; understand)

A POMDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R, \gamma)$:

- $\mathcal{S}$: states (true world, including object location, door states, robot pose)
- $\mathcal{A}$: actions (navigate, open door, inspect)
- $\mathcal{O}$: observations (camera detections, door sightings)
- $T(s'|s, a)$: transition model
- $Z(o|s, a)$: observation model
- $R(s, a)$: reward
- γ: discount

The agent does not observe s directly; it maintains a **belief** $b(s) = \Pr(s)$.

G.2 Belief update equation

Given belief b_t, action a_t, and observation o_{t+1}:

$$b_{t+1}(s') = \eta\, Z(o_{t+1} \mid s', a_t) \sum_{s \in \mathcal{S}} T(s' \mid s, a_t)\, b_t(s),$$

where η is a normalization constant so beliefs sum to 1.

G.3 Toy example (two rooms, one object)

Rooms: A and B. State is object location only: $\mathcal{S} = \{A, B\}$. Start with uniform belief $b_0(A) = b_0(B) = 0.5$.

Action: *inspect room A*. Observation: *did we see the object?* So $\mathcal{O} = \{\text{see}, \text{not see}\}$.

Assume:

$$Z(\text{see} \mid A) = 0.9, \quad Z(\text{see} \mid B) = 0.1,$$

and room inspection does not move object (T is identity).

If you observe *not see*, then:

$$b_1(A) \propto Z(\text{not see} \mid A)\, b_0(A) = 0.1 \cdot 0.5 = 0.05,$$

$$b_1(B) \propto Z(\text{not see} \mid B)\, b_0(B) = 0.9 \cdot 0.5 = 0.45,$$

Normalize: $b_1(A) = 0.1$, $b_1(B) = 0.9$.

This is exactly the logic behind "search room A, fail, then search room B": belief shifts. MoMa-LLM does not write the belief update like this in the main loop, but the DSG + history effectively encode similar belief shifts. [1, p. 3] [p. 3, Sec. III-B, ¶1; "dynamic"]

H Appendix: Mini-dossiers on post-MoMa papers (what they add)

This appendix gives short but *operational* summaries of the papers you provided. For each, we state: (i) what problem they target, (ii) the core technical idea, and (iii) what this implies for extending or critiquing MoMa-LLM.

H.1 VoroNav (Voronoi + LLM for zero-shot object navigation)

Problem focus. VoroNav targets *zero-shot object navigation*: given a target object category, navigate to where it is likely to be found without training in the target environment. [6, p. 1] [p. 1, Abstract, ¶1; "Zero-shot"]

Core idea. The method uses a *Voronoi-based* planning backbone, which implicitly favors paths with clearance and yields a compact connectivity structure. The LLM is used to inject semantic priors or decision heuristics into navigation. [6, p. 1] [p. 1, Title/Abstract, ¶1; "Voronoi-based"]

Relation to MoMa-LLM. This strengthens the case that Voronoi graphs are not a MoMa-LLM quirk; they are a reusable abstraction for language-assisted spatial reasoning. MoMa-LLM goes further by coupling the Voronoi backbone with explicit interaction (opening) and with a DSG that stores room/object structure. [1, p. 4] [p. 4, Sec. III-C, ¶2; "Voronoi graph"]

H.2 SayNav (language-instructed navigation)

Problem focus. SayNav addresses language-guided navigation: interpret a natural language instruction and produce navigation behavior. [7, p. 1] [p. 1, Title, ¶1; "SayNav"]

Core idea. The paper emphasizes reasoning over an internal state summary that can be produced from perception and map context, then using the LLM to guide navigation decisions. [7, p. 1] [p. 1, Abstract, ¶1; "navigation"]

Relation to MoMa-LLM. MoMa-LLM's prompt is similarly a state summary, but it is specialized for *object search* and includes door/room structure and object-room assignments. If you are extending MoMa-LLM, SayNav is a useful reference for how to format navigation-centric state in prompts and how to evaluate language grounding without interaction-heavy retrieval. [1, p. 6] [p. 6, Sec. IV, ¶1; "prompt"]

H.3 OrionNav (online scene updates and graph reasoning)

Problem focus. OrionNav highlights the need to update scene representations *online* during navigation and uses this structure for reasoning about goals and paths. [8, p. 1] [p. 1, Abstract, ¶1; "online"]

Core idea. Instead of assuming a static world model, the system continuously refines its representation based on new observations and uses this updated structure to guide navigation. [8, p. 1] [p. 1, Abstract, ¶1; "updates"]

Relation to MoMa-LLM. MoMa-LLM also updates a DSG, but the critique is that many dynamics are still not explicitly modeled (objects moved by humans, door states flipping). OrionNav's emphasis on online updates can be read as a direct response to this class of brittleness: your world model must be maintained as carefully as your controller. [1, p. 11] [p. 11, Conclusion, ¶1; "limitations"]

H.4 MORE (memory as a first-class module)

Problem focus. MORE argues that memory is essential for long-horizon robot behavior: storing past observations and retrieving them appropriately. [9, p. 1] [p. 1, Abstract, ¶1; "Mem

Core idea. Instead of treating history as only a prompt suffix, MORE promotes dedicated memory mechanisms (storage, retrieval, summarization) that can feed back into decision making. [9, p. 1] [p. 1, Title/Abstract, ¶1; "Memory-Augmented"]

Relation to MoMa-LLM. MoMa-LLM effectively uses two "memories": the DSG (structured state) and the action/history context fed to the LLM. A MoMa-LLM extension that integrates MORE-style memory would likely improve long-horizon search where the robot revisits rooms and must remember failed interactions and partial evidence. [1, p. 3] [p. 3, Sec. III-B, ¶1; "dynamic"]

H.5 StretchAI (search/retrieval in dynamic and concealed spaces)

Problem focus. StretchAI targets search and retrieval of objects in *dynamic and concealed spaces*—the hardest part of household robotics where objects are inside containers, behind clutter, or moved during execution. [10, p. 1] [p. 1, Title/Abstract, ¶1; "dynamic and co

Core idea. The paper emphasizes open-vocabulary and semantic-aware reasoning for search, suggesting richer semantic priors and reasoning about concealment. [10, p. 1] [p. 1, Title, ¶1; "Semantic-Aware Reasoning"]

Relation to MoMa-LLM. MoMa-LLM's room-and-door abstraction is a strong start, but concealed-space retrieval often demands reasoning about *container hierarchies* (cabinet → drawer → shelf) and about dynamic changes. StretchAI is therefore a roadmap for extending the DSG beyond rooms/doors toward a richer containment graph, while preserving the same key idea: LLM planning should operate on structured, verifiable state. [1, p. 1] [p. 1, Abstract, ¶1; "structured"]

I Appendix: From paper to code (pseudocode and data structures)

If you cannot map the paper to a concrete data-flow graph, you do not understand it.

I.1 Core data structures

A minimal MoMa-LLM-like implementation needs:

- **Map:** occupancy grid / TSDF / voxel map storing free/occupied.
- **ESDF:** scalar field $\phi(x)$ with nearest-obstacle distance.
- **Voronoi graph:** nodes with coordinates; edges with lengths.
- **DSG:** a labeled graph with nodes for rooms, doors, objects; edges for adjacency/containment.
- **Action set:** discrete templates parameterized by DSG entities.

This structure is directly aligned with the paper's stated components. [1, p. 3] [p. 3, Sec. III, ¶2; "scene graph"]

I.2 High-level loop (pseudocode)

```
Initialize map M0 from sensors
Initialize DSG G0 (empty rooms/objects)
for t = 0..T:
    Update map Mt using new sensor data
    Compute ESDF, GVD, Voronoi graph from Mt (periodically)
    Update door hypotheses via KDE -> update rooms in DSG
    Update object hypotheses -> assign to rooms -> update DSG
    Serialize DSG + history into prompt
    Ask LLM to choose action a_t from allowed action set
    Execute low-level skill for a_t (nav / open / inspect / grasp)
end
```

This pseudocode is the skeleton behind the "two-level policy" described by the paper. [1, p. 3] [p. 3, Sec. III-A, ¶2; "two-level policy"]

I.3 Where engineering time actually goes

Most time is not spent on the LLM call. It is spent on: (i) robust state estimation (doors/rooms/objects), (ii) interfaces between modules, and (iii) fallback behaviors when perception or parsing fails. MoMa-LLM's reported performance assumes these pieces are functional. [1, p. 11] [p. 11, Conclusion, ¶1; "limitations"]

J Appendix: Prompt design as an interface contract

MoMa-LLM uses a prompt that serializes structured state. Treat this as an API contract.

J.1 What must be in the prompt

At minimum, the LLM must see:

- the **goal** (target object category),
- the set of **rooms** and which are explored/unexplored,
- **connectivity** (which doors connect which rooms),
- current **candidate detections** and their room assignments,
- the list of **allowed actions** (with parameters).

The paper emphasizes that the LLM is called with structured scene information rather than raw pixels. [1, p. 6] [p. 6, Sec. IV, ¶1; "prompt"]

J.2 Common failure modes

- **Ambiguous identifiers:** if rooms are "room1/room2" without semantic labels, the LLM may mix them.
- **Too much text:** long prompts cause truncation and remove critical state.
- **Invalid entity reference:** the LLM proposes an action with an entity that does not exist.

You should handle these by deterministic validation and fallback, not by "hoping the LLM behaves". MoMa-LLM's constrained action selection is motivated by this. [1, p. 3] [p. 3, Sec. III-A, ¶2; "action space"]

K Appendix: Failure modes and diagnostics (what to log)

If you run an experiment and only log success/failure, you are wasting your time.

K.1 Perception failures

Log:

- door observations (positions + confidence) used by KDE,
- object detections (class, pose, score),
- false positives that enter the DSG.

Because the DSG is a belief state, errors propagate. [1, p. 5] [p. 5, Sec. III-D, ¶1; "door"]

K.2 Planning failures

Log:

- the serialized prompt,
- LLM raw output,
- parsed action, validation result, and fallback triggers.

This is necessary to debug whether failure was due to prompt/state vs. low-level execution. [1, p. 6] [p. 6, Sec. IV, ¶1; "prompt"]

K.3 Execution failures

Navigation and manipulation skills fail for mundane reasons (localization drift, controller saturation, gripper slip). MoMa-LLM is not immune; it relies on standard modules, so classical robotics debugging still applies. [1, p. 7] [p. 7, Sec. V-C, ¶1; "real-world"]

L Appendix: Reproducibility checklist (what a good report includes)

You asked what "proper references" mean. In robotics systems papers, reproducibility also requires *systems details*. Here is a checklist you can apply when reading MoMa-LLM or any follow-up:

1. Environment description (layout, number of rooms, door states).
2. Sensors and perception models used (classes, thresholds).
3. Map representation and resolution.
4. Graph extraction parameters (ESDF resolution, pruning thresholds).
5. Prompt template and action grammar.
6. LLM model/version and decoding settings.
7. Baselines and evaluation budgets.

MoMa-LLM provides many of these at a high level; as a reader you should still confirm which details are explicit and which are assumed. [1, p. 6] [p. 6, Sec. V, ¶1; "evaluation"]

M Appendix: Worked end-to-end example (tiny house, full loop)

This section walks through a *complete* MoMa-LLM-style loop on a tiny environment so the symbols in the main paper stop looking like decoration.

M.1 Environment

Two rooms (A and B) connected by one door D. Target object category: *mug*. The mug is in room B, but the robot starts in room A and does not know this.

M.2 Step 0: initialize belief/state

- Map: partial occupancy from initial scan.
- Door hypotheses: none yet.
- DSG: one "unknown room" node containing the robot; no door nodes.

This corresponds to the early exploration phase described in the paper. [1, p. 3] [p. 3, Sec. III-B, ¶1; "dyn

M.3 Step 1: observe a doorway and update KDE

Robot approaches the boundary and observes a doorway candidate at position x_D (noisy). Add this observation into the KDE buffer and recompute density peaks. If a peak passes a threshold, instantiate a door node D in the DSG. [1, p. 5] [p. 5, Sec. III-D, ¶1; "kernel density"]

M.4 Step 2: split into rooms

Given a door node, create room nodes A and B in DSG and connect them via D. The room adjacency graph now exists and can be serialized into the LLM prompt. [1, p. 5] [p. 5, Sec. III-D, ¶2; "connected rooms"]

M.5 Step 3: prompt and action choice

The prompt contains:

- goal: find mug,
- rooms: A explored partially, B unexplored,
- door D connects $A \leftrightarrow B$,
- allowed actions: search room, open door, explore frontier, attempt grasp.

The LLM selects an action such as `OPEN(D)` or `SEARCH(B)`. [1, p. 6] [p. 6, Sec. IV, ¶1; "prompt"]

M.6 Step 4: execute and update

After opening D and entering B, perception detects a mug candidate. The object is inserted into DSG and assigned to room B via the graph-distance cost. [1, p. 5] [p. 5, Sec. III-E, ¶1; "assign"]

M.7 Step 5: retrieval

If the mug is reachable, the low-level manipulation skill attempts grasp. If it fails, the failure becomes part of the serialized history and can change the next high-level decision. [1, p. 3] [p. 3, Sec. III-A, ¶2; "two-level policy"]

What you learned. Every step above is mundane systems engineering. The novelty is the *interface*: a structured state that is rich enough for reasoning, and a constrained high-level decision mechanism that can exploit that structure.

N Appendix: Extended comparison matrix (MoMa-LLM vs. adjacent work)

This is a higher-resolution version of Table 1. It is intentionally long so you can use it as a checklist when reading papers.

Work	World representation	Decision layer	Interaction model	What it teaches about MoMa-LLM
MoMa-LLM [1, p. 1] [p. 1, Abstract, ¶1; "Voronoi graph"]	Dynamic scene graph [1, p. 4] [p. 4, Sec. III-C, ¶2; "Voronoi graph"]	LLM chooses from constrained actions [1, p. 3] [p. 3, Sec. III-A, ¶2; "action space"]	Door opening + search + retrieval [1, p. 1] [p. 1, Abstract, ¶1; "Interactive object search"]	Baseline for structured-state LLM planning
SayCan [2, p. 1] [p. 1, Title, ¶1; "Do As I Can"]	No explicit 3D map (task-level tools) [2, p. 1] [p. 1, Abstract, ¶1; "affordances"]	LM + affordance scoring [2, p. 1] [p. 1, Abstract, ¶1; "grounding"]	Skill execution (varies)	MoMa's DSG is a different grounding mechanism (state grounding vs. affordance filtering)
VoxPoser [3, p. 1] [p. 1, Title, ¶1; "3D-Value Maps"]	Composable 3D value maps	Optimization/planning over values	Manipulation-centric interaction	MoMa could incorporate VoxPoser-style continuous spatial goals for retrieval
Hydra [4, p. 1] [p. 1, Title, ¶1; "3D Scene Graph"]	Real-time 3D scene graph	Not an LLM planner	Not task-specific	MoMa can be seen as a task-driven use of scene graphs

Work	World representation	Decision layer	Interaction model	What it teaches about MoMa-LLM
VoroNav [6, p. 1] [p. 1, Title, ¶1; “VoroNav”]	Voronoi planning + language priors	LLM-assisted navigation decisions	Mostly navigation	Validates Voronoi backbone as reusable
SayNav [7, p. 1] [p. 1, Title, ¶1; “SayNav”]	State summary for navigation	LLM reasoning for nav	No/limited manipulation	Suggests alternative prompt/state formats
OrionNav [8, p. 1] [p. 1, Title, ¶1; “OrionNav”]	Online-updated scene representation	Graph reasoning + LLM	Limited interaction	Highlights importance of non-stationarity handling
MORE [9, p. 1] [p. 1, Title, ¶1; “MORE”]	Memory (retrieval)	LLM + memory retrieval	Task dependent	Motivates explicit memory beyond DSG + prompt history
StretchAI [10, p. 1] [p. 1, Title, ¶1; “StretchAI”]	Semantic reasoning for concealed spaces	LLM + semantics	Interaction-heavy retrieval	Roadmap for extending MoMa beyond room/door abstraction

This matrix is not “the truth”. It is a tool: when you read a paper, fill in these fields. If you cannot, you did not understand the paper. [1, p. 11] [p. 11, Conclusion, ¶1; “future work”]

References

[1] Daniel Honerkamp et al. “Language-Grounded Dynamic Scene Graphs for Interactive Object Search with Mobile Manipulation”. In: *IEEE Robotics and Automation Letters* 9.2 (2024). Preprint version, pp. 1–9.

[2] Michael Ahn et al. “Do As I Can, Not As I Say: Grounding Language in Robotic Affordances”. In: *Proceedings of the Conference on Robot Learning (CoRL)*. arXiv:2204.01691. 2022.

[3] Wenlong Huang et al. “VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models”. In: *Proceedings of the Conference on Robot Learning (CoRL)*. arXiv:2307.05973. 2023.

[4] Nathan Hughes, Yun Chang, and Luca Carlone. “Hydra: A Real-time Spatial Perception System for 3D Scene Graph Construction and Optimization”. In: *arXiv preprint* (2022). arXiv: 2201.13360 [cs.RO].

[5] Fabian Schmalstieg et al. “Learning Hierarchical Interactive Multi-Object Search for Mobile Manipulation”. In: *arXiv preprint* (2023). arXiv: 2307.06125 [cs.RO].

[6] Pengying Wu et al. “VoroNav: Voronoi-based Zero-shot Object Navigation with Large Language Model”. In: *arXiv preprint* (2024). arXiv: 2401.02695 [cs.RO].

[7] Abhinav Rajvanshi et al. “SayNav: Grounding Large Language Models for Dynamic Planning to Navigation in New Environments”. In: *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*. arXiv:2309.04077. 2024.

[8] Venkata Naren Devarakonda et al. “OrionNav: Online Planning for Robot Autonomy with Context-Aware LLM and Open-Vocabulary Semantic Scene Graphs”. In: *arXiv preprint* (2024). arXiv: 2410.06239 [cs.RO].

[9] Mohammad Mohammadi et al. “MORE: Mobile Manipulation Rearrangement Through Grounded Language Reasoning”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2025). arXiv:2505.03035.

[10] Rohit Menon et al. “Open-Vocabulary and Semantic-Aware Reasoning for Search and Retrieval of Objects in Dynamic and Concealed Spaces”. In: Workshop paper (IROS 2025) / preprint. 2025. URL: https://autonomousrobots.nl/assets/images/workshops/2025_iros/accepted_papers/paper_8_Open.pdf.