VDB-based Spatially Grounded Semantics for Interactive Robots

Lorenzo Ferrini
PAL Robotics
Barcelona, Spain
lorenzo.ferrini@pal-robotics.com

Séverin Lemaignan
PAL Robotics
Barcelona, Spain
severin.lemaignan@pal-robotics.com

***Abstract*—This paper presents a new approach for representing spatially-grounded semantics in interactive robots. The method combines spatial and symbolic data to improve robot interactions in human-occupied environments. A key feature is a voxel-based data structure optimized for dynamic and sparse information, along with a global lookup table to manage and track spatially-grounded entities and their relationships. The implementation, which is integrated into a ROS 2-based framework, allows for seamless querying through semantic web APIs such as SPARQL. Initial tests demonstrate the efficiency of this system in supporting advanced scenarios in human-robot interaction. All the repositories developed as part of this contribution can be found at github.com/RepresentationMaps.**

***Index Terms*—Semantic Mapping, Knowledge Representation, Interactive Robots, Human-Robot Interaction**

I. Introduction

Humans naturally combine and mix symbolic and spatial references in their interactions to enhance communication and reduce ambiguities. This behaviour has long influenced the way people attempt to communicate with robots, aiming to use the same blend of symbolic and spatial information when referring to objects and entities in the environment [1], [2].

To achieve such interactions, robots need to develop world representation capabilities that align with those of humans. In this work, we introduce a new technique for spatially-grounded semantic representations in interactive robots. Our approach leverages a structure that incorporates both spatial and symbolic elements, modelling the mixed nature of semantics in a compact, efficient, and comprehensive manner.

Specifically, we focus on defining an architecture suited for robots that interact with humans, ensuring it can capture the continuously evolving and dynamic nature of human-affected environments at a high pace. In detail, we describe the structures and algorithms used to link spatial regions and volumes, regardless of shape, to symbolic concepts stored in an independent RDF-based knowledge base, using VDBs [3] as an efficient data structure to handle sparse 3D information and facilitate seamless integration with the knowledge base for real-time updates and queries. Additionally, we outline the programmatic interfaces, based on SPARQL, that allow for seamless insertion and retrieval of spatially-grounded semantics within a SPARQL-based framework.

This work has been funded by the H2020 PERSEO project (no. 955778).

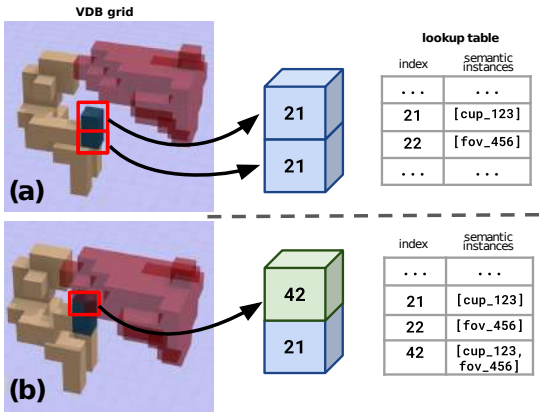

Fig. 1. An example of the lookup table update process. Picture (a) and (b) are (voxelized) representations of a person handing over a bottle, with the person's field of view represented in red. In (a), the bottle is not yet in the field of view: the bottle's voxels in the VDB grid are both attached to the same ID. When the bottle enters the person's field of view (Picture (b)), a new index (42) is created to represent regions of space attached to both the bottle and the person's field of view, and the bottle's voxels are updated accordingly.

Differently from most of the previous works in the literature, we do not focus on sensor-dependent approaches nor we limit to the representation of information related to the physical elements of the environment. The methodology presented in this works attempts to generically target the representation of spatially-grounded semantic information, either this one being associated with physical (objects, humans, etc.) or abstract (fields of view, social space, etc.) elements of the environment.

II. Related Work

Various solutions have been proposed in the literature to represent combined symbolic and spatial information for robots. In [4], the authors present a perspective-taking approach designed to resolve ambiguities in speech interactions between humans and robots. This method employs an ontology-based approach [5] to associate objects in space with semantic concepts, where the semantics may refer to object classes or the spatial relationships between objects in a scene. This framework primarily rely on pre-computed CAD models to represent the environment, making it less suitable for continuous field representations, even though extensions have been proposed [6].

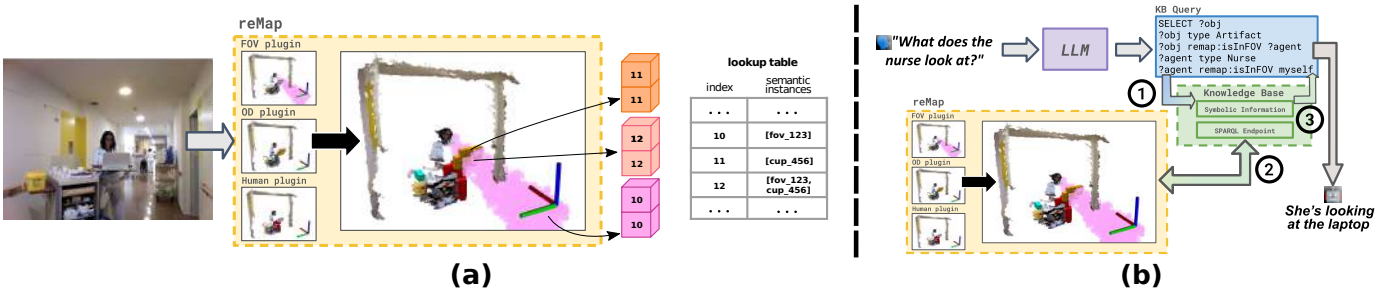

Fig. 2. On the left, a detail of how reMap builds a VDB grid and a lookup table starting from real world data. In the input frame, a nurse is looking at the laptop. We highlight the voxels representing the nurse's field of view and the laptop. On the right, an overview of a general architecture for natural speech grounding based on the methodology proposed in this work. The implementation of an LLM as a natural translator between natural speech and programmatic interface will be explored in future works.

Another approach, *SEMAP* [7], introduces a joint representation of spatial and semantic information, primarily focusing on static components of environments, such as furniture, walls, and rooms. However, it does not emphasize the dynamic, rapidly changing aspects of the environment, such as human-related components or the objects they interact with.

The *Vimantic* framework [8] offers a client-server architecture with custom plugins for managing semantic maps, integrating 2D images and 3D data (such as point clouds and lidar scans) to detect objects and create labeled 3D maps of the environment. As in [7], this work does not target spatial information that is not associated with actual objects/physical elements of the scene, such as field of views, personal space and other abstract elements that have, though, a spatial grounding in the environment.

In previous work, we introduced *reMap* [9], a framework for spatially-grounded semantics in interactive robots. *reMap* comes as a ROS-based package that employs a voxel-based approach to store semantic information directly in a discretized 3D grid. We focused on building an efficient, real-time framework for both storing and retrieving spatial information, including continuos fields. However, the proposed solution requires to build a 3D representation of the world for each different type of symbolic information (one for objects, one for rooms, etc.), limiting its scalability in scenarios where a range of different types of spatially-grounded semantic information has to be handled.

III. Joint Representation of Symbolic and Spatial Information

We define the problem of *spatially-grounded semantic representation* as building and continuously updating a function f mapping 3D regions of space to their (semantic) content. We say that semantic entities are *attached* to regions of space, noting that *attached* might mean interchangeably *contained* (eg, 'the mug is inside that particular region') or *contains* (eg, 'that particular region is inside the robot's field of view').

Considering that a given region of space might be attached to several distinct semantic entities, we can define f as:

$$f : \mathcal{P}(\mathbb{R}^3) \rightarrow \mathrm{ID}^n \tag{1}$$

with $\mathcal{P}(\mathbb{R}^3)$ the set of all subsets of $\mathbb{R}^3$ (ie, all possible volumes of space); *ID* a unique identifier of a semantic entity (in practice, an ID defined in the robot knowledge base, like and RDF identifier); n the maximum permissible number of different semantic entities in a given region of space (in our implementation, n is essentially unbounded).

As an example, consider a book laying on a table, with a robot looking at it. The region of space encompassing the book (a group of voxels in our system) might be associated to the IDs `book_326`, `fov_632`, `room_711` and `zone_943`, semantically representing in the knowledge base, respectively the book, the robot's field of view, the room in which the table is located, and the abstract space 'on top of the table'.

In the next section, we present an implementation of the data structures and the algorithms required to represent and manage the mapping defined by f.

A. Data structure

A variety of data structures for representing 3D-structured information have been proposed in the literature [10]. However, the type of information we aim to represent—spatially-grounded semantic information—has distinct characteristics. First, it tends to be sparse, and second, it can be highly dynamic, necessitating efficient structures for both insertion and update (writing), as well as for retrieval (reading). Given these requirements, we have opted for a VDB-based implementation.

VDBs [3] are particularly well-suited for efficiently handling 3D-structured information, and have already been used with the same purpose in related previous works [9]. VDBs are voxel-based structures and a key parameter is the voxel size, which can be adjusted depending on the specific use case, balancing detail and performance. The standard library for working with VDBs is OpenVDB, which supports a fixed range of data types. As such, OpenVDB does not allow for the creation of VDBs that can natively store custom data types, such as lists of semantic instances (as required to implement the representation function in 1).

To address this limitation, our algorithm creates on-demand abstract semantic classes corresponding to the union of the semantic entities found in a particular voxel, and assign a unique integer index to the abstract class. That index is then

stored in each corresponding voxel of the VDB (Fig. I). A global lookup table is used to link back a voxel index to the list of semantic entities that it represents.

B. Algorithms

Given the data structures introduced in the previous section, we now describe the algorithms for inserting and removing the spatial representation of an instance from the VDB. For the insertion algorithm, we assume that the volume grounding the instance, defined as a set of points P, has already been determined. The insertion procedure involves iterating over each point in P, and, if necessary, updating the integer value associated with that point in the VDB structure. This update is performed based on the index of the voxel. The pseudo-code for the algorithm is reported in Alg. 1.

Algorithm 1 Instance insertion algorithm

Input:

- **points**: set of points in the VDB associated with the instance.
- **instance**: The ID of the symbolic instance to be inserted.
- **register**: the lookup table.

```
for all p ∈ points do
    if p is not active then        ▷ Point not associated with any instance
        index ← register.find_index({instance})
        if index is null then
            index ← register.add_entrance({instance})
        end if
        p.active ← True
        p.value ← index
    else        ▷ Point already associated with at least one instance
        instances ← register.find_entrance(p.value)
        if instance ∉ instances then
            index ← register.find_index(instances ∪ {instance})
            if index is null then
                index ← register.add_entrance(instances ∪ {instance})
            end if
            p.value ← index
        end if
    end if
end for
```

Being p the number of points in P, a the number of active indices, Alg. 1 complexity is $\mathcal{O}(p * a)$.

Similarly, the deletion algorithm also requires to iterate over the points associated with the instance to remove, and accordingly update the lookup table. The pseudo-code implementation of the algorithm is reported in Alg. 2. Being p_a the number of active points in the VDB, a the number of active indices, the complexity of Alg. 2 is $\mathcal{O}(p_a * a)$

Algorithm 2 Instance deletion algorithm

Input:

- **instance**: The ID of the symbolic instance to be removed.
- **register**: the lookup table.
- **vdb**: VDB storing grid.

```
indexes ← register.find_indexes(instance)  ▷ Find indices containing instance
points ← vdb.find(indexes)  ▷ Retrieve voxels assigned with these indexes
local_register ← register.filter(indexes)  ▷ Create a smaller lookup table
for all p ∈ points do
    instances ← local_register.find_entrance(p.value)
    updated_instances ← instances − instance
    if updated_instances is empty then
        p.active ← False
        continue
    end if
    index ← register.find_index(updated_instances)
    if index is null then
        index ← register.add_entrance(updated_instances)
    end if
    p.value ← index
end for
```

IV. ROS 2 IMPLEMENTATION

Given some of the common aspects between this work and reMap, this work is implemented as an extension of the reMap ROS 2 package[1].

While the original implementation of reMap focused on a multiple VDBs, each one representing a different semantic aspect of the environment, the technique presented in Sec. III focuses on a single map approach. For this, we have both adapted previously existing code and *ex novo* implemented the required data structures which were not present in the original implementation of the package. As in the original implementation, the whole framework is based on ROS 2, publicly available and open-source. The following packages are part of our adaptation of the original framework:

- `representation_plugin_base` and `representation_plugins`: reMap follows a modular approach, where every module (that is, plugin) has the role to materialise a specific type of semantic information (objects, humans, rooms, etc.) into the VDB. These two packages provide the base class for the plugins and the implementation of a few of them. These modular components are implemented through `pluginlib` and subscribe to the required ROS 2 topics to receive the information to be mapped into the VDB. In `representation_plugin_base` is also available the implementation of the lookup table described in Sec. III-A.

[1] https://github.com/RepresentationMaps

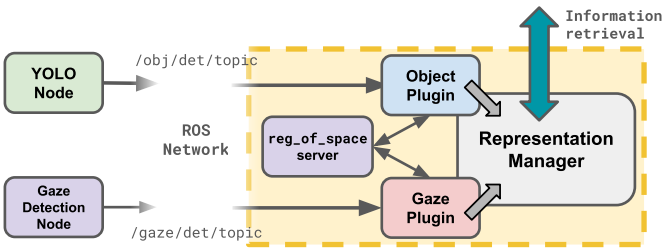

Fig. 3. Schematic representation of the extended reMap framework (in yellow) within a generic ROS 2 network. Independent nodes publish information (such as detected objects, gaze direction for detected humans). Each plugin is specialised in mapping this semantic information in both the knowledge base (not visible here) and the VDB.

- `map_handler`: a wrapper for the original integer-based VDB implementation from OpenVDB. It offers a simplified API to add and remove spatial information from the VDB, including managing the lookup table.
- `representation_manager`: this package provides the central component managing the plugins. Its role is to make sure that the results of each (independent) plugin are correctly mapped into the VDB, and exposing the information to interfaces for information querying/retrieval.
- `reg_of_space_server`: implementation of a server that provides unique IDs for newly added instances. The provided IDs will be used in the knowledge base to identify the instance, as well in the lookup table to define new voxel indices.

A schematic representation of the extended reMap framework in action inside of a ROS 2 network is presented in Fig. IV.

V. Preliminary evaluation

In order to evaluate the execution of the newly proposed implementation of the reMap framework, we have tested the spatial grounding performance on a set of rosbags depicting daily situations in healthcare and hospital environments (see the input image to Fig. IIa).

We have adapted the plugins available in the original reMap implementation to the new API, and we have tested the performance of the framework directly on the

hardware of the robot. The results obtained align with those from the original work, and enable building and updating in real time the semantic representation of the environment.

VI. Integration with semantic web APIs

By having control over the VDB representing the spatial grounding of semantic information, the Representation Manager can also compute the spatial relationships between the instances. These spatial relationships can be added to the knowledge base, and then queried through semantic web APIs such as SPARQL (in case the knowledge base exposes a SPARQL endpoint), following a *push* approach (ie, new semantic entities are pro-actively added to the knowledge base by the extended reMap framework when detected).

We also have implemented the prototype of a custom SPARQL handler to query on-demand reMap for specific spatial properties of the environment (like 'what are all the objects on top of the table' or 'what objects are currently in the field of view of this human?'). This *pull* approach scales better (spatial relations are only computed when required), but do not offer the ability for the robot to react to spatial events like 'tell me when the human looks at the book', a mechanism that is available in knowledge bases like KnowledgeCore[2], that we use on our robots. We will explore more in-depth this point in future works, where full testing of the whole architecture (including involving LLMs for natural speech-based interaction, as in Fig. IIb) will be performed.

VII. Conclusions

In this paper, we have introduced a novel methodology for representing spatially-grounded semantic information in a compact and computationally efficient manner. This contribution addresses the specific requirements for spatially-grounded semantic representation in human-robot interaction (HRI) and is built on ROS 2 to ensure compatibility with existing robotic systems.

The supplementary material provided with this paper includes a C++ implementation of the algorithms and data structures discussed in previous sections. Additionally, an interactive demo is available, allowing users to explore the management of different instances and visually assess the performance of the system.

Although the current implementation does not support multi-threading, the algorithms presented can be adapted for multi-threading and parallel processing. Moreover, incorporating a reverse lookup table and hashing mechanism could reduce the complexity of certain operations in Alg. 1 and Alg. 2, enabling asymptotically constant-time lookups. Future work will explore these enhancements to improve performance further.

Furthermore, the implementation could be optimized with discrete GPU acceleration. Existing GPU-friendly implementations of VDB structures, such as NanoVDB [11], could significantly enhance the efficiency of the system. We plan to extend our reMap-based implementation with such structures in future developments to harness these advantages.

Statement on the use of Generative AI tools

Generative AI has been used to create the first version of the abstract of this paper. The result has then been further reworked by the authors.

References

[1] S. Li, R. Scalise, H. Admoni, S. Rosenthal, and S. S. Srinivasa, "Spatial references and perspective in natural language instructions for collaborative manipulation," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2016, pp. 44–51.

[2]https://github.com/severin-lemaignan/knowledge_core

[2] M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, and D. Brock, "Spatial language for human-robot dialogs," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 2, pp. 154–167, 2004.

[3] K. Museth, "VDB: High-resolution sparse volumes with dynamic topology," *ACM transactions on graphics (TOG)*, vol. 32, no. 3, pp. 1–22, 2013.

[4] S. Lemaignan, R. Ros, E. A. Sisbot, R. Alami, and M. Beetz, "Grounding the interaction: Anchoring situated discourse in everyday human-robot interaction," *International Journal of Social Robotics*, vol. 4, pp. 181–199, 2012.

[5] S. Lemaignan, R. Ros, L. Mösenlechner, R. Alami, and M. Beetz, "ORO, a knowledge management platform for cognitive architectures in robotics," in *2010 IEEE/RSJ International conference on intelligent robots and systems*. IEEE, 2010, pp. 3548–3553.

[6] A. K. Pandey and R. Alami, "Mightability maps: A perceptual level decisional framework for co-operative and competitive human-robot interaction," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 5842–5848.

[7] H. Deeken, T. Wiemann, and J. Hertzberg, "Grounding semantic maps in spatial databases," *Robotics and Autonomous Systems*, vol. 105, pp. 146–165, 2018.

[8] D. Fernández-Chaves, J.-R. Ruiz-Sarmiento, N. Petkov, and J. Gonzalez-Jimenez, "ViMantic, a distributed robotic architecture for semantic mapping in indoor environments," *Knowledge-Based Systems*, vol. 232, p. 107440, 2021.

[9] L. Ferrini, J. Palmieri, A. Marino, D. Lee, and S. Lemaignan, "reMap: Spatially-grounded and queryable semantics for interactive robots," in *International Conference on Social Robotics*. Springer, 2024.

[10] M. Aleksandrov, S. Zlatanova, and D. J. Heslop, "Voxelisation algorithms and data structures: a review," *Sensors*, vol. 21, no. 24, p. 8241, 2021.

[11] K. Museth, "NanoVDB: A GPU-friendly and portable VDB data structure for real-time rendering and simulation," in *ACM SIGGRAPH 2021 Talks*, 2021, pp. 1–2.