From Speech-to-Spatial: Grounding Utterances on A Live Shared View with Augmented Reality

Yoonsang Kim Divyansh Pradhan Devshree Jadeja Arie E. Kaufman

Center for Visual Computing, Stony Brook University

SUPPLEMENTARY MATERIALS

On this page, we illustrate any details that can further support the claims of our main manuscript. The following details are supporting arguments of our research. Optional, but can be found valuable.

1 System Design and Implementation

1.1 Linguistic Attributes Parsing

Upon the reception of a spoken instruction, Speech-to-Spatial first transcribes the audio input, and parses the resulting text to extract its linguistic attributes. We decompose the utterance into intrinsic object descriptors (e.g., object name, appearance features)–Direct Feature, optional anchor references, relational cues (e.g., left, behind, on top of)–Relation, and contextual modifiers such as Activity or Memory-relevant phrases (e.g., "the one we used earlier")–Memory. This parsing step converts free-form natural language into a set of normalized representation to be assigned to our object-centric graph representation in later steps. To achieve this, we prompt to an LLM agent to parse and summarize, while instructing it to prevent any mal-modification to the original information. We share the core part of our pseudo-prompt:

```
"""
You are a spatial reasoning assistant. You are given a
      transcribed user query.
Parse the query to extract its linguistic attributes without
      altering the original meaning.

1) Identify the TARGET OBJECT being referred to.
For the target object, extract:
- label: the primary noun or object type (use null if
      underspecified).
- description: intrinsic attributes describing the object
  (e.g., color, size, shape, material, texture, pattern,
       functional modifiers).
- memory: references to prior interactions with the object.
  Each memory entry includes:
  - action: the verb describing the interaction.
  - description: a short phrase describing the interaction
       context.

2) Identify any ANCHOR OBJECTS used to describe the target.
For each anchor object, extract:
- label: the primary noun or object type.
- description: intrinsic attributes describing the anchor.
- relation: the spatial relationship between the target and the
      anchor
  (e.g., left_of, right_of, above, below, in_front_of, behind_of)
       .
- memory: references to prior interactions associated with the
      anchor
  (same structure as target memory).

3) Extract spatial relational cues explicitly stated in the query
       and associate them with the corresponding target-anchor
      pairs.

4) Extract contextual modifiers related to prior interactions or
      temporal references
(e.g., earlier, last week, we used before).

5) Preserve the original intent and wording of the query.
Do NOT infer missing information, add attributes, or resolve
      ambiguity.

[Example]
Query: The green box next to the blue sphere with stripes that we
       used last week

Parsed Output:
{
  "target": {
    "label": "box",
    "description": "green",
    "memory": []
  },
  "anchors": [
    {
      "label": "sphere",
      "description": "blue with stripes",
      "relation": "next_to",
      "memory": [
        {
          "action": "used",
          "description": "used last week"
        }
      ]
    }
  ]
}
"""
```

1.2 3D Object Detection and Recognition

As illustrated in Fig. 1, the visual captures of the physical environment are generated into 3D pointcloud (as GLB), along with object detection of Gemini 2.5-Flash (open-vocabulary). Gemini 2.5-Flash is given the RGB 2D image capture, and are instructed to detect any objects up to 10 with their individual bounding box. If the target description was provided, the description is summarized to list the objects contained in the descriptions, and be localized to only those target objects. Upon return, the bounding box is associated with our 3D point cloud to derive the 3D location of each localized object. We share the core part of our pseudo-prompt:

```
"""
You are given an image from an AR user's viewpoint.

1) Localize up to 10 objects.
2) For each object, return:
   - label (shortest form of class name)
   - visual descriptions (list of short phrases: color, size,
        shape, texture/material/patterns, visible text)
   - bounding box (ymin, xmin, ymax, xmax)
3) Return a concise description of the scene (up to 3 sentence).
"""
```

1.3 Object-centric Graph Building

The object-centric graph is constructed using the visual attributes (Sec. 1.2), first. Each detected object in the scene is instantiated as a

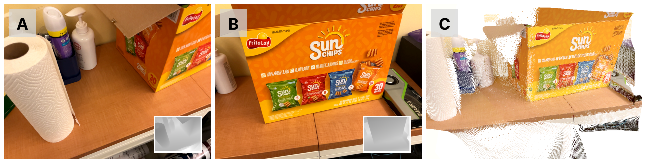

Figure 1: Illustration of 3D pointcloud generation steps. We use RGB+D (**A, B**–Small box at the bottom right indicates Depth image) image captures to generate 3D point clouds (**C**) and associate the transformational information with the detected/class instance of Gemini 2.5-Flash. We adapt the data collection pipeline of Explainable XR [2].

graph node, where each node contains its intrinsic visual attributes, including object label/class, features, 6DoF pose, timestamp, and a textual description of its surroundings. Then, the spatial relation is mapped with a graph assigned with one of the 6D relation types (e.g., left, right, above, below, in-front-of, behind-of) if two object are within the proximity threshold radius (r=50cm). These relation connect physical spatial properties to logical (code-level).

Next, upon any linguistic input such as "*The water bottle next to a cookie*", the instance 'water' (target) and 'cookie' (anchor) is mapped to the matching (or most highest similar) node class with the class/label, 'water' and 'cookie' in the logical graph. Also, the interaction history as well as action (if any) is recorded onto the object on the graph, allowing Speech-to-Spatial to capture both spatial relations of objects in the real-world and the linguistic attributes.

1.4 Graph Traversals and Reasoning

Prior to the graph traversal, we proceed by gradually narrowing the candidate referents, within our object-centric graph. Given a parsed intrinsic attribute description, the system first compares the current linguistic descriptions against the stored/existing graph nodes using cosine similarity in the embedding space (we use OpenAI's *text-embedding-3-small* model). Only the nodes with similarity exceeding a threshold (thresh=0.65) are retained as initial candidates as a list. These candidates are further filtered using contextual constraints, whether their space, activity, interaction history aligns with the current utterance. The descriptions of the user's space and activity, are generated in Sec. 1.2 via Gemini 2.5-Flash and GPT-4.1 API calls. The last filtering occurs at the client's perspective-level. The existing graph nodes that have the bounding volume outside the current viewing frustum are removed, as well as occlusion computation (Refer to the main paper). This reduced set of nodes enable our graph traversal more optimal, before iterating through each candidate graph nodes.

We traverse the graph not from a single entry, but rather from the filtered candidate nodes. By iterating each candidate node and verifying whether its neighboring nodes satisfy any of the Relation condition (if any), the candidate node is included in the final potential target list. If there is a conflict in the list, or multiple targets satisfy the conditions, we employ our fall-back mechanism where *Full* transcription is visualized to the user, minimizing the confusion the instructed user may have from an incorrect inference. If there is only single most plausible referent, we generate a visual indicator on top of the referent (Sec. 1.5).

1.5 AR Visual Guidance Anchoring

At the last stage of our pipeline, AR visual pointer is generated. This is instantiated on the client's AR device (via Unity AR Foundation–AR anchor) at the center point of the 3D position of the target object to which the verbal instruction was referring. In addition, the transcribed text of the instruction is generated on top of the visual indicator. The orientation of the indicator is transformed to face towards the user, for a better visibility.

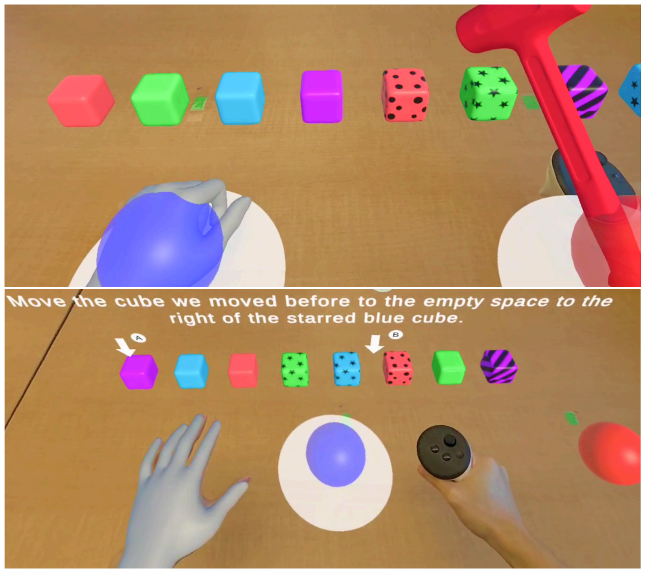
Move the cube we moved before to the empty space to the right of the starred blue cube.

Figure 2: The in-app capture of the user study view. The eight virtual cubes represent the target of reference of given verbal instructions. A participant conducting Move (Top) and Locate (Bottom) Task under *Full* condition. The environment was registered and scanned prior to the study.

2 Evaluation

2.1 Study Setup

Our evaluation study validating the quantitative and qualitative gains and losses of Speech-to-Spatial was conducted on the lab desk settings across the participants (Fig. 2). The physical scene scanning for AR, anchoring of contents, and registration were performed prior to the user study.

2.2 Synthetic Instructions

To maintain a consistent environment across the participants (uniform tone/emotion/pace of given speech instructions), and hypothesize the scenario of remote instructions, we used the OpenAI's Text-to-Speech model (*gpt-4o-mini-tts*). The voice were pre-generated and saved. Examples of the synthesized AI instructions include: "*Hammer the cube to the left of the puple cube we moved before*" (Locate Task with Chained speech pattern), "*Hammer the red cube*" (Locate Task with Direct Feature), or "*Move the cube to the left of the blue cube to the empty space to the right of the starred green cube.*" (Move Task with Relational pattern). The participants were given the instructions to perform the tasks (Locate, Move) upon the completion of the AI-generated voice, verbal instructions. Every task was given a single instruction.

2.3 Effects Per Referencing Pattern

Locate Task: For a well-specified, single-anchor language (Direct Feature, Relational), *Summary* yielded a clear time advantage over *Audio* (even over *Full*), with the largest gains when the utterance includes Relational or Memory-based pattern.

The median times of *Direct Feature* were : *Audio* 3.42s, *Full* 3.04s, *Summary* 2.13s (Friedman $\chi^2 = 18.11$, $p < .001$; *Summary*<*Audio*, $p < .001$; *Summary*<*Full*, $p < .001$). *Summary* sped up the task completion time by 38% vs *Audio* and 30% vs *Full*.

For *Relational*, the medians were : *Audio* 3.19s, *Full* 2.79s, *Summary* 2.06s (Friedman $\chi^2 = 17.44$, $p < .001$; *Summary*<*Audio*, $p < .001$; *Summary*<*Full*, $p = .012$; *Full*<*Audio*, $p = .048$), and the *Summary* was 35% faster than *Audio* and 26% faster than *Full*.

Full also outperformed *Audio Memory-based pattern*: *Audio* 3.21s and *Summary* 1.89s (Friedman $\chi^2 = 14.78$, $p < .001$;

Summary<*Audio*, $p = .003$; *Summary*<*Full*, $p = .001$). The *Summary* cut task completion time by 41% vs *Audio* and was also faster than *Full*.

The *Chained pattern* reaped: *Audio* 7.21*s*, *Full* 6.63*s*, *Summary* 5.97*s* (Friedman $\chi^2 = 14.78$, $p < .001$; *Summary*<*Audio*, $p < .001$; *Summary*<*Full*, $p = .006$). *Summary* reduced the time by 17% vs *Audio*, and 10% vs *Full*.

Move Task: *Summary* reduced time in every pattern and was especially effective for Memory and Chained language, where relational/temporal context is key. It also yielded the strongest accuracy gains when Memory cues were involved.

The median times of *Direct Feature* were : *Audio* 7.05*s*, *Full* 5.69*s*, *Summary* 5.08*s* (Friedman $\chi^2 = 14.78$, $p < .001$; *Summary*<*Audio*, $p < .001$; *Full*<*Audio*, $p = .012$). *Summary* was 28% faster than *Audio* and 11% faster than *Full*, and *Full* was faster than *Audio*.

For *Relational*, the medians were : *Audio* 7.69*s*, *Full* 5.67*s*, *Summary* 4.66*s* (Friedman $\chi^2 = 16.78$, $p < .001$; *Summary*<*Audio*, $p < .001$; *Full*<*Audio*, $p < .001$), and the *Summary* was 39% vs *Audio* and 18% vs *Full*; *Full* also outperformed *Audio*.

Memory-based pattern: *Audio* 7.41*s*, *Full* 6.20*s*, and *Summary* 4.71*s* (Friedman $\chi^2 = 21.01$, $p < .001$; *Summary*<*Audio*, $p < .001$; *Summary*<*Full*, $p < .001$). *Summary* delivered the largest gains, 36% vs *Audio*, and 24% vs *Full*. This was also where accuracy benefited most overall. *Summary* improved accuracy from 0.644 (*Audio*) to 0.731 (+13.5%).

The *Chained pattern* reaped: *Audio* 12.32*s*, *Full* 10.74*s*, *Summary* 8.99*s* (Friedman $\chi^2 = 4.33$, $p = .115$; *Summary*<*Audio*, $p = .017$). *Summary* sped up performance by 27% vs *Audio* and 16% vs *Full*; *Summary* vs *Audio* remained significant.

For accuracy, *Summary* gains the largest (*Summary*>*Audio*, $p = .025$; *Summary*>*Full*, $p = .020$), in Memory-based referencing (0.82, 0.83, 0.99; *Audio*, *Full*, *Summary*, respectively).

3 Limitation and Future Works

In this section, we extend the discussion on technical limitations and future work plans of Speech-to-Spatial.

3.1 Advancement of Reasoning LLMs and Graphs

We treat our object-centric graph more than an intermediate representation. It is a persistent memory bank that can retain object attributes and interaction histories across time. As LLM reasoning capabilities improve, we expect the "reasoning layer" to become increasingly capable of resolving under-specified speech with fewer hand-crafted heuristics. However, even with the advancement of capable reasoning LLMs, the need for structured memory remains. Maintaining information persistently across sessions, providing transparent reasoning (what gets remembered, when, and why), and interpretability of how a referent was resolved. A practical next step, would be formalizing our current approach as a graph framework where LLMs issue structured graph queries (filter, traverse, verify) similar to SQL queries, enabling the system to benefit from improved LLM models, while keeping the memory transparent and human-readable.

3.2 Agentic Architecture

Our pipeline currently executes a query, sequentially: Parse, Retrieve, Resolve, and the AR Overlay. And these are based on hand-crafted heuristics. In our future work, we envision Speech-to-Spatial to integrate the agentic pipeline, where the decision making is automated by the system reasoning agents, while maintaining the user agency (verification upon ambiguity) and transparency of our interaction history-projected graph representations.

3.3 Defining the Boundary of Referent Proximity

Our current spatial relation modeling simplifies the "Left" or "Right" relationships into discrete booleans using a fixed constant threshold (radius=50cm). If the two objects are outside the defined radius, we assume the two are not adjacent to each other. However, in practical scenarios, the definition of "to the left" or "to the right of" vary per person and by the granularity the explanation requires. For example, wayfinding instructions, to locate in a city, the context of "next to" is different from explaining the mouse "next to" the keyboard. To this end, we plan to explore the context-dependent (e.g., instruction type, distance) variation of thresholding.

3.4 Information Overload

We currently convert the textual descriptions into vector embeddings during the save, and traverse through each node following a filtering step based on a (cosine) similarity threshold. However, the latency will proportionately increase to the size of the incoming data. To address this challenges, we plan to apply similar techniques as GraphRAG [1], where a hierarchical information (community) forms an abstraction before iterating through the nodes with high semantic similarity.

3.5 Instance Detection and Tracking

Our current prototype assumes a simplified instance identity logic. If an object appears in view within the same space, it is treated as the same instance. This approach may not always succeed, when multiple similar objects exist, objects are rearranged, or sensing is intermittent. A more robust instance management may be required for re-identification across viewpoints, and sessions using a combination of appearance, geometry, and interaction history, and object tracking. We plan to adopt this approach in our future work.

References

[1] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv:2404.16130*, 2024. 3

[2] Y. Kim, Z. Aamir, M. Singh, S. Boorboor, K. Mueller, and A. E. Kaufman. Explainable xr: Understanding user behaviors of xr environments using llm-assisted analytics framework. *IEEE TVCG*, 2025. 2