electronics

Article

A Flexible Semantic Ontological Model Framework and Its Application to Robotic Navigation in Large Dynamic Environments

Sunghyeon Joo [1], Sanghyeon Bae [1], Junhyeon Choi [1], Hyunjin Park [1], Sangwook Lee [2], Sujeong You [2], Taeyoung Uhm [3], Jiyoun Moon [4] and Taeyong Kuc [1,*]

1 Department of Electrical and Computer Engineering, College of Information and Communication Engineering, Sungkyunkwan University, Suwon 16419, Korea; sh.joo@skku.edu (S.J.); shbae.skku@skku.edu (S.B.); choijunhyeon@skku.edu (J.C.); hyunjinp@skku.edu (H.P.)
2 Robot R&BD Group, Korea Institute of Industrial Technology, Ansan 15588, Korea; mcr6368@kitech.re.kr (S.L.); sjyou21@kitech.re.kr (S.Y.)
3 Korea Institute of Robotics and Technology Convergence, Pohang 37666, Korea; uty@kiro.re.kr
4 Department of Electronics Engineering, Chosun University, Gwangju 61452, Korea; jymoon@chosun.ac.kr
* Correspondence: tykuc@skku.edu

Abstract: Advanced research in robotics has allowed robots to navigate diverse environments autonomously. However, conducting complex tasks while handling unpredictable circumstances is still challenging for robots. The robots should plan the task by understanding the working environments beyond metric information and need countermeasures against various situations. In this paper, we propose a semantic navigation framework based on a Triplet Ontological Semantic Model (TOSM) to manage various conditions affecting the execution of tasks. The framework allows robots with different kinematics to perform tasks in indoor and outdoor environments. We define the TOSM-based semantic knowledge and generate a semantic map for the domains. The robots execute tasks according to their characteristics by converting inferred knowledge to Planning Domain Definition Language (PDDL). Additionally, to make the framework sustainable, we determine a policy of maintaining the map and re-planning when in unexpected situations. The various experiments on four different kinds of robots and four scenarios validate the scalability and reliability of the proposed framework.

Keywords: semantic navigation; ontology; PDDL; semantic knowledge; semantic map; robotics

check for updates

Citation: Joo, S.; Bae, S.; Choi, J.; Park, H.; Lee, S.; You, S.; Uhm, T.; Moon, J.; Kuc, T. A Flexible Semantic Ontological Model Framework and Its Application to Robotic Navigation in Large Dynamic Environments. *Electronics* **2022**, *11*, 2420. https://doi.org/10.3390/electronics11152420

Academic Editor: Ahmad Taher Azar

Received: 8 July 2022
Accepted: 1 August 2022
Published: 3 August 2022

1. Introduction

Robots now have the ability to independently navigate a variety of surroundings thanks to advanced robotics research. Nevertheless, addressing unanticipated situations while performing complex jobs remains challenging for robots. Robots should execute tasks by comprehending working environments beyond metric data and considering protective measures for various scenarios. Using semantic knowledge recently has become a research trend to solve such problems [1,2]. In cognitive robotics, the concept of "semantic knowledge" can be represented by general knowledge that people use, such as concepts, attributes, properties, and purposes of environmental elements (e.g., object and place) and their relationships.

We introduce a map for robots to explain semantic knowledge. When building a map for people, we first consider how to represent the environment according to their requirements. For example, if we make a park guide map, the map shows the park's structure. In this case, we illustrate the location of significant landmarks and paths connecting them on the map. However, maps used by robots are extremely different. We need to consider the understanding ability of the robot that uses the map. If robots have human-level intelligence, it would be enough to give them the maps we use. However, for robots that do not have human-level intelligence, it is essential to make specific maps that can be perceived and utilized. Building maps for robots is handled as a Simultaneously Localization and Mapping (SLAM) problem in robotics. The SLAM problem can be defined as generating an unknown environmental map and simultaneously estimating the robot's pose on the map.

The early research directions for SLAM were aimed at building a metric map such as a grid, feature, and point cloud map [3,4]. A metric map describes the world using low-level metric information within a fixed coordinate system. Therefore, using the metric map, robots can deduce only the metric location and geometrical structure of the environment. Furthermore, to operate the robots, an administrator should convert their instructions into a metric form that robots can perceive. For example, if we want the robots to "move to a second door", we must give them instructions such as "move to (x,y) coordinate". Controlling the robots with numerical commands is not a problem when performing simple tasks in uncomplicated environments. However, various limitations exist while performing a high-level task (e.g., fetch a package) in complex environments using only metric instructions and information.

An increasing number of recent studies have shown advancements in the map concept by including semantic knowledge on the map to overcome these limitations [2,5,6]. Many researchers have developed numerous semantic SLAM approaches that build enriched maps, known as semantic maps, by extracting semantic knowledge such as environmental elements' labels and geometrical relationships that can be obtained from various sensors [7–11]. These approaches combine extracted semantic knowledge with the metric maps. On the other hand, in the case of implicit knowledge (e.g., conceptual hierarchy, attributes, and properties) that cannot be extracted directly, researchers have defined and represented the knowledge depending on a specific domain for the robot's application [1].

A semantic navigation framework enables robots to perform tasks by utilizing semantic knowledge. The framework's core is modeling and handling knowledge to complete tasks without problems. Our previous work [12] proposed a semantic navigation framework composed of the Semantic Modeling Framework (SMF), the Semantic Autonomous Navigation (SAN) module, and the Semantic Information Processing (SIP) module. Each framework module utilizes semantic knowledge by defining the Triplet Ontological Semantic Model (TOSM). We showed the inspection scenario based on semantic knowledge in an indoor single-floor environment.

In this paper, we present a TOSM-based scalable semantic navigation framework for various conditions: robots, environments, and scenarios. We extend the previous framework with the following contributions:

- Integrating the framework for robots with different kinematics and various tasks using Web Ontology Language (OWL) and Planning Domain Definition Language (PDDL);
- Modeling a TOSM-based semantic knowledge for robots, indoor multi-floor buildings, and outdoor environments;
- Generating a semantic map containing asserted and inferred semantic knowledge of environmental elements: objects, places, and robots;
- Designing a hierarchical planning scheme that utilizes semantic knowledge in each layer;
- For maintenance, updating the semantic map whenever the robot works;
- Re-planning to ensure the framework's reliability when the plan fails.

In the rest of the paper, we introduce and compare the related works in Section 2. Sections 3 and 4 explain the extended framework's modules that define and utilize semantic knowledge. Section 5 describes the experiments that demonstrate the framework. Lastly, we discuss and conclude in Sections 6 and 7.

2. Related Work

Ontology is a method for describing concepts and the relationships between them. Many researchers have adopted ontology-based approaches to design semantic knowledge [1,2,13,14]. Ontological definitions of semantic knowledge can be specified according to each application domain: office [15,16], challenging fields [17], medical [18,19], manufacturing [20,21], domestic [22–26], and convention center [12]. We set the detailed comparison criteria as semantic knowledge (ontology, reasoner, query), planner, scalability (environment, robot type, and task), and reliability (re-planning and map maintenance). Table 1, ordered by year, shows the comparison between semantic navigation frameworks.

In Table 1, (M) represents that multiple methods are used depending on the case for semantic knowledge and planner; methods not mentioned are expressed with a hyphen (−). (In) and (Out) mean that the framework is applied indoors and outdoors. A black circle (●) and ($\surd$) mean that the framework considers various robot types and tasks and has functions for re-planning and map maintenance. An empty circle (○) and a hyphen ($\times$) mean that it does not.

Table 1. Comparison for semantic navigation frameworks.

Name	Ref.	Semantic Knowledge			Planner	Scalability			Reliability	
		Ontology	Reasoner	Query		Env	Robot	Task	Replan	Maintain
Galindo et al.	[15]	DL	–	–	Metric-FF	In	○	●	×	×
OUR-K	[22]	OWL	M	–	[27]	In	○	○	×	×
Dhouib et al.	[17]	OWL	Pellet	–	–	Out	●	●	×	×
KnowRob	[28]	OWL	Prolog	M	–	In	○	●	×	×
Goncalves et al.	[18]	OWL	M	EL	–	In	○	○	×	×
Sadik et al.	[20]	JADE/ACL	Drools	–	Drools	In	○	●	×	×
Kootbally et al.	[21]	OWL/XSDL	–	–	–	In	●	○	√	×
SEMAP	[16]	OWL/SWRL	M	M	–	Both	●	●	×	√
Sabri et al.	[25]	μConcept	DL	–	SmartRules	In	○	●	×	×
Sun et al.	[26]	OWL/SWRL	JESS	–	–	In	○	○	×	√
Chang et al.	[19]	OWL	Prolog	SPARQL	–	In	○	●	×	×
Joo et al.	[12]	OWL	Pellet	SPARQL	POPF	In	○	○	×	×
TOSMNav	Ours	OWL/SWRL	Pellet	SPARQL	POPF	Both	●	●	√	√

While most frameworks do not consider scalability and reliability, our framework, TOSMNav, manages all components of them. We explain some papers and our previous work in detail for further understanding.

Galindo et al. used a semantic map to improve the robot's task planning efficiency [15]. The utilized semantic map is defined by integrating hierarchical spatial information and semantic knowledge. The spatial information, called S-Box, contains the state of elements in the environment (e.g., area, object, and robot) and the spatial relationships between them, such as *at* and *connected*. The domain knowledge for the elements is defined as a terminological component called T-Box. The instances and assertions of the concepts are stored in the assertional component called A-Box. When the robot builds a map, the occupancy map can be classified as *area-4* in the S-Box and linked to *Room*, an ontological concept, in the T-Box. The explained semantic map is represented based on the structure of the knowledge representation system called Description Logic (DL) [29]. They demonstrated the proposed semantic map using MOVE and OBSERVE actions in a home environment.

Lim et al. proposed an Ontology-Based Unified Robot Knowledge (OUR-K) framework [22], which consists of knowledge description and knowledge association for indoor service robots. The knowledge description module separates the knowledge into the world model (objects, spaces, and contexts) and interaction methods with the world (features and actions). The object knowledge contains three hierarchical levels: O_1, O_2, and O_3. O_1 is the part-object level with functional and perceptual parts. O_2 is the object level for name, and O_3 is the compound level for similar concepts. The space knowledge includes three kinds of maps: metric (S_1), topological (S_2), and semantic (S_3). Context knowledge comprises spatial relationships (e.g., on, in, and left) and temporal concepts (e.g., before, after, and met). For the interaction knowledge, the feature knowledge has the perceptual feature and the concept level; the action knowledge has the primitive behavior, sub-task, and task level. The framework's semantic knowledge representation is based on the ontological framework of Karlsruhe Ontology (KAON) in [30]. The knowledge association module defines the relationships between knowledge descriptions using logical inference, Bayesian inference, and heuristics. The proposed framework was verified with a reception service scenario.

In another research work, Tenorth et al. presented the KnowRob-Map system that combines encyclopedic knowledge with instances of an object in the environment [31]. The object-based map is built using the method in [32] and processed to semantically represent the knowledge based on the system, called KnowRob [33]. The KnowRob-Map system integrates both systems to generate knowledge-linked semantic object maps. The maps are constructed using a triple $SemObjMap = (D, O, R)$, consisting of data, ontology, and rules, represented by SWI Prolog [34]. Kunze et al. applied the system to search for objects in large-scale indoor environments [35]. They presented the decision-theoretic search algorithm to maximize the probability of finding the object's location. Other applications using the system are described in detail in the paper [28].

Deeken et al. introduced the Semantic Environment Mapping (SEMAP) framework [16,36]. The framework aims to generate and maintain spatial relationships using Geometrical Information Systems (GIS), specifically PostGIS, and to support query languages for reasoning based on an ontological framework, Apache JENA. The ontological model for the environment representation consists of a core ontology for general knowledge and a domain-specific ontology for various applications. Environmental instances' geometrical information (e.g., points, lines, and polygons) are stored in the PostGIS database and connected with the SEMAP's knowledge database using the properties such as *semap:hasDbId*. The framework utilizes these links to manage and query the semantic map. They applied the framework to the real robot by implementing an interface to the Robot Operating System (ROS) in the office environment.

We proposed the semantic navigation framework consisting of the SMF, the SIP module, and the SAN module [12], based on the TOSM integrating three models: explicit, implicit, and symbolic. The explicit model describes the information obtained from sensors, such as metric state, geometrical characteristics, and image information. The implicit model contains relationships and facts for environmental elements; we classify the environmental elements as *Object*, *Place*, and *Robot*. The environmental elements are symbolized using the symbolic model. The TOSM utilizes OWL terminologies defined in the OWL reference [37] to represent the world. The terminologies consist of the class, object property, and data property. The class defines the hierarchy of the environmental elements that provide "is-a" knowledge of elements. The classes' relationship knowledge and attributes are represented using object and data properties, respectively. We demonstrated the framework using an inspection scenario in an indoor convention center environment.

3. Semantic Knowledge Representation

We propose the SMF based on the TOSM in our previous work [12]. This work extends semantic knowledge for various environments and mobile robots by defining new classes and properties. We explain which semantic knowledge is added compared with the previous work and how semantic maps are generated in the following sections by targeting an experimental environment. For clarity, classes and properties are represented in italic, and individuals as usual.

3.1. Semantic Knowledge for Working Environment

Determining the robot's path and traversable doorways to reach its goal is required when planning a task. The decision inside the semantic navigation framework depends on semantic knowledge of environmental elements.

In this work, we selected a complex experimental environment, illustrated in Figure 1, composed of an office-like multi-floor building, two experimental facilities, and outdoor roads connecting them. In the environment, the office building has an automatic door that has fixed-access time in the main lobby, and the facilities have closed, large gates. Additionally, the building has three floors with two kinds of elevators. One is a small passenger elevator that connects each floor from the inside; the other is a large freight elevator with two doors that connect indoors and outdoors on the first floor and one door on the other floors.

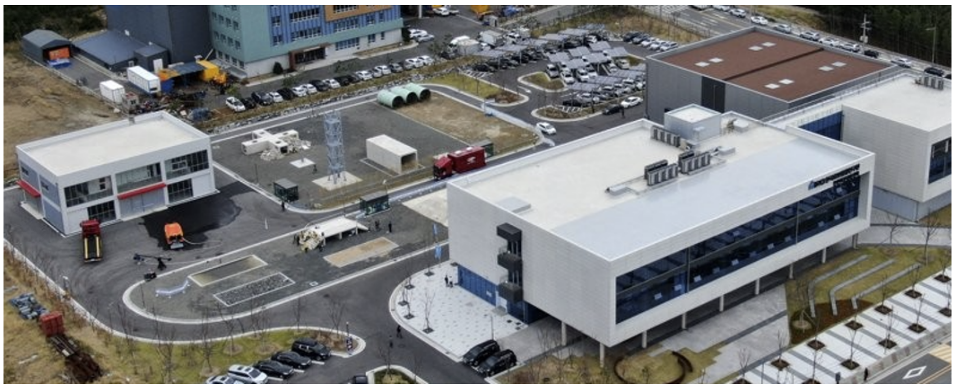


Figure 1. The experimental environment including an office building, experimental facilities, and outdoor roads.

To utilize the environmental information, first, we define data properties for the elements according to the target environment, as described in Table 2. *availableStartTime* and *availableEndTime*, respectively, represent a doorway's open and close time, at the robot can go through; *assignedTaskAt* is the time when the robot receives a task. Moreover, since the robot should open the elevator to take, we include the *canBeOpenedByRobot* property that indicates which elevator the robot can enter. *isLeafPlace* is utilized to represent the lowest hierarchical level of places. In other words, the leaf places do not have any child place represented by the *isInsideOf* property. The leaf places are the criterion for judging navigability when planning. The properties for size in the *explicitModel* are defined for determining whether the robot can pass. When passing the doorway while executing tasks, the robot needs to check whether the door inside the doorway is open or not. At this time, the robot selects sensors to check using *material* knowledge of the door.

Table 2. Data properties for *EnvironmentalElement*.

Data Property Hierarchy		Domains	Ranges
implicitModel	availiableStartTime	Doorway	double
	availiableEndTime	Doorway	double
	assignedTaskAt	Robot	double
	canBeOpenedByRobot	Elevator	boolean
	isLeafPlace	Place	boolean
	material	Door	string
explicitModel	entranceSize	Doorway	double
	footPrintSize	Robot	double

Second, we define the *Robot*'s object properties to utilize semantic knowledge of places. Table 3 shows the definition of the properties. *canWorkingAt* expresses the robot's working area by considering the robot's specification. For example, we can use this property to distinguish whether it is for indoor or outdoor use. *canNotGoThrough* and *canNotRotateInPlace* define the knowledge between robots and leaf places; *canGoThrough* is disjoint with *canNotGoThrough*. *LeafPlace* in ranges means "*isLeafPlace* value *true*" in the class expression. These object properties are used for generating a specific robot's on-demand semantic map.

Table 3. Object properties for *Robot*.

Object Property Hierarchy		Domains	Ranges
robotRelationship	canWorkingAt	Robot	Place
	canGoThrough	Robot	LeafPlace
	canNotGoThrough	Robot	LeafPlace
	canNotRotateInPlace	Robot	LeafPlace

Finally, we use the Semantic Web Rule Language [38] (SWRL) to allow the robot to infer semantic knowledge based on the asserted properties. The SWRL is a language for expressing rules that allow the inferring of new knowledge from existing OWL ontologies. The defined SWRL rules for the environmental elements are as follows:

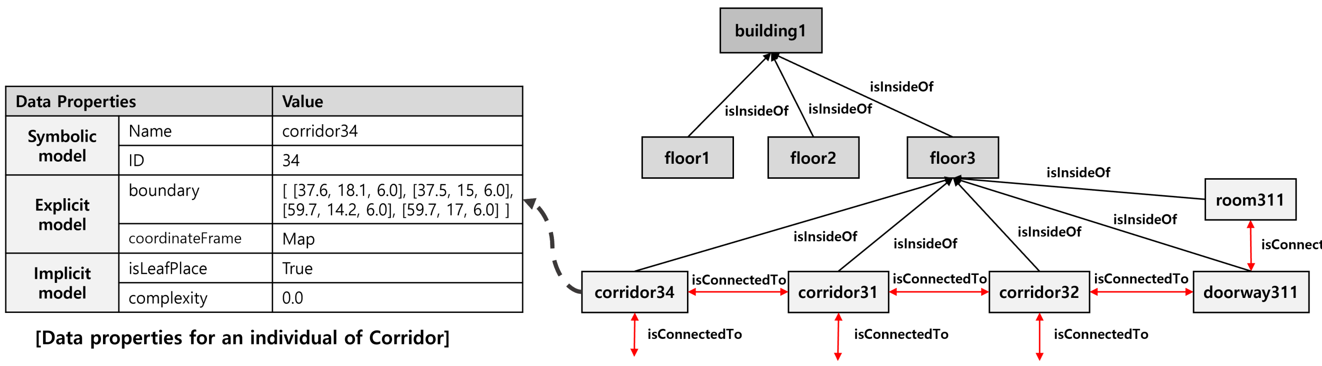
- **Rule 1:**
  Robot(?r) ˆ Place(?p) ˆ canWorkingAt(?r, ?p) ˆ Place(?lp) ˆ
  isLeafPlace(?lp, true) ˆ isInsideOf(?lp, ?p) -> canWorkingAt(?r, ?lp)
- **Rule 2:**
  Robot(?r) ˆ assignedTaskAt(?r, ?amt) ˆ Doorway(?dw) ˆ
  availableStartTime(?dw, ?st) ˆ swrlb:lessThan(?amt, ?st) -> canNotGoThrough(?r, ?dw)
- **Rule 3:**
  Robot(?r) ˆ assignedTaskAt(?r, ?amt) ˆ Doorway(?dw) ˆ
  availableEndTime(?dw, ?et) ˆ swrlb:greaterThan(?amt, ?et) ->
  canNotGoThrough(?r, ?dw)
- **Rule 4:**
  Robot(?r) ˆ footPrintSize(?r, ?fs) ˆ Doorway(?dw) ˆ entranceSize(?dw, ?es) ˆ
  swrlb:greaterThan(?fs, ?es) -> canNotGoThrough(?r, ?dw)
- **Rule 5:**
  Robot(?r) ˆ Elevator(?ev) ˆ canBeOpenedByRobot(?ev, false) ->
  canNotGoThrough(?r, ?ev)
- **Rule 6:**
  Robot(?r) ˆ canNotRotateInPlace(?r, elevator2) ->
  canNotGoThrough(?r, doorway115)

These rules' primary purpose is to generate the robot's on-demand database for task planning. Rule 1 infers the *canWorkingAt* property from places to leaf places. For example, if the semantic knowledge base has the fact that the robot can work for building1, the robot can deduce all workable leaf places that are inside of building1 (e.g., corridor313 and room311). Rule 2, 3, and 4 define the *canNotGoThrough* property between the robot and doorways using each doorway's available time and size. Rule 5 and 6 determine whether the robot can take the elevator based on the robot's locomotion and ability, defined as the implicit model. In the case of Rule 6, environmental-specific knowledge is applied.

3.2. Semantic Map Generation

Generating a semantic map consists of an assertion and inference step. The assertion step involves individuals and their relationships, called assertion components (A-box), from the TOSM-based semantic knowledge definition, called terminology components (T-box). The inference step uses reasoners to generate an inferred knowledge graph based on the characteristics of properties and SWRL rules.

In the assertion step, first, we divide the environment into top place individuals; building1, building2, building3, and outdoor1. Each top place consists of child places such as floors, corridors, or doorways. Figure 2 shows part of the place individuals inside of building1. The fact that three floors are inside of building1 is defined by the *isInsideOf* property. Second, we build metric maps of each floor to divide and make leaf places. The leaf places' boundary is determined based on the metric map. The relationships between leaf places are defined by the *isConnectedTo* property. Similarly, the *isInsideOf* property between each floor and leaf place is defined.



Data Properties		Value
Symbolic model	Name	corridor34
	ID	34
Explicit model	boundary	[[37.6, 18.1, 6.0], [37.5, 15, 6.0], [59.7, 14.2, 6.0], [59.7, 17, 6.0]]
	coordinateFrame	Map
Implicit model	isLeafPlace	True
	complexity	0.0

[Data properties for an individual of Corridor]

Figure 2. An example of place individuals' asserted object and data properties.

After defining places and their relationships for entire environments, we add object individuals using the object detector, YOLOv5 [39]. The detector is trained using 26,078 labeled images that cover all environments. The training is conducted on 30 epochs with the initial learning rate of 5×10^{-4} and the rate reduction of 0.95 factor for plateau loss. The detected objects' 3D position in the camera frame is decided by their central position, calculated using the following equations:

$$X_c = \frac{1}{N} \sum_{i=1}^{N} \frac{d_i}{fx} \left(u_i - \frac{w-1}{2} \right) \tag{1}$$

$$Y_c = \frac{1}{N} \sum_{i=1}^{N} \frac{d_i}{fy} \left(v_i - \frac{h-1}{2} \right) \tag{2}$$

$$Z_c = \frac{1}{N} \sum_{i=1}^{N} d_i \tag{3}$$

where (X_c, Y_c, Z_c) is the object's position in the camera frame, N is the number of pixels inside of the bounding box, (u_i, v_i, d_i) are a pixel and its depth, (f_x, f_y) is the focal length of each coordinate, and (w, h) is the image size. The calculated position is transformed into the global coordinate and tracked using the Kalman Filter (KF) to cluster points for sequential scenes. We add the detected objects every time a robot passes through a leaf place. Each object and leaf place are linked with the *isInsideOf* property by checking the places' boundary and objects' position. Figure 3 visualizes the generated objects and places on building1 using their position and boundary in the explicit model.

Lastly, we define subclasses of *Robot*, as illustrated in Figure 4. The subclasses (*TricycleRobot*, *OmnidirectionalRobot*, and *DifferentialRobot*) are distinguished by a robot's kinematics. We create a robot individual used in our experiments per each subclass: *Kirobot*, *NRlab02*, *SmartCookie*, and *NRlab04*. Figure 4 describes the asserted properties of each robot (*canNotRotateInPlace* and *canWorkingAt*). The individual of *NRlab04* can work in building1 and outdoor1, but cannot rotate in elevator2 because of its kinematics. This knowledge can be defined by the following SWRL rule: *TricycleRobot(?r) -> canNotRotateInPlace(?r, elevator2)*. The individuals of *Kirobot* and *NRlab02* can work only indoors, but the individual of *SmartCookie* can work only outdoors.

The inference step uses the Pellet reasoner [40] to reason logical knowledge based on the A-box from the previous step. The reasoner finds all relationships between environmental elements (objects, places, and robots). For example, using the asserted relationships "corridor31 is inside of floor3", we can infer "corridor31 is inside of building1". These kinds of inferred knowledge are utilized in semantic navigation.

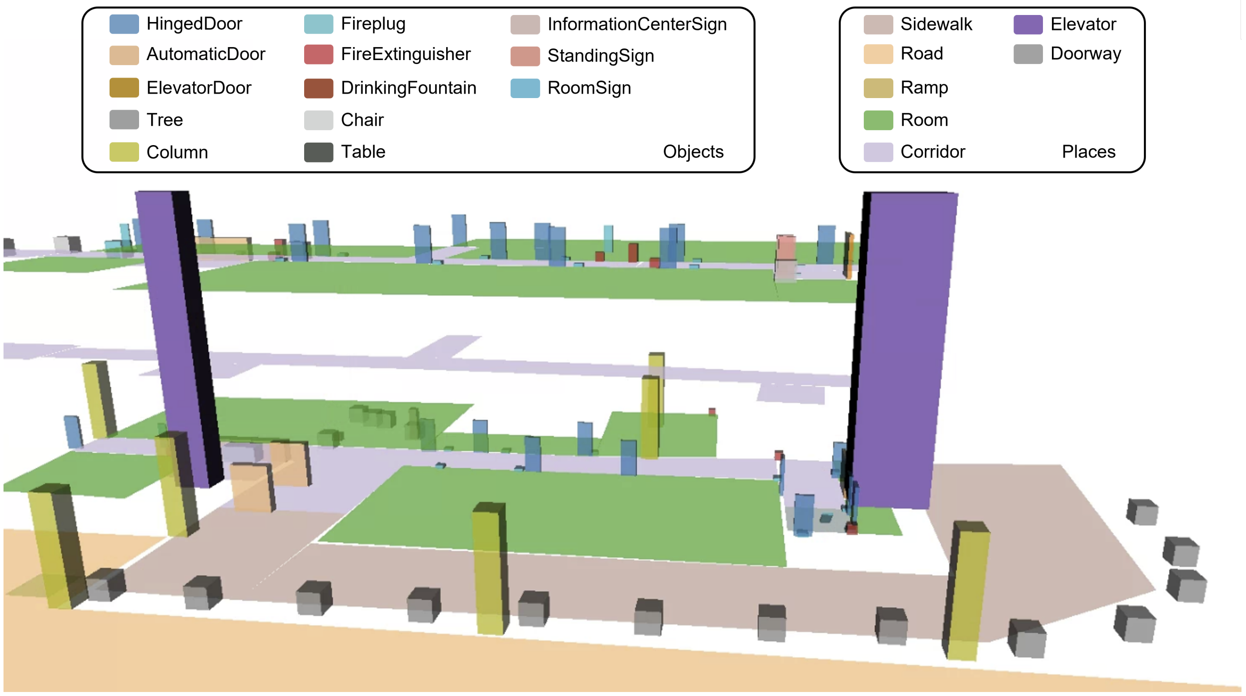


Figure 3. The explicit model-based visualization of objects and places on the generated semantic map.

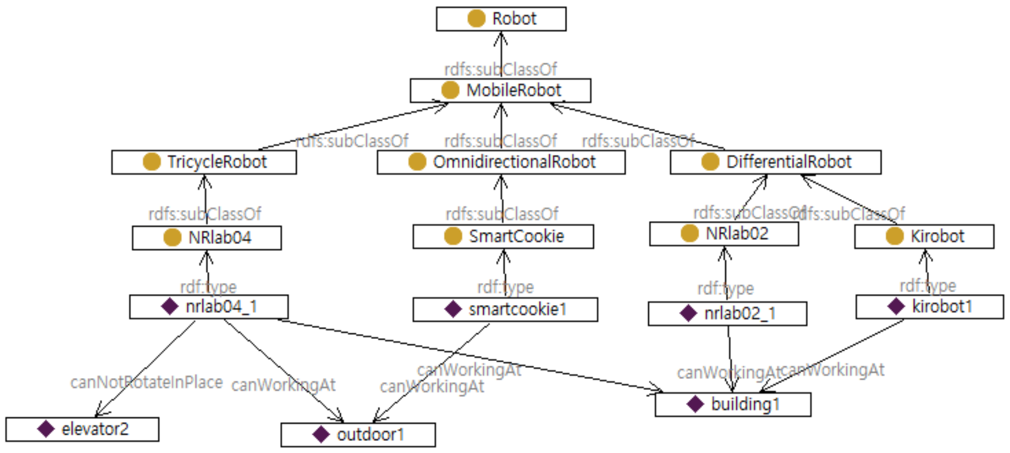


Figure 4. Subclasses of *Robot* and some of their individuals' asserted object properties.

4. Semantic Navigation

To make a robot complete an assigned task based on semantic knowledge, we presented the Semantic Autonomous Navigation (SAN) module and the Semantic Information Processing (SIP) module in [12]. The SAN module adopts the hierarchical planning scheme composed of task, behavior, and action planning. When a robot works, the SIP module semantically recognizes the environment to handle the robot's behaviors and actions. This work enhances the modules to take miscellaneous robots and tasks in indoor and outdoor fields. The following sections explain how the modules are implemented organically and make the framework sustainable in the selected environment, as shown in Figure 1.

4.1. Hierarchical Planning

The main components of hierarchical planning are task, behavior, and action planning. The task planner generates a behavior sequence using PDDL, the behavior planner decides an action sequence for each behavior by exploiting the behavior database, and the action planner makes direct robot control signals for each action. For each planner, we offer criteria to distinguish them. First, “task” is a set of goal states that a robot needs to accomplish. The state can be visiting somewhere, delivering coffee, or finding something. Second, “behavior” is a concept of acting to target goals with prior static knowledge. For example, when we go to a convenience store, we plan paths such as “going to an elevator” and “using the elevator”, then “going through some places” to get to the goal. These instances can be behaviors. Lastly, on the other hand, “action” needs to make real-time decisions by considering dynamic situations while executing.

For a clear understanding, we describe the hierarchical planning procedure for different robots using a delivery task example. When a robot receives a task, such as “Deliver a stuff to corridor32”, the task planner makes the goal state “(deliver robot_name stuff_name corridor32)” in the problem file of PDDL by parsing the task. The predicates of goal states are defined in the domain file of PDDL.

Next, the SAN module queries the robot’s on-demand semantic database to the SMF. Exploiting SPARQL, the SMF queries all semantic knowledge related to the robot to the database using the robot’s name. After acquiring the on-demand database, the task planner can realize the robot’s locations, workable and untraversable leaf places defined as *isLocatedAt*, *canWorkingAt*, and *canNotGoThrough*, respectively. For example, if the task is assigned to nrlab04_1, part of the obtained semantic relationships is as follows: “nrlab04_1 *isLocatedAt* corridor11”, “nrlab04_1 *canWorkingAt* sidewalk2”, “nrlab04_1 *canWorkingAt* corridor39”, and “nrlab04_1 *canNotGoThrough* doorway115”. These relationships are stored in the problem file as init states, and environmental individuals’ classes are saved in the object field of the domain file. Then, the task planner generates behavior sequences through the Partial Order Planning Forwards (POPF) planner [41] using the built problem file and pre-defined domain file.

As expected, assigning the same task to different robots causes different results, as illustrated in Figure 5. Recall the characteristics of two robots: *NRlab04* and *Kirobot*. Since *NRlab04* cannot directly go to floor3 indoors, the plan for *NRlab04* guides the robot outdoors via sidewalks, whereas *Kirobot* takes the elevator indoors. The task planner’s results for each robot are represented in Listing 1 and Listing 2, respectively. There are two kinds of “goto” behavior in the listings: goto_place and goto_place_through_doorway. goto_place, defined in Listing 3, handles general navigation behaviors between connected leaf places. Otherwise, goto_place_through_doorway specifies behaviors for two connected leaf places with a doorway. The parameter field of goto_place_through_doorway is defined as (?r - robot ?from ?to - leafplace ?dw - doorway) in the PDDL domain file. In line 8 of Listing 1, the robot moves between floors using move_floor_using_elevator behavior with the parameter field (?r - robot ?from ?to - leafplace ?dw_from ?dw_to - doorway ?ev - elevator).

Listing 1. A behavior sequence for *NRlab04* generated by the task planner.

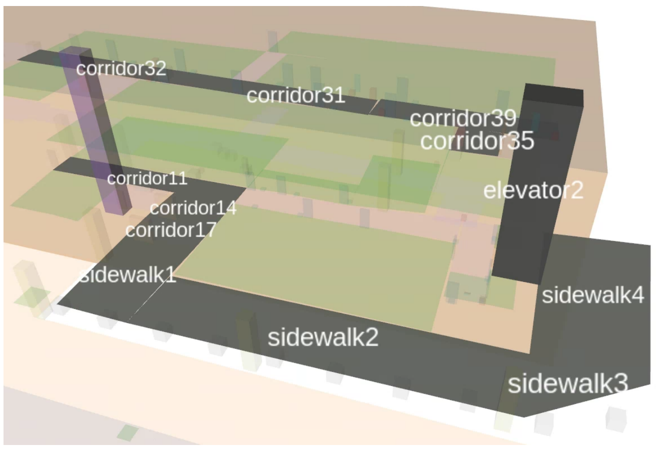
```
1  (get_stuff nrlab04_1 stuff1)
2  (goto_place nrlab04_1 corridor11 corridor14)
3  (goto_place_through_doorway nrlab04_1 corridor14 corridor17
      doorway114)
4  (goto_place_through_doorway nrlab04_1 corridor17 sidewalk1
      doorway121)
5  (goto_place nrlab04_1 sidewalk1 sidewalk2)
6  (goto_place nrlab04_1 sidewalk2 sidewalk3)
7  (goto_place nrlab04_1 sidewalk3 sidewalk4)
8  (move_floor_using_elevator nrlab04_1 sidewalk4 corridor35
      doorway113 doorway317 elevator2)
9  (goto_place nrlab04_1 corridor35 corridor39)
10 (goto_place nrlab04_1 corridor39 corridor31)
11 (goto_place nrlab04_1 corridor31 corridor32)
12 (give_stuff nrlab04_1 stuff1)
```

Listing 2. A behavior sequence for *Kirobot* generated by the task planner.

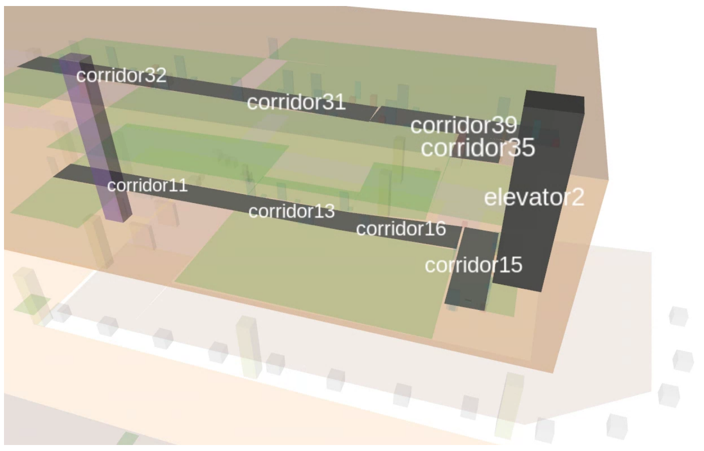
```
(get_stuff kirobot1 stuff1)
(goto_place kirobot1 corridor11 corridor13)
(goto_place kirobot1 corridor13 corridor16)
(goto_place kirobot1 corridor16 corridor15)
(move_floor_using_elevator kirobot1 corridor15 corridor35
    doorway115 doorway317 elevator2)
(goto_place kirobot1 corridor35 corridor39)
(goto_place kirobot1 corridor39 corridor31)
(goto_place kirobot1 corridor31 corridor32)
(give_stuff kirobot1 stuff1)
```

Listing 3. goto_place behavior.

```
(:durative-action goto_place
    :parameters (?r - robot ?from ?to - leafplace)
    :duration (= ?duration (/ (distance ?from ?to)(velocity ?r
        )))
    :condition (and
        (over all (canGoThrough ?r ?from))
        (over all (canGoThrough ?r ?to))
        (over all (canWorkingAt ?r ?from))
        (over all (canWorkingAt ?r ?to))
        (over all (isConnectedTo ?from ?to))
        (at start (isLocatedAt ?r ?from))
    )
    :effect (and
        (at start (not (isLocatedAt ?r ?from)))
        (at end (isLocatedAt ?r ?to))
    )
)
```



(**a**) A deliver task planning result for *NRlab04*.



(**b**) A deliver task planning result for *Kirobot*.

Figure 5. Different task planning results according to different classes of *Robot*.

After generating the behavior sequence, each behavior is dispatched to its behavior planner. The behavior planner can determine sophisticated action sequences using semantic knowledge of environmental elements. In the case of goto_place_through_doorway behavior, the planner obtains the individual name of the doorway that connects two leaf places. Then, the planner queries a door that is inside the doorway. If there is no door, the robot goes through via the doorway, but if an automatic door or a hinged door exists, the robot should check the door before passing. Therefore, the planner includes a checking action for this case.

Additionally, due to the robot's ability, we select a wireless elevator control solution for move_floor_using_elevator behavior. The following sequences can solve the behavior: "Find the elevator", "Wait until opened", "Call the elevator", "Enter the elevator", "Select the floor", "Wait until arrived", and "Leave the elevator". If the robot has a manipulator that can touch the elevator buttons, the "Call" and "Select" action is converted into "Push the button" actions.

Finally, the action planner directly controls the robot according to the given action sequence from the behavior planner. We separate actions into moving and others. For moving actions, the robot navigates to the goal by changing the navigation parameters considering the type of the goal. For example, the robot can speed up in wide places; otherwise, it can be careful in narrow or complex places such as doorways or crowded hallways. Other actions, such as handling the elevator, depend on domain-specific knowledge.

4.2. Plan Execution

The hierarchical planning scheme follows a top-down approach; conversely, the robot executes the plan bottom-up. After completing the action, the action planner returns the action result to the behavior planner. Likewise, obtaining all results of the action sequence, the behavior planner sends the behavior result to the task planner.

While the robot executes the plans, the SIP module semantically recognizes the working environment as the following sequences.

First, the object detector, explained in Section 3.2, finds objects and their positions to identify them. Based on the robot's location (place), represented as *isLocatedAt*, the SIP module queries objects using the *isInsideOf* relationship with the place name, and then compares the positions to determine the ID of the detected objects. The SIP module updates and adds objects whenever the robot finishes each behavior because, after finishing a behavior, the robot moves to another leaf place. At this time, the matched objects' position is updated, and others and their relationships are added to the semantic map in the same way as in semantic map generation in Section 3.2.

Second, the SIP module recognizes the states of objects and places defined in the TOSM. In our environments, the most critical information about objects that the SIP module needs to check is whether the door is open or not. The SIP module uses ultrasonic sensors for the undetectable glass doors (automaticdoor114 and automaticdoor121); for others, such as wood and steel doors, it uses images and point clouds. The sensors to use are determined based on the door's implicit knowledge (*material*), queried by the object detector's identification results. In addition, the SIP module inspects whether the place is blocked; this information is related to the *canNotGoThrough* property. These results make the action planner decide to go, wait, or fail.

When the action planner returns a fail result with reason, the behavior planner sends the information to the task planner. Then, after updating the on-demand database, the task planner re-plans to find other behavior sequences. We explain the re-planning scenario in detail in Section 5.

5. Experiment

In this section, we introduce experiments considering various conditions, such as robot kinematics, environment, and scenarios. The purpose of the experiments is to verify the scalability of the described semantic navigation framework.

5.1. Experimental Environment

We conducted the experiments in an environment containing multi-floor buildings and outdoors; the detailed description of the environment is given in Section 3. The OWL-based semantic database for the environment is generated using Protege [42]. Figure 6 illustrates the generated semantic map of the environment based on the explicit model. Table 4 represents the metric information of the semantic database used for the experiments. In the first column of the table, the number of defined OWL terminologies includes definitions of our previous work; we utilize 66 classes, 13 object properties, and 32 data properties. On the other hand, the second column only counts the individuals and asserted properties for the current environment; the database has 114 objects, 152 places, 4 robots, 432 asserted object properties, and 2096 asserted data properties.

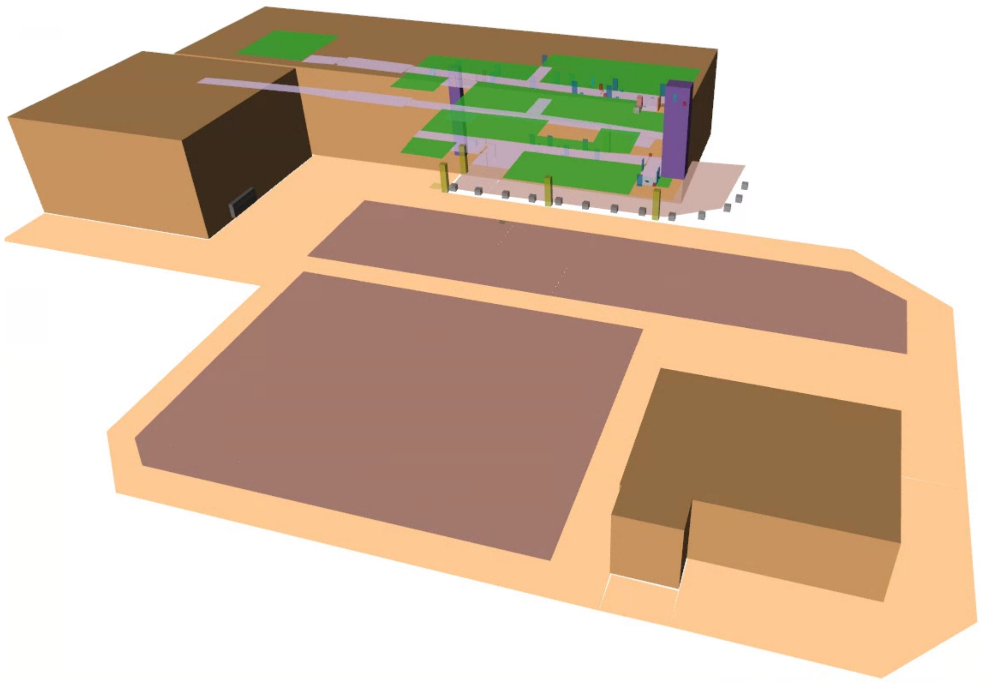


Figure 6. The explicit model-based visualization of the generated semantic map for the entire environment.

Table 4. Metric information for the semantic database.

Definitions			Individuals				
Class	**Object Property**	**Data Property**	**Object**	**Place**	**Robot**	**Object Property Assertion**	**Data Property Assertion**
63	13	32	114	152	4	432	2096

5.2. On-Demand Database

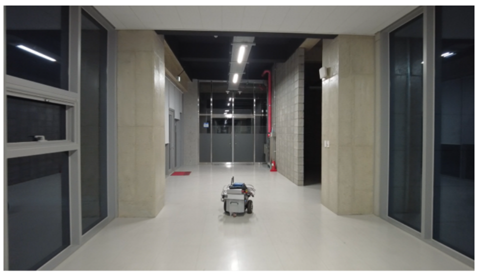
To validate the scalability according to robot types, we used four robots, illustrated in Figure 7. *NRlab04* and *SmartCookie* can work in outdoor1 because they are waterproof; otherwise, *Kirobot and NRLab02* can work only indoors, in building1. Moreover, *SmartCookie* cannot work indoors due to its size. These facts are stored as *canWorkingAt* in the semantic database.

Since the robots have different characteristics, they do not need the entire semantic database for their tasks. Therefore, we made an on-demand semantic database for each robot to obtain efficiency by considering their features. Using the Pellet reasoner, first, we inferred relationships based on the asserted semantic knowledge and the SWRL rules. The reasoning step took 1346 ms. Second, we used SPARQL, the query language, to generate their on-demand database. We adopted Owlready [43], which utilizes RDF quadstore in an optimized database (SQLite3) to handle the OWL-based database.

The results for their on-demand database are described in Table 5. In the relationships column, *canWorkingAt* (a) and *canWorkingAt* (i) mean asserted knowledge and inferred knowledge, respectively, from Rule 1, which finds the leaf places where the robot can work. The number of *canNotGoThrough* can be changed according to *assignedTaskAt*. The table assumes that the robots are assigned when the doorway is available. The number of relationships and individuals is queried based on their workable places. The total number of relationships contains relationships between queried individuals such as *isInsideOf*, *isConnectedTo*, and *isLocatedAt*. We describe assigned tasks for each robot in the next section.

(**a**) *NRLab04*.

(**b**) *Kirobot*.


(**c**) *NRLab02*.

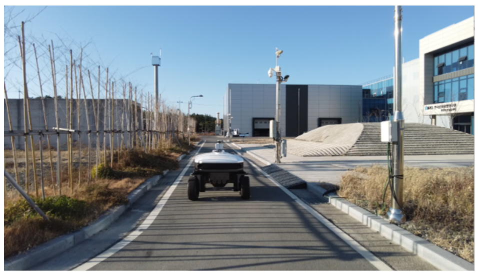

(**d**) *SmartCookie*.

Figure 7. *Robot*'s individuals used for the experiment. (**a**) is *TricycleRobot*, (**b**,**c**) are *DifferentialRobot*, and (**d**) is *OmnidirectionalRobot*.

Table 5. On-demand semantic database for each subclass of *Robot*.

Classes	Relationships				Individuals		Assigned Tasks
	***canWorkingAt* (a)**	***canWorkingAt* (i)**	***canNotGoThrough***	**Total**	**Objects**	**Places**	
NRLab04	building1, outdoor1	138	2	1924	114	152	Delivery
Kirobot	building1	95	1	1572	94	110	Guidance
NRLab02	building1	95	2	1573	94	110	Surveillance
SmartCookie	outdoor1	43	0	255	20	42	Patrol

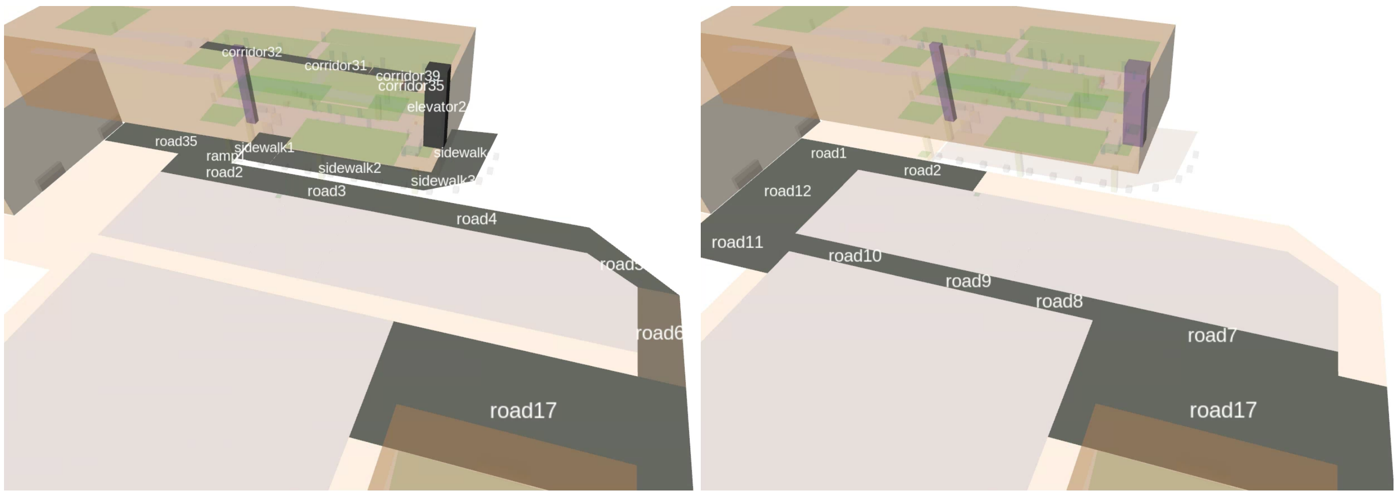
5.3. Scenario

We demonstrate the semantic navigation framework by showing various scenarios: delivery, guidance, surveillance, and patrol. Each task is assigned as shown in Table 5. When the robot receives the task, the SAN module generates the problem file using knowledge in its on-demand database; the initial location of the robot is defined as the *isLocatedAt* property. We perform the scenario by changing the robot's starting and goal place to evaluate the framework. The goal can be set using our user interface, which has selectable lists for each task.

5.3.1. Delivery

The first scenario, delivery, was assigned to *NRLab04*, which can work for entire environments. As explained in Section 4.1, the delivery task can be defined as the predicate (deliver_complete ?r - robot ?s - stuff ?p - place), which becomes the goal state of the problem file. To introduce a building-to-building navigation scenario, we set the robot's location to corridor32 and the goal to road17 connected to building2. In this scenario, querying for the on-demand database took 348 ms, generating the problem file took 622 ms, and task planning took 49 ms. The behavior sequence has 16 behaviors similar to Listing 1, and the total sequence duration time used as the planner's optimization cost was 563 s.

Using the scenario, we added an unexpected situation in which we blocked the place when the robot was working. To create the re-planning signal, we barricaded road3 using large boxes. In this situation, the re-planning proceeds as follows. First, the SIP module decides that the robot cannot go through road3. Using the result, the action planner for the goto_place behavior sends feedback to the task planner through the behavior planner. Second, the task planner updates the on-demand database by removing the *isConnectedTo* property between the current leaf place and road3. Finally, the task planner re-plans based on the updated knowledge until all sequences fail. The first generated and re-planned behavior sequences are illustrated in Figure 8.



(**a**) The first generated behavior sequence.

(**b**) The re-planned behavior sequence.

Figure 8. The behavior sequence generated by the task planner. (**a**) The plan result from corridor32 to road17. (**b**) The re-plan result from road2 to road17.

Furthermore, we included a predicate for the object's location to expand the scenario. Using the predicate, the robot can also deliver an object that is not in the robot's starting place. In this case, the user should select the goal and the object's location.

5.3.2. Guidance

The guidance scenario is simpler than the deliver scenario. The task assigned to *Kirobot* requires only behaviors related to moving: goto_place, move_floor_using_elevator, and goto_place_through_doorway. Moreover, the goal of the task is converted using the *isLocatedAt* property. For the metric comparison, we set the goal to corridor11 when the robot was located at corridor32. This case took 280 ms to query for the on-demand database, 430 ms to generate the problem file, and 36 ms for task planning. The behavior sequence had 7 behaviors, and the total time was 172 s.

5.3.3. Surveillance

We assigned a surveillance scenario that can have multiple goals to inspect sequentially to *NRLab02*. The task can have multiple goals to inspect sequentially. To complete the task, we define an inspection behavior with the parameters (?r - robot ?p - place) that can be generated using the task goal. For the behavior, we implemented an action that detects people while rotating. Unlike other scenarios, we set the goals as "inspect corridor38 and 39, then come your initial place" without moving floors. The result of the task planner contains the inspection behavior between moving behaviors (goto_place). The planner took 285 ms for querying, 442 ms for the problem file, and 42 ms for task planning. The result has 12 behaviors and a total of 259 s.

5.3.4. Patrol

The final scenario is a patrol task assigned to *SmartCookie* for outdoors. The goal of the task is to check all leaf places inside of outdoor1. We devise a checking behavior for the task with the predicate (check ?r - robot ?p - place). Based on the predicate, the goals are represented as a set of predicates with queried individuals. We set the robot's initial location as road1 and included (smartcookie1 *isLocatedAt* road1) in the goal state for return. The planner took 122 ms and 184 ms for querying and the problem file. Because the outdoors has roads connecting many roads at once, the patrol task can be performed in various behavior orders. Therefore, the task planner took 1120 ms to find the best sequence with 53 behaviors and a total of 626 s.

5.3.5. Comparison

We compare the experiments using the metrics of four scenarios (delivery, guidance, surveillance, and patrol) in Figure 9.

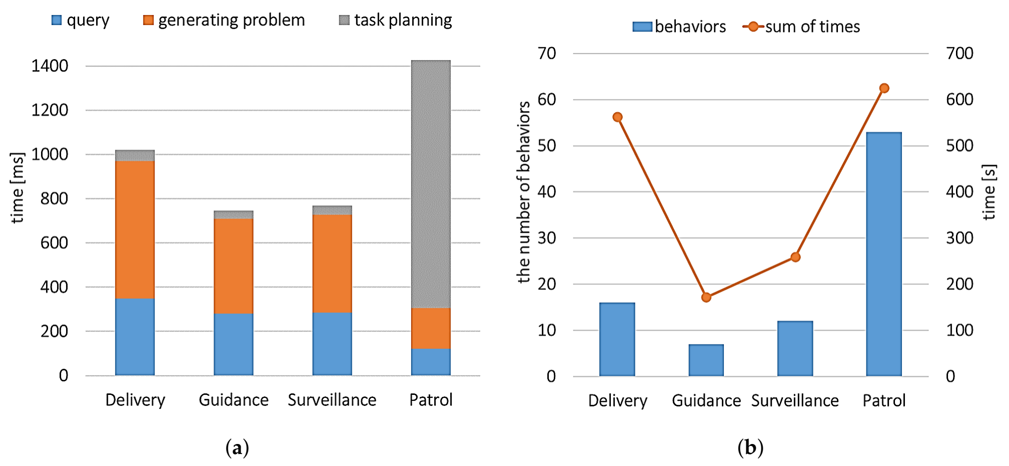


Figure 9. Comparison of four scenarios: delivery, guidance, surveillance, and patrol. (**a**) Query time for the on-demand database (blue), generating the problem file time (orange), and task planning time (gray). (**b**) The number of behaviors (blue) and the sum of times for the sequence (orange).

Figure 9a illustrates each scenario's time spent querying the semantic database (blue), generating the problem file (orange), and task planning (gray). Recall the on-demand database for each robot, described in Table 5. Because building1 had more individuals and relationships than outdoor1, the patrol scenario took less time when querying and generating the file than other scenarios. This means that the times for querying and generating are proportional to the size of the on-demand database. On the other hand, since the patrol task could be solved using several sequences, the task planner took the longest to find the best sequence; the total times spent were 1019 ms, 746 ms, 769 ms, and 1426 ms for the order of scenarios.

Figure 9b compares the results of the task planner for each scenario. The behavior sequence for the delivery and guidance task consists of moving and using elevator behaviors; the surveillance has moving and inspecting behaviors; the patrol has moving behaviors only. We specified the duration of goto_place by dividing the distance between places by the robot's velocity; we set the duration of inspect_place to 60 s. In the case of move_floor_using_elevator, since we cannot predict the duration, we assumed the duration as 60 s. Accordingly, the total times for the behavior sequences were 563 s, 172 s, 259 s, and 626 s. The time of the patrol task was relatively short compared to others because of the robot's speed.

We evaluated the framework's reliability using the delivery scenario by changing the robot's initial and goal place. The task success was determined by whether the robot arrived at the destination accurately. We achieved a 96.7% success rate in a total of 30 experiments.

Most of the experiments were successful, but one experiment failed when communication to open the door was lost inside the elevator.

5.4. Changing the Environmental Domain

We validated the scalability and effectiveness for various environments by applying the proposed framework to the corporate collaboration center at Sungkyunkwan University (SKKU). The building, consisting of seven floors, has a cafe on the first floor; three elevators connect each floor, and outdoor roads connect the first floor. We selected the first and the seventh floor for the coffee delivery scenario. The semantic database for the environment has 20 objects, 62 places, 1 robot, 826 asserted object properties, and 531 asserted data properties. Figure 10 visualizes the environment's semantic database.

We set the robot's start and goal as corridor717 on the seventh floor. The robot must stop by cafe1 on the first floor to bring a coffee to complete the task. Therefore, the robot should take the elevator twice. In this scenario, querying for the on-demand database took 232 ms, generating the problem file took 368 ms, and task planning took 73 ms. The behavior sequence had 14 behaviors, and the total sequence duration time was 412 s.

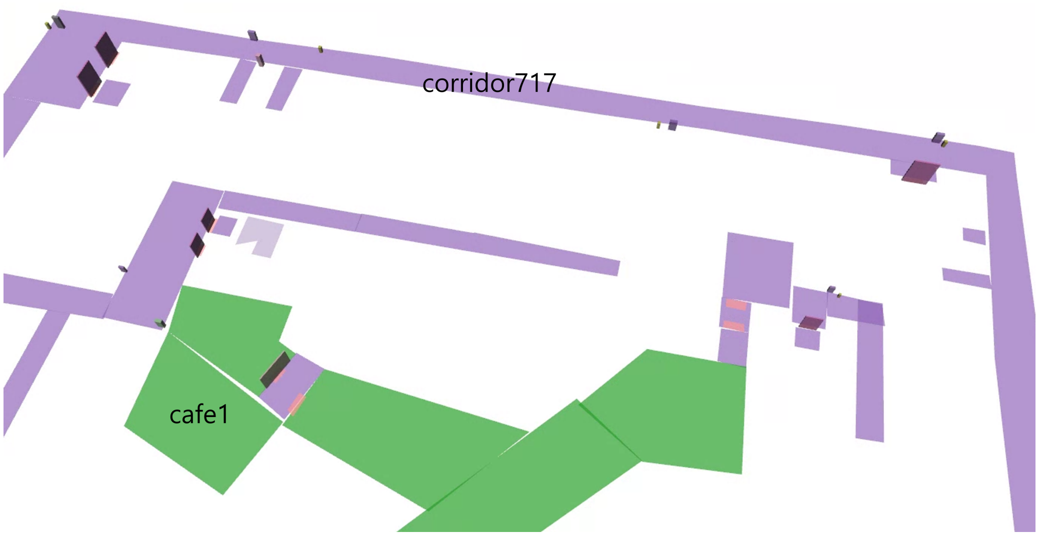


Figure 10. The explicit model-based visualization of the generated semantic map for the corporate collaboration center at Sungkyunkwan University (SKKU).

6. Discussion

The development of semantic navigation frameworks for robots has been advanced in recent years. The semantic knowledge enables robots to become intelligent, to realize the world and act in a human way. However, their abilities can be further improved in various applications. Therefore, we suggest several future research directions with associated open issues.

Executing complicated tasks. Nowadays, robots are applied for many tasks, such as surveillance, guidance, delivery, and disinfection. However, these tasks consist of behaviors that can be implemented without complex behavior knowledge. The final goal of using semantic knowledge is beyond completing simple repetitive works. For complicated tasks, knowledge of each task should be defined and utilized. Beetz et al. presented the KNOWROB 2.0 framework [24] for complex manipulation tasks (e.g., making pizza and setting tables). To increase the capabilities of robots' manipulation skills, they defined the task knowledge about them. Consequently, designing more sophisticated task models will enable the robots to execute various tasks, such as doing the laundry.

Improving recognition skills. State-of-the-art methods to recognize environments have achieved remarkable performance in many domains. However, the current recognition methods using only sensory inputs are insufficient for complex decisions in numerous situations. For example, when a robot is guarding a particular area, there are countermeasures for each case, such as specific domain knowledge. Therefore, we believe that improving recognition skills based on semantic knowledge in various situations is needed for reasonable determinations.

Standardization of semantic knowledge. There are several semantic knowledge representation approaches in [1]. Each approach defines semantic knowledge for its specific domain. However, to share and accumulate semantic knowledge, semantic knowledge representations should be standardized. Schlenoff et al. discussed the IEEE-RAS working group, entitled Ontologies for Robotics and Automation (ORA WG), that developed a standard ontology for knowledge representation [44]. They have tried to unify the format of entire knowledge representation terminologies. Their works proposed a Core Ontology for Robotics and Automation (CORA), described in [45–48]. Furthermore, the RoboEarth project [49] was aimed at presenting a system for sharing knowledge between robots. The project was the first implementation of a World Wide Web for robots. They demonstrated that sharing knowledge between robots can accelerate the speed of learning. Further research on the standardization of semantic knowledge will offer a basis for robot knowledge.

7. Conclusions

In this work, we presented the TOSM-based semantic navigation framework. We expanded the previous framework to accommodate robots with varying kinematics and tasks in indoor and outdoor environments by defining semantic knowledge based on the TOSM, OWL, and PDDL. We also generated a semantic map including asserted and inferred semantic knowledge using SWRL and the reasoner. The on-demand database generated by considering each robot's class allowed the robots to execute tasks according to their characteristics. The hierarchical planning and re-planning strategy made the framework reliable even in unexpected situations. Moreover, we maintained the semantic map by adding and updating processes whenever the robots worked. The experiments on four scenarios (deliver, guidance, surveillance, and patrol) and four different robots demonstrated that our framework could be used for various robots, environments, and scenarios. Our future work will focus on handling the issues described in the discussions and extending the framework to a multi-robot system.

Author Contributions: Conceptualization, S.J. and T.K.; methodology, S.J., S.B., T.U. and J.M.; software, S.J., S.B. and J.C.; validation, S.J., S.B., J.C., S.L., S.Y., T.U. and J.M.; formal analysis, S.J., H.P. and T.K.; investigation, S.J.; resources, S.J., S.L., S.Y., T.U. and J.M.; data curation, S.J.; writing—original draft preparation, S.J.; writing—review and editing, S.J.; visualization, S.J.; supervision, H.P. and T.K.; project administration, T.K.; funding acquisition, T.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Korea Evaluation Institute of Industrial Technology (KEIT), funded by the Ministry of Trade, Industry & Energy (MOTIE) (No. 1415178566 and 1415172051).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Manzoor, S.; Rocha, Y.G.; Joo, S.H.; Bae, S.H.; Kim, E.J.; Joo, K.J.; Kuc, T.Y. Ontology-Based Knowledge Representation in Robotic Systems: A Survey Oriented toward Applications. *Appl. Sci.* **2021**, *11*, 4324. [CrossRef]
2. Crespo, J.; Castillo, J.C.; Mozos, O.M.; Barber, R. Semantic Information for Robot Navigation: A Survey. *Appl. Sci.* **2020**, *10*, 497. [CrossRef]
3. Aulinas, J.; Petillot, Y.R.; Salvi, J.; Lladó, X. The slam problem: A survey. *CCIA* **2008**, *184*, 363–371.

4. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*; The MIT Press: Cambridge, MA, USA, 2005.
5. Kostavelis, I.; Gasteratos, A. Semantic mapping for mobile robotics tasks: A survey. *Robot. Auton. Syst.* **2015**, *66*, 86–103. [CrossRef]
6. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [CrossRef]
7. Dubé, R.; Cramariuc, A.; Dugas, D.; Sommer, H.; Dymczyk, M.; Nieto, J.; Siegwart, R.; Cadena, C. SegMap: Segment-based mapping and localization using data-driven descriptors. *Int. J. Robot. Res.* **2020**, *39*, 339–355. [CrossRef]
8. Chen, X.; Milioto, A.; Palazzolo, E.; Giguère, P.; Behley, J.; Stachniss, C. Suma++: Efficient lidar-based semantic slam. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macu, China, 3–8 November 2019; pp. 4530–4537.
9. Rosinol, A.; Abate, M.; Chang, Y.; Carlone, L. Kimera: An open-source library for real-time metric-semantic localization and mapping. *arXiv* **2019**, arXiv:1910.02490.
10. Rosinol, A.; Gupta, A.; Abate, M.; Shi, J.; Carlone, L. 3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans. *arXiv* **2020**, arXiv:2002.06289.
11. Li, J.; Zhang, X.; Li, J.; Liu, Y.; Wang, J. Building and optimization of 3D semantic map based on Lidar and camera fusion. *Neurocomputing* **2020**, *409*, 394–407. [CrossRef]
12. Joo, S.H.; Manzoor, S.; Rocha, Y.G.; Bae, S.H.; Lee, K.H.; Kuc, T.Y.; Kim, M. Autonomous Navigation Framework for Intelligent Robots Based on a Semantic Environment Modeling. *Appl. Sci.* **2020**, *10*, 3219. [CrossRef]
13. Zander, S.; Ahmed, N.; Frank, M. A Survey about the Usage of Semantic Technologies for the Description of Robotic Components and Capabilities. In Proceedings of the SAMI@ iKNOW, Graz, Austria, 18–19 October 2016.
14. Sun, X.; Zhang, Y. A review of domain knowledge representation for robot task planning. In Proceedings of the 2019 4th International Conference on Mathematics and Artificial Intelligence, Chegndu, China, 12–15 April 2019; pp. 176–183.
15. Galindo, C.; Fernández-Madrigal, J.A.; González, J.; Saffiotti, A. Robot task planning using semantic maps. *Robot. Auton. Syst.* **2008**, *56*, 955–966. [CrossRef]
16. Deeken, H.; Wiemann, T.; Hertzberg, J. Grounding semantic maps in spatial databases. *Robot. Auton. Syst.* **2018**, *105*, 146–165. [CrossRef]
17. Dhouib, S.; Du Lac, N.; Farges, J.L.; Gerard, S.; Hemaissia-Jeannin, M.; Lahera-Perez, J.; Millet, S.; Patin, B.; Stinckwich, S. Control architecture concepts and properties of an ontology devoted to exchanges in mobile robotics. In Proceedings of the 6th National Conference on Control Architectures of Robots, Grenoble, France, 24–25 May 2011; p. 24.
18. Gonçalves, P.J.; Torres, P.M. Knowledge representation applied to robotic orthopedic surgery. *Robot. Comput. Integr. Manuf.* **2015**, *33*, 90–99. [CrossRef]
19. Chang, D.S.; Cho, G.H.; Choi, Y.S. Ontology-based knowledge model for human-robot interactive services. In Proceedings of the 35th Annual ACM Symposium on Applied Computing, Brno, Czech Republic, 30 March–3 April 2020; pp. 2029–2038.
20. Sadik, A.R.; Urban, B. An ontology-based approach to enable knowledge representation and reasoning in worker–cobot agile manufacturing. *Future Internet* **2017**, *9*, 90. [CrossRef]
21. Kootbally, Z.; Kramer, T.R.; Schlenoff, C.; Gupta, S.K. Implementation of an ontology-based approach to enable agility in kit building applications. *Int. J. Semant. Comput.* **2018**, *12*, 5–24. [CrossRef]
22. Lim, G.H.; Suh, I.H.; Suh, H. Ontology-based unified robot knowledge for service robots in indoor environments. *IEEE Trans. Syst. Man Cybern. Part A Syst. Humans* **2010**, *41*, 492–509. [CrossRef]
23. Pangercic, D.; Pitzer, B.; Tenorth, M.; Beetz, M. Semantic object maps for robotic housework-representation, acquisition and use. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 4644–4651.
24. Beetz, M.; Beßler, D.; Haidu, A.; Pomarlan, M.; Bozcuoğlu, A.K.; Bartels, G. Know rob 2.0—A 2nd generation knowledge processing framework for cognition-enabled robotic agents. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 512–519.
25. Sabri, L.; Bouznad, S.; Rama Fiorini, S.; Chibani, A.; Prestes, E.; Amirat, Y. An integrated semantic framework for designing context-aware internet of robotic things systems. *Integr. Comput. Aided Eng.* **2018**, *25*, 137–156. [CrossRef]
26. Sun, X.; Zhang, Y.; Chen, J. High-level smart decision making of a robot based on ontology in a search and rescue scenario. *Future Internet* **2019**, *11*, 230. [CrossRef]
27. Eshghi, K. Abductive Planning with Event Calculus. In Proceedings of the Fifth International Conference on Logic Programming ICLP/SLP, Seattle, WA, USA, 15–19 August 1988; MIT Press: Cambridge, MA, USA, 1998; pp. 562–579.
28. Tenorth, M.; Beetz, M. KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *Int. J. Robot. Res.* **2013**, *32*, 566–590. [CrossRef]
29. Baader, F.; Calvanese, D.; McGuinness, D.; Patel-Schneider, P.; Nardi, D.; Patel-Schneider, P.F. *The Description Logic Handbook: Theory, Implementation and Applications*; Cambridge University Press: Cambridge, MA, USA, 2003.
30. Bozsak, E.; Ehrig, M.; Handschuh, S.; Hotho, A.; Maedche, A.; Motik, B.; Oberle, D.; Schmitz, C.; Staab, S.; Stojanovic, L.; et al. KAON—Towards a large scale Semantic Web. In *International Conference on Electronic Commerce and Web Technologies*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 304–313.

31. Tenorth, M.; Kunze, L.; Jain, D.; Beetz, M. Knowrob-map-knowledge-linked semantic object maps. In Proceedings of the 2010 10th IEEE-RAS International Conference on Humanoid Robots, Nashville, TN, USA, 6–8 December 2010; pp. 430–435.
32. Rusu, R.B.; Marton, Z.C.; Blodow, N.; Dolha, M.; Beetz, M. Towards 3D point cloud based object maps for household environments. *Robot. Auton. Syst.* **2008**, *56*, 927–941. [CrossRef]
33. Tenorth, M.; Beetz, M. KnowRob—Knowledge processing for autonomous personal robots. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 10–15 October 2009; pp. 4261–4266.
34. Wielemaker, J.; Schrijvers, T.; Triska, M.; Lager, T. Swi-prolog. *Theory Pract. Log. Program.* **2012**, *12*, 67–96. [CrossRef]
35. Kunze, L.; Beetz, M.; Saito, M.; Azuma, H.; Okada, K.; Inaba, M. Searching objects in large-scale indoor environments: A decision-theoretic approach. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 4385–4390.
36. Deeken, H.; Wiemann, T.; Lingemann, K.; Hertzberg, J. SEMAP—A semantic environment mapping framework. In Proceedings of the 2015 European Conference on Mobile Robots (ECMR), Lincoln, UK, 2–4 September 2015; pp. 1–6.
37. Bechhofer, S.; Van Harmelen, F.; Hendler, J.; Horrocks, I.; McGuinness, D.L.; Patel-Schneider, P.F.; Stein, L.A. OWL web ontology language reference. *W3C Recomm.* **2004**, *10*, 1–53. .
38. Horrocks, I.; Patel-Schneider, P.F.; Boley, H.; Tabet, S.; Grosof, B.; Dean, M. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Memb. Submiss.* **2004**, *21*, 1–31.
39. Jocher, G.; Chaurasia, A.; Stoken, A.; Borovec, J.; NanoCode012; Kwon, Y.; Xie, T.; Fang, J.; imyhxy; Michael, K.; et al. ultralytics/yolov5: v6.1—TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference. 2022. Available online: https://zenodo.org/record/6222936#.YunrIHkRWU (accessed on 7 July 2022). [CrossRef]
40. Sirin, E.; Parsia, B.; Grau, B.C.; Kalyanpur, A.; Katz, Y. Pellet: A practical owl-dl reasoner. *J. Web Semant.* **2007**, *5*, 51–53. [CrossRef]
41. Coles, A.; Coles, A.; Fox, M.; Long, D. Forward-chaining partial-order planning. In Proceedings of the International Conference on Automated Planning and Scheduling, Toronto, ON, Canada, 12–16 May 2010; Volume 20, pp. 42–49.
42. Musen, M.A. The protégé project: A look back and a look forward. *AI Matters* **2015**, *1*, 4–12. [CrossRef] [PubMed]
43. Lamy, J.B. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artif. Intell. Med.* **2017**, *80*, 11–28. [CrossRef]
44. Schlenoff, C.; Prestes, E.; Madhavan, R.; Goncalves, P.; Li, H.; Balakirsky, S.; Kramer, T.; Migueláñez, E. An IEEE standard Ontology for Robotics and Automation. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 1337–1342. [CrossRef]
45. Carbonera, J.L.; Fiorini, S.R.; Prestes, E.; Jorge, V.A.; Abel, M.; Madhavan, R.; Locoro, A.; Gonçalves, P.; Haidegger, T.; Barreto, M.E.; et al. Defining positioning in a core ontology for robotics. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 1867–1872.
46. Prestes, E.; Carbonera, J.L.; Fiorini, S.R.; Jorge, V.A.; Abel, M.; Madhavan, R.; Locoro, A.; Goncalves, P.; Barreto, M.E.; Habib, M.; et al. Towards a core ontology for robotics and automation. *Robot. Auton. Syst.* **2013**, *61*, 1193–1204. [CrossRef]
47. Fiorini, S.R.; Carbonera, J.L.; Gonçalves, P.; Jorge, V.A.; Rey, V.F.; Haidegger, T.; Abel, M.; Redfield, S.A.; Balakirsky, S.; Ragavan, V.; et al. Extensions to the core ontology for robotics and automation. *Robot. Comput. Integr. Manuf.* **2015**, *33*, 3–11. [CrossRef]
48. Jorge, V.A.; Rey, V.F.; Maffei, R.; Fiorini, S.R.; Carbonera, J.L.; Branchi, F.; Meireles, J.P.; Franco, G.S.; Farina, F.; Da Silva, T.S.; et al. Exploring the IEEE ontology for robotics and automation for heterogeneous agent interaction. *Robot. Comput. Integr. Manuf.* **2015**, *33*, 12–20. [CrossRef]
49. Waibel, M.; Beetz, M.; Civera, J.; d'Andrea, R.; Elfring, J.; Galvez-Lopez, D.; Häussermann, K.; Janssen, R.; Montiel, J.; Perzylo, A.; et al. Roboearth. *IEEE Robot. Autom. Mag.* **2011**, *18*, 69–82. [CrossRef]