Human-Machine Teaming with Scene Graphs: LLM Tool Agents for User Interaction

Ho Chit Siu*
hochit.siu@ll.mit.edu
Massachusetts Institute of Technology Lincoln Laboratory
Lexington, MA, USA

Laura Niss*
laura.niss@ll.mit.edu
Massachusetts Institute of Technology Lincoln Laboratory
Lexington, MA, USA

Carlyn Dougherty*
Massachusetts Institute of Technology Lincoln Laboratory
Lexington, MA, USA

Ashley Suh*
ashley.suh@ll.mit.edu
Massachusetts Institute of Technology Lincoln Laboratory
Lexington, MA, USA

Abstract

The development of real-time hierarchical 3D scene graphs allows a machine to quickly build a model of its environment. This model is machine-readable and designed for inference efficiency from a robotics perspective, but lacks a well-designed interface for non-expert users to team with the machine. We introduce a system structure that relies on large language model (LLM) tool agents to interact with, query, and update both the underlying world model and to guide the machine's interactions with the environment. Through two user scenarios derived from discussions with emergency incident responders, we analyze the capabilities of our visual and language interface. Finally, we give recommendations on what information should be collected or derived from the 3D scene graphs to support human-machine teaming (HMT), and the successes and limitations of the current iteration of LLMs to support user interaction.

CCS Concepts

• **Computing methodologies** → **Robotic planning**; • **Human-centered computing** → **Natural language interfaces**; **Human computer interaction (HCI)**.

Keywords

Scene graphs, human-robot interaction, visualization, agentic AI, large language models

ACM Reference Format:
Ho Chit Siu, Laura Niss, Carlyn Dougherty, and Ashley Suh. 2026. Human-Machine Teaming with Scene Graphs: LLM Tool Agents for User Interaction.

*Equal contributions

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Under Secretary of War for Research and Engineering under Air Force Contract No. FA8702-15-D-0001 or FA8702-25-D-B002. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of War for Research and Engineering.

The 3rd InterAI Workshop at CHI 2026, Barcelona, Spain

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In *The 3rd InterAI Workshop: "Interactive AI for Human-Centered Robotics at ACM CHI 2026, April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Motivation

Although domain experts laud the perceived capabilities of robots in helping mitigate active threats and perform urban search and rescue, operating a robot requires in-depth expertise that first responders are not typically trained for. This limits their timely implementation, as both the robot and an expert human controller must be present for use, both of which are rare commodities. Consequently, tools that rely on minimal technical expertise would empower nonexpert-roboticists, removing significant training and organizational overhead when implementing robotics for public safety. In this paper, we present an LLM-based methodology that bridges human and robot world models, supporting non-expert users in teaming with a robot in a similar way to how they would interact with their own human teammate.

2 Background & Related Work

2.1 Scene Graphs in Robotics

3D scene graphs provide a compact data structure that contains geometric and sematic spacial information about scenes that allow for much faster processing of many robotics tasks than representations that are closer to the sensor data. They have been applied in both indoor and outdoor scenes, across a range of different scales, implemented with hierarchical relations, and with open and closed vocabularies for labeling graph nodes and edges [5, 8, 14]. An example set of common attributes found in robotic scene graphs is shown in Table 3 in Appendix C.

2.2 Scene Graphs and LLMs for Task Planning

LLMs have shown promise in planning over scene graphs, as the semantics in node/edge relations helps them effectively plan movements and actions over the scenes, even over long horizons [4, 10, 12]. LLM chain-of-thought reasoning can similarly improve robots' ability to navigate scenes [15].

2.3 Scene Graphs and LLMs for HMT

Scene graphs have primarily been used from a purely robotics perspective, or for simple object-context descriptions (e.g., "the TV is

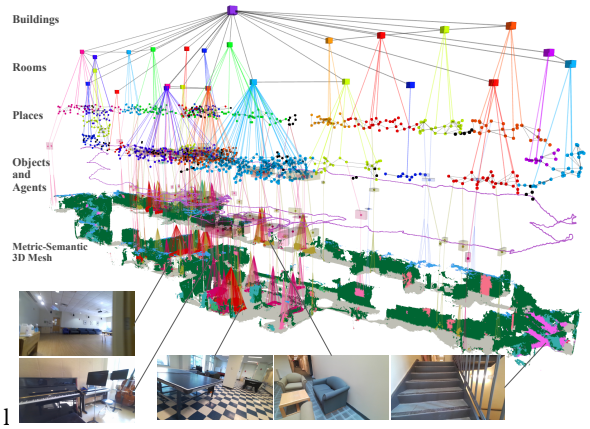

Figure 1: Typical visualization, presented to a user, of a hierarchical scene graph used by robot planners [5].

opposite the sofa" [15]). The complexity of scene graph data makes visualizing them and supporting user interactions difficult – an example is shown in Figure 1. Consequently, most workflows that leverage scene graphs require users to already know what is in the graph, or rely on simple semantic searches to find particular nodes of interest [9]. Much of the work in this domain comes from the robot-developer perspective, e.g., by using unified foundation models to drive robot input/output [16] or by using an LLM to propose what skills a robot could use to address a user's command [1]. In contrast, consideration for how *human users* might use the rich, well-organized data in scene graphs to interact with robots remains an open problem. Our approach in this paper provides a starting point to address this gap.

3 SceneChat System

3.1 Approach

We rely on 3D scene graphs to be the world model for robot teammates. In this work, we assume the scene graph to be both accurate and stable. Real-time creation of hierarchical 3D scene graphs give robots a compact world-state from which to operate alongside current sensor data. These graphs typically generate nodes that information about spatial dimensions of objects, rooms, buildings, and the edges define semantic relationships (*"object A is in room B"*), creating an easily searchable data structure. However, while 3D scene graphs make robot interpretation easier for autonomous systems, the JSON representation and complex graphics supported in current software result in unintuitive human-machine teaming (see Figure 1).

Our system, SceneChat, addresses the world model gap by designing for natural language speech commands and limiting visual outputs to highly relevant information, reducing cognitive load. SceneChat uses tool-calling LLMs that have the ability to execute pre-written and verified code to query a robot's world model (in our case a 3D hierarchical scene graph) to visualize subcomponents of the graphs in a digestible manner, and command robot movement through the scene in a reliable way. To combat the still-significant issue of hallucination, especially when used to summarize or extract information from structured data, we limit the use of raw LLM output and mainly employ the LLM as a tool-calling agent. This ensures that information relayed to the user is the result of verified code rather than a stochastic process. Importantly, SceneChat was developed based on discussions with domain experts as well as the real-world deployment of robots in public safety contexts [3, 11].

3.2 User Design Considerations

In designing SceneChat, we first considered who would be using the system and in what type of scenarios. Across our review of the literature, conversations with domain experts, and real-world deployments of robots in public safety contexts [3, 11], we identified two primary user personas (*Dismounted First Responder* and *Incident Commander*) and response scenarios (*Mass Casualty* and *Active Threat*) as the basis for our design choices. These were developed in consultation with a veteran of a United States military police unit with unrelated robotics experience, a member of the New York City Fire Department's Robotics Unit (which uses a Boston Dynamics Spot robot), and one author's perspective from training for wilderness first response, including mass casualty triage.

3.2.1 User Personas. The **dismounted first responder** is a person who is responding to a scene on foot (e.g. a paramedic, police officer) and likely does not have access to or attention available for a visual display. For these people, commands and queries would be made and received verbally, and interactions may include physically following a robot or performing handoffs with such.

The **incident commander** is a person at a management level of the response, managing multiple subordinates persons or groups. We assume that this person would be at an incident command post and not actively moving around in a scene, and thus able to use a visual display. A person in this role may or may not have direct remote control of an on-site robot, but would not be directly physically interacting with it.

3.2.2 Usage Scenarios. In an **urban mass casualty event**, responders may need to search an area for survivors and perform triage. *Dismounted first responders* could use SceneChat to task robots with searching specific areas of a building for signs of survivors, scouting ahead of first responders, and/or simply increasing the coverage of an area. Instead of relying on a complex control interface that requires continuous monitoring, SceneChat would identify the relevant area from the 3D scene graph, then instruct the robot to begin its search given the found set of coordinates. If a survivor is located, the robot can report immediately back through SceneChat, which could include a visual map of the environment that highlights a recommended safe route for human rescuers to reach the survivor.

An **active threat** scenario involves time-critical responses to ongoing threats, such as a live crime scene or public safety emergency. Here, an *incident commander* may be directing a team of responders who are moving quickly through a building. To prevent overloading the team with constant updates, the commander could instead query a robot on the scene directly via SceneChat. Consequently, SceneChat can provide actionable information at the right level of detail, supporting coordinated decision-making without overloading communication channels.

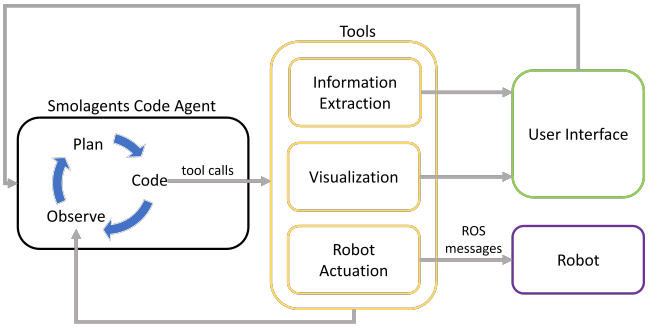

Figure 2: System diagram for SceneChat. The HuggingFace Smolagents framework is used to interact with the user interface and robot via LLM tool calls.

In both usage scenarios, a question arises of how users would have access to a relevant scene graph in the first place, given an emergency. Our interviews with operators indicated that emergency response units often already have maps of potential incident locations in their jurisdictions (e.g. major event venues, schools, large buildings, etc), and in some cases, have the opportunity to rehearse in them. Given work done in dynamic scene graph creation (e.g. [5]), it is not unreasonable for units with robots to build these scene graphs on site as a preliminary step.

3.3 Backend Design

The overall system design of SceneChat is shown in Figure 2. We use the HuggingFace `Smolagents` framework to support agentic, code-based responses to natural language queries [13]. In particular, we use the CodeAgent subcomponent, which takes any large language model and requires that it responds to a query via a three-step process of planning, writing code, and observing the outputs of the code. Code written by the agent may use pre-written functions called "tools" that broadly fall into the categories of 1) information extraction, 2) visualization, and 3) robot actuation.

To facilitate information extraction from the scene graph, we provide the system with tools that list object IDs by object name, provide a scene summary (number of objects of each type), and finding paths between coordinates or object instances. In the latter case, paths can be returned as a set of waypoints (for robot use) or as a set of directions in natural language (for human use). The natural language versions directs the user from room to room, providing a cardinal direction in the first room (door on the north, south, east, or west) since we do not know the user's starting orientation, followed by body-relative directions for all subsequent rooms, since we know the direction of entry. Where possible, the steps also include object cues to allow a user to verify a direction, such as indicating objects that are near doors that the user should take. An example of a set of directions is shown in Figure 3.

Visualization: The primary goal of visualization is to only provide the information that is most needed for a particular prompt. We focus on two types of visualizations: showing object instances, and showing paths, both on a top-down 2D map of the scene at the relevant floor. These maps are presented on the user interface described in Section 3.4 and an example is shown in Figure 3.

Figure 3: The SceneChat interface supports question-answering as well as command and control in natural language. Here, the operator asks about the statuses for both survivors in the scene, directions for how to reach the most injured one, and sends a command to the robot to go to the less injured survivor. In SceneChat, we bypass operators having to interpret a complex 3D scene graph generally produced for roboticists (e.g. Figure 1).

Robot Actuation: Robot actuation is accomplished via commands in Robot Operating System (`ROS`), and have been implemented and tested using a Boston Dynamics Spot robot, as well as a 2D `PyGame` interface for software-only testing of movement control.

The environment was modeled as a graph (distinct from the hierarchical scene graph representation), where each node represented a possible waypoint and edges were paths the robot could take between those waypoints. When a prompt is provided that requires robot movement, the information extraction tools are used to first determine a target set of waypoints that the robot then takes.

We used traditional graph search methods to create a proposed path between the robot's current position to the target location. Once a valid path was found, the system sent the waypoint command to the Spot wrapper. We translated the high-level waypoint into movement commands for Spot. As the robot traveled, the wrapper monitored its position and obstacle sensors to keep it safe and on track, while also updating the system with its current status. For our experiments, Spot traveled in the virtual environment. Rather than map and navigate the real environment, we assumed that we had a pre-existed structured map of the environment as well as the Spot pose. Of course, this assumption would allow us to use other mapping and localization techniques that were not the focus of this effort to create an end-to-end system in the future.

3.4 Frontend Design

The user interface for SceneChat is implemented in `Vue.js` and uses Flask to pass messages back and forth with Smolagent. Figure 3 shows an exemplar scenario of using the frontend for robot-scene graph communication.

There are three visual panels as part of the frontend design. The first is a Summary Panel, which provides an overview of the scene graph uploaded by the user. Currently the summary includes the

Table 1: Scene graph sizes for the small and large evaluation scenes.

	Small Scene	Large Scene
Number of Rooms	7	35
Number of Object Types	12	12
Total Number of Objects	27	88

number of rooms in the scene, object classes, and the ordinal count for each of those classes.

The primary view in the frontend is the Chat Panel, which supports question-answering over the scene graph via Smolagents. The chat accepts both text and speech-to-text, allowing users to ask questions via microphone. User's questions are then sent to the backend and Smolagents runs the appropriate tool(s) to answer the question. Whenever the user asks a question that requires a chart as part of the answer (e.g., a map displaying the robot's path to a survivor), SceneChat shows it in the Visualization Panel. All charts are stored in this panel and a contextual description is included for situational awareness.

Additionally, to meet the requirement that spoken commands be usable for the system, a speech-to-text interface was also used. For users with access to the GUI, the speech-to-text populates the chat input panel and allows for user confirmation that the prompt was correct before entry, though we recognize that such a method would not be possible for a user without a GUI, such as some cases of dismounted responders working directly with a robot.

In the future, we would like to include a visualization of the user's scene graph as part of the interface design. To effectively display the scene graph, we posit that only essential portions of the graph should be displayed in a given moment, and that cognitive load needs to be consistently managed for practical use cases of SceneChat. Past work has investigated optimizing 3d scene graphs for visualization purposes [2, 5], although it is unknown whether these visuals are easily interpretable to end-users – prior research has shown that multi-attributed graph visualizations can be overwhelming [7]. We discuss further in Appendix B.

4 System Evaluation

We perform system evaluation by measuring prompt response accuracy in the SceneChat workflow. We consider two different scenes a *small* and a *large* scene as described in Table 1. The small scene is the iGibson "Soldier" dataset [6], augmented with additional "objects" with the name "survivor" and which had one of three status levels: red, yellow, or green, corresponding to the colors used in medical triage. For consistency, the large scene is a version of the same dataset with the building and a subset of the objects copied and repeated five times in one direction, with doors connecting adjacent copies. All evaluations were performed using the November 20th, 2024 checkpoint of GPT-4o.

In this evaluation, we primarily aim to show the utility of our tool-calling approach for scene graph interaction. Since LLM performance, particularly in tool calling, is expected to improve rapidly, it is less important for us to evaluate on the most up-to-date models. Similarly, SceneChat would likely be used in an edge device (e.g. on a mobile robot or laptop), but we expect that future small language models that are able to run on such hardware to perform on par with older frontier models such as GPT-4o.

Table 2: Task performance (correctness) on typical user interactions. "Strict correctness" requires coordinates or room numbers for "where is [object]" prompts. "Allows directions" accepts both of the above, or directions from a given location to the target object.

Strictness	Small Scene [95% CI]	Large Scene [95% CI]
strict correctness	83.3% [77.8%, 88.8%]	86.1%[81.0%, 91.2%]
allow directions	97.22% [94.8%, 99.65%]	92.8%[89.0%, 96.6%]

We generated six test prompts in each of six categories and manually verified expected outputs based on the data. LLM outputs from five replicates of each specific prompt were compared against human evaluators examining the scene graph and/or writing code to answer the question (without an LLM).

The following types of prompts were used:

(1) Location: "If I am at location (0, 0), where is the nearest [object type]?"
(2) Path: "How do I get from the first to the second [object type]?"
(3) Count "How many [object type] are in the scene?"
(4) Map: "Show me a map of the [object type]."
(5) Survivor:
    - "What is the status of the survivors?"
    - "Where is the [most/least] critically-injured survivor?"
    - "Where are all the [green/yellow/red]-status survivors?"
(6) Compound: These are prompts that more explicitly involve multiple reasoning steps, such as counting the number of number of objects in the scene that can be used to carry water or finding the path between the two survivors that are furthest from each other. See Appendix C.1 for the full list.

4.1 Evaluation Results

Overall evaluation results (Table 2) show generally high performance, though we note some discrepancies with the expected answers for particular prompt types, we discuss below. See Figures 4 and 5 for the full results.

Many of the location-based questions returned what was originally considered incorrect responses (Figure 4 location-type prompts). Instead of returning a location as an (x, y) coordinate or as a room number, the response would be provided in terms of directions to the object. We observed that the LLM was answering the question of "How do I get to the nearest [object]?" that is generally implied by the explicit location-based question of "Where is the nearest [object]?" This ambiguity also affected a small number of questions in other categories. Thus, we present the results under both a strict interpretation where "where" questions require a location to be returned, and an "allow directions" interpretation where such questions are considered correct if a correct path is provided.

5 Conclusion

In this paper we introduce a framework employing tool-calling LLMs to retrieve information from 3D scene graphs for robot control. We contribute two user personas and usage scenarios, informed via interviews with domain experts, leading to the design of SceneChat. Our system includes a user interface to support natural language question-answering for operators without prior scene graph or robot expertise. We evaluated the response accuracy of SceneChat on six test prompts and six categories compared to human evaluators, showing that an LLM-mediated approach has the potential to support human understanding of these data structures for operationally-relevant tasks such as scene assessment, planning, direction-giving, and triage.

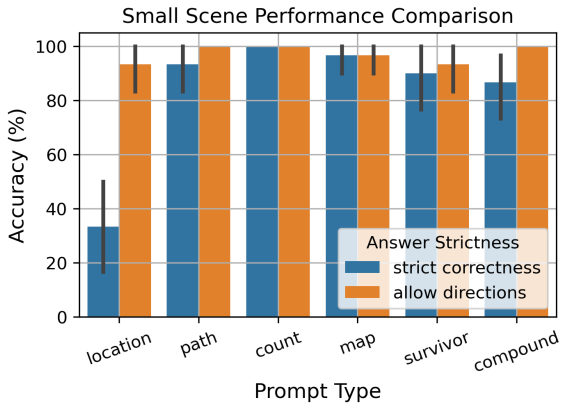

Figure 4: SceneChat performance on the small scene. Error bars represent 95% confidence intervals. "Strict correctness" requires coordinates or room numbers for "where is [object]" prompts. "Allows directions" accepts both of the above, or directions from a given location to the target object.

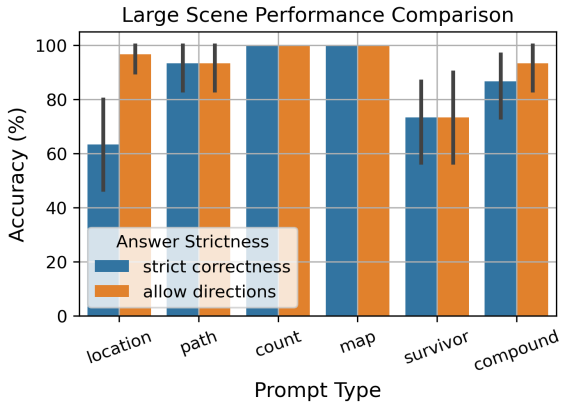

Figure 5: SceneChat performance on the large scene. Error bars represent 95% confidence intervals. "Strict correctness" requires coordinates or room numbers for "where is [object]" prompts. "Allows directions" accepts both of the above, or directions from a given location to the target object.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691* (2022).

[2] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 2019. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5664–5673.

[3] Marco da Silva. 2024. The many ways robots keep people safe. https://www.police1.com/police-products/police-technology/robots/the-many-ways-robots-keep-people-safe.

[4] Zhirui Dai, Arash Asgharivaskasi, Thai Duong, Shusen Lin, Maria-Elizabeth Tzes, George Pappas, and Nikolay Atanasov. 2024. Optimal scene graph planning with large language model guidance. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 14062–14069.

[5] Nathan Hughes, Yun Chang, and Luca Carlone. 2022. Hydra: A Real-time Spatial Perception System for 3D Scene Graph Construction and Optimization. In *Robotics: Science and Systems*.

[6] Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain, et al. 2021. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272* (2021).

[7] Harry Li, Gabriel Appleby, Camelia Daniela Brumar, Remco Chang, and Ashley Suh. 2024. Knowledge Graphs in Practice: Characterizing their Users, Challenges, and Visualization Opportunities. *IEEE Transactions on Visualization and Computer Graphics* 30, 1 (2024), 584–594. doi:10.1109/TVCG.2023.3326904

[8] Xinghang Li, Di Guo, Huaping Liu, and Fuchun Sun. 2022. Embodied Semantic Scene Graph Generation. In *Proceedings of the 5th Conference on Robot Learning*. PMLR, 1585–1594.

[9] Dominic Maggio, Yun Chang, Nathan Hughes, Matthew Trang, Dan Griffith, Carlyn Dougherty, Eric Cristofalo, Lukas Schmid, and Luca Carlone. 2024. Clio: Real-time task-driven open-set 3d scene graphs. *IEEE Robotics and Automation Letters* (2024).

[10] Zhe Ni, Xiaoxin Deng, Cong Tai, Xinyue Zhu, Qinghongbing Xie, Weihang Huang, Xiang Wu, and Long Zeng. 2024. GRID: Scene-Graph-based Instruction-driven Robotic Task Planning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 13765–13772. doi:10.1109/IROS58592.2024.10801291

[11] Margaret Osborne. 2023. Robot Dog Surveys Collapsed New York Parking Garage. https://www.smithsonianmag.com/smart-news/robot-dog-surveys-collapsed-new-york-parking-garage-180982028/.

[12] Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. 2023. SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Robot Task Planning. In *Conference on Robot Learning*. PMLR, 23–72.

[13] Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 2025. 'smolagents': a smol library to build great agentic systems. https://github.com/huggingface/smolagents.

[14] Jared Strader, Nathan Hughes, William Chen, Alberto Speranzon, and Luca Carlone. 2024. Indoor and outdoor 3d scene graph generation via language-enabled spatial ontologies. *IEEE Robotics and Automation Letters* 9, 6 (2024), 4886–4893.

[15] Hang Yin, Xiuwei Xu, Zhenyu Wu, Jie Zhou, and Jiwen Lu. 2024. Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation. *Advances in neural information processing systems* 37 (2024), 5285–5307.

[16] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*. PMLR, 2165–2183.

A Limitations

The framework used by SceneChat is generally applicable to hierarchical scene graphs, but assumes the existence of node labels and coordinates. Practical implementation of the tools may vary slightly across scene graph datasets depending on the specifics of each format (e.g. iGibson [6] vs Hydra [5] implementations), and modifications would have to be made to meet each format. The more widespread a standard is, the easier it would be to use it for such a system.

Additionally, direction-giving tools provided in SceneChat assume that rooms are arranged with orthogonal walls and doors, so the current implementation of those tools need modification

for more unusual or complex geometries. From our conversations with operators, different domains have different conventions for describing buildings and rooms, so user domain adaptation may also be appropriate.

B Future Work

Although our use cases and user personas were developed with expert consultation, this work did not include formal user evaluation of either the software or its use in robotic hardware. Of particular interest would be the task performance (speed, accuracy) of human operators on information retrieval and command and control tasks, as well as the utility of the retrieved information, particularly given the question ambiguities we observed in our current testing.

There is also potential for a generalization of the SceneChat framework – beyond 3D scene graphs – for similar data structures, such as non-hierarchical graphs and non-spatial data, such as social or computer networks. In the current implementation of SceneChat, we limit the visualizations provided to users to connections between objects in the graph. Other generic visualizations are also supported like situated plots of all instances for a particular object. That said, these visualizations are generated using tools (e.g., custom matplotlib functions) in the SceneChat framework. Extending SceneChat to other data types would require manually changing the other functionality of SceneChat, like the way it plots data. An LLM could also be tasked with generating charts, however, we believe embedded visualization tools for an agent to use is the most trustworthy way to display data to end-users.

The structure of scene graphs and SceneChat's method of interacting with them also affords users interesting ways to impose belief states on the graphs. All the scene graphs used in our current evaluation had set nodes that were either sensed or simulated, and therefore had not only semantic relations with other nodes through edges, but were also fixed in space. However, a natural extension of the present work may utilize humans' mental models of space and object relations to add "hypothesized" edges and nodes that have sematic and hierarchical relations to each other (and to any existing graph structure), but that do not have any grounding to specific positions until they are explored.

For example, with such "belief graphs" it is possible to specify that a (sensed) hallway has two rooms to the right, two doors to the left, and that the second door on the left leads to a room with a fire extinguisher. Such a specification adds edge relations to the hallway, additional nodes for each room, and an object-level node beneath one of the room nodes, all without grounding the nodes. Nonetheless, the graph relations may be sufficient for a robot to navigate it based on the grounded portions of the graph, and the existence of particular hypothetical portions of the graph may be confirmed or denied upon exploration. Such a capability would allow for far more effective use of partially-explored scenes, as human knowledge may be combined with robot sensing to drive further exploration and action, rather than purely relying on robot sensing (which may be inefficient without guidance) or human specification (which may be difficult for specifics such as precise positions).

C Scene graph attributes

Scene graphs contain nodes pertaining to rooms and objects, and may additionally have nodes pertaining to the robots taken path and nodes pertaining to open space (where a robot could traverse). These nodes contain information such as is listed in Table 3 with a typical structure as shown in Table 4

C.1 Compound Question List

- If I am at location (0,0), where are the two clocks that are closest to the least critically-injured survivor near me?
- If I am at location (0,0), where is the sink and the vase that are closest to the most critically-injured survivor near me?
- If I am at location (0,0), tell me how to get to the nearest sink and then to the nearest vase from that sink.
- If I am at location (0,0), find the two closest survivors to me, then tell me how to get to the closest one, then the next closest one.
- Find the two survivors that are furthest from each other and give me the directions to get from one to the other.
- How many objects in the scene can I use to carry water from one place to another and what are they?

Table 3: Definitions for attributes found in a typical scene graph.

Attribute	Definition
id	unique identifier
position	3D position in space relative to known starting point
dimensions	3D dimensions of the room or object
parent room	room an object is located within
label	semantically meaningful label if an object
room edge	ids of rooms that it is connected to

Table 4: Typical structure of a scene graph.

nodes	room nodes	id position dimensions
	place nodes	id position
	object nodes	id dimensions position parent room label
edges	room nodes	id place edges room edges
	place nodes	id place ids edges room edges