Research Article

Time Is on My Side: Scene Graph Filtering for Dynamic Environment Perception in an LLM-Driven Robot

Simone Colombani[1], Luca Brini[2], Dimitri Ognibene[2], Giuseppe Boccignone[1]

1. University of Milan, Italy; 2. University of Milan - Bicocca, Italy

Robots are increasingly being used in dynamic environments like workplaces, hospitals, and homes. As a result, interactions with robots must be simple and intuitive, with robots' perception adapting efficiently to human-induced changes.

This paper presents a robot control architecture that addresses key challenges in human-robot interaction, with a particular focus on the dynamic creation and continuous update of the robot's state representation. The architecture uses Large Language Models to integrate diverse information sources, including natural language commands, robotic skills representation, real-time dynamic semantic mapping of the perceived scene. This enables flexible and adaptive robotic behavior in complex, dynamic environments.

Traditional robotic systems often rely on static, pre-programmed instructions and settings, limiting their adaptability to dynamic environments and real-time collaboration. In contrast, this architecture uses LLMs to interpret complex, high-level instructions and generate actionable plans that enhance human-robot collaboration.

At its core, the system's Perception Module generates and continuously updates a semantic scene graph using RGB-D sensor data, providing a detailed and structured representation of the environment. A particle filter is employed to ensure accurate object localization in dynamic, real-world settings.

The Planner Module leverages this up-to-date semantic map to break down high-level tasks into sub-tasks and link them to robotic skills such as navigation, object manipulation (e.g., PICK and PLACE), and movement (e.g., GOTO).

By combining real-time perception, state tracking, and LLM-driven communication and task planning, the architecture enhances adaptability, task efficiency, and human-robot collaboration in dynamic environments.

Corresponding authors: Simone Colombani, simone.colombani@studenti.unimi.it; Luca Brini, l.brini@campus.unimib.it; Dimitri Ognibene, dimitri.ognibene@unimib.it; Giuseppe Boccignone, giuseppe.boccignone@unimi.it

1. Introduction

Immediacy is crucial in assistive robotics[1][2][3]. In a typical human-robot interaction scenario, users may provide commands in natural language, such as "*Pick the blue bottle on the table and bring it to me*". To such aim, the use of Large Language Models (LLM) allows robots to interpret natural language requests and "translate" instructions into plans to

achieve specific goals; yet, these models need to know the environment in which they operate so to generate accurate plans[4]. The need for translation arises from the complexity of human language and the variability in instructions. Users may express commands differently or exploit ambiguous terms that the robot must comprehend. To address these challenges, robotic architectures must integrate natural language processing with environmental understanding.

The chief concern of the work is to exploit scene graphs as semantic maps providing a structured representation of spatial and semantic information of robot's environment. This enables LLMs to generate plans based on this information. Indeed, via scene graphs robots can map the relationships between objects, their properties, and their spatial arrangements.

Here we address such limitations by representing the environment as a graph endowed with updatable semantics that language models can interpret. More precisely, the dynamics of the update is achieved via particle filtering to enhance the reliability and precision of real-time semantic mapping. The model adopted (PSGTR) is lightweight and can be easily utilized, making it suitable for live applications and accessible even on less powerful hardware. Using RoBee, the cognitive humanoid robot developed by Oversonic Robotics, the system dynamically updates the environment graph and replans in case of failure, overcoming challenges in long-term task planning.

2. Related works

A scene graph captures detailed scene semantics by explicitly modeling objects, their attributes, and the relationships between paired objects (e.g., "blue bottle on the table")[5]. 3D scene graphs[6] extend this concept to three-dimensional spaces, representing environments like houses or offices, where each piece of furniture, room, and object is a node. The edges between these nodes describe their relationships, such as a vase on a table or a chair in front of a sofa.

Recent works, such as[7] and[8] have proposed to generate 3D scene graphs from RGB-D images, combining geometric and semantic information to create detailed environmental representations. Scene graphs have been widely used in computer vision and robotics to improve scene understanding, object detection, and task planning. For example, SayPlan[9] integrates 3D scene graphs and LLMs for task navigation and planning, performing semantic searches on the scene and instructions to create accurate plans, further refined through scenario simulations. DELTA[10] utilizes 3D scene graphs to generate PDDL files, employing multiple phases to prune irrelevant nodes and decompose long-term goals into manageable sub-goals, enhancing computational efficiency for execution with classical planners. SayNav[11] constructs scene graphs incrementally for navigation in new environments, allowing the robot to generate dynamic and appropriate navigation plans in unexplored spaces by passing the scene graph to a LLM, thus facilitating effective movement and execution of user requests.

In a crude summary, the main limitations of the above mentioned approaches to build environment representations lie in their reliance on computationally heavy vision-language models (VLMs) and computer vision models. Such models are not designed for precision and often demand significant resources, while lacking the ability to be updated in real time, and thus limiting their practical application.

3. Architecture

Our system is based on two components:

- **Perception Module:** it is responsible for sensing and interpreting the environment and building a semantic map in the form of a directed graph that integrates both geometric and semantic information. Its architecture is explained in detail below.
- **Planner Module:** it takes the information provided by the Perception Module to formulate plans and actions that allow the robot to perform specific tasks. It is composed by the following:
  - Task Planner: Translates user requests, expressed in natural language, into high-level skills.
  - Skill Planner: Translates high-level skills into specific, low-level executable actions.
  - Executor: Executes the low-level actions generated by the Skill Planner.
  - Controller: Monitors the execution of actions and manages any errors or unexpected events during the process.
  - Explainer: Interprets the reasons of execution failures by analyzing data received from the Controller and provides suggestions to the Task Planner on how to adjust the plan.

These components interact to allow the robot to understand its environment and act accordingly to satisfy user requests. In what follows we specifically address the Perception Module while details on the planner will be provided in a separate article.

Robot Hardware. The system was implemented using RoBee, the cognitive humanoid robot developed by Oversonic Robotics. RoBee, shown in Figure 1, stands 160 cm tall and weighs 60 kg. It features 32 degrees of freedom, and is equipped with cameras, microphones, and force sensors.

3.1. Perception module

The Perception Module is the component responsible for building a representation of the environment, which the robot can use for task planning. The representation takes the form of a semantic map, a graph that integrates both geometric and semantic information about the environment. To generate the semantic map, the perception module uses data from various sensors. It requires RGB-D frames obtained from the camera which are then processed using a scene graph generation model, such as PSGTR[12] to extract objects masks, label and relationships. Also it uses data on the camera position relative to the geometric map to determine the location of the objects identified by the model. More formally, a Semantic Map is represented as a directed graph $G_m = (V_m, E_m)$ where:

- A node $v \in V_m$ can be one of the following types:
  - Room node: Defines the different semantic areas of the environment, such as “kitchen,” “living room,” or “bedroom.” Each room node contains information about its geometric boundaries and the object nodes it contains;
  - Object node: Represents physical objects in the environment, such as “table,” “chair,” or “bottle.” Each object node contains information about its 3D position, semantic category, dimensions, and other relevant properties:
- An edge $e \in E_m$ can represent:
  - The relationship between two objects;
  - The connection between two rooms;
  - The belonging of an object to one and only one room.

The presence of room nodes is important because it facilitates the categorization of objects based on their respective rooms, which helps distinguish between objects with the same name and enhances the natural language description of the task, while room nodes enable the application of graph search algorithms for planning paths to objects. Room nodes are created based on the geometric map, while object nodes are generated following the steps explained below.

As to edges, more specifically:

- **Edges between rooms** directly connect two rooms and facilitate navigation between them.
- **Edges between objects** represent the relationships between objects and are directed, the direction capturing the influence of one object on another; the label associated with each edge is derived from the inferences made by the PSGTR model.

Figure 1 shows an example of a semantic map of an office, built with the room node 'Office' (italian, 'Ufficio') and the object nodes connected to each other by relationships and linked to the room node.

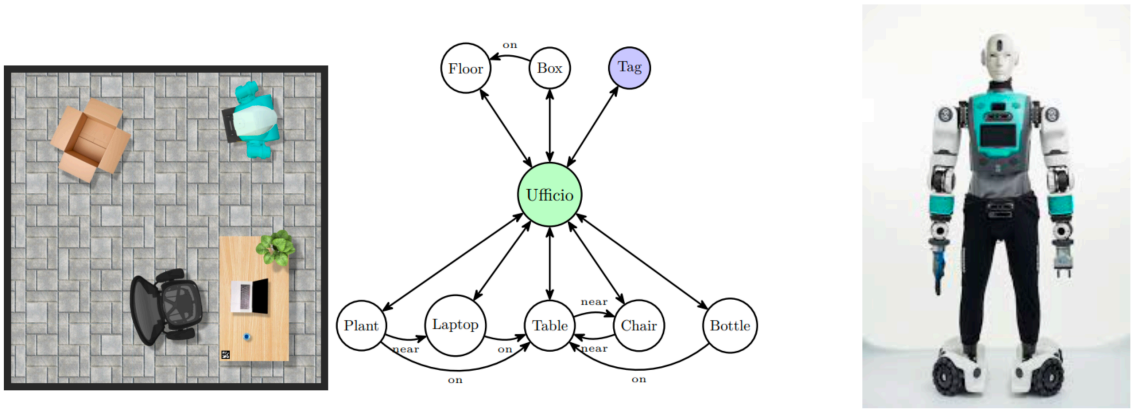

Figure 1. The figure on the left showcases an example of a semantic map in an office environment, while the image on the right shows RoBee, the humanoid robot developed by Oversonic Robotics.

Generating and updating the semantic map

The scene graph generation process is based on the PSGTR model, a single-stage model built on the Transformer architecture[13]. This model generates a graph representation of a scene given its panoptic segmentation. PSGTR does not achieve the highest quality in panoptic segmentation compared to better models, but it provides reasonable inference times for real-time applications, taking about 400 ms to process a 480p image on a machine with access to an NVIDIA T4 GPU.

The Perception Module uses the result of PSTGR and builds the semantic map following the steps below:

1. **Reading RGB-D frames:** The video frames from the robot's cameras are sent to the model to be analyzed and used to generate the scene graph.
2. **Reading robot poses:** To generate the scene and semantic map, it is necessary to know the robot's position relative to the geometric map, the camera's position relative to the map, and the camera's mounting position on the robot.

3. **Inference:** Each received frame is processed by the model. Results are information about detected objects, such as labels and masks, and the relationships between them, such as relationship labels and associated probabilities.
4. **Graph construction:** This step involves extracting data from the object returned by the model and computing values dependent on the robot system, such as the position of objects. At a finer level it consists of three sub-steps:
    1. **Node construction:** Classes and masks of detected objects are extracted. Next, the 3D position of each object is computed, starting in the pixel coordinate system, then transforming to the camera system, and finally to the robot's map coordinate system. Nodes for the semantic scene and the semantic map are instantiated using the appropriate 3D coordinates. A distance-based filter is applied to prune objects that are too far from the robot to avoid issues with object detection and tracking.
    2. **Edge construction:** Data about relationships between objects are extracted. For each relationship, the source and target object indices are identified. If both objects meet distance constraints and the relationship probability exceeds a defined threshold, an edge is created between the corresponding nodes.
    3. **Inference improvement through Particle Filter (PF):** As the model's output is not accurate regarding mask inference, this leads to errors in calculating the object's centroid for obtaining its position relative to the map. A PF based on previous observations is applied to improve the accuracy of the result.

At the end of the process, the semantic map is updated with the new information, and the semantic scene is generated and provided to the planner module.

The PF is used to track the object masks in real-time, provided as output by the PSGTR model, and to improve the estimation of their position in space. During the update process, the filter uses information from frames acquired to refine the position estimate of the objects. The last object masks identified by the PSGTR model are compared with previous ones using the Intersection over Union (IoU) metrics and by applying the motion model, which can be defined as a transformation of the camera position relative to the map between two time instances. Denote the transformation matrices describing the camera position at time $t-1$ and at subsequent time t, $\mathbf{T}_{t-1}$ and $\mathbf{T}_t$, respectively; then, the change in position and orientation can be expressed by the transformation matrix $\mathbf{\Delta T} = \mathbf{T}_t \mathbf{T}_{t-1}^{-1}$. To associate objects between successive frames, we use an IoU matrix computed over segmentation masks. For two masks A and B, IoU is defined as $\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}$, where $|A \cap B|$ represents the area of intersection between masks A and B, and $|A \cup B|$) represents the area of their union. To compare segmentation masks between two successive frames, we denote the segmentation mask at time $t-1$ as M_{t-1} and at time t as M_t. The transformation matrix $\mathbf{\Delta T}$ is applied to the previous mask to obtain a transformed mask M'_{t-1} such that $M'_{t-1} = \mathbf{\Delta T} \cdot M_{t-1}$. The Intersection over Union (IoU) is then computed between the transformed mask M'_{t-1} and the current mask M_t as follows: $\text{IoU}(M'_{t-1}, M_t) = \frac{|M'_{t-1} \cap M_t|}{|M'_{t-1} \cup M_t|}$. This allows us to identify the same object across successive frames based on their masks.

- $\mathbf{T}_t$ is the transformation matrix that describes the camera position at time t.
- $\mathbf{T}_{t-1}$ is the transformation matrix that describes the camera position at time $t-1$.
- $\mathbf{\Delta T}$ is the transformation matrix representing the change in camera position between the two time instances.

More formally, each object is represented by a set of N particles, where each particle s_i^t at time t is a 3D vector representing a hypothesis about the object's position: $s_i^t = [x_i, y_i, z_i]^T$, where $i = 1, \ldots, N$. The particles are initialized with a normal

distribution around the initially observed position $\mu_0 = [x_0, y_0, z_0]^T$: $s_i^0 \sim \mathcal{N}(\mu_0, \Sigma_0)$, where $\Sigma_0 = \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_z^2)$ is the initial covariance matrix. Initial weights are uniform: $w_i^0 = \frac{1}{N}$, where $i = 1, \ldots, N$. Prediction takes into account the camera motion. If $T_{t-1,t}$ is the transformation matrix from frame $t-1$ to frame t, each particle is updated as $s_i^t = T_{t-1,t} \cdot s_i^{t-1} + s_i^0$, where s_i^0 represents the noise added to account for uncertainties in motion, maintaining the same distribution structure used for initial particle initialization. Given a new observation s_{new}, the particle weights are updated based on the Euclidean distance between the predicted position and the observed one: $d_i^t = \|s_i^t - s_{new}\|_2$ and $w_i^t = \frac{1}{1+d_i^t}$. Weights are then normalized: $w_i^t = \frac{w_i^t}{\sum_{j=1}^N w_j^t}$. The final position of the object $\hat{s}_t$ is estimated as the weighted mean of all the particles: $\hat{s}_t = \sum_{i=1}^N w_i^t s_i^t$.

Table 1 shows the improvement obtained over 30 measurements using particle filter.

The overall process for updating the semantic map using the particle filter can be summarized by the algorithm 1.

Property	No Particle	Particle
Real position [m]	(0.67, 0.10, 0.95)	(0.67, 0.10, 0.95)
Mean position [m]	(0.74, -0.08, 0.93)	(0.65, 0.08, 0.94)
Mean of absolute error [m]	(0.07, 0.18, 0.02)	(0.02, 0.02, 0.01)
Error standard deviation [m]	(0.35, 0.24, 0.03)	(0.17, 0.12. 0.02)

Table 1. Comparison of position data

Algorithm 1 Semantic Map update using Particle Filter

for each frame t **do**
  for each object k **do**
    Apply transformation: $M'_{t-1} = \Delta\mathbf{T} \cdot M_{t-1}$ ▷ Transform previous masks
  end for
  Compute IoU$(M'_{t-1}, M_t) = \frac{|M'_{t-1} \cap M_t|}{|M'_{t-1} \cup M_t|}$ ▷ Compute IoU between nodes and inference results
  for each object k **do**
    if IoU $> \lambda_{\text{IoU}}$ **then**
      Update weights: $d_i^t = \|s_i^t - s_{\text{new}}\|_2$, $w_i^t = \frac{1}{1+d_i^t}$
      Normalize: $w_i^t = \frac{w_i^t}{\sum_{j=1}^N w_j^t}$
      Estimate: $\hat{s}_t = \sum_{i=1}^N w_i^t s_i^t$
    end if
  end for
  for each unmatched observation **do**
    Init new object: $s_i^0 \sim \mathcal{N}(\mu_0, \Sigma_0)$
  end for
  Update semantic map with $\hat{s}_t$
end for

Algorithm 1.

4. Conclusions

Scene graphs provide a structured representation that captures geometric and semantic information about the environment. This comprehensive understanding enables improved task planning with large language models, allowing robots to execute commands.

In this article we have shown how to use real-time sensor data to dynamically update semantic maps, thus enabling the robot to adapt to ongoing changes in their environment, particularly in collaborative settings influenced by human actions. Here, particle filtering is applied to improve geometric data precision and semantic map accuracy. This can be particularly important also for social interaction and intention prediction[14][15] other than physical interaction with the environment.

The issues addressed in this work are cogent. Indeed, the effectiveness of planners in translating complex instructions into actionable plans relies on a robust state representation. Without an accurate semantic map, planners risk generating plans that misalign with the actual environment, potentially leading to task failures. The integration of semantic and geometric insights permits robots to reason about their environment in a more informed and adaptive way, ensuring that they can operate effectively and responsively in dynamic environments.

The adoption of a semantic map containing rich spatial information combined with a flexible LLM based planner can easily allow to explore in the future the introduction of new spatial relationships, e.g. wrapped, stuck under, surrounding, aligned, that could support specific novel robot skills[16].

Notes

The Planner Module mentioned in the article is discussed in the following paper: https://arxiv.org/abs/2411.15033)

___

Workshop on Advanced AI Methods and Interfaces for Human-Centered Assistive and Rehabilitation Robotics (a Fit4MedRob event) - AIxIA 2024, November 25–28, 2024, Bolzano, Italy.

Statements and Declarations

Online Resources

More information about RoBee and Oversonic Robotics are available:

- RoBee,
- Oversonic Robotics

Acknowledgments

Special thanks to Oversonic Robotics for enabling the implementation of the system using their humanoid robot, RoBee.

References

1. ^Di Napoli C, Ercolano G, Rossi S (2023). "Personalized home-care support for the elderly: a field experience with a social rob ot at home". User Modeling and User-Adapted Interaction. **33** (2): 405–440.
2. ^Lucignano L, Cutugno F, Rossi S, Finzi A (2013). "A dialogue system for multimodal human-robot interaction". Proceedings of the 15th ACM on International conference on multimodal interaction. pp. 197–204.
3. ^Ognibene D, Mirante L, Marchegiani L. "Proactive intention recognition for joint human-robot search and rescue missions t hrough monte-carlo planning in pomdp environments". In: Social Robotics: 11th International Conference, ICSR 2019, Madri d, Spain, November 26--29, 2019, Proceedings 11. Springer; 2019. p. 332--343.
4. ^Galindo C, Fern\uooe1ndez-Madrigal JA, Gonz\uooe1lez J, Saffiotti A (2008). "Robot task planning using semantic maps". R obotics and autonomous systems. **56** (11): 955--966.
5. ^Zhu G, Zhang L, Jiang Y, Dang Y, Hou H, Shen P, Feng M, Zhao X, Miao Q, Shah SAA, et al. Scene graph generation: A compre hensive survey. arXiv e-prints. 2022:arXiv--2201.
6. ^Armeni I, He ZY, Gwak J, Zamir AR, Fischer M, Malik J, Savarese S. "3d scene graph: A structure for unified semantics, 3d spac e, and camera". In: Proceedings of the IEEE/CVF international conference on computer vision. 2019. p. 5664–5673.
7. ^Gu Q, Kuwajerwala A, Morin S, Jatavallabhula KM, Sen B, Agarwal A, Rivera C, Paul W, Ellis K, Chellappa R, et al. Conceptgra phs: Open-vocabulary 3d scene graphs for perception and planning. In: 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE; 2024. p. 5021–5028.
8. ^Chang H, Boyalakuntla K, Lu S, Cai S, Jing E, Keskar S, Geng S, Abbas A, Zhou L, Bekris K, et al. Context-aware entity groundi ng with open-vocabulary 3d scene graphs. arXiv preprint arXiv:2309.15940. 2023.
9. ^Rana K, Haviland J, Garg S, Abou-Chakra J, Reid I, Suenderhauf N (2023). "Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning". 7th Annual Conference on Robot Learning.
10. ^Liu Y, Palmieri L, Koch S, Georgievski I, Aiello M. "DELTA: Decomposed Efficient Long-Term Robot Task Planning using Lar ge Language Models". arXiv e-prints. 2024; arXiv--2404.
11. ^Rajvanshi A, Sikka K, Lin X, Lee B, Chiu HP, Velasquez A. Saynav: Grounding large language models for dynamic planning to navigation in new environments. Proceedings of the International Conference on Automated Planning and Scheduling. 2024; **34**:464–474.
12. ^Yang J, Ang YZ, Guo Z, Zhou K, Zhang W, Liu Z. "Panoptic scene graph generation". In: European Conference on Computer Vi sion. Springer; 2022. p. 178–196.
13. ^Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017). "Attention is all you need". I n: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17. Red Hook, NY, USA: C urran Associates Inc.; 2017. p. 6000–6010.
14. ^Ognibene D, Chinellato E, Sarabia M, Demiris Y (2013). "Contextual action recognition and target localization with an active allocation of attention on a humanoid robot". Bioinspiration & biomimetics. **8** (3): 035002.
15. ^Rossi S, Staffa M, Bove L, Capasso R, Ercolano G. User's personality and activity influence on HRI comfortable distances. In: S ocial Robotics: 9th International Conference, ICSR 2017, Tsukuba, Japan, November 22-24, 2017, Proceedings 9. Springer; 201 7. p. 167–177.

16. ^ *Marocco D, Cangelosi A, Fischer K, Belpaeme T (2010). "Grounding action words in the sensorimotor interaction with the world: experiments with a simulated iCub humanoid robot". Frontiers in neurorobotics. 4: 1308.*

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.