ROB 498/599 · 3D Robot Perception · Fall 2026

L13 3D scene graphs, open-set and task-driven

From a dense map to a graph of objects, places and rooms: what a scene graph stores, how ConceptGraphs builds one, and how a task decides what an object is.

Wed Oct 21, 2026 • ANNOTATED build

3D scene graphs, open-set and task-driven

L13 · Unit 2

L11 put a feature in every voxel. A robot asked to fetch a mug does not want 50 million voxels. It wants the mug, the room it is in, and how to get there.

Today: the **scene graph**, a map of things and places joined by relations, and the open-set systems that build one from foundation models.

Today

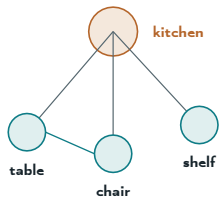
1. **The data structure:** nodes, edges, and the layered hierarchy: mesh, objects, places, rooms, building.
2. **ConceptGraphs, in depth:** how SAM, CLIP and an LLM become an open-set object graph. Association worked by hand.
3. **What counts as an object depends on the task:** Clio in brief (Monday's guest goes deeper), then HOV-SG and OpenGraph.
4. **Why graphs:** memory, bandwidth between robots, and noticing that the world changed.
5. **Graphs and memories for language models**, from L15: SayPlan, ReMEmbR, and embodied question answering.

From a dense map to a graph

dense: a feature in every voxel

(L07, L11)

group



a graph: a few hundred nodes

- A dense map answers *what is at this point?* A robot more often asks *where is the mug?*, *which room has a printer?*, *how do I get to the kitchen?*
- Answering those from voxels means scanning all of them, every time.
- A **scene graph** stores the answer once: a node per thing or place, an edge per relation.

Nodes and edges

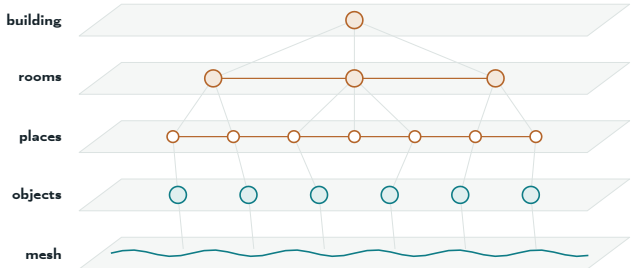
what a scene graph stores

Part	What it carries
Object node	a pose or centroid, a 3D bounding box, its points or mesh, a label or a feature vector, a caption
Place node	a position in free space and its distance to the nearest obstacle
Room node	a centroid, a label or feature, the places it contains
Intra-layer edge	*traversable* between places; *on*, *next to*, *inside* between objects; *adjacent* between rooms
Inter-layer edge	*contains*: building to room, room to place, place to object

> *Nodes are things you can name. Edges are the relations a planner needs.*

The layered hierarchy

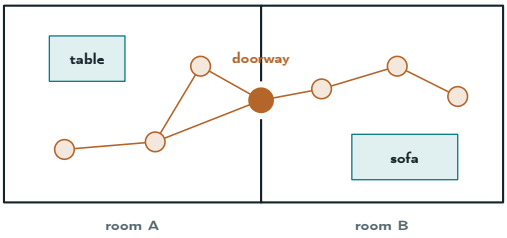
Rosinol et al., RSS 2020; Hughes et al., RSS 2022



- **Mesh:** the metric-semantic surface (L07).
- **Objects** (and agents) on it.
- **Places:** a sparse graph of free space.
- **Rooms:** clusters of places.
- **Building:** the root.

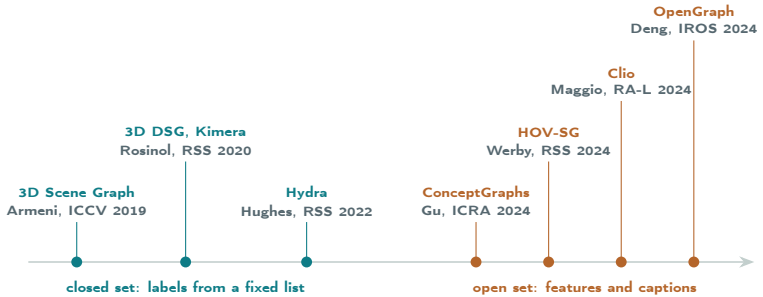
Each layer is coarser than the one below and answers a different question: *what is here, where can I stand, which room am I in.*

Places: a graph of free space



- Recall L07: an ESDF stores, at every free voxel, the distance to the nearest obstacle.
- Its ridges, the points equally far from two obstacles, form a skeleton of the free space. **Places** are nodes sampled along it, each with its clearance.
- Edges between places are paths a robot can drive. A narrow cut in the graph is a **doorway**; cutting there splits places into **rooms**.

Closed set, then open set



- **Closed set:** every object gets a label from a fixed list (chair, table, . . .). Queries are only as good as the list.
- **Open set:** every object keeps a **feature vector** (CLIP, from L11) and often a **caption**. A query is any text, compared to the features or read by a language model.

ConceptGraphs

Gu et al., ICRA 2024

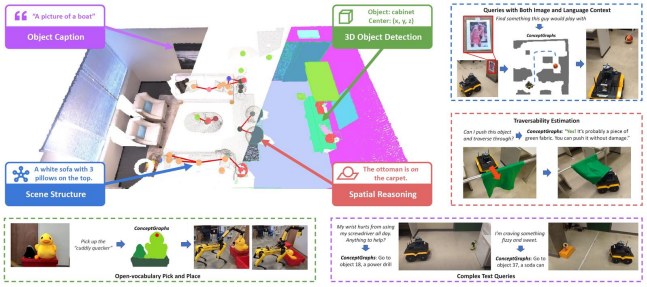
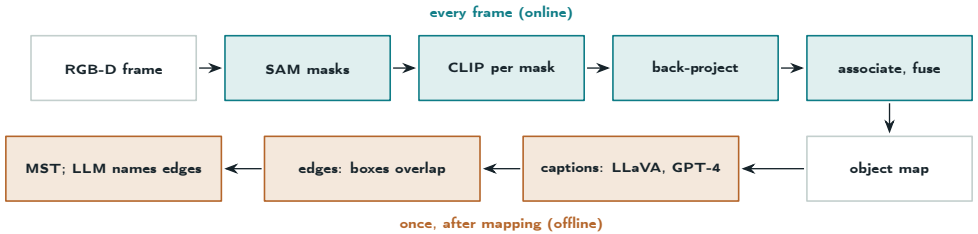


Figure: ConceptGraphs (Gu et al.), MIT licence

An object-level map in which every node has a CLIP feature and a caption, and every edge a relation in words, used for open-vocabulary queries, pick-and-place and traversability.

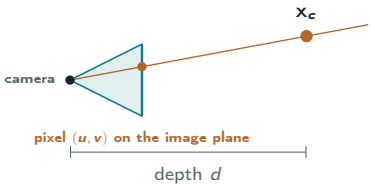
The pipeline



- **Per frame:** SAM proposes class-agnostic masks; each mask gets a CLIP image feature and is back-projected to a point cloud with depth and pose. Each one is then matched to an existing object or starts a new one.
- **After mapping:** each object is captioned and the edges are built. These steps call large models and are not real-time.

From a mask to 3D points

L02 and L03, once per pixel



$f_x = f_y = 525$, $(c_x, c_y) = (319.5, 239.5)$. Pixel $(400, 300)$ reads depth 1.50 m. Where is it, in the camera frame and in the world, with the camera at $(2, 1, 0.8)$ and no rotation?

$$X_c = d\,\frac{u - c_x}{f_x} = 1.50 \cdot \frac{400 - 319.5}{525} = 0.230$$

$$Y_c = 0.173, \quad Z_c = 1.50$$

$$\mathrm{X}_w = \mathrm{R}_{wc}\mathrm{X}_c + \mathrm{t}_{wc} = (2.230,\ 1.173,\ 2.30)$$

- Up to 307,200 points per VGA frame, so each object's cloud is voxel-downsampled at once.
- T_{wc} comes from the SLAM system underneath; drift in it moves the whole object, and breaks association.

Association: is this the same object?

A new detection i with points p_i and unit CLIP feature f_i; an existing object j with p_j, f_j.

$$\varphi_{\mathrm{sem}}(i,j) = \tfrac{1}{2}\,\mathrm{f}_i^{\top}\mathrm{f}_j + \tfrac{1}{2}, \qquad \varphi_{\mathrm{geo}}(i,j) = \text{share of } \mathrm{p}_i \text{ with a neighbour in } \mathrm{p}_j \text{ within } \delta_{nn}$$

$$\varphi(i,j) = \varphi_{\mathrm{sem}}(i,j) + \varphi_{\mathrm{geo}}(i,j), \qquad \text{merge with } \arg\max_j \varphi(i,j) \text{ if it exceeds } \delta_{\mathrm{sim}} = 1.1$$

- Both terms lie in $[0,1]$, so $\varphi \in [0,2]$. The paper uses $\delta_{nn} = 2.5$ cm, and the threshold 1.1 cannot be reached on semantics alone unless the features nearly match.
- Detections in one frame are assigned greedily to their best object.

Association by hand

two identical mugs



Cosines 0.80 to mug A and 0.86 to mug B; overlaps 0.62 and 0.00. Which, if either, does it join? And a plant beside mug A: cosine 0.30, overlap 0.20?

$$\varphi_A = \tfrac{1}{2}(0.80) + \tfrac{1}{2} + 0.62 = 1.52 > 1.1 \;\Rightarrow\; \text{merge with A}$$
$$\varphi_B = \tfrac{1}{2}(0.86) + \tfrac{1}{2} + 0.00 = 0.93 < 1.1$$
$$\varphi_{\text{plant}} = 0.65 + 0.20 = 0.85 < 1.1 \;\Rightarrow\; \text{a new object}$$

Mug B *looks* more similar ($0.93 > 0.90$). Only geometry tells the two mugs apart.

Fusing a match into the object

$$\mathrm{f}_j \leftarrow \frac{n_j\,\mathrm{f}_j + \mathrm{f}_i}{n_j + 1}, \qquad \mathrm{p}_j \leftarrow \mathrm{downsample}(\mathrm{p}_j \cup \mathrm{p}_i)$$

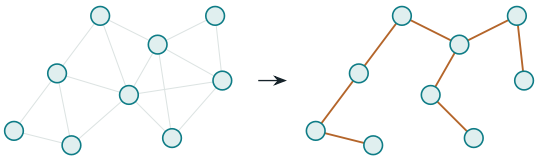
Detection i matched object j. f: CLIP feature; p: 3D points; n_j: detections merged into j so far, the k-th with feature $\mathrm{f}^{(k)}$.

Show that after n merges f_j is the plain average of all n detections' features

Induction: if $\mathrm{f}_j = \frac{1}{n}\sum_{k=1}^{n} \mathrm{f}^{(k)}$, then $\frac{n\mathrm{f}_j + \mathrm{f}^{(n+1)}}{n+1} = \frac{1}{n+1}\sum_{k=1}^{n+1} \mathrm{f}^{(k)}$. With $n_j = 4$, the next view moves the feature by only $\frac{1}{5}$ of the difference.

- Points are merged and voxel-downsampled at 2.5 cm, so an object seen a thousand times does not grow.
- Averaging many views is L11's multi-view fusion, done per object instead of per voxel.
- Trap: the average of unit vectors is shorter than 1. Renormalise before taking a cosine.

Captions and edges



- **Captions:** LLaVA-7B describes each object from its 10 best views; GPT-4 condenses those into one caption.
- **Candidate edges:** objects whose 3D bounding boxes overlap.
- **Prune** to a minimum spanning tree, then ask an LLM what each kept edge is: *on*, *in*, . . .

With $N = 100$ objects, how many LLM calls to label every pair, and how many for a spanning tree?

All pairs: $\binom{100}{2} = 4{,}950$. A spanning tree: $N - 1 = 99$, fifty times fewer.

Asking the graph a question

- **Simple queries:** embed the text with CLIP, take the cosine with every node's feature, return the best node and its 3D position.
- **Complex queries** (*my wrist hurts from using a screwdriver all day*): serialise the graph as a list of nodes with captions and positions, give it to an LLM, and ask which node.
- The answer is a node id, so the robot gets a 3D box to navigate to or grasp.

The context-window limit

A few hundred captions fit in an LLM's context. A five-storey building with every object does not. Choosing *which* nodes to keep is the next part of the lecture.

What the language model actually sees

```
[ ...
  {"id": 18, "caption": "a cordless power drill on a shelf",
   "centroid": [2.41, -0.87, 0.95], "extent": [0.24, 0.08, 0.20]},
  {"id": 37, "caption": "a can of soda on the floor by the door",
   "centroid": [0.62, 3.10, 0.06], "extent": [0.07, 0.07, 0.12]},
... ]
User: My wrist hurts from using a screwdriver all day. Anything to help?
LLM:  {"object_id": 18, "why": "a power drill does the screwdriver's job"}
```

- The graph is serialised as a list of nodes: id, caption, where it is and how big. The LLM never sees an image or a point.
- So the answer is only as good as the **caption**: a drill captioned *grey object on a shelf* is invisible to this query.

Two ways to get the objects

ConceptGraphs and ConceptGraphs-Detect

	ConceptGraphs	**ConceptGraphs-Detect**
Proposals	SAM: every region, no names	RAM tags the image, Grounding DINO finds boxes for the tags
Walls, floor	split into many pieces	recognised as background and merged
Misses	nothing large, but many fragments	anything the tagger did not name
Choose it when	you cannot list what matters in advance	scenes are cluttered and structure gets in the way

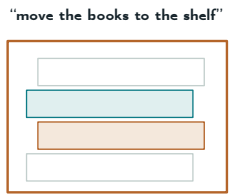
> *Class-agnostic finds everything and names nothing. A detector names things, and only the things it was asked about.*

Where ConceptGraphs struggles

- **Granularity:** SAM returns a book, a page, or the whole shelf depending on the view. The map inherits whatever the masks were.
- **Duplicates** when pose drift breaks the geometric overlap; **merges** of touching objects when it does not.
- **Static world:** nothing removes an object that has moved.
- **Offline:** captioning and edges run once, after mapping, with large models.
- **Relations are LLM guesses** from captions and boxes, not measured.

> *The first problem is the deepest: what is an object?*

What counts as one object?





- There is no task-free right answer. The stack is one object, or four, or one book and some clutter.
- A fixed segmentation threshold picks one granularity for every task. **Clio** lets the task list pick.

Clio

Maggio et al., RA-L 2024

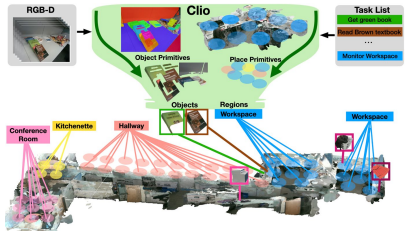


Figure: Clio (Maggio et al.), MIT SPARK Lab, BSD 2-Clause

- RGB-D and a list of tasks in. Fine **primitives** (SAM masks with CLIP features; places) are clustered into the objects and regions the tasks need; the rest is dropped.
- **Monday:** Yun Chang, a Clio author, on how the clustering works and where it went next.

HOV-SG: floors, rooms, objects

Werby et al., RSS 2024

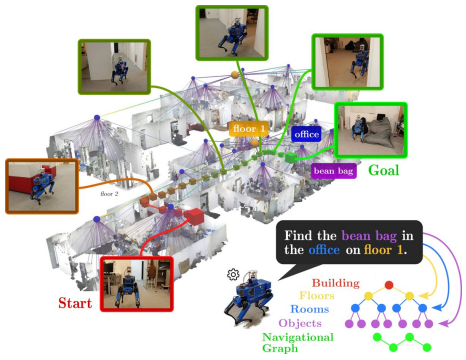
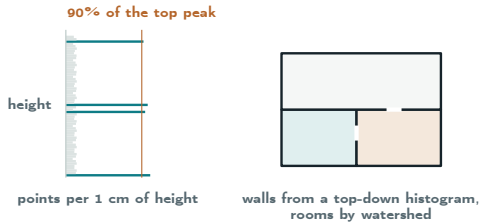


Figure: HOV-SG (Werby et al.), MIT licence

- A hierarchy of **floors, rooms and objects**, every node with an open-vocabulary feature aggregated from the level below.
- A **cross-floor Voronoi graph** for navigation: places again, now including stairs.
- About **75% smaller** than a dense open-vocabulary map: 143 MB against 568 MB for VLMaps.
- Real multi-storey navigation from *find the bean bag in the office on floor 1*.

How HOV-SG finds floors and rooms



- **Floors:** histogram every point's height in 1 cm bins; floors and ceilings are the peaks above 90% of the highest (illustrative, left).
- **Rooms:** a top-down histogram of each floor's points shows the walls; the watershed algorithm fills the rooms between them.
- **Features:** objects carry CLIP features; each room keeps 10 representative views chosen by k-means; each floor is just the text *floor 1*, *floor 2*.

Querying top-down

an illustrative building



flat: score every object



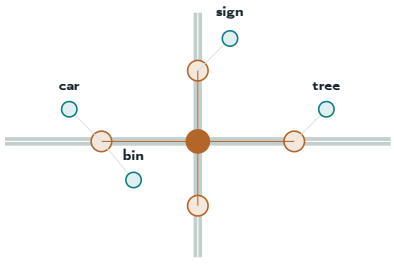
top-down: pick a floor, a room, then an object

3 floors, 40 rooms, 2,000 objects, spread evenly. How many feature comparisons for *bean bag, office, floor 1*, flat and top-down?

Flat: 2,000. Top-down: 3 floors + 14 rooms on that floor + 50 objects in that room = 67. But a wrong choice at the floor is never undone.

OpenGraph: the same idea outdoors

Deng et al., IROS 2024



lane segments and a junction, objects hung on them

- **Front end:** RAM tags each image, Grounding DINO boxes the tags, and TAP segments and captions each instance. Captions are embedded with Sentence-BERT.
- **Mapping:** masks are projected onto the **LiDAR** scan and fused incrementally, matching on 3D overlap, caption similarity and feature similarity: ConceptGraphs' rule with a caption term.
- **Layers:** points, a **lane graph**, instances, road **segments** cut along the lane graph, and the environment. Evaluated on SemanticKITTI.

Four systems side by side

	ConceptGraphs	Clio	HOV-SG	OpenGraph
Sensors	RGB-D	RGB-D	RGB-D	LiDAR and camera
Layers	objects	objects, places, regions	floors, rooms, objects	objects, lane segments
Open set via	CLIP, LLaVA and GPT-4 captions	CLIP against a task list	CLIP features at every level	captions, encoded as text
Granularity set by	SAM's masks	the tasks	SAM's masks	the captioner
Association	semantic + geometric overlap	3D primitives from masks, then IB clustering	mask overlap across frames	3D IoU, caption and feature similarity
Upper layers from	none	the tasks	walls and floor heights	the lane graph
Online?	objects yes, captions and edges no	yes, onboard	no	yes

Which would you pick?

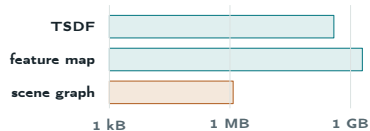
Situation	Pick	Why
One apartment, questions not known in advance	ConceptGraphs	keeps every object with a caption; any later question can be asked of it
A fixed list of jobs, compute on the robot	Clio	builds only what the tasks need, online, at the right granularity
A multi-storey building, *go to X in room Y*	HOV-SG	floors and rooms make the search short and the instruction's structure explicit
Streets, LiDAR, a vehicle	OpenGraph	LiDAR geometry, and segments along the roads it plans over
Tasks that change hourly	ConceptGraphs, then filter	Clio's map cannot answer a task that was not on its list

How much smaller?

an illustrative building

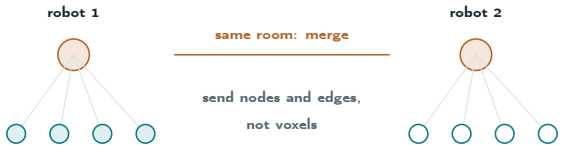
A $50 \times 40 \times 3$ m building at 5 cm. A TSDF, a feature map with a 1024-d float16 CLIP feature on surface voxels (1 in 50), and a graph of 500 objects, 3,000 places, 40 rooms.

$$\begin{aligned} \text{voxels} &= 1000 \cdot 800 \cdot 60 = 48{,}000{,}000 \\ \text{TSDF} &= 48{,}000{,}000 \times 8 \text{ B} = 384 \text{ MB} \\ \text{features} &= 960{,}000 \times 2{,}048 \text{ B} = 2.0 \text{ GB} \\ \text{graph} &\approx 500 \times 2{,}136 \text{ B} + \ldots = 1.20 \text{ MB} \end{aligned}$$



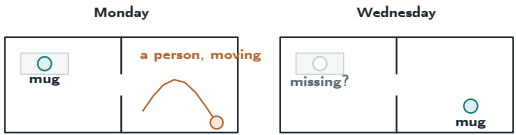
The graph is about 1,640× smaller than the feature map.

Two robots, one map



- Over a 2 Mbit/s radio the graph above takes 4.8 s to send; the dense feature map takes 2.2 hours.
- So robots exchange graphs, and after the first exchange only the **changes**: new nodes, moved nodes, new edges.
- Merging needs the two robots' frames aligned: a shared room or object is a loop closure between their factor graphs (L04). Hydra-Multi does this for scene graphs.

The world changes



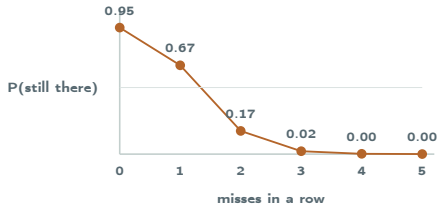
- **Agents** move constantly: people and robots are nodes with a trajectory (Rosinol et al., 2020).
- **Objects** move occasionally. A map that never forgets sends the robot to an empty table.
- **Change detection:** an object that *should* be in view, but is not detected, may have gone.

> *One miss is noise. Several misses in a row are news.*

How many misses before we delete it?

illustrative detector

Prior 0.95 that the mug is there. The detector misses a present mug 10% of the time and reports an absent one 5% of the time. Belief after one miss? After k?



$$P(\text{there} \mid \text{miss}) = \frac{0.10 \cdot 0.95}{0.10 \cdot 0.95 + 0.95 \cdot 0.05} = 0.67$$

Each miss multiplies the odds by $0.10/0.95$. After k misses: 0.67, 0.17, 0.02, 0.00, 0.00.

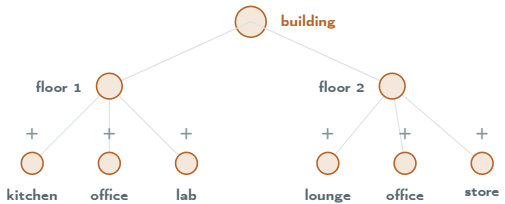
From answering to planning

ConceptGraphs answers which object. A robot told to refrigerate the orange on the kitchen bench needs a sequence of actions, across rooms, that it can actually execute.

- Hand the graph to an LLM as its context, and ask for a **plan**, not a node id.
- Two problems from earlier today come back: the graph is too big for the context, and the LLM writes plans that cannot run.
- This is the start of L15, spatial memory and language.

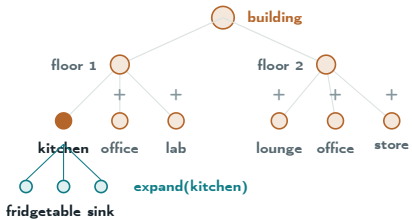
SayPlan

Rana et al., CoRL 2023



1. **Search:** the LLM starts from a collapsed graph and opens only the rooms it needs.
2. **Paths:** a classical planner, not the LLM, finds the route between rooms.
3. **Check:** a scene-graph simulator runs the plan and reports what fails; the LLM tries again.

1 of 3: start from a collapsed graph



- The LLM first sees only the top of the hierarchy: floors and rooms. Every object is hidden.
- `expand_node(room)` reveals what is in a room; `contract_node(room)` hides it again.
- The LLM calls these until it decides it has the **subgraph** the task needs, then stops.

```
Task: refrigerate the orange left on the kitchen bench.
LLM:  expand_node(kitchen)      # reveals bench, fridge, sink
LLM:  expand_node(lab)          # nothing relevant
LLM:  contract_node(lab)
LLM:  terminate                 # subgraph: kitchen and what is in it
```

What the search saves

an illustrative building

The building from *How much smaller?*: 500 objects, 3,000 places, 40 rooms. Say 30 tokens per object, 8 per place, 10 per room. Tokens for the full graph, without the places, collapsed, and after the search opens 2 rooms of 12 objects?

$$\begin{aligned} \text{full} &= 500 \cdot 30 + 3{,}000 \cdot 8 + 40 \cdot 10 = 39{,}400 \\ \text{no places} &= 15{,}400 \\ \text{collapsed} &= 40 \cdot 10 = 400 \\ \text{after search} &= 400 + 2 \cdot 12 \cdot 30 = 1{,}120 \end{aligned}$$

The search hands the LLM about $14\times$ fewer tokens than the graph without places. The places never go in: they are for the path planner.

What the search buys, measured

SayPlan, Tables 1 and 2

- Office graph: 6,731 tokens in full, 878 collapsed, 87% fewer before the search begins.
- Finding the right subgraph with **GPT-4**: 86.7% of simple searches and 73.3% of complex ones, in both environments.
- With **GPT-3.5**: 6.6% and 0%. It invented nodes to open, or opened the same one over and over.

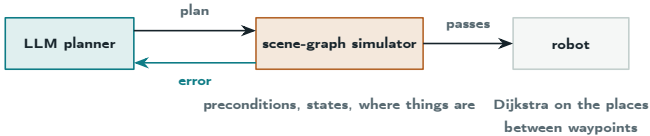
> *The hierarchy lets the model read the graph a piece at a time, and only a strong model can use it.*

2 of 3: the LLM does not plan paths

```
# an LLM asked for the whole plan writes every waypoint
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(kitchen_bench) pickup(orange1) ...
# SayPlan: the LLM names the room, Dijkstra fills in the poses
goto(kitchen) access(kitchen_bench) pickup(orange1) ...
```

- Waypoints are the places layer. Writing them out is long, and one wrong pose breaks the plan.
- So the LLM names the destination room or object, and **Dijkstra on the places graph** fills in the route.
- The LLM keeps the part a graph search cannot do: which objects, in which order, and why.

3 of 3: check the plan before the robot runs it



- `verify_plan` runs the plan forward **on the graph**: each action against the objects' states, predicates and affordances.
- A failure comes back as text, *cannot pick up banana*, if the fridge holding it is closed. The LLM rewrites the plan and the loop repeats.

One round of replanning

illustrative

```
plan 1: goto(kitchen) access(kitchen_bench) pickup(orange1)
        access(fridge) release(orange1)
check:  "cannot release orange1: fridge is closed"
plan 2: goto(kitchen) access(kitchen_bench) pickup(orange1)
        access(fridge) open(fridge) release(orange1) close(fridge)
check:  success
```

The first plan reads well and would fail on the robot. The check catches the missing `open(fridge)` on the graph, in milliseconds, before anything moves.

What each part buys

SayPlan, Table 3

	Simple, correct	Long, correct	Long, executable
LLM+P	93.3%	33.3%	0.0%
LLM as planner	93.3%	66.7%	13.3%
SayPlan	93.3%	73.3%	86.6%

- **Correctness:** the plan would do the task. **Executability:** every action can actually run.
- On short tasks all three are right as often. On long ones the others write plans that are *right in spirit* and cannot run; the path planner and the check fix that.

Where the LLM still fails

- **Negation:** *find me an office that does not have a cabinet* failed every time.
- **Counting and distance** across several nodes: the LLM miscounts, or guesses which room is nearer from coordinates.
- **A fixed map:** the graph is built beforehand and assumed static, so a moved orange breaks the plan.

> *The graph supplies the facts. Reasoning about space with them is still up to the LLM, and it is weak at it.*

Three places to decide what matters

When	System	What it costs
When the map is built	Clio	a task not on the list cannot be answered later
When a question is asked	ConceptGraphs	every object kept; long prompts
While a plan is made	SayPlan	the LLM must search well; only strong models can

A map, or a diary?

A scene graph holds the world as it is now. Some questions are about what the robot saw, and when.

- *Where did you last see my keys? How long was the loading dock blocked this morning? Take me somewhere with a nice view.*
- These need **time**, and things that are not objects: events, people, views.
- Second L15 topic: **retrieval over a spatial memory**.

ReMEmbR

Anwar et al., ICRA 2025

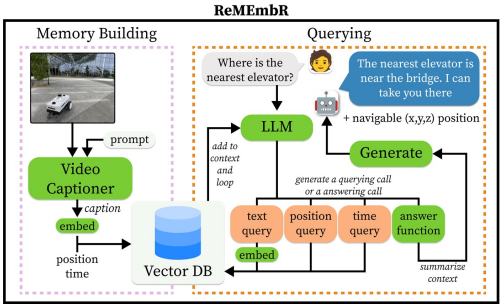


Figure: ReMEmbR (Anwar et al.), CC BY 4.0

Build: every 3 s of video becomes a caption, stored with where and when. **Query:** an LLM searches the store with tools, up to three rounds, then answers.

1 of 2: why not put it all in the prompt?

illustrative sizes

One caption every 3 s for an 8-hour day, 60 tokens each. How many tokens to hand an LLM the whole day? Storage, at a 1024-d float32 embedding, 300 B of text, and x, y, z, t per entry?

$$\begin{aligned} \text{entries} &= 8 \cdot 3600/3 = 9{,}600 \\ \text{tokens} &= 9{,}600 \cdot 60 = 576{,}000 \\ \text{one entry} &= 1024 \cdot 4 + 300 + 16 = 4{,}412 \text{ B} \\ \text{storage} &= 9{,}600 \cdot 4{,}412 \text{ B} \approx 42 \text{ MB} \end{aligned}$$

Storing a day is easy. Reading it is not: half a million tokens per question, every question, and the history keeps growing.

2 of 2: answering with three tools

```
Q:    Where can I get some chips?
LLM:  text_query("snacks")      # nearest captions by embedding
      14:02:11 (31.0, -4.2)  "shelves of chips and drinks by a counter"
      14:20:45 (2.5, 17.8)   "a vending machine in a hallway"
LLM:  text_query("cafeteria")   # a second look before answering
LLM:  answer({"position": [31.0, -4.2, 0.0],
              "text": "the cafeteria shelves, past the lobby"})
```

- Tools: **text** (nearest captions by embedding), **position** (memories near (x, y, z)), **time** (memories near a clock time). Up to three rounds.
- The answer is typed: a position to drive to, a time, a duration, yes or no, or a description.

Does retrieval beat reading everything?

ReMEmbR, NaVQA, Table I

Position error (m)	Short	Medium	Long
ReMEmbR	5.1	27.5	46.3
LLM with every caption	5.1	33.3	56.0
Video VLM, 2 frames/s	7.5	×	×

- NaVQA: 210 questions on campus drives, in clips under 2 min, 2 to 7 min, and over 7 min.
- On short clips reading everything is as good. As the history grows, retrieval pulls ahead, and the VLM cannot take the video at all (×).

ReMEmbR on a robot

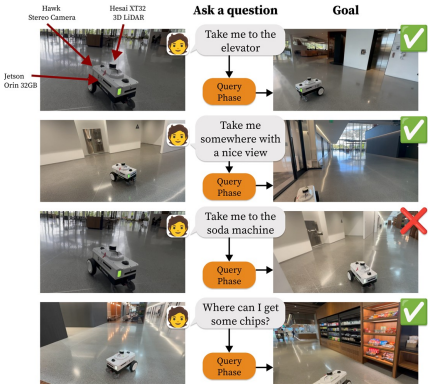


Figure: ReMEmbR (Anwar et al.), CC BY 4.0

- 25 minutes of driving in an office builds the memory, captioned on the robot's own GPU.
- *Where can I get some chips?* goes to the snack shelf. *Somewhere with a nice view* goes to a glass-walled corridor.
- *Take me to the soda machine* goes to a water fountain. The small on-board captioner had called it a **silver machine**.

Scene graph or memory log?

	Scene graph	**Caption memory**
Holds	objects, places, rooms, now	what was seen, where, when
Grows with	the size of the building	time spent driving
Good at	where is X, plans across rooms	when did, how long, what happened
Weak at	events, history	geometry, counting, anything not captioned

Embodied question answering

OpenEQA: Majumdar et al., CVPR 2024

- Third L15 topic: ask the robot a question about a place it has been, in plain language, and grade the answer.
- **Episodic memory (EM-EQA):** answer from a recorded walk through the scene. **Active (A-EQA):** the agent is dropped in the scene and must explore before answering.
- 1,636 questions written by people, in over 180 real homes and offices from ScanNet and HM3D.

> *A benchmark for the question every map in this unit claims to answer: does the robot understand the place?*

Seven kinds of question

examples from the paper

Category	Example
Object recognition	What is left of the kitchen pass-through?
Attribute recognition	What colour is the kitchen backsplash?
Object state	Is the microwave door propped open?
Object localisation	Where is the checkers board?
Spatial understanding	Can another cookie jar fit on the cookie jar shelf?
Functional reasoning	Where can I store the house key?
World knowledge	Does this house have forced-air heating?

Grading an open answer

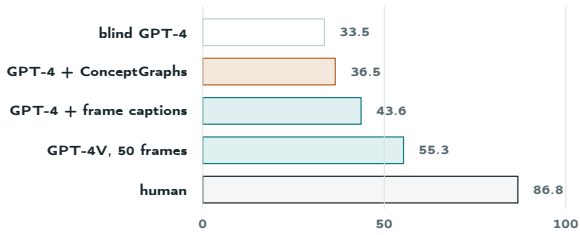
- Answers are free text, so exact match fails: *entryway table* and *the table by the front door* are both right.
- **LLM-Match:** GPT-4 compares each answer with the human one and scores it $\sigma \in \{1, \ldots, 5\}$.

$$C = \frac{1}{N} \sum_{i=1}^{N} \frac{\sigma_i - 1}{4} \times 100\%$$

Three answers scored $5, 3, 1$ give $C = 50\%$. Against four human graders on 300 questions, Spearman $\rho = 0.909$, as high as the graders agree with each other.

How well do they do?

OpenEQA, EM-EQA



- **Blind:** GPT-4 given only the question, no memory at all.
- + **ConceptGraphs:** today's map, serialised as on the earlier slide.
- The best system is about 32 points short of people.

A blind model scores a third

- Homes are regular. Asked the colour of a staircase railing with no memory at all, GPT-4 says *brown*, and is often right.
- So 33.5 is the floor, and ConceptGraphs adds only 3.0 points to it.
- On **spatial understanding** the scene graph scores 29.1, below the blind model's 31.4.

> *A map has to beat the guess a language model makes without one.*

Why the coordinates did not help

- The authors deleted every box and size from the serialised graph. The scores **barely moved**.
- So GPT-4 was answering from the captions, and not using the 3D numbers next to them.
- Same finding as SayPlan's failures on distance and counting: geometry written out as text is mostly ignored.

Wednesday, L15

What a language model can and cannot infer about space, and what to compute outside it: distances, containment and free space done by geometry, handed to the model as answers rather than coordinates.

What the graph cannot do

- **Collision checking:** a box per object is not the free space. Keep the mesh or ESDF underneath.
- **Uncertainty:** most systems store one feature and one caption per node, with no confidence.
- **Relations** named by an LLM from captions and boxes are plausible, not measured.
- **Evaluation** is hard: there is rarely a ground-truth graph, so papers report object recall and query success instead.

Learn it again, differently

the explanations worth your time

- Hughes, Chang et al., Foundations of Spatial Perception for Robotics (IJRR 2024). The long, careful account of layered scene graphs, places and rooms, by the Hydra team.
- MIT News, Helping robots focus on the objects that matter (Sep 2024). Clio's Information Bottleneck in plain language, with the book-stack and condiment-packet examples.
- The ConceptGraphs project page. Videos of the map being built and queried by a robot.
- The SayPlan project page. The robot carrying out long plans across an office floor.
- The ReMEmbR project page and the OpenEQA project page, with videos and the benchmark data.
- Rotondi et al., 3D Scene Graphs: Open Challenges and Future Directions (2026), with 3dscenegraphs.com: a survey that fixes the terminology across the field.
- The 3D Scene Graph project (Armeni et al., ICCV 2019). Where the layered idea started.

What you should be able to do

- Say what a scene graph's nodes and edges store at each layer, and why places and rooms come from free space.
- Compute ConceptGraphs' association score for a detection and decide merge or new object; explain what each term prevents.
- Explain why the spanning tree is a cost decision, and what relations it loses.
- Explain why Clio's map depends on its task list, and what that costs when a new task arrives.
- Estimate the memory and bandwidth of a dense map against a graph, and update an object's existence belief after misses.
- Count the tokens a collapsed graph saves, and say what SayPlan hands to the LLM, to Dijkstra and to the simulator.
- Size a day of caption memory, explain why ReMEmbR retrieves instead of reading it all, and read an OpenEQA score against the blind baseline.

Next

L14 Mon Oct 26

- **Response due Sun Oct 25:** ConceptGraphs. Clio and HOV-SG are also assigned.
- **Project proposals due Fri Oct 23**, naming a target paper and an evaluation.
- **L14, Mon Oct 26: guest lecture by Yun Chang.** Response paper: Hydra.
- He will cover Hydra (building the layers in real time), Clio (task-driven granularity), ASHiTA (breaking a task into subtasks grounded in the scene graph) and FOUND-IT (task-driven scene graphs from one monocular camera, built on foundation models).
- **L15, Wed Oct 28:** spatial memory and language, what a language model can and cannot infer about space, and anything from today's last section we did not reach.