Integrating Semantic Information in Navigational Planning

Henning Deeken, Osnabrück University, hdeeken@uni-osnabrueck.de, Germany
Sebastian Pütz, Osnabrück University, spuetz@uni-osnabrueck.de, Germany
Thomas Wiemann, Osnabrück University, twiemann@uni-osnabrueck.de, Germany
Kai Lingemann, DFKI Robotics Innovation Center, Osnabrück Branch, kai.lingemann@dfki.de, Germany
Joachim Hertzberg, Osnabrück University and DFKI Robotics Innovation Center, Osnabrück Branch, joachim.hertzberg@uni-osnabrueck.de, Germany

Abstract

Executing high-level tasks in a fetch-delivery-scenario (e.g.,"Robot, bring me a muffin!") requires a robot to reason about *what* the desired object is and *where* it is located. At best, the underlying map to plan and execute such tasks allows for a close linkage of geometric and semantic information and efficient look ups. Geographic Information Systems combine relational and geometric data in one database and can serve as powerful backends for semantic mapping, because of their in-built ability to handle spatial queries. In this paper, we present a GIS-based approach to semantic mapping that allows semantic labels to be integrated closely with geometric environment maps. We show that our semantic map is suited to store and maintain information that supports complex task planning, as well as other tasks. As an example, we show how robot navigation can be improved by including semantic information.

1 Introduction

Fetch-delivery-scenarios are commonly posed benchmarks for autonomous robots, because they require the integration of different domains. In the following, such a scenario triggered by high-level commands, like "Robot, bring me a muffin from the kitchen!", will be considered. To execute such a command, a robot has to solve a sequence of problems: Besides understanding the given command, it is most crucial to reason about *what* the desired object is and *where* it is located with respect to the robot and the destination. Based on this information a suitable decomposition of individual actions can be executed. For this, an extensive set of techniques ranging from localization, navigation, spatial reasoning, task planning to object identification and manipulation, has been developed in the past. Usually, these functionalities are individual components that, though interacting with each other, operate on separated data.

The idea of *semantic maps* promises to bring semantic, topological and geometric information closer together by building maps that merge this information to support task planning and reasoning. However, the notion of semantic maps can be extended to include a variety of other robotic functionalities. We contribute to this topic by presenting a new architecture that suites this extended notion. To illustrate the capabilities of our approach, we chose robot navigation as a showcase project for our experiments, since navigation is essential to fetch-delivery-tasks. **Figure 1** shows how our architecture combines semantic mapping and robot navigation within this paper. The structure of the text follows the depicted data flow. At first we introduce *Geographic Information Systems* (GIS) as database backends for semantic mapping (A) and discuss how they can be used to combine the geometry, topology and semantics of an environment (B). We will describe how our semantic map is structured and how it can provide the necessary information for planning a robot's path on a topological level (C1). To subsequently execute the path, we show how the geometric and semantic content of our robot map is used to extract augmented navigation maps (C2). As these maps conform with standard robot navigation techniques used in the ROS framework [1], we demonstrate how the extracted maps are used to implement navigation with in both 2D and 3D collision awareness (D).

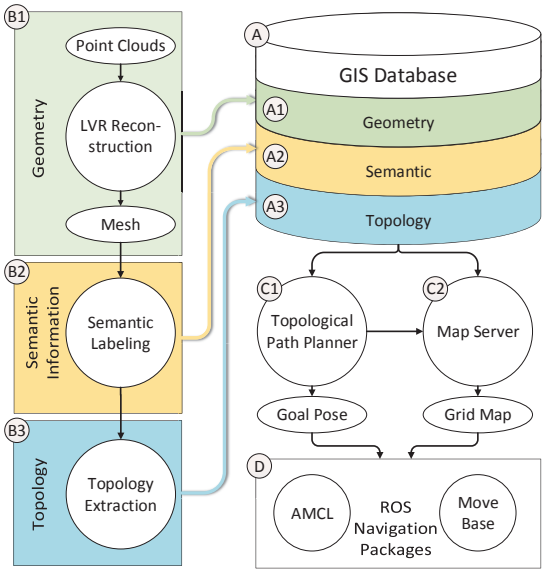


Figure 1: The proposed semantic mapping framework is based on a GIS database that holds geometric, semantic and topological information and can be used to extract semantically annotated grid maps used in robot navigation.

ISBN 978-3-8007-3601-0

2 Related Work

Robot navigation is a well-studied field with efficient methods publicly available. Two-dimensional grid map and grid map based cost maps have become standard solutions for self-localization, and motion planning and execution, respectively. Within the ROS framework, the `gmapping` package based on work by Grisetti et al. [2] solves the SLAM problem to create grid maps from 2D laser data. These grid maps are used in the `amcl` and `move_base` packages based on [3, 4], that implement localization and navigation. Despite their robustness and high performance, these 2D approaches have two major drawbacks: First, the used maps are only 2D and can represent 3D information at best indirectly. Secondly, these approaches are most often detached from 3D data processing, e.g., object recognition and manipulation pipelines.

Addressing the first issue, Hornung et al. [5] proposed a method to surpass the 2D limitation by extracting multiple 2D layers out of an octree based 3D representation [6]. By dividing the 3D navigation problem into multiple 2D sub-problems that take the robot's 3D geometry into account, they were able to efficiently solve planning tasks in 3D. The underlying octree can be used to represent the environment for both navigation and manipulation. However, the presented approach relies on a geometrical environment representation only. Semantic mapping addresses this issue by adding topological and semantic information.

Various approaches to semantic mapping have been developed, but have not yet merged to a standard solution. Some approaches concentrate on integrating topological information within a semantic map. Bastianelli et al. [7] recently showed how to combine semantics and topology to support high-level task execution like global path planning or object manipulation. Their topological navigation is based on semantically annotated 2D maps acquired via 2D SLAM and a human instructor, who labels objects and locations. Other approaches concentrate on the automatic collection of semantics. Works like [8] showed how different types of rooms can be identified by means of analyzing visual cues and geometric features, like area and shape in 2D.

Semantic mapping in 3D is often done by using a 3D model of the environment with semantic annotations to solve complex manipulation tasks [9] or focuses on automatic symbol grounding of complex geometric objects, like furniture recognition [10], CAD-matching [11] or automatic extraction of a room's geometry [12]. Within this line of work, choosing a 3D data format is of importance. Traditionally, point-based data formats dominate the robotics community. Creating geometric maps of large-scale environments with point data can be demanding in both memory and time consumption since a large number of point samples has to be processed. Therefore using other data formats is beneficial. Especially mesh-based geometric models can increase efficiency by modeling continuous surfaces. The Las Vegas Reconstruction Toolkit (LVR) [13] is able to calculate compact, yet geometry preserving triangle meshes from point clouds. In the semantic mapping context, this data format is of advantage, as it can be efficiently clustered and segmented due to the internally used linked data structures.

All these advances towards semantic mapping cover fruitful approaches, but combining the different aspects remains to be done. We want to contribute to this by introducing a new database backend to combine different layers of data. Here we show how semantic mapping in 3D can be a tool to make the various robotic functionalities more consistent by using one common data pool and exemplify this by improving robot navigation using semantic information.

3 Semantic Mapping Framework

Classically, semantic mapping is understood to support task planning and reasoning: "A semantic map for a mobile robot is a map that contains, in addition to spatial information about the environment, assignments of mapped features to entities of known classes. Further knowledge about these entities, independent of the map contents, is available for reasoning in some knowledge base with an associated reasoning engine." [12].

In this paper, we promote the idea of using semantic maps as data source for a multitude of tasks. To support this idea, we directly address the first part of the definition by Nüchter et al. [12] and focus on how to represent, store and manage the content of semantic maps. We will show how our system supports both practical robotic tasks, as well as knowledge-bound reasoning, mentioned in the second part of the definition. We combined semantic mapping, topological path planning and robot navigation in an integrated system implemented in ROS.

3.1 Requirements

A semantic map is intended to provide information about an environment by joining geometric, semantic and topological information. The stored information must be retrievable in a way that enables the robot to plan interactions with the environment. An ideal map must support dynamic updates and is independent of the robot's geometry. It is a multi-purpose map with several different information levels, hence the different layers must be accessible quickly. Queries that ask for just one type of information, as well as complex combinations that involve different modalities must be enabled. To fulfil these requirements, we use the Geographic Information System PostGIS.

3.2 Geographic Information Systems

Geographic Information Systems (GIS) are designed to manage content of geometric nature in combination with additional data. PostGIS, for example, is a derivate of PostgreSQL that adds specific data structures to efficiently store and query geometric information. It can

ISBN 978-3-8007-3601-0

combine standard SQL-queries with geometric queries and allows to link geometric and contextual information in a common database. The built-in functionalities allow to derive spatial relations between stored facts such that more complex information can be derived from the database than the explicitly stored facts. The ability to create and execute geometric queries make GIS-type databases very suitable within the semantic mapping context, as they match the given requirements we defined before.

3.3 Database Content

We designed our PostGIS database (A) in a way that every entry inherits from a base structure that binds geometric, semantic and topological information. This structure is two-fold. At first, it consists of an *ID* that uniquely identifies the entry within the database. The ID is used to link entries with each other or to communicate about the database content in modules outside of the GIS architecture. Secondly, to form a model of the environment within the semantic map each entry contains a triple of the following components:

(A1) A *geometric model* that defines the shape of the entry and its pose with respect to the map origin.

Geometry can be represented by several different data types, e.g., triangle meshes or point clouds can be used to describe objects and rooms. Even more abstract geometries like bounding boxes or references to other geometric sources like a database of CAD-models can be used, as long as they can be expressed in relation to the map's origin.

(A2) A *semantic type* that describes the semantic meaning of the entry and optionally a set of *properties* that additionally specify an entry.

The semantic type ties perceived real world objects to concepts represented in the background knowledge. For example, a classical T-Box can be used to give background knowledge and a comprehensive description of concepts named by the semantic type. A matching A-Box can store all the features of the distinct object instances. The properties allow to change the robot's behaviour with respect to the specific semantics of a concept's instance. The database ID can be used to link between the semantic map and the A-Box entries. This representation allows to reason over real world instances that were grounded by sensory perception.

(A3) A set of *topological links* that defines the spatial relations between the entries in the semantic map.

Due to the nature of the GIS, one can differentiate between implicit and explicit topological information. PostGIS comes with mechanisms to derive spatial relations by analyzing the geometric models. For example, the *ST_CONTAINS* function can be used to check if the geometric model of a object is contained in the geometric model of a room. This information is implicitly stored. To increase the accessibility of topological information it is further possible to model the relations between entries explicitly, by connecting entries and specifying the nature of this connection, as described in the next section.

To fill the semantic map with content, data of the three different modalities needs to be provided (B). Note that each entry should be instantiated on all three domains to be completely functional. However, it is possible to have empty domains, e.g., due to missing information during map creation. The fact that an entry is not fully specified alone is helpful, as it can be used to encourage further exploration of the environment.

3.4 Environment Model

Based on this structure, we defined a set of object classes and implemented a small, but fully functional robot domain that allows to execute commands like "move to the kitchen and fetch a muffin". For sake of simplicity, in this paper we consider an office environment that consists of only one floor. The full setup of our semantic map is capable of representing a multi-storey building with various topological connections (e.g., elevators).

Topological Information A floor consists of several entities with the semantic type `Room`, which separates the environment. Each room is associated with a triangle mesh that represents its geometry and can be associated with a function by giving a property label, e.g., `kitchen`. Elements of the semantic type `Door` are used to connect a room to its neighbours. Each `Door` holds detailed information on how to go from one room to another within its semantic properties. This information defines the topological relations, which can be used to plan robot navigation on a topological level. See **Figure 2** for an example. For more complex environments, other connector types have been modeled to represent the different topological transitions which are bound to the predefined robot behaviours to handle the transition, cf. Section 5.2.

Objects To make the environment interactive, dynamic objects of different semantic types like `Table` and `Muffin` were defined. The geometric model of an object is relevant for guiding object identification or grasp planning and is either grounded in sensor data or given by a detailed model retrieved through database-driven object recognition. Each object belongs to the inventory of a room or an object that can have an inventory itself, like tables. By binding objects to inventories, they are explicitly grounded in the topology of the environment. Even though they are already implicitly bound by their pose, we decided to explicitly store objects in inventories as it allows for quicker access, e.g., by using a room's ID to look up contained objects.

ISBN 978-3-8007-3601-0

Regions Another capability of our framework is identifying regions the robot can interact with. In addition to the overall semantic types of rooms or objects, it is further useful to model the semantics of sub-components of these entities. We consider a *region* to be a subset of the geometric model of either a room or an object. Regions come from the geometric properties and unite these in a more abstract way, e.g., surfaces or clusters. Regions can be found by clustering and surface algorithms. As regions are entries in the GIS, it is possible to classify regions with a semantic type (e.g., `Staircase`) and to denote their properties. The properties of a region can invoke specific robot behaviours. For example, a staircase that is annotated as `blocked` can prohibit the robot to navigate into this part of the room. *Navigationally relevant regions* like this become important in Section 4.2.

Objects can be used to define regions, too. For example, a table stored in the semantic map can be surrounded by regions labeled as `observational area`, which guide the object recognition.

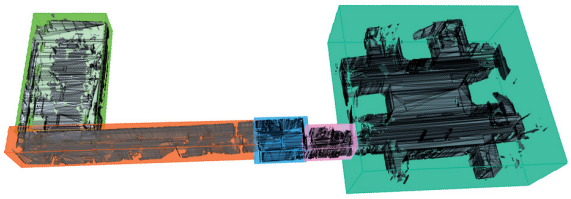


Figure 2: A visualization of separated rooms that constitute the topology of the environment. Separated rooms are wrapped with colored bounding boxes.

3.5 Data Retrieval

Using a GIS database as backend for semantic mapping is especially beneficial when it comes to reading information from the map. Using SQL query statements it is possible to design individual look up procedures for different applications. Due to the relational links between semantic labels, topological information and geometric primitives, special queries can be defined to derive useful information for specific tasks.

The SQL query in **Figure 3** demonstrates how to retrieve information across the different layers in the database. The statement (a) finds all instances of the semantic type `Table` that have a `Muffin` object in their inventory and (b) returns the property label of the containing room together with the table's name and bounding box.

The query includes a SQL join of tables *face*, *fa_to_ob*, *object* and *inventory*. To get all `Table` objects which hold a muffin the query contains a join of the SQL tables *ob_a* and *ob_b* over *inventory*. It fetches the faces of all affected objects and the PostGIS function *Box3D* returns the axis-aligned bounding box from the respective triangle collections.

```
SELECT ob_a.label, ob_b.label,
   Box3D(ST_Collect(face.triangle))
FROM object ob_a, inventory inv_ab,
    object ob_b, inventory inv_bc,
    object ob_c, face, fa_to_ob
WHERE ob_a.type = 'room'
AND inv_ab.inv_id = ob_a.inv_id
AND ob_b.id = inv_ab.ob_id
AND ob_b.type = 'table'
AND inv_bc.inv_id = ob_b.inv_id
AND ob_c.id = inv_bc.ob_id
AND ob_c.type = 'muffin'
AND fa_to_ob.ob_id =  ob_b.id
AND face.id = fa_to_ob.face_id
GROUP BY ob_a.label, ob_b.label
```

(a) An example PostGIS query.

Room	Object	Box 3D
office4	desk4	Box3D(-1.1 3.8 -1.4, ...)
lab2	desk1	Box3D(-1.9 0.3 -1.4, ...)

(b) Database response for the requested data.

Figure 3: All table objects which contain a muffin in their inventory are extracted from the PostGIS back end.

This example shows the advantages of using a GIS for semantic mapping. The information necessary for processing high level tasks can be retrieved from the database. The given statement also demonstrates how explicitly modeled semantic information, as well as implicitly stored geometric information, can be extracted by one combined query. Generally it is possible to retrieve all the geometric primitives and bounding boxes of all objects by their semantic type or properties. Vice versa all objects with certain properties within a given bounding box can be queried.

4 Topology Based Navigation

To accomplish a fetch-delivery-task, a robot has to navigate from its current position to the desired object and back again. The following section shows how the semantic map is used to achieve safe robot navigation invoked by topological assignments. Hereby the data flow Ⓐ to Ⓓ of Figure 1 will be addressed.

4.1 Planning & Execution

Even though robot navigation finally comes down to sending the robot from one metric pose to another, it is helpful to plan a robot's path on a topological level first. Especially during the execution of complex tasks, planning in topological chunks is preferable, since they can be executed step-by-step to better account for the involved transitions, like opening a door before moving through it or using an elevator.
At first, a command like "Move to the kitchen!" needs to be fed into the planning system. We assume that the planning unit Ⓒ1 is provided with at least the following information: the room where the robot resides, which object it

ISBN 978-3-8007-3601-0

has to fetch and to which room it has to deliver to. Based on this information the planning unit calls the semantic map to get the required topological and geometric information, e.g., the `Room` instances that contain instances of the desired object type. Once all sub goals are identified, a path connecting them needs to be found.

Any graph planning algorithm can be used to do this, once the topological structure of rooms and connecting doors is extracted from the semantic map. For example, we used an *answer set programming* approach [14] to plan a sequence of predefined actions that can be executed to solve the given task. The necessary symbolic information, like the room instances and their connecting doors, is extracted from the GIS database and fed into the planner. **Figure 4** shows an example of how the robot can be asked to fetch a `Muffin` object and bring it to a distinct `Table` instance `desk1`. Note that the exemplified task uses our extended environment model, which includes an elevator so that the robot can to move from one floor to another.

```
goal(t) :- on(t, muffin, desk1)
```

```
 1, pass_door, (door1, "lab1", "corridor1")
 3, pass_door, (door2, "corridor1", "foyer-1st")
 4, call_elevator
 5, pass_door, (edoor1, "foyer-1st", "elevator-1st")
 6, use_elevator, ("floor-1st", "floor-3rd")
 7, pass_door, (edoor3, "elevator-3rd", "foyer-3rd")
 8, pass_door, (door5, "foyer-3rd", "corridor3")
 9, pass_door, (door6, "corridor3", "office4")
10, pickup, (muffin, desk4)
11, pass_door, (door7, "office4", "corridor3")
12, pass_door, (door8, "corridor3", "lab2")
13, place, (muffin, desk1)
```

Figure 4: An example task that commands the robot to pick up a `Muffin` and bring it to the `Table` instance `desk1`.

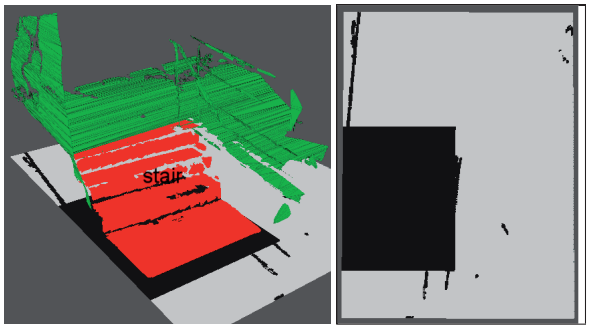
As solution to the given problems, the planner returns a sorted set of `Door` connectors, that specify the transitions that bring the robot from one room to another. The individual transitions through a door or from one door to another can be used to divide the execution of the entire path into smaller parts. As each door is specified by two poses that help the robot navigate through the door, each stage of the path can be executed by handing the goal poses down to the module that moves the robot (D). In complex transitions, like calling an elevator or passing a locked door, the behaviour is handled by appropriate modules designed to deal with the given requirements.

For the navigation between two metric poses, we use the ROS package `move_base`, in combination with the `amcl` package that provides sensor-driven localization (D). Both modules operate on grid maps. `move_base` interprets the occupied space within a grid map as obstacles and creates a cost map based on this information. By inflating obstacles with a suitable padding it is able to calculate a collision aware path between two poses. Further it manages the execution of the calculated path. `amcl` uses a grid map for matching against sensor data.

Usually the underlying maps are created by 2D SLAM methods (e.g., `gmapping`) and thus represent only a section of the environment's geometry. By using 3D geometry stored in the semantic map, we can derive the same kind of grid maps. But the semantic map allows to generate even more expressive grid maps by augmenting them with semantic information. To use this kind of semantically annotated navigation maps, we implemented a map extraction method (C2) that we will discuss in the next section. Further we use this map extraction method to extend the standard ROS components to perform collision awareness in 3D. We present this approach in Section 4.3.

4.2 Map Extraction

To derive grid maps from the semantic map, it is necessary to identify the components that are relevant for navigating the robot. Here the semantic information is of advantage, because it allows to segment the environment into navigationally relevant and irrelevant parts. This segmentation can also be used to distribute the different parts onto individual grid maps, so that they can be treated independently. For example different obstacle padding can be applied to different objects. In our implementation we make use of different layers, that are later joined into one grid. We use a *Room Geometry Layer* that contains only the fixed geometry of the environment (e.g., walls and pillars), as well as a *Blocked Regions Layer*, which contains all regions that are completely blocked for the robot. Further the *Admissible Objects Layer* is designed to include objects the robot is allowed to approach closely objects by using a low padding, whereas the *Lethal Obstacles Layer* includes all objects the robot must avoid and applies a high padding (e.g., `Tables` that are labeled as `human workplace`).



(a) A `Staircase` region. **(b)** The resulting grid map.

Figure 5: The figure shows how a region within the environment can be semantically annotated to be blocked for the robot and how this information is translated into occupied space in the extracted grid map.

The grid maps are created by projecting the geometric primitives of the relevant parts onto the grid. They can be retrieved from the GIS by using the semantic types of the desired components. Depending on the geometric model of the involved components the projection onto the grid map slightly differs. Map components that are represented by triangle meshes, e.g., a room's geometry, are included in a 2D occupancy grid by computing all inter-

ISBN 978-3-8007-3601-0

sections of the mesh with planes parallel to the the ground plane. Depending on the use case it is possible to either "slice" a mesh once, for example to generate a localization map, or multiple times to comprise a 3D geometry into a 2D grid map. In the latter case, the map is created by calculating the intersections of multiple planes along the Z axis between an interval $[a, b]$ with a fixed increment. The segments that are created when a plane cuts a face of the mesh are subsequently drawn onto the grid. A map created by slicing along the full height of the robot can, for example, be used for collision aware path planning. Both the slicing of the meshes and the drawing of line segments onto the grid map includes a discretization, but can be refined arbitrarily to the needed resolution since the base data is continuous. If the geometric model is a box, as in case of `blocked` regions, the entire area of the bounding box can be transformed into *occupied space* in the grid map.

In conclusion, the extraction of navigation maps from the geometric information stored in the semantic map leads to a more consistent approach to robot navigation, as the grid maps root in the same data that supports other functionalities, like grasp planning, task planning et cetera. This distinguishes our procedure from other systems that rely on grid maps derived from 2D data, which are not necessarily synchronized with the pipelines that process 3D data. Further the ability to augment the maps with semantic information is clearly of additional value. The staircase depicted in **Figure 5**, for example, could not be included in a navigational map by considering the environment's geometry along the height of the robot alone, because it is below the ground floor. Since its semantic label implies that it is `blocked`, including this region makes the map more accurate and thus navigation safer. Another feature of this extraction process is that the scope of the maps can be set dynamically. The topological planning divides a given route into distinct parts, thus maps that span only the currently relevant part can be generated. As a result of this, only the necessary segments can be swapped into working memory to reduce memory consumption.

4.3 3D Navigation

With the grid generation method at hand it is possible to overcome another problem of classical robot navigation. Since classically the collision checks rely on grid maps, it is either incomplete, in the case of maps created from pure 2D data, or overprotective, in case of maps that have been created by projecting the entire 3D geometry onto one grid. Since our grid extraction is fast and can be restricted to cover only the local area around the robot, it is cheap to create multiple navigation maps which can be used to implement 3D collision awareness.

Hornung et al. introduced the idea that multiple grid maps can be used to reduce the problem of checking for collisions in full 3D to several 2D sub problems [5]. Their approach is based on the idea that a robot's geometry can often be segmented into several segments that can be represented by their own collision map. Such a decomposition allows to better account for the robot's 3D geometry, as each robot part can be checked individually.

We followed these ideas and slightly extended the standard ROS components to allow for 3D aware planning with multiple 2D layers extracted from our GIS database. We sorted the kinematic links that make up our robot into functional units, e.g., the *arm*, the *base structure* and the *pole* that mounts the sensors. These units can be used to dynamically derive the "slicing" intervals $[a, b] \subset Z$ by using the upper and lower bound of such a unit.

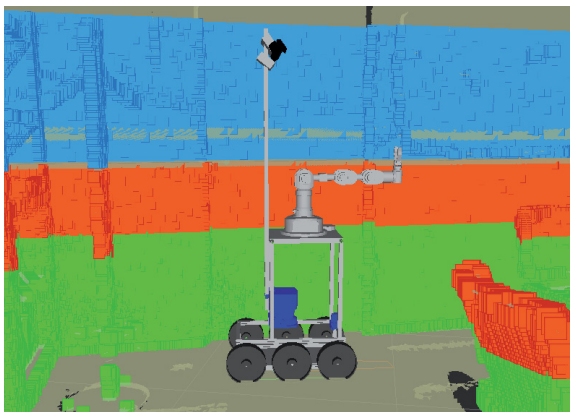


Figure 6: A geometric decomposition of the robot can be used create multiple cost map layers to respect the 3D geometry during motion planning. The colors of the different sections match the ones in Figure 7.

The resulting intervals are used generate grid maps of the environment's geometry, so that only the geometry that affects this part of the robot will be included in the collision map of this functional unit. An illustration of this segmentation can be found in **Figure 6**. Here we also include the semantic information relevant for the different sections. The resulting grid maps are used to create individual cost maps for each robot component.

Usually, `move_base` operates on a single cost map that can consist of different grid layers, but allows for only one type of obstacle inflation. We extended the implementation to allow cost maps that consists of different cost maps, which all have their individual inflation level and set of grid map components (e.g., the layers defined in Section 4.2). To correctly inflate the obstacles in the cost map of each robot part, it is necessary to derive the individual footprints of the different functional units.

The respective footprints can be calculated dynamically from the robot's current kinematic state. Given a geometric description of the robot and its current configuration, the geometry of each functional unit can be projected onto a 2D plane. The convex hull of this projection can serve as footprint for the respective robot part. **Figure 7** shows how we divided the robot into three functional units and show the different footprints that were derived from the robot's kinematic state.

Each footprint is used to calculate the inflation radius that expands the obstacles in the respective cost map layer, as discussed in [4]. Note that if the robot's configuration

ISBN 978-3-8007-3601-0

changes, the footprints and the corresponding cost maps will be updated automatically. Finally, the different cost map layers are combined to one master cost map that is used for motion planning. Since for each robot part was accounted for by inflating its respective layer with a suitable radius, the master map can be seen as the terrain a dot-shaped robot is allowed to navigate in. We use this master cost map to implement a simple 3D aware robot navigation with the standard `move_base` package.

Our setup is not as expressive as the one by Hornung et al., since it does not take a robot's motion primitives into account during the planning process. The approach in the original paper allowed holonomic robots to slide through highly cluttered terrain. However our simpler approach suits the need of our non-holonomic robot. Due to the different cost maps it is possible to navigate the robot arm above the table, while part of the base structure is moved below the table top, as can be seen in **Figure 9**.

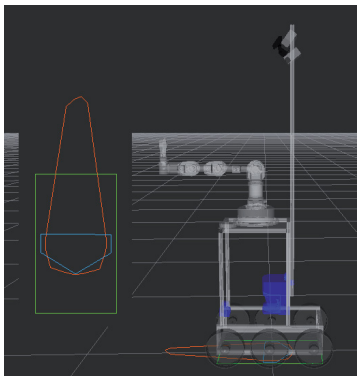


Figure 7: The figure shows the different robot parts and their individual footprints. The base structure mounting the arm is represented in green. The arm's footprint is shown in red, whereas the footprint of the pole that mounts the robot's sensors is visualized in blue.

5 Experiments

As mentioned before the presented approach to semantic mapping was implemented in ROS. By defining interfaces between the PostGIS database, our topological planning module the custom-made map extraction mechanism and the standard ROS navigation components, we created an integrated system. In the following, we will briefly describe how we use a pair of robots to, first, build the semantic map and, second, to fetch and deliver small objects.

5.1 Building the Semantic Map

To gather data for the semantic map, a mobile robot platform equipped with a rotating 2D laser scanner that periodically outputs 3D point clouds of the environment was used. The original data was collected in a semi-automatic mapping procedure that consists of sending the mapping robot to a sequence of scan poses. Using `3DTK`'s SLAM algorithms [15], the different scans are fused into a common geometric map. The nodes of the topological graph are created during the mapping run by annotating the incoming data on-the-fly with the `Room` instance's ID. To get the full topological representation of the environment the topological connections (e.g., the `Door` instances) are specified manually. To annotate rooms, objects and region with their semantic types and properties, we developed a set of plug-ins for the ROS visualization tool RViz. Note that this information can in principle be extracted without user interaction as well. For example by using the automatic segmentation processes, like described in [10, 11] or [12].

Figure 8 shows an examples of how a part of the semantic map can be visualized. Note that the colored parts denote different objects. The semantic types of the objects are visualized as well.

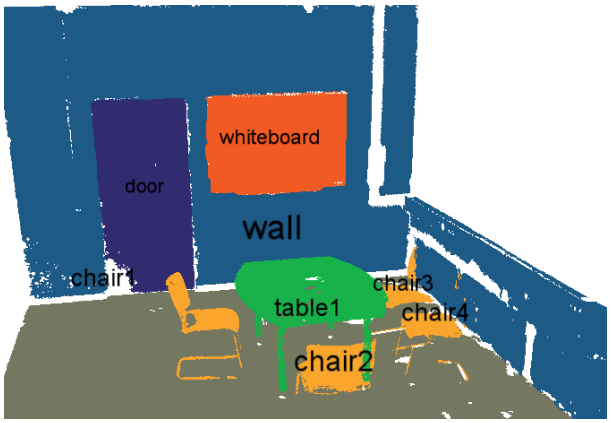


Figure 8: The figure shows a 3D mesh annotated with semantic labels. The labeling was done by hand using our RViz label tools.

5.2 Using the Semantic Map

As our semantic map is robot-independent, it can be used by multiple robots. So to test out our semantic map we commanded the robot shown in Figure 7 to perform different fetch-delivery-tasks, since its 5 DOF manipulator is able to pick and place small objects. We used the robot within a multi-storey building and to support transition between different floors, with additional topological connections to enter and operate the elevator. To enable the robot to manipulate small objects, we further implemented a simple pick and place application. Within this extended domain, we commanded the robot to perform tasks of different complexity.

To provide missions to the robot, we rely on a state machines that waits for user input, calls the planning unit and delegates the execution of individual actions, which are identified by the planning unit. We implemented different protocols that implement certain kind of task by using different retrieval queries to get the relevant information from the GIS database. One on these is designed to used the topological navigation module only, for example, by sending the robot to distinct rooms naming

ISBN 978-3-8007-3601-0

their ID ("Move to `office2`."). Another commands the robot to move to a certain type of rooms by using the semantic properties during database retrieval ("Move to a `kitchen`"). We also implemented a homing behavior by labeling a regions within our lab map as `home`. If the robot is commanded to go home, a path to the lab planned and finally the robot moves into it's home by drawing an admissible pose from the given region. To include the other robot functionalities, we further implemented query statements to determine the topological location and exact poses of objects. This allows the robot to first move to the desired room and at arrival to start the object manipulation pipeline with the absolute location of the object.
In order to come up with a solution to these given tasks, the controlling state machine extracts the entire topology and object content of the environment from the semantic map and provides them to the answer set programming solver, which subsequently calculates a plan, c.f. Figure 4. The derived actions are sequentially handed down to several subordinate state machines that enable the robot to perform the required action.
During the execution of the different tasks, our semantic map proved to be a reliable source to provide all the necessary data for the different components.

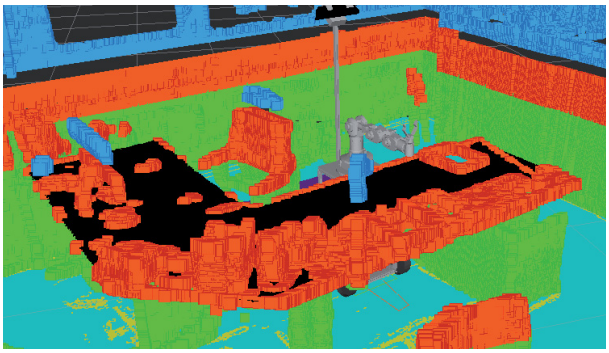


Figure 9: Due to the different collision maps it is possible to move the robot's base under the table.

6 Conclusion

This paper presented a general architecture for semantic mapping that supports planning and executing complex tasks by drawing all necessary information from a semantically annotated 3D representation. We showed how GIS databases can lead to a desirable link between geometric, topological and semantic information and how they can be use to retrieve data across the different domains by using mixed query statements. We further proved that semantic mapping can be useful beyond knowledge-based applications, by using the map to provide the necessary grid maps for the navigation modules. We demonstrated how the mesh data that is used to represent the environment's geometry is suitable to extract grid maps from the semantic map. By including semantic information into the grid maps we exemplified that semantic maps bring useful information into the domain of robot navigation.
For future work, we intend to extend the connection between the semantic map and the robot navigation modules to support the object manipulation pipeline. Using the 3D collision aware navigation coupled together with grasp planning techniques will allow to directly infer suitable approaches towards an object. Further we intend to couple our GIS back end to a full-fledged reasoner with a suitable background knowledge to make our semantic map more expressive.

References

[1] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: A open-source robot operating system," in *Proc. ICRA 2009 Workshop on Open Source Software*, 2009.

[2] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[3] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, pp. 343–349, 1999.

[4] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Proc. ICRA 2010*, May 2010.

[5] A. Hornung, E. Phillips, E. G. Jones, M. Bennewitz, M. Likhachev, and S. Chitta, "Navigation in three-dimensional cluttered environments for mobile manipulation." in *Proc. ICRA 2012*, 2012, pp. 423–429.

[6] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, 2013.

[7] E. Bastianelli, B. D. D., F. Cossu, G. Gemignani, L. Iochhi, and D. Nardi, "On-line semantic mapping," in *Proc. ICAR 2013*, 2013.

[8] A. Pronobis and P. Jensfelt, "Large-scale semantic mapping and reasoning with heterogeneous modalities," in *Proc. ICRA 2012*, 2012.

[9] N. Blodow, L. C. Goron, Z. Marton, D. Pangercic, T. Ruhr, M. Tenorth, and M. Beetz, "Autonomous semantic mapping for robots performing everyday manipulation tasks in kitchen environments," in *Proc. IROS 2011*, 2011.

[10] M. Günther, T. Wiemann, S. Albrecht, and J. Hertzberg, "Building semantic object maps from sparse and noisy 3d sata," in *Proc. IROS 2013*, 2013.

[11] S. Albrecht, T. Wiemann, M. Günther, and J. Hertzberg, "Matching cad object models in semantic mapping," in *Proc. ICRA 2011 Workshop on Semantic Perception, Mapping and Exploration*, 2011.

[12] A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 11, 2008.

[13] T. Wiemann, K. Lingemann, A. Nüchter, and J. Hertzberg, "A toolkit for automatic generation of polygonal maps – las vegas reconstruction," in *Proc. ROBOTIK 2012*. VDE Verlag, 2012.

[14] M. Gelfond, *Handbook of Knowledge Representation*. Elsevier Science, 2008, ch. Answer Sets.

[15] A. Nüchter, K. Lingemann, D. Borrmann, and J. Elseberg, "3DTK - The 3D Toolkit," 2012, http://slam6d.sourceforge.net.

ISBN 978-3-8007-3601-0