UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería Informática

Creación y Aprovechamiento de Mapas Semánticos en Robótica Empleando Modelos de Gran Escala

Building and Exploiting Semantic Maps in Robotics Using Large Models

Realizado por
Jesús Moncada Ramírez

Tutorizado por
Antonio Javier González Jiménez
José Raúl Ruiz Sarmiento

Departamento
Ingeniería de Sistemas y Automática

MÁLAGA, junio de 2024

Abstract

Mobile robots are increasingly being deployed in diverse applications across fields such as home assistance, industry, healthcare, or education. A fundamental requirement for these applications, especially when it comes to Human-Robot Interaction (HRI) scenarios, is the robot's ability to interpret and reason about its environment. This ability is often achieved using semantic maps, which enrich the typical geometric/topological representation of the robot workspace with the semantics of its constituent elements (properties, functionalities, relationships, etc.). Traditional methods for creating these maps rely on object detectors that can only detect those objects previously seen during training, and predefined knowledge bases like ontologies, limiting the robot's adaptability and functionality. This project presents a method for semantic mapping based on large models, including Large Vision-Language Models (LVLMs) and Large Language Models (LLMs), to handle an open set of object categories and dynamic semantic information.

The proposed method, based on ConceptGraphs, utilizes state-of-the-art techniques including Segment Anything (SAM) for object segmentation, CLIP for feature extraction, Gemini 1.5 Pro for generating textual descriptions, and ChatGPT-4o for describing the relationships among objects. The pipeline processes located RGB-D images to create a semantic map represented as a scene graph, where nodes correspond to objects and edges denote their semantic relationships.

The method has been validated using both synthetic (Replica) and real-world (ScanNet) datasets, demonstrating its effectiveness in varied environments. The semantic maps generated are further exploited in an HRI scenario, where an LLM-based chatbot, powered by ChatGPT-4o, assists a mobile robot in helping users perform tasks based on their requests. The use of a self-reflection technique ensures the accuracy and relevance of the robot's actions by refining the LLM's responses. This research improves mobile robot adaptability and intelligence, which is demonstrated by success in controlled and real-world environments.

Keywords: Intelligent Robotics, Semantic maps, Machine learning, Large models.

Resumen

Los robots móviles están siendo utilizados en una variedad cada vez mayor de aplicaciones, como la asistencia en el hogar, la industria, la salud y la educación. Un requisito fundamental para estas aplicaciones, especialmente en escenarios de Interacción Humano-Robot (HRI por sus siglas en inglés, *Human-Robot Interaction*), es la capacidad del robot para interpretar y razonar sobre su entorno. Esta capacidad se logra habitualmente mediante mapas semánticos, que enriquecen la típica representación geométrica/topológica del espacio de trabajo del robot con la información semántica de sus elementos constituyentes (propiedades, funcionalidades, relaciones, etc.). Los métodos tradicionales para crear estos mapas dependen tanto de detectores de objetos que solo pueden detectar aquellos objetos vistos previamente durante el entrenamiento, como de bases de conocimiento predefinidas como las ontologías, limitando la adaptabilidad y funcionalidad del robot. Este proyecto presenta un método para el mapeo semántico basado en modelos de gran escala, incluyendo modelos de visión y lenguaje a gran escala (LVLMs por sus siglas en inglés, *Large Vision-Language Models*) y modelos de lenguaje a gran escala (LLMs por sus siglas en inglés, *Large Language Models*), para manejar un conjunto abierto de categorías de objetos e información semántica dinámica.

El método propuesto, basado en ConceptGraphs, utiliza técnicas del estado del arte que incluyen Segment Anything (SAM) para la segmentación de objetos, CLIP para la extracción de vectores de características, Gemini 1.5 Pro para generar descripciones textuales y ChatGPT-4o para describir las relaciones entre objetos. El flujo de trabajo procesa imágenes RGB-D junto con sus poses para crear un mapa semántico representado como un grafo de escena, donde los nodos corresponden a objetos y las aristas denotan sus relaciones semánticas.

El método ha sido validado utilizando tanto conjuntos de datos sintéticos (Replica) como del mundo real (ScanNet), demostrando su efectividad en entornos variados. Los mapas semánticos generados se explotan además en un escenario de HRI, donde un chatbot basado en un LLM, en concreto ChatGPT-4o, asiste a un robot móvil en ayudar a los usuarios a realizar tareas según sus solicitudes. El uso de

una técnica de autorreflexión asegura la precisión y relevancia de las acciones del robot refinando las respuestas del LLM. Esta investigación mejora la adaptabilidad e inteligencia de los robots móviles, lo cual se demuestra con éxito en entornos controlados y en el mundo real.

Palabras Clave: **Robótica Inteligente, Mapas semánticos, Aprendizaje automático, Modelos a gran escala.**

Contents

1 Introduction

1.1 Motivation

Mobile robots are increasingly being deployed in more diverse applications in fields such as home assistance, industry, healthcare, or education [22]. Common to these applications is the robot's need to possess the cognitive capabilities required to interpret its environment and reason about it, a fundamental requirement to be able to perform and complete its tasks successfully. One commonly employed strategy to achieve these capabilities is the use of **semantic maps** [24], which are models of the working environment that represent both the elements within it (objects, rooms, etc.) and their semantics (properties, functionalities, relationships, etc.). For example, using a semantic map, a robot could manage an inventory by organizing cleaning products, bathroom supplies, office items, etc., considering their properties (e.g., remaining quantity, components, chemical properties, expiration dates, etc.), functionalities, and relationships. These relationships could include their usual locations (e.g., cabinets, shelves, drawers, etc.) and incompatibilities, allowing for the maintenance of necessary safety measures and issuing alerts in situations such as violations of these measures, depleting products, etc.

Traditionally, the workflow of semantic mapping techniques includes the use of an object detection method to identify the elements of the environment and include them in the map [2]. These detectors are usually adjusted using a certain dataset during their training stage [23], so they will only be able to detect a closed set of object categories defined by that repository. For example, the Microsoft COCO dataset [12] is widely used for training object detectors, such as the popular YOLO series since version 3 [28], or Mask R-CNN [8], whose recent implementation in Detectron2 incorporates the popular *transformers*. COCO considers 80 categories divided between outdoor objects (traffic lights, trees, etc.) and indoor objects

(sofa, TV, etc.), as shown in Figure 1. These categories include, for example, three types of fruit: banana, apple, and orange, so the rest of the fruits will not be detectable, and therefore the robot will not be able to operate with them. This limitation arises because a **closed vocabulary** of detectable objects is considered.

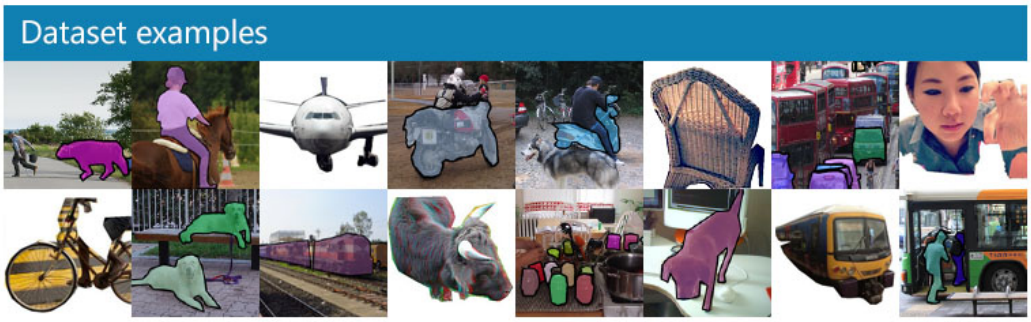

Figure 1: Sample images from the COCO dataset, showing various object categories and instance segmentation annotations.

Once the objects have been identified and introduced into the map, known as factual information, this is accompanied by semantic information encoded in a knowledge base. This knowledge base usually takes the form of an ontology, where a series of concepts (e.g., microwave, refrigerator, table, etc.), their properties (e.g., microwaves are box-shaped, medium-sized, need electricity, etc.), their functionalities (e.g., they are used to heat food, can tell the time, etc.) and their relationships (e.g., they are usually found in kitchens on furniture like tables, etc.) are defined. Thus, the objects present in the map are connected with the semantic information associated with them, which allows the robot to use them when performing its tasks [5, 17]. Typically, the knowledge base is coded from information provided by experts in the domain in question, a tedious process that requires precisely defining concepts and relationships, resulting in a finite and possibly biased representation. This implies a limitation of a **closed knowledge base** that, similarly to the closed vocabulary, restricts the robot's operation to work with the semantic information encoded there.

Very recently, new AI approaches have emerged addressing both limitations using large-scale models. Perhaps the most popular work is presented by [7], introducing the ConceptGraphs representation. A ConceptGraph is a semantic map that captures the working environment through what is called a *scene graph*, in which the nodes model the objects identified in the environment, and the edges define the geometric relationships between them (e.g., on

top of, inside, etc.). The authors consider an open vocabulary of object categories using a Large Vision-Language Model (LVLM) that describes segmented objects in images, and they use a Large Language Model (LLM) to obtain the geometric relationships between objects. Upon a certain user request, the graph is utilized by the LLM as an open knowledge base to generate the task for the robot to execute to solve it. One of the limitations of this approach is the reliance on LLMs and their responses, which are well known to sometimes include incorrect, inaccurate, or invented information [32], resulting in erroneous robot behavior.

In this TFG we propose to address the previous limitations by employing different deep-learning techniques. Specifically, we will study and integrate techniques that consider an open set of categories when detecting objects, such as those used to build ConceptGraphs [7], as well as the integration of geometric data (obtained from conventional 3D mapping systems) with semantic data (obtained from LVLMs). These semantic data are mainly textual tags and captions that describe objects in a semantically rich way. We intend these techniques to be based on state-of-the-art large-scale languages, for example, Gemini 1.5 as LVLM and ChaptGPT-4o as LLM. It is known that the evolution of these models is rapid, so using their latest versions can be crucial in terms of the quality of their responses.

Once the semantic map of the working environment has been generated, we propose to use it to perform tasks by a mobile robot in a Human-Robot Interaction (HRI) scenario. Specifically, given a request made by the user, we intend to use an LLM to, by designing a suitable prompt in which the semantic map is a central element, obtain as a response the tasks to be performed by the robot to complete the request. We will also incorporate a stage of LLM response refinement using the Reflection technique [14]. Specifically, this refinement is based on a self-reflection process where the LLM questions the validity of its initial result, obtaining feedback that it uses to refine its response. Exploitation's success lies in defining appropriate model inputs, commonly called prompts, allowing the minimization of incorrect responses.

This proposal and some of its results have been synthesized in a paper that has been accepted for publication at the "XLV Jornadas de Automática", to be held in Málaga on September 2024 [16]. This project method's implementation code is published in a GitHub repository, available in the Appendix A.

1.2 Objectives

The main objective of this project is the study, development, and implementation of a pipeline based on state-of-the-art techniques and large-scale models (LLMs and LVLMs) for the generation of semantic maps that consider an open set of object categories. In addition, an exploitation of these semantic maps is contemplated so that a mobile robot can execute tasks by interacting with a human.

After a careful study of the state of the art, the objectives have been defined as follows:

1. **Pipeline design and implementation**. A pipeline for the generation of semantic maps considering an open set of categories will be designed and implemented, selecting previously studied deep learning techniques. This pipeline may be based on one of the state-of-the-art methods that have shown the best results, such as ConceptGraphs [7].

2. **Validation**. A validation of the designed pipeline will be performed using different datasets, to ensure the accuracy and consistency of the generated semantic maps.

   (a) First, it will be validated on a sequence of posed RGB-D images from the adaptation proposed by Nice-SALM [34] of the Replica dataset [27]. This dataset contains images obtained in a simulator.

   (b) Secondly, it will be validated on a sequence of posed RGB-D images from the ScanNet dataset [3], which contains real-world images.

3. **Exploitation**. Using the developed pipeline, a semantic map of some image sequences will be created and used to enable a mobile robot to perform tasks that require environmental knowledge, in a human-robot interaction context.

1.3 Structure of the document

This project report is divided into six chapters, organized as follows:

Chapter 1: Introduction. This chapter introduces the main topic of the project, outlining the motivation and the objectives proposed in the preliminary draft. It sets the stage for the subsequent discussions.

Chapter 2: Background. This chapter sets the stage by providing essential context for the project. It includes explanations and definitions interesting in the field of semantic mapping, the primary focus of the proposed method. This background information is vital for understanding what will be the results of the proposed method.

Chapter 3: Used technologies. This chapter discusses the tools employed throughout the project. It covers all stages of the project, from the platforms where we approximated the problem with preliminary experiments, to the deep learning models that are part of the developed method.

Chapter 4: Semantic mapping method. This central chapter presents the proposed semantic mapping method. For clarity and understanding, a description of each stage in the method is included, specifying their inputs and outputs, together with the implementation details. Where possible, explanations are accompanied by images and examples illustrating the results of each stage.

Chapter 5: Semantic mapping validation and testing. In this chapter, first of all, the semantic mapping method is validated using some datasets suitable for this. Secondly, the generated semantic maps are exploited in an HRI scenario, where a user asks a mobile robot to perform some tasks using natural language and referring to objects in the scene. The results exposed in this chapter demonstrate the proper functioning of the developed method.

Chapter 6: Conclusions. This chapter summarizes the work done in the project and discusses some results, as well as their causes and implications. It also suggests directions for future work. This final section encapsulates the entire study, providing a clear and concise conclusion based on the outcomes.

2 Background

2.1 Semantic maps

A semantic map is an advanced form of spatial and knowledge representation used in various fields, particularly robotics and artificial intelligence, to model and understand environments. Unlike traditional maps that primarily focus on geometric and topological information, semantic maps enrich this data with additional layers of meaning, including the functionality and relationships of the elements within the environment. Figure 2 shows an illustrative example of a semantic map. Simply, a semantic map links geometric information about the elements identified in the scene (objects, rooms, etc.) with their semantics (properties, functionalities, relations, etc.).

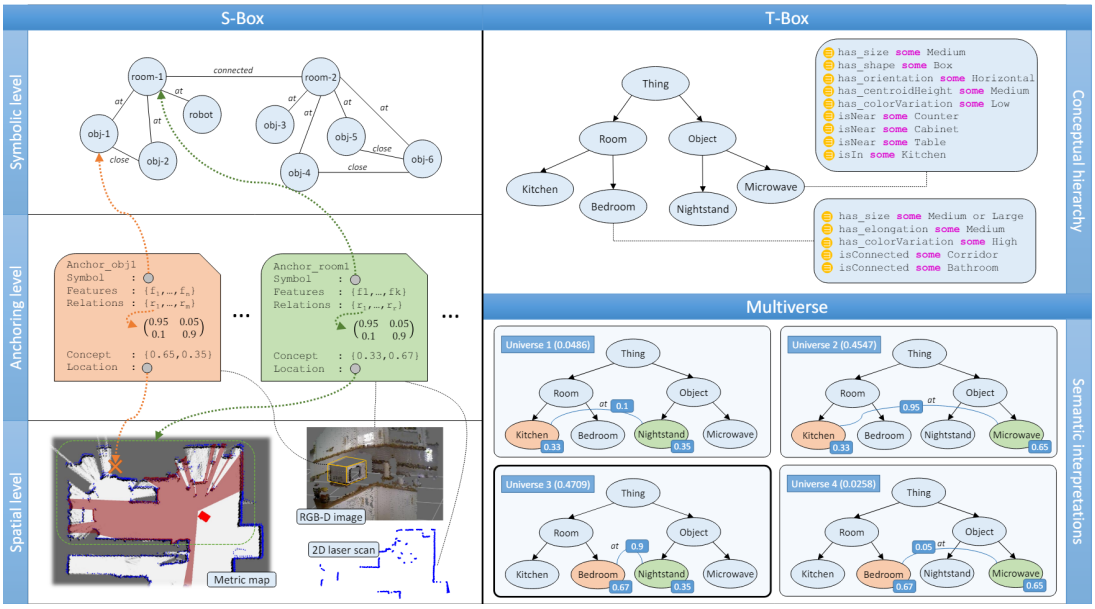

Figure 2: Example of a Multiversal Semantic Map [24] representing a simple domestic environment.

Some authors criticize traditional maps, especially metric maps, for lacking human-centric

understanding and interaction capabilities [10]. While traditional mapping methods like SLAM provide accurate geometric representations and localization, its outputs are limited to metric data, which contrasts with the intuitive, contextual navigation humans perform. Semantic maps, which incorporate human concepts and their spatial arrangements significantly enhance navigation, task planning, and human-robot interaction. By integrating these human concepts, called semasiological attributes, semantic maps bridge the cognitive gap between robots and humans, fostering more effective and natural interactions.

2.1.1 Characteristics

At its core, a semantic map contains geometric details, which describe the spatial occupancy of the environment, and topological information, which details the connectivity between different points. These foundational layers allow for basic navigation and obstacle avoidance. However, the principal feature of semantic maps is their enrichment with semantic information. This includes categorizing objects and understanding their functionalities and relationships. For instance, a microwave in a kitchen is not just recognized as an object but also associated with the function of heating food and its typical location within a household setting. Thus, one of the primary characteristics of semantic maps is the integration of spatial and semantic information.

Semantic information in these maps is often structured using ontologies [29]. Ontologies provide a framework for codifying objects and their properties, facilitating complex reasoning about the environment. This structured approach enables robots to understand and interact with their surroundings in a more human-like manner. For example, an ontology for a household robot might include categories like “furniture”, “appliances”, and “utensils”, with properties such as size, shape, and function.

Semantic maps often have a hierarchical structure, providing multiple levels of detail. They can offer detailed information at different levels, from individual objects to rooms to entire buildings. Additionally, semantic hierarchies establish relationships between different objects and categories. This makes a robot understand, for example, that a “sink” is not an object on the same level as a “kitchen” but is located within it.

Another crucial feature of semantic maps is their dynamic and updatable nature, allowing for the integration of information over time. This temporal aspect ensures that the map re-

mains up-to-date and reflects the current state of the environment, accommodating changes and new observations. As the robot encounters new objects or changes are made in the environment, the maps can be updated in real-time, enabling adaptive learning and continuous improvement. This real-time updating ensures that the robot's understanding of the environment remains accurate and relevant.

Multi-modal data integration is another important characteristic of semantic maps. By combining data from various sensors, such as cameras, LiDAR, and ultrasonic sensors, robots can create a comprehensive map. Cross-modal understanding further enhances semantic understanding by integrating visual, auditory, and other sensory information.

2.1.2 Applications

Semantic maps in robotics are essential for improving the autonomy and intelligence of robotic systems. They have numerous applications in robotics, including:

Figure 3: Tesla's FSD system demonstrates semantic mapping by detecting and classifying pedestrians and vehicles in an urban environment, utilizing camera inputs and LiDAR to navigate complex traffic scenarios.

Autonomous navigation. Autonomous vehicles use semantic maps to navigate urban environments. These maps contain information about road types, traffic signs, pedestrian crossings, and other critical elements. For instance, Google's Waymo and Tesla's Full Self-

Driving (FSD) (see Figure 3) systems rely on detailed semantic maps to understand the driving environment and make real-time decisions.

Figure 4: An elderly man joyfully interacts with a Giraff robot, designed to assist elderly individuals by providing remote communication and support.

Home assistance. Robots designed to assist elderly individuals, such as the Giraff robot (see Figure 4), can utilize semantic maps to provide navigation aids, identify obstacles, and offer contextual information about their environment [13]. For example, these robots can guide users to specific locations within their homes, alert them to potential hazards like uneven floors or stairs, and inform them about nearby amenities such as shops, parks, or healthcare facilities. Some works have even leveraged users' feedback to perform technical improvements on these kinds of robots [6].

Healthcare robotics. In hospitals, robots equipped with semantic maps assist in tasks such as delivering medications, guiding visitors, or transporting medical supplies. A robot like the TUG from Aethon (see Figure 5) uses semantic maps to navigate hospital corridors and deliver supplies to the correct locations.

Industrial automation. In manufacturing, robots equipped with semantic maps can navigate complex factory environments, avoid obstacles, and perform tasks such as assembly,

Figure 5: Autonomous TUG robots navigating a hospital corridor, assisting healthcare staff by transporting supplies and reducing the workload in a clinical environment.

Figure 6: An automated guided vehicle (AGV) transports a stainless steel coil at an Acerinox factory located in Los Barrios, Cádiz.

quality control, and logistics. For example, a robot in an industrial plant might use a semantic map to locate and transport components to the assembly line (see Figure 6).

These are just a few examples, and new applications for semantic maps in robotics are constantly emerging as technology advances. Other applications could include object recognition and manipulation, human-robot interaction, environmental monitoring, search and rescue operations, security and surveillance, and smart home integration.

2.2 Semantic mapping and exploitation

Semantic mapping refers to the creation of semantic maps. The primary goal of semantic mapping is to enhance human-robot interaction. By enabling robots to understand and describe their surroundings in human-centric terms, semantic maps bridge the communication gap between humans and robots. Instead of merely processing coordinates or positions, robots can recognize and interact with spaces like "kitchens" or "living rooms", and locate objects such as "tables" or "chairs" within these spaces [10].

The traditional workflow for semantic mapping includes the following steps:

1. **Data collection**. This initial step involves gathering raw data from the environment using sensors such as cameras, LiDAR, and other perception tools. This data provides the necessary geometric and appearance information.

2. **Object detection and recognition**. By using advanced machine learning techniques, particularly those based on CNNs and *transformers*, we can detect and recognize objects within the collected data. This process is essential for populating the map with identifiable objects, hence the importance of being able to detect an open set of object categories.

3. **Semantic annotation**. Detected objects are annotated with semantic labels, which include not only the identification of the objects but also their attributes and relationships with other objects. The semantic information for this annotation can be sourced from ontologies or from more recent approaches, such as using LVLMs.

4. **Integration and mapping**. The annotated data is integrated into a coherent map structure. This process involves aligning the detected objects with their geometric positions and updating the map continuously as new data is collected.

5. **Updating and maintenance**. Environments are dynamic, so semantic maps must be regularly updated. This step involves incorporating new data to reflect changes in the environment, ensuring the map remains accurate and reliable.

All in all, the semantic mapping process transforms raw sensory data into a rich, interactive model of the environment, enabling robots to understand and operate within human-

centric spaces effectively. This capability is fundamental for advanced robotic applications, from domestic assistance to complex industrial tasks.

Early works focused on semantic mapping built the 3D maps using online algorithms like SLAM [18, 31, 30] or offline methods like structure-from-motion (SfM) [26, 1]. Subsequently, new approaches emerged that focused on using deep-learning-based object detection and segmentation to reconstruct the 3D scene. Instead of reconstructing the 3D geometry directly, these methods employed dense semantic mapping [15, 25] or object-level decomposition [19, 20]. Most of them use a closed vocabulary to identify and represent the elements in the workspace, so they greatly restrict its use to object categories seen during training. This project addresses that limitation by using an object detector considering an open set of categories.

Once the semantic map is built, it can be leveraged by the robot to effectively operate in its workspace. For that, a knowledge base codifying the semantics of the domain at hand is used, which is typically retrieved from experts. Logic reasoners are in charge of, given a request by the user, returning a plan to solve it. This project also employs a large-scale model to carry out said reasoning without explicitly defining a knowledge base and also proposes a mechanism to improve its answers.

3

Used technologies

This section describes the technologies used to build the method outlined in Section 4. These technologies include, among others, the primary programming language used for implementation, the platforms where the experiments were conducted, and various deep learning models incorporated into the pipeline, including some Large Language Models like Gemini and ChatGPT.

3.1 Python

Python[1] is one of the most widely used programming languages in the field of computer vision and artificial intelligence, renowned for its simplicity and versatility. It offers a vast set of open-source libraries and frameworks including implementations for state-of-the-art models and algorithms. Python's extensive ecosystem enables researchers and developers to efficiently implement and experiment with advanced techniques, making it the preferred choice for both academic research and industry applications.

This project uses Python as the main programming language to implement the described semantic mapping method. Given the aforementioned appealing features, and that the authors of ConceptGraphs [7] also used Python for their code, we decided to maintain consistency by using the same language. By leveraging Python's robust libraries, our method integrates various deep-learning models and algorithms to build accurate and efficient semantic maps.

3.1.1 Conda virtual environment

Given the extensive number of required libraries for this project (see Figure 7), it's crucial to have detailed management of the dependencies. To achieve this, we have utilized a Conda[2]

[1] https://www.python.org/

[2] https://conda.io/projects/conda/en/latest/index.html

virtual environment. Conda facilitates the creation of an isolated environment with precise versions of libraries and dependencies, ensuring consistent code execution across various systems. This strategy improves reproducibility and simplifies dependency management, making the software easier to maintain and update.

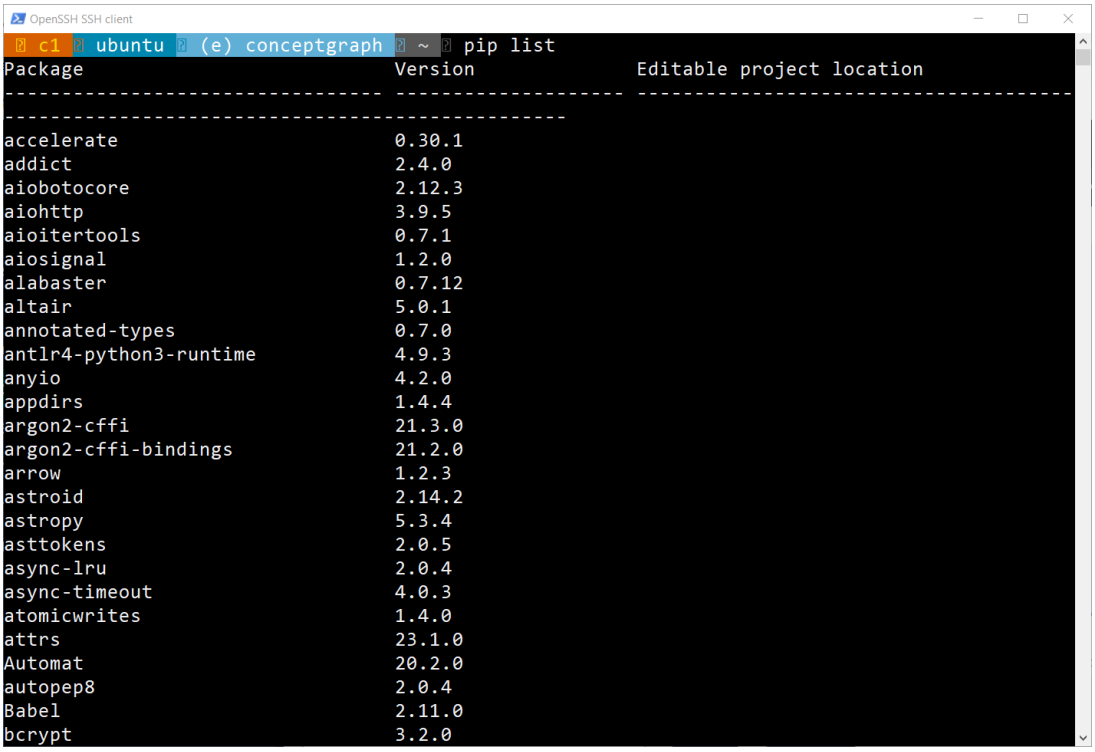

Figure 7: The output of the *pip list* command executed in the used virtual environment contains a large number of libraries, including those directly needed to run the method and their dependencies.

3.2 GitHub

GitHub[3] is a widely used platform built on top of Git, a powerful version control system that offers numerous advantages for code developers. It facilitates collaboration through features like branching, pull requests, and code reviews, ensuring seamless teamwork and high code quality. GitHub also integrates with various development tools and CI/CD pipelines, enhancing workflow efficiency. With robust issue tracking and project management capabilities,

[3] https://github.com/

GitHub, combined with Git's underlying version control, is essential for modern software development, enabling effective teamwork and maintaining high-quality codebases.

In this project, GitHub was used to maintain an online repository for all the developed code (accessible at Appendix A.1), as can be seen in Figure 8. The main feature used was version control, which allowed tracking changes and managing different versions of the code effectively. This ensured that the project maintained a clear history of all modifications, facilitating easier debugging and code review.

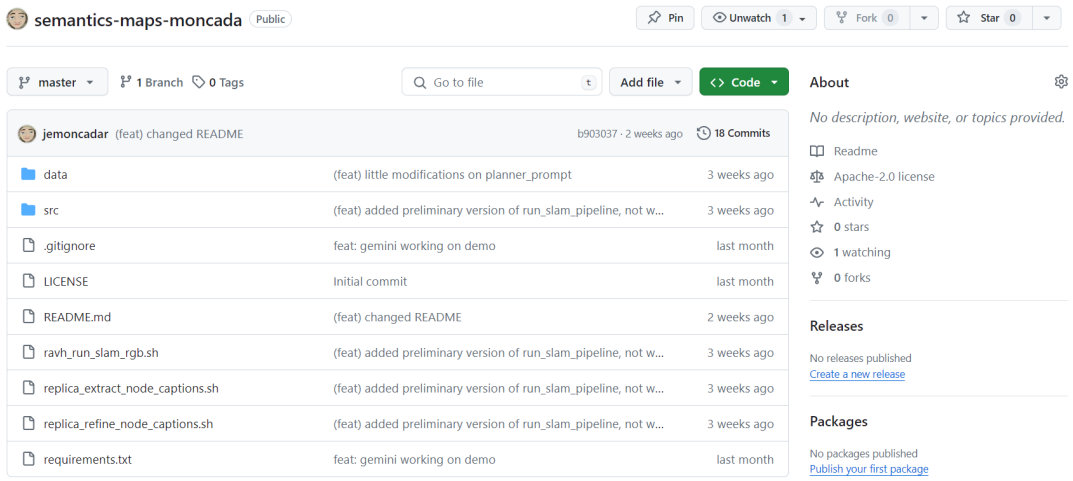

Figure 8: Screenshot of this project's GitHub repository showcasing its directory structure and recent commit history.

3.3 Computing environments

The semantic mapping method requires substantial hardware resources, particularly GPU RAM, making it impractical to run on a typical personal computer or laptop. Therefore, we utilized platforms with greater resource availability for execution, including Google Colab and a server operating with a Linux container-based architecture.

3.3.1 Google Colab

Google Colab[4] is a cloud-based platform designed for interactive coding and data analysis. It provides a Jupyter Notebook environment that allows users to write and execute Python

[4] https://colab.research.google.com/

code directly in their browsers, with no setup required. Colab is particularly intended for machine learning, data science, and artificial intelligence tasks, offering free access to powerful computing resources like GPUs and TPUs. It supports collaborative work, enabling multiple users to share and edit notebooks in real time, making it an ideal tool for both educational purposes and professional research.

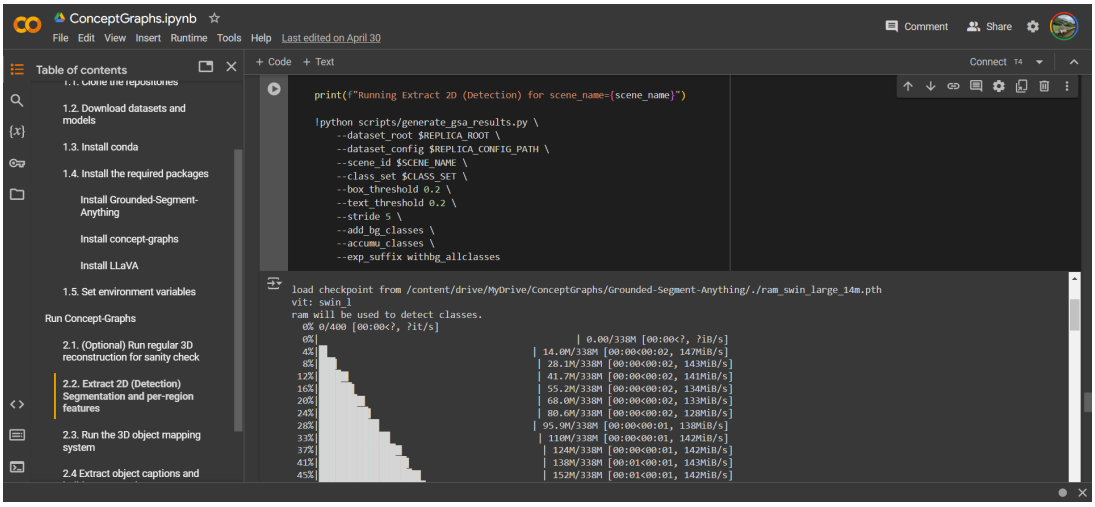

Figure 9: A screenshot of the Google Colab notebook used in the project's initial stages. The screenshot specifically shows a cell running SAM object detection on images from the dataset.

During the initial stage of the project, we used Google Colab for evaluations and tests. In particular, a Jupyter Notebook compressing all the ConceptGraphs method's pipeline was created (accessible at Appendix A.2), as can be seen in Figure 9. This platform provides an accessible and powerful environment for coding and data analysis, even on the free plan. However, the free plan has some drawbacks, such as an unstable connection that can result in session disconnections based on the server's computational load, requiring users to restart their work. Given the extensive number of libraries needed to run the method, this instability was a major setback. Consequently, for the first two months, we subscribed to Google Colab Pro at $11 per month, which provided access to faster, more powerful GPUs and a more stable connection, significantly enhancing our computational performance. Overall, Google Colab was essential for the early development and validation of the project.

3.3.2 Linux container-based architecture

A Linux container-based architecture is a virtualization method that enables multiple isolated user-space instances, known as containers, to run on a single Linux operating system kernel. Each container encapsulates an application and its dependencies, ensuring consistent performance across various computing environments. This architecture provides lightweight, efficient, and scalable deployment solutions by sharing the host system's kernel and resources while maintaining isolation between containers.

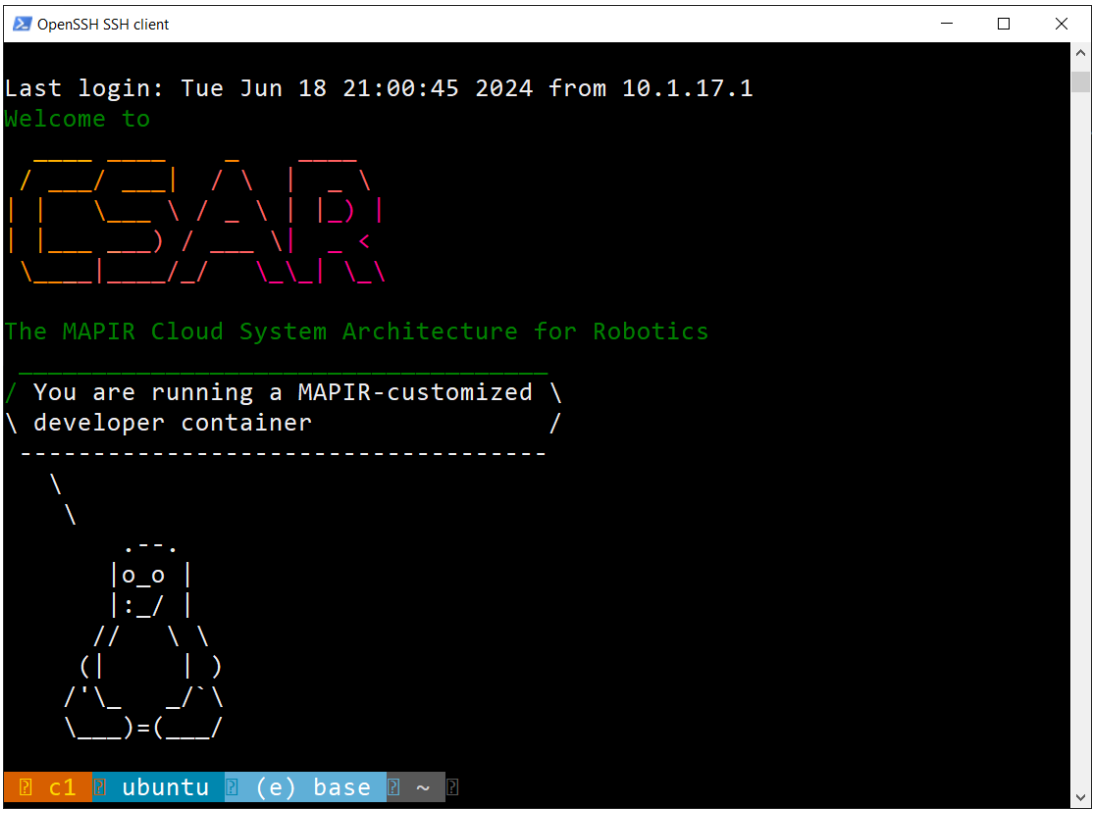

Figure 10: Welcome screen of the MAPIR Cloud System Architecture for Robotics (CSAR), accessed via SSH.

One of the challenges with Google Colab, even with the Pro plan, is that it does not retain the created environments, including installed libraries or cloned repositories. Each time users log in, they encounter a completely clean environment, requiring them to reinstall all necessary requirements before running their code. The installation time for all the libraries and dependencies required by our method was approximately 35 minutes, making it impractical to perform this process with every execution. In this project, to address these issues, we have

implemented our method using a Linux container-based architecture called CSAR, operating in a distributed manner, as maintained by the MAPIR Research Group[5] at the University of Málaga. The server where this architecture currently operates has a new generation i7 processor, 128GB of RAM, 4TB of NVME storage, 8TB of SATA storage for backups, an NVIDIA GeForce RTX 3060 GPU with 8GB of video memory, and an NVIDIA Titan X with 12 GB, as well as a 2.5 Gb ethernet interface that will allow its future integration in a cluster with other servers. This server is currently running the Ubuntu Server LTS 22.04 operating system, in addition to the updated drivers that manage the GPUs. This approach has allowed the execution of different stages on various devices, such as edge devices, without compromising the typically limited resources of a mobile robot.

3.4 Segment Anything (SAM)

The Segment Anything Model (SAM) [9] is an advanced object detection model designed to identify and segment objects within images. Developed by Meta AI, SAM is capable of recognizing and delineating objects with high precision, regardless of the object's category. A key feature of SAM is its open-vocabulary object detection, which allows it to detect and segment objects based on descriptions that may not be part of its initial training set. This capability is critical for applications that require high adaptability and flexibility, such as ours, as SAM can generalize across diverse and unseen objects.

In this project, SAM was utilized to detect and segment objects in the input sequence of RGB images, resulting in a set of detections characterized by masks for each image. Following this initial object detection phase, detections that likely originate from the same object are associated and combined into real objects, which then become the nodes of the semantic map.

3.5 Contrastive Language-Image Pretraining (CLIP)

The CLIP (Contrastive Language-Image Pre-Training) [21] model, developed by OpenAI, is designed to extract features from images by leveraging the relationship between textual descriptions and visual content. By training on a vast dataset of images paired with corresponding text, CLIP learns to generate highly informative and discriminative features that can be

[5] https://mapir.isa.uma.es/mapirwebsite/

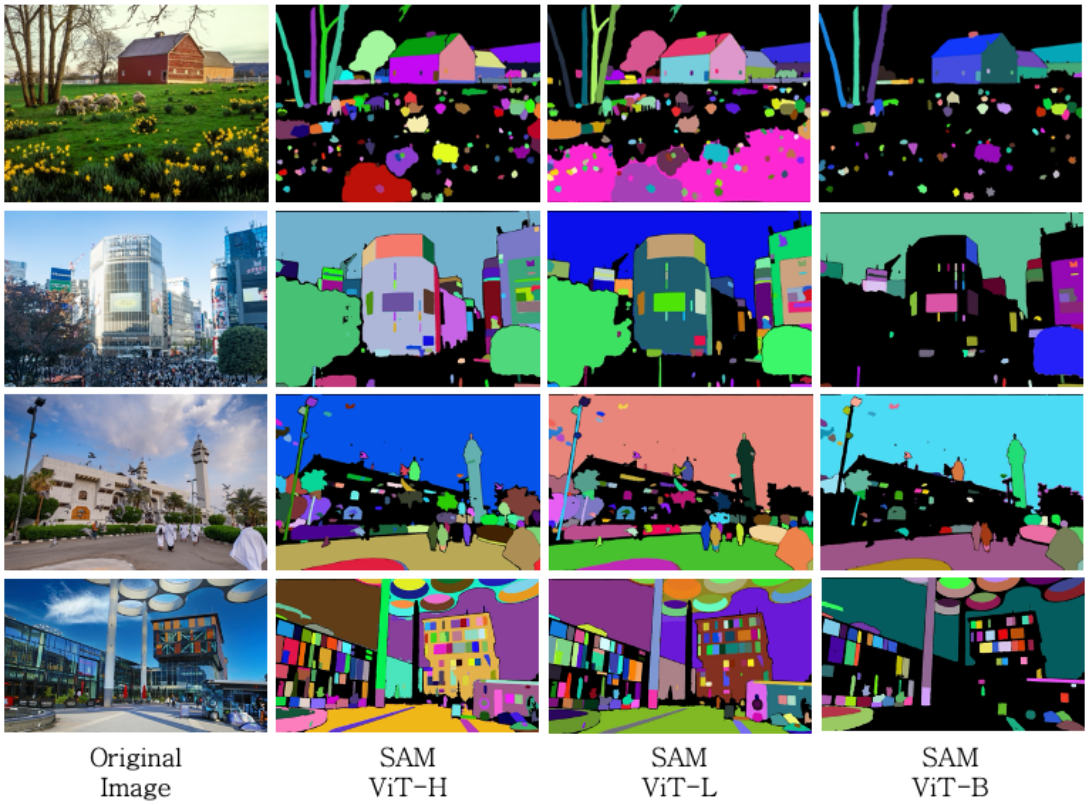

Figure 11: Comparison of segmentation results from the Segment Anything model based on different model sizes (ViT-H, ViT-L, ViT-B) [33]. The first column shows the original images, followed by the segmented outputs from three versions of the model, highlighting variations in segmentation quality and detail depending on the model size.

used for various downstream tasks. CLIP's ability to understand and connect visual and textual information makes it a powerful tool for bridging the gap between computer vision and natural language processing.

This project used CLIP to compute a feature vector for each object detection obtained by SAM. Figure 12 shows a diagram of this model's approach for illustrative purposes. These descriptors are crucial for associating detections of the same object across different images, as this association relies on a similarity measure that considers both geometric and semantic similarities. Geometric similarity ensures that detections close to each other on the map are more likely to be the same object. In contrast, semantic similarity, determined by comparing the CLIP feature vectors of the detections, ensures that they refer to the same object by evaluating the visual and textual characteristics captured in the feature vectors.

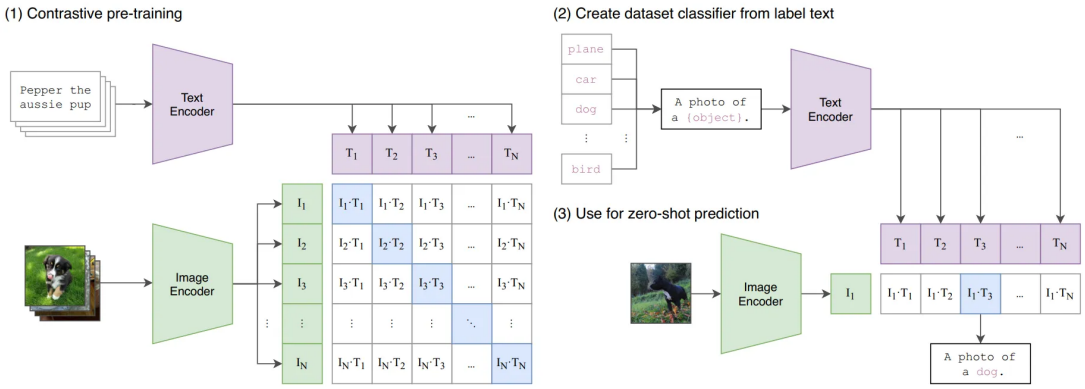

Figure 12: The diagram of the CLIP model's approach, as taken from the original paper [21]. Note that each green box represents an image embedding, i.e., a feature vector derived from an image.

3.6 Large models

Large-scale models or simply large models, encompassing both Large Language Models and Large Vision-Language Models, are advanced AI systems designed to understand and generate human language as well as interpret and analyze images. Trained on vast datasets of text and images, these models can perform tasks like translation, summarization, conversation, and image recognition. Recent rapid advancements in deep learning, computational power, and data availability have significantly enhanced their capabilities, making them increasingly versatile and widely used across various fields.

Due to the significant GPU RAM required to run these models on a local computer, this project opted to use proprietary LLMs and LVLMs accessible via an API, such as ChatGPT and Gemini. Some of the advantages of this approach are the speed of text generation responses, the need to install no software, and the quality of these proprietary models, which is often superior to that of open-source models. However, the main drawback is the cost, as these models require payment based on the number of tokens sent in the request and received in the response.

3.6.1 Gemini

The Gemini[6] language model, launched by Google in December 2023, is an advanced language model designed for natural language understanding and generation. Utilizing state-of-the-art AI technology, Gemini provides powerful and efficient language processing capabilities. Building on this foundation, the Gemini 1.5 Pro version introduces enhanced features and improvements. This upgraded, multi-modal model offers even more sophisticated natural language processing, delivering superior performance and increased efficiency in handling complex language tasks across various modes, including text, images (see Figure 13), and audio.

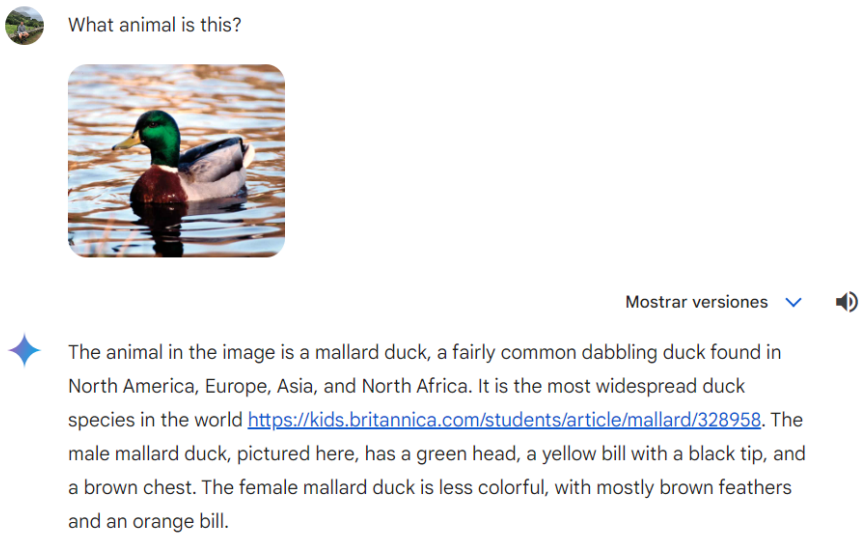

Figure 13: Example of an object detection interaction with Gemini 1.5 Pro: a user asks for the identification of an animal in a picture, and receives a detailed response explaining that the animal is a male mallard duck, including a description of its appearance and a link to additional information.

In this project, we use the multi-modal capabilities of the Gemini Pro 1.5 to convert object images into textual descriptions, including language tags and captions. This transformation enables the nodes of the semantic map, originally made up of object images from the input sequence, to become textual descriptions. These textual descriptions can then be easily processed by another language model to generate the edges of the semantic map. Overall, this approach enables the transition from a visual representation of objects to a textual one.

[6] https://gemini.google.com/app

3.6.2 ChatGPT

The ChatGPT[7] language model by OpenAI is an advanced language model designed for natural language understanding and generation, enabling tasks such as conversation, text completion, and summarization. Utilizing deep learning and extensive training data, ChatGPT excels in creating coherent and contextually appropriate responses (see Figure 14), making it suitable for various applications, from virtual assistants to content creation. The latest version, ChatGPT-4o, launched in May 2024, offers significant improvements over its predecessors. It features enhanced contextual understanding for more coherent and relevant conversations, better handling of nuanced language, and improved integration with various APIs and tools. These advancements make ChatGPT-4o a powerful tool for personal, professional, and academic use, setting a new standard for conversational AI.

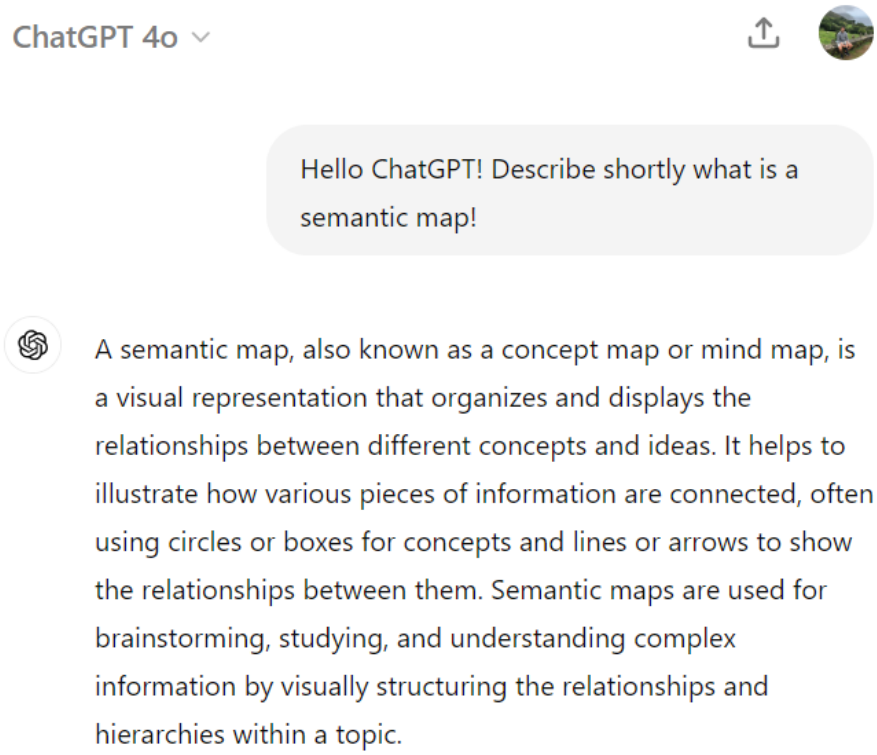

Figure 14: Example of an informative interaction with ChatGPT-4o: a user inquires about the definition of a semantic map and receives a concise explanation highlighting its purpose and structure.

In this project, ChatGPT-4o was used to construct the edges of the semantic map, playing

[7] https://chatgpt.com/

a crucial role in the semantic mapping process. The choice of this model for this task responds to the fact that it is widely recognized as the model that currently delivers the best results across numerous evaluation metrics. This model, excelling at chatting, was also used in the exploitation experiments of the generated semantic maps, making it perfect for HRI scenarios.

4

Semantic mapping method

The developed method, based on ConceptGraphs [7], takes as **input** a sequence of localized RGB-D images captured in an indoor environment. From this input, it produces as **output** a semantic map of the environment built using large-scale models including LLMs and LVLMs. This map is presented as a scene graph, where each node represents an object and each edge reflects the semantic relationship between two objects.

This method's pipeline is divided into a set of stages that must be executed sequentially (see Figure 15). Initially, each input image undergoes an object detection process capable of identifying a broad set of categories (see Section 4.1). This results in multiple views or detections of each object present in the environment. In a subsequent step, a data association stage is performed, where detections belonging to the same physical objects are grouped using an algorithm that considers both geometric and semantic similarity, thus forming the **nodes** of the semantic map (see Section 4.2).

Subsequently, **textual descriptions** are generated by an LVLM for each object (see Section 4.3). Specifically, these descriptions are generated based on the N observations of the object with the highest confidence value in detection provided by the object detector. These individual textual descriptions are then processed by an LLM to generate a single, comprehensive, and coherent textual description of the object.

To complete the construction of the map, the interactions and relationships between adjacent nodes are inferred using another LLM, which, considering the general world knowledge learned during training, establishes the pertinent semantic connections between them (see Section 4.4). These connections are formalized as the **edges** of the semantic map, thus completing the semantic graph resulting from the method.

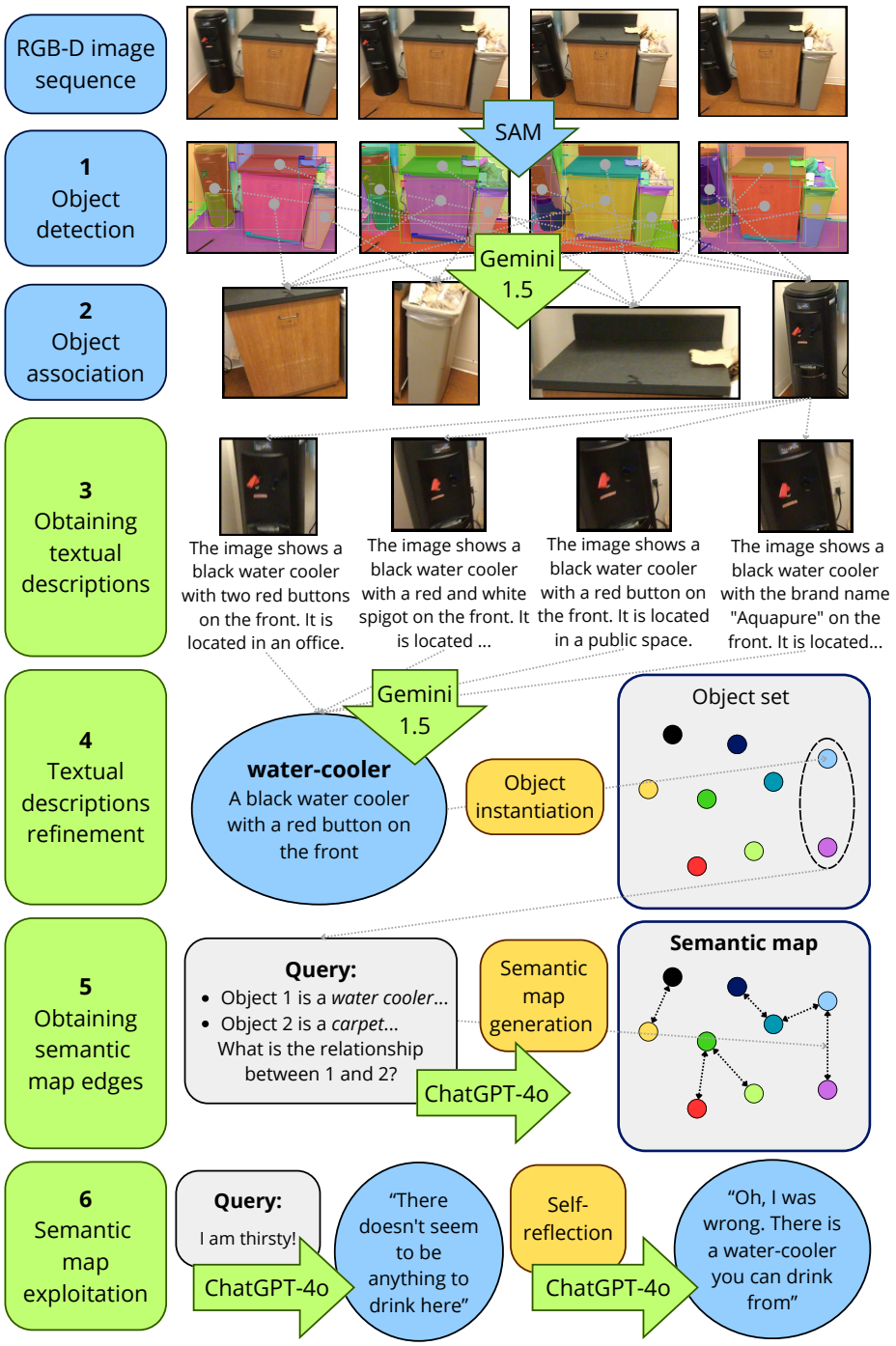

Figure 15: Overview of the proposed method based on ConceptGraphs, illustrated on the sequence *scene0003_02* from the ScanNet dataset. The stages highlighted in green have been modified and/or added to include our proposals related to the original ConceptGraphs work.

Finally, the built map can be utilized by a mobile robot to complete its tasks. To achieve this, the robot is equipped with an LLM-based chatbot that includes a response refinement

stage using the Reflection technique. This enhances the robustness of the responses, thereby improving the robot's overall operation (see Section 4.5).

4.1 Object detection

The object detection process is carried out in two stages: i) object segmentation and ii) object description. First, to segment the objects present in each RGB image $\mathcal{I}_t^{RGB}$, Segment Anything (SAM) [9], a state-of-the-art method that allows category-agnostic object detection, is used. As output, SAM provides a series of **masks** $\mathbf{m}_{t,i}, i = 1...M$, corresponding to each of the M detected objects. Additionally, it provides a set of **confidence scores** $c_{t,i}, i = 1...M$ which indicate the certainty that each detected object occupies its respective region. Next, each mask is applied to the image $\mathcal{I}_t^{RGB}$ to extract the fragment of the image where each object appears, which is then processed with a feature extractor. Specifically, in this case, the CLIP model [21] is used, which returns a vector of semantic features $\mathbf{f}_{t,i}$ that describes the processed image fragment. These feature vectors will be used in a later stage to perform data association between multiple observations of the same physical object.

4.2 Object association

So far, each detection in an image is characterized by a mask $\mathbf{m}_{t,i}$ and a feature vector $\mathbf{f}_{t,i}$. Additionally, each detection is annotated with its point cloud $\mathbf{p}_{t,i}$, which is obtained by projecting the pixels that fall within $\mathbf{m}_{t,i}$ into 3D space, using the camera intrinsic matrix K and the depth image $\mathcal{I}_t^D$. The obtained point cloud is filtered to eliminate spurious points using a density-based clustering method, specifically, DBSCAN [4].

Once all detections obtained in $\mathcal{I}_t^{RGB}$ have been characterized with the mask $\mathbf{m}_{t,i}$, the feature vector $\mathbf{f}_{t,i}$, and the corresponding point cloud $\mathbf{p}_{t,i}$, the data association stage is carried out to group the different observations of the same physical object in different images. To do this, a similarity is calculated between each object detected in the current image $\langle \mathbf{p}_{t,i}, \mathbf{f}_{t,i} \rangle$ and the existing objects in the map $\langle \mathbf{p}_{\mathbf{o}_j}, \mathbf{m}_{\mathbf{o}_j} \rangle$ with which some geometric overlap is shared. The total similarity is calculated based on a dual criterion, taking into account both geometry and semantics:

1. The **geometric similarity** $s_{\text{geo}}(i, j) = \text{nnratio}(\mathbf{p}_{t,i}, \mathbf{p}_{\mathbf{o}_j})$ is calculated as the proportion

of points in the point cloud of the detected object that have nearest neighbors in the point cloud of the object in the map, given a certain distance threshold.

2. The **semantic similarity** $s_{\text{sem}}(i,j) = \mathbf{f}_{t,i}^T \mathbf{f}_{\mathbf{o}_j}/2 + 1/2$ is calculated as the normalized cosine distance between the corresponding visual descriptors (obtained in the previous step).

The **total similarity** $s(i,j) = s_{\text{geo}}(i,j) + s_{\text{sem}}(i,j)$ is calculated as the sum of the geometric and semantic similarity. The association to a particular object is performed using a greedy algorithm, that is, each detection is associated with the existing object with the highest similarity. If a detection does not have a minimum similarity δ_{sim} with any of the present objects, a new object is instantiated.

When associating a detection $\langle \mathbf{f}_{t,i}, \mathbf{p}_{t,i} \rangle$ with an already instantiated object, it is necessary to update both the feature vector and the point cloud of the new object. The feature vector is updated as $\mathbf{f}_{\mathbf{o}_j} = (n_{\mathbf{o}_j} \mathbf{f}_{\mathbf{o}_j} + \mathbf{f}_{t,i})/(n_{\mathbf{o}_j} + 1)$ where $n_{\mathbf{o}_j}$ is the number of detections that have been associated with the object $\mathbf{o}_j$ so far. The point cloud is updated by including all points from the new detection, i.e., $\mathbf{p}_{t,i} \cup \mathbf{p}_{\mathbf{o}_j}$.

4.3 Object textual descriptions

Once the nodes of the semantic map have been obtained (i.e., the objects), the next step is to evolve from a representation based on visual detections in images to a definition of the objects through **textual descriptions**. These textual descriptions (also called object captions) are first generated individually for each object's observation and then combined into a single global textual description of the object, which is more complete and coherent.

4.3.1 Generation of individual textual descriptions

To obtain the first textual descriptions, for each object i, the $N = 10$ image fragments where the object is observed with the highest confidence provided by SAM $c_{t,i}$ are selected. Each one of these image fragments is then combined with the text "*Describe with 2 or 3 sentences maximum the central element of the image*" to form a prompt, which is provided to an LVLM, specifically Gemini 1.5 Pro (*gemini-1.5-pro-preview-0409*). As a result, an individual textual description is obtained for each of the processed fragments.

4.3.2 Extraction of the global textual description

Once the $N = 10$ individual textual descriptions of a given object have been obtained, these descriptions are combined into a single prompt that is passed to an LLM to obtain a single global description of the object (see Appendix B.1). Specifically, the main instruction of this prompt is to *refine* and *summarize* the given descriptions, extracting the common characteristics of the individual descriptions and discarding those that differ radically from the majority. The prompt also requests the LLM to generate a semantically rich **linguistic label** (also called object tag), composed of two or three words, that describes the object based on the textual descriptions. Additionally, the LLM is given the option to *invalidate* the generation of a global textual description for an object, in case the individual descriptions differ radically from each other, so the object will not be taken into account in the resulting semantic map. In this project, the complete prompt is provided to Gemini 1.5 Pro (*gemini-1.5-pro-preview-0409*) to obtain the global textual description of the object.

For a better understanding, examples of the step-by-step process of creating semantic maps and their subsequent exploitation are provided in Section 5. These examples are accompanied by snapshots of the actual execution whenever possible.

4.4 Semantic map generation

Up to this point, the method has identified the nodes of the semantic map, which represent the objects in the environment. Next, it is necessary to establish the **edges**, which represent the spatial relationships between the objects. To do this, the connectivity between all objects is first estimated by considering their spatial overlaps. This estimation involves calculating the Intersection over Union (IoU) between the bounding boxes of each pair of objects, thereby obtaining a similarity matrix. A minimum spanning tree is estimated from this similarity matrix, whose edges will connect the potentially related nodes.

A prompt is created from each pair of potentially connected objects, whose main instruction asks to describe the spatial relationship between the objects in question (see Appendix B.2). The objects are represented in the prompt with their three-dimensional location (coordinates and dimensions of their bounding boxes) and the textual description generated in the previous step, which is essential to provide the language model with context about

their nature. In this project, this prompt is passed to GPT-4o (*gpt-4o*), which responds by not only identifying the relationship between the objects but also providing a detailed explanation of the underlying reasoning. To describe the relationship between two objects, the prompt provides the LLM with several examples, such as one object being inside another (*object1 in object2*) or one object being on top of another (*object1 on object2*). However, the LLM can also express the relationship using alternative or more detailed terms, such as an object that *may be stored inside* another, an object that *may be holding* another, etc. This flexibility is essential to capture the diversity and complexity of spatial interactions in real-world scenarios.

Once the method pipeline is completed, the semantic map includes the objects, their descriptions, and the relationships between them. To facilitate its subsequent use, it is beneficial to express this map in a format that other large language models can easily integrate and understand, such as JSON, as employed here.

4.5 Semantic map exploitation

Once the semantic map is constructed, it can be leveraged by the robot to effectively operate in its workspace. For example, based on a semantic map, the robot could recognize and locate objects, understand their properties and relationships, navigate efficiently, and perform tasks accurately. To illustrate this, in this project, we propose the exploitation of these semantic maps in an HRI scenario. In particular, an LLM-based chatbot is employed, with a system prompt where the semantic map is the central element (see Appendix B.3), to respond to the requests of a user who asks a mobile robot for help to perform a specific action. The instruction of this prompt asks the LLM to calculate which object in the semantic map the mobile robot should navigate to, in order to help the user complete the task. Additionally, the LLM is requested to provide a list of other potentially useful objects ordered by relevance, an explanation of the query made by the user according to its understanding, and a justification for the final decision.

The inclusion of these detailed explanations and justifications serves multiple critical purposes in the context of HRI and the use of semantic maps. By providing an explanation of the user's query and a justification for the final decision, the system enhances transparency, allowing users to understand why certain decisions are made, which builds trust in the robot's capabilities and decision-making processes. In a real robot, these explanations could be emitted

by a sound emission system, allowing users to receive immediate auditory feedback, further enhancing the interaction experience and usability of the robot in various practical scenarios.

4.5.1 Self-reflection

To refine and adjust the responses obtained by the LLM in this process, the **self-reflection** technique [14] is used on the initial response. By reflecting on the initially provided plan, the LLM can offer clearer explanations and even correct errors that could have led to an incorrect execution of the task. Specifically, this self-reflection takes place with a specialized prompt that includes the first response and has as its main instruction to evaluate that response in terms of correctness (the inferred query and relevant objects are correctly identified), relevance (the identified objects are truly ordered by relevance and none are omitted), and clarity (the explanations are straightforward and unambiguous) (see Appendix B.4). Based on the previous evaluation, it is asked to return a list of clear and concise suggestions for pertinent modifications to improve the initial response.

Using the self-reflection technique is crucial for enhancing the accuracy, relevance, and clarity of the LLM's responses. It ensures that the inferred query and identified objects are correctly aligned with the user's intentions, reducing the likelihood of errors in task execution. Assessing relevance prioritizes the most pertinent objects, optimizing the robot's efficiency. Clarity evaluation ensures explanations are understandable, facilitating better user interaction. The iterative self-reflection process helps the LLM continuously improve, maintaining high-performance standards in dynamic environments. The structured suggestions enable quick and efficient implementation of improvements, enhancing the system's autonomy and effectiveness in HRI scenarios.

4.5.2 Response correction

Finally, the reflection generated in the previous step is materialized in a correction of the response. This correction is accomplished using another prompt that incorporates information about all the previous responses. The prompt instructs the LLM to generate a new response that closely mirrors the original one but incorporates the corrections identified during the self-reflection process (see Appendix B.5). This step ensures that the final response is not only aligned with the user's original query but also enhanced for accuracy, relevance, and

clarity. By systematically addressing the noted improvements, the corrected response aims to provide a more precise and understandable solution, thus improving the overall quality and effectiveness of the interaction.

5

Semantic mapping validation and testing

To validate the proposals presented in this work, a series of experiments have been conducted on two different datasets: Replica [27] and ScanNet [3]. The first, Replica, was used to validate the method and compare it with the results obtained by ConceptGraphs, as the authors published their results on it. Once its correct operation was confirmed, additional experiments were carried out with ScanNet, a dataset derived from real-world environments. From this dataset, various sequences have been used, differing in scene size and number of visible objects. This variety has enabled testing the method in both large, complex scenes and small, controlled ones, allowing for precise control over the number and arrangement of objects.

5.1 Validation on Replica

The first validation of the semantic mapping method took place in a set of posed RGB-D image sequences from the adaptation proposed by Nice-SLAM [34] of the Replica dataset [27]. For simplicity, this report will comment on the results in one of the dataset sequences, namely the *room0* sequence.

Before executing the described method, it is advisable to perform a preliminary coherence check on the dataset to be used. This involves applying a Simultaneous Localization and Mapping (SLAM) algorithm to reconstruct the 3D scene. In this project, the GradSLAM method [11] was specifically used, and the obtained results are shown in Figure 16. Since the scene reconstruction is correct, the validity of the data is ensured, allowing the continuation of the subsequent stages of the method.

Figure 16: Result of applying the GradSLAM method to the *room0* sequence from Replica.

In the next step, which involves object detection in the images (see Section 4.1), the images are segmented, and the objects present are identified. An example of this process for one frame of the sequence can be seen in Figure 17.

Figure 17: A frame from the sequence *room0* from Replica (left) and the result after applying the SAM object detector (right).

Following this, a similarity threshold of $\delta_{\text{sim}} = 1.2$ was used for data association on objects (see Section 4.2), resulting in a total of 75 objects for the sequence concerned, *room0*. Once the objects are identified and their observations across different images are associated, the individual textual descriptions are obtained (see Section 4.3.1) and subsequently refined into a pair <global description, linguistic label> (see Section 4.3.2).

Finally, the generation of the semantic map (see Section 4.4) resulted in coherent objects

consistent with expectations. For illustrative purposes, here is a fragment of the objects from the *room0* sequence's semantic map in JSON format:

```
[
  ...
  { "id": 5,
    "bbox_extent": [0.4, 0.3, 0.2],
    "bbox_center": [-0.6, 1.8, -0.7],
    "possible_tags": ["vase", "ceramic vase", "jug", "table",
   "wooden table", "console table", "sideboard"],
    "object_tag": "vase",
    "caption": "A round ceramic vase with a narrow neck sits on a
   wooden table."
  }, ... ,
  { "id": 9,
    "bbox_extent": [1.0, 0.9, 0.0],
    "bbox_center": [-0.8, 1.1, 0.0],
    "possible_tags": ["window", "square panes", "black frame",
   "white wall"],
    "object_tag": "window",
    "caption": "a large window with small square panes set in a
   black frame mounted on a white wall"
  }, ... ,
  { "id": 17,
    "bbox_extent": [0.2, 0.1, 0.0],
    "bbox_center": [1.2, -1.1, -0.3],
    "possible_tags": ["electrical outlet", "outlet", "wall",
   "electrical switch", "switch"],
    "object_tag": "electrical outlet",
    "caption": "a white electrical outlet with three switches on a
   white wall"
  }, ...
]
```

5.2 ScanNet semantic maps

The following execution of the method took place using some sequences from the ScanNet [3] dataset. In this report, two of them have been chosen to show the results: *scene0000_01*, which comes from an apartment with a kitchen and bathroom, thus representing a large and complex scene, and *scene0003_02*, which represents a small kitchen, allowing for thorough control over the number and arrangement of present objects.

The preliminary coherence check resulted in a correct reconstruction, as shown in Figure 18, confirming the proper integration of the dataset into the semantic mapping method. Following this sanity check, the method execution could proceed.

Figure 18: Result of applying the GradSLAM method to parts of the sequences *scene0000_01* (left) and *scene0003_02* (right) from ScanNet.

In the subsequent step, which focuses on object detection within images (see Section 4.1), images are segmented, and the objects within them are identified. An example of this process for one frame of the sequences can be seen in Figure 19.

Afterward, a similarity threshold of $\delta_{\text{sim}} = 1.0$ was applied for data association of the objects (see Section 4.2), resulting in 535 objects for the *scene0000_01* sequence and 43 for *scene0003_02*. Precisely in this difference in the number of detected objects, we can notice the variation in size between these two scenes.

In the next step, the individual textual descriptions are obtained (see Section 4.3.1) and then refined into a pair <global description, linguistic label> (see Section 4.3.2). For illustrative purposes, Figure 20 shows four fragments where the same physical object has been observed, the textual descriptions from top to bottom, from left to right were:

Figure 19: Frames from two ScanNet sequences (left) and the result after applying the SAM object detector (right). Images on top are taken from the *scene0000_01* sequence, and images on bottom are taken from the *scene0003_02* sequence.

```
- The image shows a black water cooler. It has a red button on the
   front and a small cabinet at the bottom.
- The image shows a black water cooler with a red button on the
   front. The water cooler is located in an office or other public
   space.
- The image shows a black water cooler with the brand "Primo" on the
   front. There is a red button on the front to dispense hot water
   and a white button to dispense cold water.
- The image shows a black water cooler with the brand "Oasis" on it.
   It has two spouts, one for hot water and one for cold water.
```

The final refined textual description resulted in "*A black water cooler with a red button on the front*" and the inferred linguistic label in "*Water cooler*", as shown in the corresponding LLM response:

```
{
  "summary": "a black water cooler with a red button on the front",
  "possible_tags": ["water cooler", "spigot", "button", "cabinet",
   "office"],
  "object_tag": "water cooler"
}
```

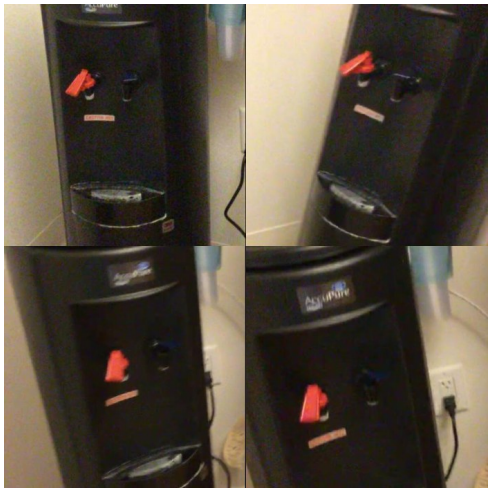

Figure 20: Four image fragments, cropped based on object detections from frames in the *scene0003_02* sequence from ScanNet. In a subsequent step, an LLM will receive each fragment tasked with generating a textual description.

Finally, the generation of the semantic map (see Section 4.4) resulted in coherent objects consistent with expectations. Below is a fragment of the objects from the semantic map in JSON format, showing only some objects from the *scene0003_02* sequence:

```
[
  { "id": 1,
    "bbox_extent": [1.1, 0.6, 0.5],
    "bbox_center": [2.7, 0.9, 0.5],
    "possible_tags": ["water cooler", "spigot", "button", "cabinet",
   "office"],
    "object_tag": "water cooler",
    "caption": "a black water cooler with a red button on the front"
  }, ... ,
```

```
{ "id": 6,
  "bbox_extent": [0.9, 0.5, 0.3],
  "bbox_center": [1.4, 1.1, 0.5],
  "possible_tags": ["trash can", "garbage can", "plastic bag", "cat", "paper", "office"],
  "object_tag": "trash can",
  "caption": "a gray plastic trash can with a black plastic bag inside"
}, ... ,
{ "id": 8,
  "bbox_extent": [0.8, 0.6, 0.2],
  "bbox_center": [1.5, 0.9, 0.9 ],
  "possible_tags": ["coffee cup", "paper cup", "Starbucks cup", "trash can", "table", "paper towel roll"],
  "object_tag": "coffee cup",
  "caption": "a white paper coffee cup with a green Starbucks logo, possibly sitting on a table or a trash can"
}, ... ,
{ "id": 30,
  "bbox_extent": [0.2, 0.1, 0.1],
  "bbox_center": [0.9, 2.5, 1.8],
  "possible_tags": ["house", "model house", "toy house"],
  "object_tag": "toy house",
  "caption": "a small pink house with a red roof"
},
{ "id": 31,
  "bbox_extent": [3.5, 1.8, 1.5],
  "bbox_center": [1.0, 2.3, 1.5],
  "possible_tags": ["door", "fridge", "refrigerator", "wall", "paper", "note", "magnet", "magnets"],
  "object_tag": "refrigerator",
  "caption": "a black fridge with a white paper note and magnets attached to it"
```

```
  }, ...
]
```

Figure 21: A frame from the *scene0003_02* sequence from ScanNet showing the geometrical overlap between object 6 (a trash can, circled in red) and object 8 (a coffee cup, circled in blue).

The semantic map's edges also resulted in coherent relationships between objects. For instance, objects with index 6 and index 8 were marked as "potentially connected" because their bounding boxes geometrically overlapped (as seen in Figure 21). Consequently, they were included in the corresponding prompt (see Appendix B.2); object 6 was referred to as *object a* and object 8 as *object b*. The LLM response indicating their semantic relationship was as follows:

```
{
  "object_relation": "b on a",
  "reason": "A coffee cup is commonly placed on top of a trash can,
   and the description and bounding box centers suggest the coffee
   cup is positioned above the trash can."
```

```
}
```

Figure 22: A frame from the *scene0003_02* sequence from ScanNet showing the geometrical overlap between object 30 (a red house sticker, circled in red) and object 31 (a black refrigerator, circled in blue).

Another example where the semantic relationship inferred is not so good can be seen between objects with index 30 and index 31, a red house sticker, and a refrigerator respectively. It is easy for the human eye to detect that the sticker is attached to the refrigerator (as seen in Figure 22), but this was not so obvious for the LLM tasked with inferring the relationship based on the previously generated textual descriptions. The LLM response containing the semantic relationship was:

```
{
  "object_relation": "none of these",
  "reason": "The objects are a toy house and a refrigerator, which
   do not have a typical relationship where one is commonly placed
   on or inside the other."
```

```
}
```

Note that the relationship is marked as "*none of these*". In this case, the error in the semantic relationship identification was due to a mistake in the generation of the text description (see Section 4.3.1) of object 30. The LVLM classified it as a *toy house* instead of classifying it as a *sticker of a red house*.

5.3 ScanNet semantic map exploitation

Finally, using the generated semantic maps, a series of experiments were conducted where certain queries of interest about the information included in the map were provided to an LLM (specifically, ChatGPT-4o), simulating a user wishing to perform different specific tasks in the given scene (see Section 4.5). Among the various experiments, one is particularly noteworthy, where the query was: "*I need to heat my food!*" on the semantic map of the *scene0003_02* ScanNet sequence. The response provided by the LLM was as follows:

```
{
  "inferred_query": "Find objects related to heating food",
  "relevant_objects": [],
  "query_achievable": false,
  "final_relevant_objects": [],
  "explanation": "There are no objects in the scene explicitly
   related to heating food, such as a microwave or an oven.
   Therefore, the query cannot be achieved with the objects in the
   scene."
}
```

This response is incorrect because the scene includes a microwave (as shown in Figure 23), an object that could be used to complete the task required by the user. In these circumstances, the self-reflection technique (see Section 4.5.1) was applied to the LLM, including a stage where it must be self-critical of its response and check if it was accurate. The result of this stage was as follows:

```
Correctness:
- The query "Find objects related to heating food" is accurate.
```

Figure 23: A frame from the *scene0003_02* sequence from ScanNet showing a microwave oven placed on a wooden shelf.

```
- The response incorrectly states that there are no objects related
   to heating food, while a "microwave oven" and a "refrigerator"
   are present.
- The query's achievable status is incorrectly marked as false due
   to this error.
- The explanation is not clear as it incorrectly claims no relevant
   objects are present.
Relevance:
- The response fails to identify all relevant objects related to
   heating food, such as the "microwave oven" and "refrigerator."
- Including and sorting these objects by relevance would improve
   accuracy.
Clarity:
- The response is clear in language and structure but may cause
   confusion by incorrectly asserting no relevant objects are
   present.
Suggestions for Improvement:
```

```
- Include "microwave oven" and "refrigerator" in the
   relevant_objects field.
- Revise the explanation to acknowledge the presence of relevant
   objects.
- Prioritize the most relevant objects in the final list.
- Provide more context on the relevance of these objects to heating
   food.
```

Among other corrections, it is noted that the original response "incorrectly claims that there are no objects related to heating food, although there is a microwave and a refrigerator in the semantic map." Finally, in the last stage of correcting the original response based on the self-reflection feedback (see Section 4.5.2), the following corrected and more accurate response was obtained:

```
{
  "inferred_query": "Find objects related to heating food",
  "relevant_objects": ["11", "26", "31"],
  "query_achievable": true,
  "final_relevant_objects": ["11", "26", "31"],
  "explanation": "While not all objects in the scene are explicitly
   related to heating food, there are relevant objects like the
   'microwave oven,' 'refrigerator,' and 'paper note with magnets'
   that can be associated with food heating processes. The
   'microwave oven' is used for heating food directly, the
   'refrigerator' aids in preserving food that may be later heated,
   and the 'paper note with magnets' possibly contains information
   related to food heating instructions or recipes."
}
```

As demonstrated in previous experiments, incorporating the self-reflection technique in the use of semantic maps (see Section 4.5) is crucial for ensuring accurate and reliable responses from language models. This enhancement allows the robot to perform high-level tasks more effectively. In this example, the initial response from the LLM incorrectly stated that there were no objects related to heating food in the scene, despite the presence of a mi-

crowave. By applying self-reflection, the model identified its mistake, recognized the microwave and refrigerator as relevant objects, and corrected its response accordingly. On the other hand, the self-reflection technique not only corrects incorrect answers but also improves the quality of correct answers. This ability to critically evaluate and refine outputs enables the LLM to provide more precise and useful information, thereby enhancing the effectiveness of semantic map utilization for practical tasks, as the one proposed in this project.

6
Conclusions

6.1 Conclusions

This project focused on the development of a method for semantic mapping in mobile robotics, leveraging state-of-the-art large-scale models, specifically Large Vision-Language Models (LVLMs) and Large Language Models (LLMs). Traditional semantic mapping techniques rely on object detectors with closed vocabularies and knowledge bases, which hampers their applicability. The proposed method, which is based on ConceptGraphs, addresses these limitations by detecting an open set of objects and incorporating dynamic semantic information. Concretely, in addition to integrate the techniques used to build ConceptGraphs, we have adapted them to work with recent LVLMs and LLMs, and have incorporated a self-reflection stage to improve the quality of the responses of large models when exploiting the map.

The validation of the proposed semantic mapping method was conducted using two distinct datasets: Replica, which images obtained in a simulator, and ScanNet, containing real-world scenes. The first validation involved the Replica dataset, where the method was compared against the results offered by the original ConceptGraphs work. Our method demonstrated comparable results, as the resulting semantic maps were coherent and consistent with the expected objects and their relationships. Further validation was performed on both small and large scenes taken from the ScanNet dataset. This is another contribution of this work since the original ConcepGraphs pipeline was only tested with simulated data. The results were acceptable, but slightly worse than those obtained in the first validation.

The exploitation of the generated semantic maps was demonstrated in a Human-Robot Interaction (HRI) scenario, where a mobile robot equipped with an LLM-based chatbot assists users who make requests in natural language. This LLM-based chatbot, powered by ChatGPT-4o, interprets user requests and generate corresponding robot actions. The inclusion

of a self-reflection technique further enhanced the reliability and accuracy of the robot's responses. This iterative refinement process has demonstrated an improvement of the responses produced by the LLM by critically evaluating and improving the initial responses, leading to more accurate and relevant actions by the robot.

The experiments carried out have corroborated the good performance of the semantic mapping method, but have also revealed some limitations. One of them was the economic cost derived from the use of proprietary LLMs and LVLMs, which can be prohibitively expensive for extensive or long-term projects, limiting accessibility for smaller research groups or applications with budget constraints. The computational cost of running all phases of the model is also high due to the use of models such as SAM and CLIP, which require substantial amounts of GPU RAM in the different stages of the pipeline. Additionally, errors have been observed in the generated semantic maps and their relationships, which may be attributed to two factors. First, the object detector used, SAM, often detects numerous objects in real-world scenes, including parts of physical objects as separate detections. Second, small objects in the scenes are frequently not correctly identified, likely due to the difficulty of the LVLM in describing small-sized objects at low resolution.

6.2 Future lines of research

The implemented semantic mapping method, while effective, has several potential avenues for improvement and further testing:

1. Conducting a detailed quantitative analysis of the obtained semantic maps is crucial. This analysis will precisely evaluate the areas for potential improvement in our proposed semantic maps and determine how they compare to those in the original ConceptGraphs paper. Such an evaluation will highlight specific strengths and weaknesses, guiding targeted enhancements.

2. Exploring alternative object detectors beyond SAM, provided they support an open set of object categories, could significantly enhance the quality of the generated semantic maps. Different detectors might offer better accuracy, fewer false positives, or improved handling of complex scenes, thereby improving overall performance.

3. Utilizing large-scale open-source models that can run on local computing resources would help reduce the high costs associated with proprietary models. This approach would make the method more accessible and feasible for broader applications, including those with limited budgets.

4. Implementing the human-robot interaction scenario on a real mobile robot would allow for a practical evaluation of the system's performance. This real-world testing would provide valuable insights into the system's applicability, responsiveness, and the effectiveness of its semantic understanding and task execution capabilities. Such implementation would reveal any practical challenges and help refine the system for robust real-world operations.

References

[1] Johannes L. ”Schönberger et al. ““Pixelwise View Selection for Unstructured Multi-View Stereo””. In: *”Computer Vision – ECCV 2016”*. Ed. by Bastian ”Leibe et al. ”Cham”: ”Springer International Publishing”, 2016, ”501–518”. ISBN: ”978-3-319-46487-9”. DOI: https://doi.org/10.1007/978-3-319-46487-9_31.

[2] David Chaves et al. “Integration of CNN into a robotic architecture to build semantic maps of indoor environments”. In: *IWANN 2019*. Springer. 2019, pp. 313–324. DOI: https://doi.org/10.1007/978-3-030-20518-8_27.

[3] Angela Dai et al. “ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes”. In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*. 2017.

[4] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[5] D. Fernandez-Chaves et al. “ViMantic, a distributed robotic architecture for semantic mapping in indoor environments”. In: *Knowledge-Based Systems* 232 (2021), p. 107440. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.107440.

[6] J. González-Jiménez, C. Galindo, and J.R. Ruiz-Sarmiento. “Technical improvements of the Giraff telepresence robot based on users’ evaluation”. In: *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*. 2012, pp. 827–832. DOI: https://doi.org/10.1109/ROMAN.2012.6343854.

[7] Qiao Gu et al. “ConceptGraphs: Open-Vocabulary 3D Scene Graphs for Perception and Planning”. In: *arXiv* (2023). DOI: https://doi.org/10.48550/arXiv.2309.16650.

[8] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969. DOI: https://doi.org/10.48550/arXiv.1703.06870.

[9] Alexander Kirillov et al. *Segment Anything*. 2023. DOI: https://doi.org/10.48550/arXiv.2304.02643. arXiv: 2304.02643 [cs.CV].

[10] Ioannis Kostavelis and Antonios Gasteratos. "Semantic mapping for mobile robotics tasks: A survey". In: *Robotics and Autonomous Systems* 66 (2015), pp. 86–103. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2014.12.006. URL: https://www.sciencedirect.com/science/article/pii/S0921889014003030.

[11] Jatavallabhula Krishna Murthy et al. "gradSLAM: Dense SLAM meets automatic differentiation". In: *arXiv*. 2020. DOI: https://doi.org/10.48550/arXiv.1910.10672.

[12] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *Computer Vision–ECCV 2014: 13th European Conference*. 2014, pp. 740–755. DOI: https://doi.org/10.48550/arXiv.1405.0312.

[13] Matteo Luperto et al. "Towards Long-Term Deployment of a Mobile Robot for at-Home Ambient Assisted Living of the Elderly". In: *2019 European Conference on Mobile Robots (ECMR)*. 2019, pp. 1–6. DOI: 10.1109/ECMR.2019.8870924.

[14] Aman Madaan et al. "Self-refine: Iterative refinement with self-feedback". In: *Advances in Neural Information Processing Systems* 36 (2024). DOI: https://doi.org/10.48550/arXiv.2303.17651.

[15] John McCormac et al. *SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks*. 2016. DOI: https://doi.org/10.48550/arXiv.1609.05130. arXiv: 1609.05130.

[16] Jesús Moncada-Ramírez et al. "Large models for semantic mapping in mobile robotics". In: *XLV Jornadas de Automática* (2024).

[17] Javier Monroy et al. "A semantic-based gas source localization with a mobile robot combining vision and chemical sensing". In: *Sensors* 18.12 (2018), p. 4174. DOI: https://doi.org/10.3390/s18124174.

[18] Richard A. Newcombe et al. "KinectFusion: Real-time dense surface mapping and tracking". In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. 2011, pp. 127–136. DOI: https://doi.org/10.1109/ISMAR.2011.6092378.

[19] Jingxing Qian et al. *POCD: Probabilistic Object-Level Change Detection and Volumetric Mapping in Semi-Static Scenes*. 2022. DOI: https://doi.org/10.48550/arXiv.2205.01202. arXiv: 2205.01202.

[20] Jingxing Qian et al. *POV-SLAM: Probabilistic Object-Aware Variational SLAM in Semi-Static Environments*. 2023. arXiv: 2307.00488.

[21] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. DOI: https://doi.org/10.48550/arXiv.2103.00020. arXiv: 2103.00020 [cs.CV].

[22] Francisco Rubio, Francisco Valero, and Carlos Llopis-Albert. "A review of mobile robots: Concepts, methods, theoretical framework, and applications". In: *International Journal of Advanced Robotic Systems* 16.2 (2019).

[23] J. R. Ruiz-Sarmiento, Cipriano Galindo, and Javier González-Jiménez. "Robot@Home, a Robotic Dataset for Semantic Mapping of Home Environments". In: *International Journal of Robotics Research* (2017). DOI: https://doi.org/10.1177/0278364917695640.

[24] Jose-Raul Ruiz-Sarmiento, Cipriano Galindo, and Javier Gonzalez-Jimenez. "Building multiversal semantic maps for mobile robot operation". In: *Knowledge-Based Systems* 119 (2017), pp. 257–272. DOI: https://doi.org/10.1016/j.knosys.2016.12.016.

[25] Martin Rünz, Maud Buffier, and Lourdes Agapito. *MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects*. 2018. DOI: https://doi.org/10.48550/arXiv.1804.09194. arXiv: 1804.09194.

[26] Johannes L. Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4104–4113. DOI: https://doi.org/10.1109/CVPR.2016.445.

[27] Julian Straub et al. "The Replica dataset: A digital replica of indoor spaces". In: *arXiv preprint arXiv:1906.05797* (2019). DOI: https://doi.org/10.48550/arXiv.1906.05797.

[28] Juan Terven and Diana Cordova-Esparza. "A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond". In: *arXiv preprint arXiv:2304.00501* (2023). DOI: https://doi.org/10.48550/arXiv.2304.00501.

[29] Mike Uschold and Michael Gruninger. "Ontologies: principles, methods and applications". In: *The Knowledge Engineering Review* 11.2 (1996), pp. 93–136. DOI: https://doi.org/10.1017/S0269888900007797.

[30] Bowen Wen et al. *BundleSDF: Neural 6-DoF Tracking and 3D Reconstruction of Unknown Objects*. 2023. DOI: https://doi.org/10.48550/arXiv.2303.14158. arXiv: 2303.14158.

[31] Thomas Whelan et al. "ElasticFusion: Dense SLAM Without A Pose Graph". In: July 2015. DOI: https://doi.org/10.15607/RSS.2015.XI.001.

[32] Jia-Yu Yao et al. "Llm lies: Hallucinations are not bugs, but features as adversarial examples". In: *arXiv preprint arXiv:2310.01469* (2023). DOI: https://doi.org/10.48550/arXiv.2310.01469.

[33] Chaoning Zhang et al. *A Survey on Segment Anything Model (SAM): Vision Foundation Model Meets Prompt Engineering*. May 2023. DOI: https://doi.org/10.13140/RG.2.2.14928.17923.

[34] Zihan Zhu et al. "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022. DOI: https://doi.org/10.48550/arXiv.2112.12130.

Appendix A
Project code

A.1 Semantic mapping method code

All the code developed for the semantic mapping pipeline is published in a GitHub repository, accessible at https://github.com/jemoncadar/semantics-maps-moncada.

A.2 ConceptGraphs Google Colab notebook

The Jupyter Notebook developed in the early stages of the project to experiment with the ConceptGraphs method is publicly accessible at https://colab.research.google.com/drive/1Ydb4moJfYwB8Twm5OJ1R39EI0-4P41tn?usp=sharing.

Appendix B
Prompts

This section outlines the various prompts used in the semantic mapping method and in the subsequent use of the generated semantic maps (see Section 4). These prompts are essential for guiding interactions with language models; in fact, some techniques of prompt engineering have been used to ensure outputs are accurate, relevant, and coherent.

B.1 Textual descriptions refinement prompt

The following prompt is responsible for refining and summarizing multiple individual textual descriptions obtained by an LVLM (see Section 4.3.1) of objects into both a single comprehensive and coherent description and a linguistic label. The prompt is used as described in Section 4.3.2. Note that this prompt contains examples that are not included in this appendix.

```
Identify and describe objects in scenes.
Input and output must be in JSON format.
The input field 'captions' contains a list of image captions aiming
   to identify objects.
Output 'summary' as a concise description of the identified
   object(s).
An object mentioned multiple times is likely accurate. If various
   objects are repeated and a container/surface is noted such as a
   shelf or table, assume the (repeated) objects are on that
   container/surface.
For unrelated, non-repeating (or empty) captions, summarize as
   'conflicting (or empty) captions about [objects]' and set
   'object_tag' to 'invalid'.
Output 'possible_tags' listing potential object categories.
Set 'object_tag' as the conclusive identification.
Focus on indoor object types, as the input captions are from indoor
   scans.
```

B.2 Edges generation prompt

The following prompt generates the semantic relationships between objects in the semantic map by describing their spatial relationships. The prompt is used as described in Section 4.4.

```
The input is a list of JSONs describing two objects "object1" and
   "object2". You need to produce a JSON string (and nothing else),
   with two keys: "object_relation", and "reason".

Each of the JSON fields "object1" and "object2" will have the
   following fields:
1. bbox_extent: the 3D bounding box extents of the object
2. bbox_center: the 3D bounding box center of the object
3. object_tag: an extremely brief description of the object

Produce an "object_relation" field that best describes the
   relationship between the two objects. The "object_relation" field
   must be one of the following (verbatim):
1. "a on b": if object a is an object commonly placed on top of
   object b
2. "b on a": if object b is an object commonly placed on top of
   object a
3. "a in b": if object a is an object commonly placed inside object b
4. "b in a": if object b is an object commonly placed inside object a
5. "none of these": if none of the above best describes the
   relationship between the two objects

Before producing the "object_relation" field, produce a "reason"
   field that explains why
the chosen "object_relation" field is the best.
```

B.3 Semantic map exploitation prompt

The following prompt is used to exploit the semantic maps generated by the method. It includes the semantic map in JSON format and responds to a user's queries by indicating the object to which a mobile robot should navigate to complete the user's proposed task. The prompt is used as described in Section 4.5.

```
### INSTRUCTION ###
The input to the model is a 3D scene described in a JSON format (###
   3D SCENE ###). Each entry in the JSON describes one object in the
   scene, with the following five fields:
1. "id": a unique object id
2. "bbox_extent": extents of the 3D bounding box for the object
3. "bbox_center": centroid of the 3D bounding box for the object
4. "object_tag": a brief (but sometimes inaccurate) tag categorizing
   the object
5. "caption": a brief caption for the object
Once you have parsed the JSON and are ready to answer questions
   about the scene, say "I'm ready".
The user will then begin to ask questions, and the task is to answer
   various user queries about the 3D scene. For each user question,
   respond with a JSON dictionary with the following fields:
1. "inferred_query": (String) Your interpretation of the user query
   in a succinct form
2. "relevant_objects": (List) List of relevant object ids for the
   user query (if applicable)
3. "query_achievable": (Boolean) Whether or not the user-specified
   query is achievable using the objects and descriptions provided
   in the 3D scene.
4. "final_relevant_objects": (List of String) A final list of
   objects relevant to the user-specified task. Sort all objects in
   this list such that the most relevant object is listed first,
   followed by the second most relevant, and so on.
```

```
5. "explanation": (String) A brief explanation of what the most
   relevant object(s) is(are), and how they achieve the
   user-specified task.

### 3D SCENE ###
{{object_list_str}}
```

Note that *object_list_str* should be replaced with the semantic map in JSON format.

B.4 Self-reflection prompt

The following prompt is responsible for the self-reflection process, where an LLM evaluates and critiques its own responses to improve accuracy, relevance, and clarity. The prompt is used as described in Section 4.5.1.

```
### CONTEXT ###
We have received a response from another LLM tasked with
   interpreting and answering questions about a 3D scene described
   in a JSON format. Your task is to reflect on this response,
   providing constructive feedback to help refine and improve it.

The INSTRUCTION was:
| The input to the model is a 3D scene described in a JSON format
   (### 3D SCENE ###). Each entry in the JSON describes one object
   in the scene, with the following five fields:
| 1. "id": a unique object id
| 2. "bbox_extent": extents of the 3D bounding box for the object
| 3. "bbox_center": centroid of the 3D bounding box for the object
| 4. "object_tag": a brief (but sometimes inaccurate) tag
   categorizing the object
| 5. "caption": a brief caption for the object
| Once you have parsed the JSON and are ready to answer questions
   about the scene, say "I'm ready".
| The user will then begin to ask questions, and the task is to
   answer various user queries about the 3D scene. For each user
```

```
   question, respond with a JSON dictionary with the following
   fields:
| 1. "inferred_query": your interpretation of the user query in a
   succinct form
| 2. "relevant_objects": list of relevant object ids for the user
   query (if applicable)
| 3. "query_achievable": whether or not the user specified query is
   achievable using the objects and descriptions provided in the 3D
   scene.
| 4. "final_relevant_objects": A final list of objects relevant to
   the user-specified task. Sort all objects in this list such that
   the most relevant object is listed first, followed by the second
   most relevant, and so on.
| 5. "explanation": A brief explanation of what the most relevant
   object(s) is(are), and how they achieve the user-specified task.

The 3D SCENE was:
{{object_list_str}}

The ORIGINAL RESPONSE was:
{{planner_response}}

### INSTRUCTION ###
Your task is to:
- Carefully review the original response for correctness, relevance,
   and clarity.
- Provide constructive criticism, focusing on areas where the
   response could be improved.
- Offer specific suggestions for how to refine the response,
   ensuring it better meets the requirements and addresses any
   potential issues.

REFLECTION TASK
```

```
Correctness:
- Are the inferred query and relevant objects accurately identified?
- Is the query achievable status appropriately determined?
- Is the explanation logically sound and coherent?
Relevance:
- Are all relevant objects identified, and are they sorted by
   relevance accurately?
- Are any crucial objects or details omitted?
Clarity:
- Is the response clear and easy to understand?
- Are there any ambiguities or vague descriptions?

CRITIQUE TEMPLATE
Please review the response carefully for correctness, relevance, and
   clarity, and provide constructive criticism for how to improve
   it. Follow the structure provided below:

Correctness:
[Your comments on correctness]
Relevance:
[Your comments on relevance]
Clarity:
[Your comments on clarity]

SUGGESTIONS FOR IMPROVEMENT
Based on your critique, please suggest specific improvements. These
   can include corrections to factual errors, inclusion of missing
   relevant objects, reordering of objects by relevance, or
   clarifications to enhance understanding.
```

Note that *object_list_str* should be replaced with the semantic map in JSON format, and *planner_response* with the first semantic map exploitation response obtained in the previous step.

B.5 Reflection correction prompt

The following prompt applies corrections to the original planner response based on self-reflection feedback, ensuring the final response is accurate, relevant, and clear. The prompt is used in Section 4.5.2.

```
### CONTEXT ###
We have received two responses from other LLMs. The first one was
   tasked with interpreting and answering questions about a 3D scene
   described in a JSON format. The second one was tasked with
   reflecting on the first response, providing constructive feedback
   to help refine and improve it.

### FIRST LLM RESPONSE ###

The INSTRUCTION was:
| The input to the model is a 3D scene described in a JSON format
   (### 3D SCENE ###). Each entry in the JSON describes one object
   in the scene, with the following five fields:
| 1. "id": a unique object id
| 2. "bbox_extent": extents of the 3D bounding box for the object
| 3. "bbox_center": centroid of the 3D bounding box for the object
| 4. "object_tag": a brief (but sometimes inaccurate) tag
   categorizing the object
| 5. "caption": a brief caption for the object
| Once you have parsed the JSON and are ready to answer questions
   about the scene, say "I'm ready".
| The user will then begin to ask questions, and the task is to
   answer various user queries about the 3D scene. For each user
   question, respond with a JSON dictionary with the following
   fields:
| 1. "inferred_query": your interpretation of the user query in a
   succinct form
```

```
| 2. "relevant_objects": list of relevant object ids for the user
   query (if applicable)
| 3. "query_achievable": whether or not the user specified query is
   achievable using the objects and descriptions provided in the 3D
   scene.
| 4. "final_relevant_objects": A final list of objects relevant to
   the user-specified task. Sort all objects in this list such that
   the most relevant object is listed first, followed by the second
   most relevant, and so on.
| 5. "explanation": A brief explanation of what the most relevant
   object(s) is(are), and how they achieve the user-specified task.

The 3D SCENE was:
{{object_list_str}}

The RESPONSE was:
{{planner_response}}

### SECOND LLM RESPONSE ###

The INSTRUCTION was:
| Your task is to:
| - Carefully review the original response for correctness,
   relevance, and clarity.
| - Provide constructive criticism, focusing on areas where the
   response could be improved.
| - Offer specific suggestions for how to refine the response,
   ensuring it better meets the requirements and addresses any
   potential issues.
| REFLECTION TASK
| Correctness:
| - Are the inferred query and relevant objects accurately
   identified?
```

```
| - Is the query achievable status appropriately determined?
| - Is the explanation logically sound and coherent?
| Relevance:
| - Are all relevant objects identified, and are they sorted by
   relevance accurately?
| - Are any crucial objects or details omitted?
| Clarity:
| - Is the response clear and easy to understand?
| - Are there any ambiguities or vague descriptions?
| CRITIQUE TEMPLATE
| Please review the response carefully for correctness, relevance,
   and clarity, and provide constructive criticism for how to
   improve it. Follow the structure provided below:
| Correctness:
| [Your comments on correctness]
| Relevance:
| [Your comments on relevance]
| Clarity:
| [Your comments on clarity]
| SUGGESTIONS FOR IMPROVEMENT
| Based on your critique, please suggest specific improvements.
   These can include corrections to factual errors, inclusion of
   missing relevant objects, reordering of objects by relevance, or
   clarifications to enhance understanding.

The RESPONSE was:
{{self_reflection_response}}

### INSTRUCTION ###
Your task is to apply the feedback provided in the critique to
   produce a refined and corrected response.
Note that the response must again be a JSON, with the same keys as
   the first LLM call, but with the values corrected according to
```

```
   the feedback.
1. "inferred_query"
2. "relevant_objects"
3. "query_achievable"
4. "final_relevant_objects"
5. "explanation"
```

Note that *object_list_str* should be replaced with the semantic map in JSON format, *planner_response* with the first semantic map exploitation response, and *self_reflection_response* with the self-reflection response obtained in the previous step.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA