applied sciences

MDPI

Article

Autonomous Navigation Framework for Intelligent Robots Based on a Semantic Environment Modeling

Sung-Hyeon Joo [1,†], Sumaira Manzoor [1,†], Yuri Goncalves Rocha [1], Sang-Hyeon Bae [1], Kwang-Hee Lee [2], Tae-Yong Kuc [1,*] and Minsung Kim [3]

1 Department of Electrical and Computer Engineering, College of Information and Communication Engineering, Sungkyunkwan University, Suwon 16419, Korea; sh.joo@skku.edu (S.-H.J.); sumaira11@skku.edu (S.M.); yurirocha@skku.edu (Y.G.R.); shbae.skku@skku.edu (S.-H.B.)

2 Robot R&D Group, Korea Institute of Industrial Technology (KITECH), Ansan 15588, Korea; leekh@kitech.re.kr

3 Department of Electronic and Electrical Engineering, Dongguk University-Seoul Campus, Seoul 04620, Korea; mkim@dongguk.edu

* Correspondence: tykuc@skku.edu

† These authors contributed equally to this work.

Received: 17 March 2020; Accepted: 27 April 2020; Published: 5 May 2020

Abstract: Humans have an innate ability of environment modeling, perception, and planning while simultaneously performing tasks. However, it is still a challenging problem in the study of robotic cognition. We address this issue by proposing a neuro-inspired cognitive navigation framework, which is composed of three major components: semantic modeling framework (SMF), semantic information processing (SIP) module, and semantic autonomous navigation (SAN) module to enable the robot to perform cognitive tasks. The SMF creates an environment database using Triplet Ontological Semantic Model (TOSM) and builds semantic models of the environment. The environment maps from these semantic models are generated in an on-demand database and downloaded in SIP and SAN modules when required to by the robot. The SIP module contains active environment perception components for recognition and localization. It also feeds relevant perception information to behavior planner for safely performing the task. The SAN module uses a behavior planner that is connected with a knowledge base and behavior database for querying during action planning and execution. The main contributions of our work are the development of the TOSM, integration of SMF, SIP, and SAN modules in one single framework, and interaction between these components based on the findings of cognitive science. We deploy our cognitive navigation framework on a mobile robot platform, considering implicit and explicit constraints for autonomous robot navigation in a real-world environment. The robotic experiments demonstrate the validity of our proposed framework.

Keywords: intelligent robot; autonomous navigation framework; triplet ontological semantic model; environment modeling; on-demand database; knowledge-based recognition; hierarchical planning

1. Introduction

Environment modeling, recognition, and planning are fundamental components for intelligent systems, such as mobile robots, that enable them to understand and perceive the complex environment in the same way a human does and perform the task reliably. When mobile robots are deployed in the real-world, numerous challenges arise: acquiring spatial models of the continuously changing environment that leads to the problem of robotic mapping, a radically changed scene appearance that is one of the significant factors towards the visual recognition failure [1], and the dynamic nature of

the operating environment that is a critical challenge for robot planning [2]. Many approaches have been proposed to handle these issues [3]. However, among them, a growing world-wide trend is introducing semantic information into robotic systems using ontology [4], which endows the mobile robot with human-like mapping [5], recognition [6] and planning [7] capabilities. Inspired by these recent advances, our framework exploits the semantic knowledge by defining a Triplet Ontological Semantic Model (TOSM).

Recent developments in robotics have heightened the need for semantic information processing techniques to design object and place recognition systems based on the findings of cognitive science. Semantic recognition systems based on Convolutional Neural Network (CNN) models have made groundbreaking advances to tackle scene understanding tasks with an emphasis on object and place recognition [8]. The complex representation of visual information and their processing by biologically inspired design of CNN model has shown unprecedented advancements in visual recognition. The layered structure in the CNN model is more reliable to represent the lower area of the human visual cortex [9]. Motivated by these developments, we combine the strengths of a deep learning-based CNN model with an on-demand database for semantic object-based place recognition and robot localization.

When planning a task, an agent requires a description of the environment and of itself, containing the possible states, actions, and events. Classical planners, such as Planning Domain Definition Language (PDDL) planners [10], make use of a set of predicates and actions to generate a valid action sequence. Those planners have been vastly used by the research community and integrated into powerful frameworks [11]. However, they tend not to perform well on unpredictable and dynamic environments, usually described by Partially Observed Markov Decision Processes (POMDP). Aiming to solve POMDPs on a variety of domains, reinforcement learning approaches try to use a large amount of data to approximate a policy capable of obtaining an optimal action based solely on the current state. Those approaches have shown promising results, but they still underperform when dealing with long-term scenarios. Our framework tries to overcome those shortcomings by combining classical PDDL planners, sampling-based planners, and reinforcement learning algorithms into a hierarchical scheme.

Autonomous navigation frameworks based on knowledge base have become essential for complex applications of robotic systems in real-world environments. In this direction, the Ontology-Based Unified Robot Knowledge (OUR-K) framework [12] emphasized at robot intelligence to perform navigation, recognition, planning, and manipulation tasks using knowledge description (for contexts, features, objects, spaces, and actions) and knowledge association for establishing relationships between descriptions by supporting inference. The OUR-K framework considered small static indoor environments such as an office room; however, our framework concentrates on robot navigation in large dynamic environments using the TOSM-based architecture. The RoboEarth framework [13] designed a knowledge-based system to generate, share, and reuse data using a cloud database. In contrast, our framework uses the on-demand database to ensure real-time capabilities. Furthermore, it integrates semantic modeling, information processing, and autonomous navigation modules to perform a task. Our prior [14] work has persuaded us to introduce the navigation framework in this study.

Our framework makes major contributions to the semantic robotic research area as follows:

- Builds a knowledge base comprising of the advanced ontological environment model.
- Creates an on-demand database considering the robot's capabilities and missions.
- Combines state-of-the-art deep learning-based CNN recognition model with an on-demand database.
- Defines a multi-layer hierarchical planning scheme that integrates semantic knowledge, a classical planner, and reinforcement learning to enable multi-story building navigation.

The rest of our paper is divided into five sections as follows: Section 2, gives a brief overview of related work in other frameworks, environment modeling, recognition, and planning. Section 3, explains the overall framework and describes each component in detail. Section 4,

illustrates experiments and discusses the results. Finally, conclusions and future directions follow in Section 5.

2. Related Work

In this section, we first discuss the related work for knowledge-based navigation frameworks (Section 2.1). After that, we review the literature that deals with the individual modules of our autonomous navigation framework distinguishing into environment modeling for robots (Section 2.2), recognition (Section 2.3), and planning (Section 2.4).

2.1. Knowledge-Based Navigation Frameworks

Recently, there is a growing interest in integrating the knowledge-based approaches into robotic systems, which use ontologies for knowledge representation or semantic web technologies for linking local robotic knowledge with web resources [15]. Therefore, the focus of robotic research community has been shifted to use these approaches for developing efficient robot navigation systems. Suh et al. [16] proposed Ontology-based Multi-layered Robot Knowledge Framework (OMRKF). It aimed at improving the robot's intelligence with its four levels of knowledge related to the model, context, perception, and activity. Its knowledge representation helped the robots to recognize objects with incomplete information and execute a task by only determining high-level service. It modeled the objects, actions, and concepts using Prolog [17]. Later, Lim et al. extended OMRKF and built OUR-K [12] knowledge framework for robot navigation, object recognition, action selection, manipulation, and planning tasks by applying ontology. It consisted of knowledge description for low-level data integration with high-level information and knowledge association in conjunction with bidirectional and unidirectional rules to enable the robot with reasoning capabilities when only partial information about an object was available.

With the rise of cloud-robotics, the web-scale approaches for knowledge processing are used by robotic frameworks. Few researchers suggest reusing and sharing abstract representations of high-level knowledge across different platforms. However, the problem is that most data depends on hardware-specific configurations. Besides, current database systems usually store only one type of data in isolation. These constraints limit the data reuse efficiently. Beetz et al. [13] addressed this problem in the RoboEarth, in which data collection, storage, and sharing were independent of specific robot hardware. It allowed the robots to access and share real-world environment and object models. In its web-based knowledge base, the robot's skills, surrounding environment, and objects were defined as generic action recipes. It generated the semantic maps for robot localization and navigation by integrating the geometrically precise visual SLAM methods with semantically enriched models of recognized objects that could be downloaded from the database. It relied on reference ontology and provided the semantic representation of actions along with the support for reasoning capabilities and rule-based learning [18]. Similarly, the KnowRob [19] knowledge processing system was a semantic framework for performing manipulation tasks that were previously vaguely described. It reduced the gap between detailed description and vague task description. It modeled the environment while integrated the common sense and encyclopedic knowledge with task and object description, along with robot capabilities. It also supported reasoning capabilities and used inference methods to query the knowledge base.

2.2. Environment Modeling for Robots

In this section, we present the literature review for environment modeling, focusing on two key areas: map representation and ontology-based knowledge models.

2.2.1. Map Representation

In the Simultaneous Localization and Mapping (SLAM) field, filter-based state estimation techniques and graph-based optimization methods are well studied for building environment

maps. Thrun et al. [20] presented a follow-up research work for an offline SLAM problem by introducing GraphSLAM, which represented the log-likelihood of the data and generated the maps with more than 10^8 features in urban environments. Ref. [21], used selective resampling operations to reduce the number of particles in Rao-Blackwellized [22] particle filter for building accurate map of environment. Ref. [23], addressed the motion estimation and distortion problem in lidar cloud by proposing a real-time approach divided into two algorithms that run in parallel. The lidar odometry algorithm estimated transform between consecutive frames at a higher frequency, while the lidar mapping algorithm performed fine matching and registration of the point clouds at a lower frequency. Subsequently, Parallel Tracking and Mapping (PTAM) [24] was considered the most popular method to implement visual based SLAM because of its ability to handle large number of 3D points. Later, Mur-Artal et al. [25] extended the adaptability of PTAM to intractable environments by designing a new feature-based monocular ORB-SLAM system from scratch.

The semantic mapping problem has attracted the researchers' attention in recent years. Pronobis et al. [26] proposed a probabilistic system for extracting semantic information from different heterogeneous modalities and combined it with common-sense conceptual knowledge. It made the semantic maps more descriptive based on the concept of spatial properties that were represented by the chain graph. Kostavelis et al. [27] gave a qualitative description to enhance the representation of the environment and enabled the robot to perceive its surroundings similar to the human. McCormac et al. [28] used CNNs and dense SLAM system ElasticFusion for providing dense correspondence from the 2D frame into the 3D map. The semantic predictions from CNN were obtained from multiple viewpoints and then fused into a dense map. Yang et al. [29] proposed an incremental semantic 3D mapping system with a scrolling occupancy grid map while CNN was used for segmentation and computation of pixel label distribution. Nakajima et al. [30] presented an approach for incrementally building dense 3D maps that were semantically annotated. The class probabilities were assigned to each region of the 3D map and segmented using geometric-based techniques, while CNN enhanced frame-wise semantic segmentation. Xiao et al. [31] presented a Dynamic-SLAM by taking advantage of deep learning-based recognition technique, known as SSD, to recognize the dynamic objects at semantic level while recall rate was improved by proposing a missed detection compensation algorithm.

2.2.2. Ontology-Based Knowledge Model

Ontology-based knowledge systems are required to transform a simple robot into an intelligent one. For this, Riazuelo et al. [32] proposed a system by integrating the knowledge base with a visual SLAM to provide a precise and continuous perception of the environment and accurate recognition of objects. He designed the semantic mapping method based on scene geometry and object locations by combining both the visual SLAM and the RoboEarth ontology. The knowledge base methods used prior information of the landmarks' locations in the semantic map to determine the potential locations of objects for recognition and robot guidance when searching for a specific object. A robot can efficiently perform its task if it has semantic knowledge. The explicit semantic representation of a real-world environment enables it to decide that an object model is suitable to perform the assigned task; if not, then the robot can determine to select among other alternative object models using semantic information. Tenorth et al. [33] worked on this challenge and proposed semantic representation language to speed-up this task by describing actions, articulation models, object recognition, semantic maps of the environment, and reasoning methods about these pieces of information.

It is a challenging task to establish a relationship between representations and their entities, known as grounding. Johnston et al. [34] extended the robotic grounding systems using semantic web technologies. Its architecture was composed of an ontology-based vision subsystem that was mainly focused on object recognition, reasoning, categorization, communication, and collaboration. Autonomous household robots rely on accurate mapping and navigation that need spatial information. Tenorth et al. [35] proposed a system that combined spatial information, encyclopedic knowledge,

and general knowledge with activity observations knowledge for environmental map building. Crespo et al. [36] proposed a semantic relational model based on a physical and conceptual representation of real-world objects, places, and semantic relationships among them to endow the robot with the capabilities of making queries about its surroundings for planning and navigation tasks.

2.3. Recognition

In this section, we focus on reviewing the object and place recognition approaches, along with localization techniques described in the literature.

2.3.1. Object Recognition

Visual object recognition is a significant problem in mobile robots. Advanced research efforts in object recognition [37,38] are focused on understanding how a cognitive representation, from the filtered output of sensor data, can be achieved to empower the robot with human-like perception [39]. In [40], human-like object recognition was developed for information processing using three hypotheses based on cognitive psychology. Deep neural networks (DNNs) inspired by neurons in the brain have recently come out as powerful algorithms to investigate the biological vision. The neural basis of DNNs has motivated the researchers to use them as models for information processing for recognition tasks [41]. CNN-based object recognition models are categorized into two-stage and one-stage detectors, in which You Only Look Once (YOLO) [42] is considered extremely fast and performs the detection task as a regression problem. Ref. [43], presented a robust method for object recognition based on a semantic model graph in which structural and geometrical attributes along with semantics as intrinsic properties of objects were used to establish a link between high-level semantics and low-level image features. Zhang et al. [44] proposed SPDA-CNN architecture consisting of two sub-networks and introduced mid-level layers for extracting part-specific features to improve the recognition performance. Detecting and recognizing partially occluded parts is a challenging task for modern object detectors because they cannot deal with occlusions. Wang et al. [45] addressed this issue by accumulating the confidence of local visual cues, also known as visual concepts, which were extracted from the internal states of deep networks using clustering.

Many researchers have considered the ontology-based knowledge-driven approaches for the semantic recognition task. In an early study, Choi et al. [46] proposed an approach of ontology inference for constructing the semantic contexts extracted from robot sensors to perform object recognition tasks in real-world environment. Web Ontology Language (OWL) was used for the representation of object ontology in which semantic contexts were generated using axiomatic rules. Another study [47] represented a cognitive vision approach for recognizing the categories of complex objects. This approach was proposed for the semantic interpretation of individual objects. The visual concept of ontology was composed of spatial, color, and texture data. These concepts were used as an intermediate layer, while the link between symbols and sensory information was established using machine learning techniques. Allani et al. [48] introduced a semantic information retrieval approach to build a modular ontology and graph-based model for visual features. A seminal study in this area was the Adaptive Bayesian Recognition framework introduced by Lee et al. [49], which endowed the robot with semantic understanding of 3D objects in home environment. A novel FER-CNN model was incorporated with this framework to extract and reconstruct 3D features by semantically linking to the ontology.

2.3.2. Place Recognition

Visual place recognition (VPR) plays a vital role in achieving long term autonomy for mobile robotic applications [1]. Several studies have shown that features extracted by CNN models outperform traditional vision-based approaches such as SIFT. This success of deep learning-based CNN models has motivated the researchers to use CNN based approaches for the VPR task. Generally, VPR methods are divided into image acquisition, feature extraction, and image search [50].

In a study, Schönberger et al. [51] proposed an approach based on 3D geometric and semantic understanding of real-world environment by training the generative model on the semantic scene to improve the descriptor learning for missing observations. Garg et al. [52] developed a set of semantics and appearance-based methods for efficient VPR in a challenging scenario by extracting convolutional features from a dense semantic segmentation network using the proposed Local Semantic Tensor (LoST) descriptor of images. Chen et al. [53] generated condition and viewpoint-invariant features by training two CNN models to recognize specific places with a multi-scale feature encoding method. Ref. [54], used CNN to perform the task of recognizing revisited locations under different environmental variations and mapping visual information to low dimensional space with Euclidean distance that corresponds to place dissimilarity.

2.3.3. Localization

The robotic research community is making steady progress in determining the autonomous robot localization concerning the environment using recognition techniques. Mitsuhashi et al. [55] proposed a system for mobile robot localization with an appearance-based place recognition approach in conjunction with the dead reckoning and gyrodometry model. Ref. [56], presented a vision-based localization approach in the outdoor environment against a known map by combining a place-dependent feature detector with prior experience of place. Instead of representing edges and corners, these features represented mid-level patches of different elements like windows. Ref. [57], presented the Visual Landmark Sequence-based Indoor Localization (VLSIL) approach and addressed environmental variations using semantic information of steady indoor objects (landmarks) while performed the localization using objects' occurrence order in the video. Zhong et al. [58] proposed a mobile robot's self-localization method using recognition of artificial visual landmarks which were identified using a bilayer recognition algorithm. The geometric relation was established between image and environmental frames to get the orientation and position of the camera for robot localization.

2.4. Planning

Planning is one of the main study areas in the robotics field, especially after robots started performing central roles in modern industry and our daily life [59,60]. Moreover, the Defense Advanced Research Projects Agency (DARPA) challenge [61] provided real-world conditions and pushed the boundaries of the Self-Driving Vehicle (SDV) researches. It also showed that sophisticated planning solutions were paramount for SDVs being able to handle a broad span of driving scenarios.

Planning is referred to as a sequence of actions necessary to accomplish a goal, given an initial state. To complete such a goal, the agent requires a description of the environment, its actions, and its capabilities. A planner can perform different roles depending on how general they are. There are many ways of dividing those roles, such as Global/Local planner or Mission/Behavior/Motion planner. The Global planner, generally performed offline, generates a high-level low-resolution sequence of actions, based on the a priori known environment, to accomplish the final goal. The local planner, on the other hand, focuses on a low-level high-resolution sequence of actions, based on the online sensor data, that focus on a portion of the global path [62]. In this paper, however, the Mission/Behavior/Motion division is explored [59].

2.4.1. Mission Planning

On robots, the mission planner generally performs as a "task planner," while on SDV, it focuses on map-level navigation or route planning [3,63].

Task planners usually follow one of two different approaches: classical planning or decision-theoretic planning. One of the first works to integrate classical planners into robotics was the Stanford Research Institute Problem Solver (STRIPS) [64], first implemented on a real robot by Nilson [65]. The planning community standardized STRIPS-based planning by introducing the PDDL, which was later expanded to support temporal planning as well as [10]. Cashmore et al. introduced

the ROSPlan framework [11] integrating PDDL 2.1 with the Robot Operating System (ROS), which is vastly used by the robotics research community. By using a Knowledge Base, ROSPlan is able to build the initial domain state automatically, validated the generated plan, and decide when to re-plan. Estivill-Castro et al. [66] showed that PDDL-planners could also work on dynamic environments when other agents follow a deterministic behavior. Researchers have been attempting to integrate Hierarchical Task Network (HTN) with task planning using different techniques. Dvorak et al. [67] proposed the first Flexible ANML Actor and Planner (FAPE) that combined most of the Action Notation Modeling Language (ANML) [68] with HTN task decomposition. It also integrated acting that followed Procedural Reasoning System (PRS) refinements, plan repair and re-planning. However, it used non-recursive decomposition hierarchy which decreased the expressiveness of HTN planning.

Nonetheless, classical planners generally do not perform well in a dynamic and unpredictable environment due to its linear nature, making it unable to reason about multiple outcomes. Decision-theoretic planning, on the other hand, tries to maximize the probability of success (also known as a maximum expected utility, MEU) for every possible action outcomes from any state the robot reaches during the plan execution. It can solve Markov Decision Processes (MDP) and even POMDPs, but its high computational cost makes decision-theoretic approaches prohibitive. Most of the research done in this field was trying to overcome this issue. Boutilier et al. [69] tried to solve an approximated optimal policy by identifying which propositions are relevant and using this information to reduce the state-space. On a similar fashion, Dean et al. [70] used a known initial state to model the world nondeterminism and restrict the planner to focus only on states that might be encountered during the plan execution.

Some newer approaches tried to combine ontology and semantic knowledge into planning. Hence, ontological knowledge eases semantic, temporal, and spatial representations of the environment [4]. Galindo et al. [71] used semantic maps to extend the planner reasoning capabilities, aiding high-level tasks understanding. KnowRob [72] is a knowledge processing framework that combines episodic memory, mental simulation, and a generalized ontology to help a robot to reason, plan, and learn. This work, however, focused only on manipulation tasks [73].

2.4.2. Behavior Planning

The behavior planner receives the next goal/task information from the mission planner combined with the local sensorial data and uses such information to check feasibility, adjust the local level plan based on environmental constraints, and re-plan when necessary. Earlier works used an Finite State Machines (FSM) to make behavioral decisions [74]. Koo et al. [75] used a Strategy/Behavior/Motion hybrid planning approach to control an Unmanned Aerial Vehicle (UAV) in which an FSM was used to define the next action based on a given strategy and the perceived state. Shi et al. [76] combined an FSM with fuzzy logic transition rules obtained from human experience and heuristics. FSMs are, however, limited by the manually designed states, which may lead to livelocks or deadlocks whenever the agent encounters a situation that was not explicitly encoded into the FSM structure [3]. More sophisticated planning architectures tried to overcome those issues. Wei et al. [77] used a Prediction and Cost-Based algorithm to generate action candidates, predict the environment state, and evaluate the best action based on several cost values.

2.4.3. Motion Planning

Motion planners receive a goal point coupled with sensor data and try to find a valid motion sequence to reach the goal without violating any constraint, such as not hitting any obstacle along the way. Usually, motion planners are compared based on their computational efficiency (the process run time and its scalability with the environment growth) and completeness (its ability to find the optimal solution in a finite time) [3]. The most notable motion planning methods are the Cell Decomposition Methods (CDMs), the Potential Field Methods (PFMs), and the Sampling-Based Methods (SBMs).

In cell decomposition methods, the environment's free space is divided into small cells, and the output is a sequence of adjacent cells that leads the initial position to the goal while avoiding obstacles. Arney [78] used approximate cell decomposition, which does not set a fixed cell shape and size, to navigate efficiently. Lingelbach [79] combined CDM with probabilistic sampling to improve its scalability. Cell decomposition approaches, however, suffer from limited granularity, combinatorial explosion, and often generating infeasible solutions [80]. In potential field methods, an attractive potential is attached to goals while a repulsive one is attached to obstacles, and the robot follows the gradient direction generated by the combination of those fields. Both Pradhan et al. [81] and Kang et al. [82] showed that such approach is suitable for multi-robot navigation. Potential fields, however, suffer from often converging to local minima, trapping the robot midway [80].

Sampling-based algorithms have attracted considerable attention due to their ability to scale well while also being probabilistic complete, namely if the number of samples converges to infinity, the probability of finding the optimal solution converges to 1. The most notable SBMs are Probabilistic Roadmaps (PRM) and Rapidly-exploring Random Trees (RRT) [83]. Despite both being based on connecting points sampled randomly, the way both algorithms connect those points largely differ. PRM [84] maintains multiple graph expansions simultaneously, and have been shown to perform well on high dimensional spaces, while RRT [85] rapidly explores a single graph, making it more suitable for local planning on small sized maps. Karaman et al. [86] showed that under mild conditions, original PRM and RRT algorithms, most of the time, returned non-optimal values. The author then introduced asymptotically optimal versions of both methods, namely PRM* and RRT*. This advantage, however, is overshadowed by their relatively slow convergence, which hinders their usage in real-time scenarios. They also provided no theoretical guarantees for obtaining an optimal solution [87].

2.4.4. Deep Reinforcement Learning

Reinforcement Learning (RL) algorithms try to map a set of sensory inputs to actions by formulating this task as a POMDP. Deep RL uses Deep Neural Networks (DNN) to approximate such mapping. Mnih et al. [88] created a Deep Q-Network (DQN), namely a DNN, to estimate Q-values for a value-based RL approach. Shah et al. [89] combined a DQN with several other deep structures to map a natural language command, a segmented semantic image, and a depth image to a motion command, aiming end-to-end map-less navigation. DQN, however, is only suitable for discrete action spaces, thus making it less practical for navigation tasks. Lillicrap et al. [90] proposed an actor-critic RL approach called Deep Deterministic Policy Gradients (DDPG), which used separated neural networks for action generation and Q-value approximation. This hierarchical structure was able to output continuous actions. Tai et al. [91] expanded this usability by creating the asynchronous DDPG (ADDPG), which used several threads to collect experience and a centralized gradient learner. This method was used to perform mapless navigation, and it was shown that it has a faster convergence than the original DDPG.

Our work utilizes a similar approach to Faust et al. [92], who combined a PRM algorithm with an RL policy to address the shortcomings of each method.

3. TOSM-Based Autonomous Navigation Framework

This section introduces a novel autonomous navigation framework that endows robots with the capacity to perform complex missions using high-level and human-like knowledge. The framework illustrated in Figure 1 is composed of semantic modeling framework (SMF, Section 3.1), semantic information processing (SIP, Section 3.2) module, and semantic autonomous navigation (SAN, Section 3.3) module.

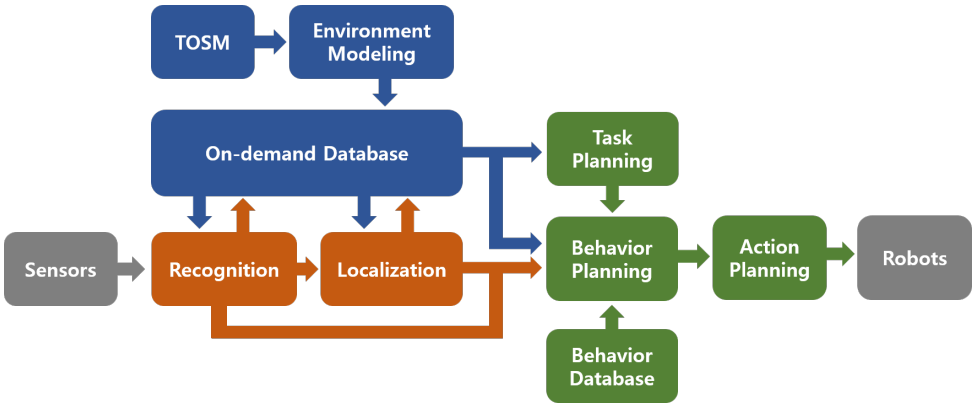


Figure 1. The proposed framework consists of semantic modeling framework (SMF) (blue blocks), semantic information processing (SIP) module (orange blocks), and semantic autonomous navigation (SAN) module (green blocks).

3.1. Semantic Modeling Framework

For robots to perform complicated missions, they must have the ability to comprehend the essence of each mission and their working environment. However, recognition methods based on traditional modeling techniques that utilize low-level geometric and semantic information are not suitable for performing complex missions. In other words, robots need a novel high-level environmental model that combines advanced knowledge information to apprehend environments. In this section, we propose the SMF based on the TOSM that imitates the human ability to express and understand the environment.

3.1.1. TOSM-Based Environment Modeling

To begin with, we divide the elements that make up the environment into an object, place, and robot. Each environmental element is represented by the explicit, implicit, and symbolic models based on the TOSM, as shown in Figure 2, to include high-level environmental information. The explicit model refers to information that can be obtained directly by processing sensor modalities. It encompasses metric information (pose, velocity, size), geometrical features (shape, boundary), and image information (image features, colors) that can be measured using logical sensors based on sensor models. In contrast, the implicit model consists of induced information based on knowledge models and databases. The relations between environmental elements (e.g., The blue chair is inside of the meeting room.) and facts (e.g., a car is movable.) such as human semantic memory defined in the field of cognitive science [93] can be considered implicit model components. Finally, the symbolic model symbolizes environmental elements so that robots and humans can interact efficiently.

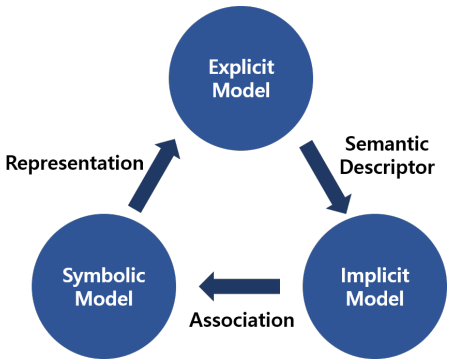


Figure 2. Triplet ontological semantic model.

Each model of TOSM is described based on classes, object properties, and datatype properties, which are terminologies defined in the OWL reference [94]. In the following tables, we explain the environment modeling based on TOSM using the OWL terminologies.

We use classes to group resources with similar characteristics for providing an abstraction mechanism. Table 1 lists the example of hierarchical classes for environmental elements. *Object*, *Place*, and *Robot* are defined as subclasses of the *EnvironmentalElement*, and each class is defined as superclass of the classes expressed in its right column (e.g., *Door* is a superclass of *AutomaticDoor*). Subclasses inherit the characteristics of superclasses, and the relation between two classes can be defined as "is-a." For example, *AutomaticDoor* inherits all the characteristics of *Door, Object*, and *EnvironmentalElement*; the relationship between *AutomaticDoor* and *Door* can be described as "An *AutomaticDoor* is a *Door*." The classes are used to determine domains and ranges of object properties and datatype properties. Therefore, subclasses can define more specific properties rather than superclasses, such as *AutomaticDoor* describes *Door* in more detail. In the case of subclasses for *Robot*, we use the robot description ontology model as defined in [95].

Table 1. Classes.

Environmental Element	Object	Door	AutomaticDoor HingedDoor ElevatorDoor
		Sign	GuideSign SafetySign
		Device	VendingMachine WaterPurifier
		Furniture	Table Chair
		Person	Occupant Pedestrian Driver
	Place	IndoorPlace	Corridor Doorway Room Elevator Staircase Floor Cafe
		OutdoorPlace	Building Road Sidewalk
	Robot		

Object properties are used to represent relations between environmental elements corresponding to the implicit model; in the OWL, object properties provide relations between individuals that are members of the classes. The object properties can be defined using object property hierarchy, descriptions, and characteristics. The object property hierarchy in Table 2 expresses the nodes corresponding to the object properties in a tree-like structure; the right child node is a subproperty of the left parent node. We use *SubPropertyOf*, domains, and ranges among the various descriptions of object properties. *SubPropertyOf* is used to represent the subproperty relations between nodes in the object property hierarchy (e.g., *isFrontOf SubPropertyOf spatialRelation*). The object properties can set domains and ranges of each property using the classes. The individual using the property is inferred as an instance of the domain and range class. For example, if "CIR752 *isLookingAt* Table34" and *Robot* and *Object* are domain and range of *isLookingAt* respectively, CIR752 is inferred as *Robot* and Table34 as *Object*. We use symmetric and transitive characteristics of the object properties. "The property is symmetric" means that it is reasonable to exchange the domain and range of the property. For example, "B *isNextTo* A" can be inferred from "A *isNextTo* B", because *isNextTo* is symmetric. In case of *isInsideOf*,

the property is transitive because we can realize "A *isInsideOf* C" using "A *isInsideOf* B" and "B *isInsideOf* C".

Table 2. Object properties.

Object Property Hierarchy		Domains	Ranges	Characteristics
spatialRelation	isInFrontOf	Object	Object	symmetric
	isNextTo	Object	Object	
	isAboveOf	Object	Object	
	isBehindOf	Object	Object	
	isOn	Object	Object	
	isInsideOf	Object or Place	Place	transitive
	isConnectedTo	Place	Place	symmetric
robotRelation	isLookingAt	Robot	Object or Place	
	isLocatedAt	Robot	Place	

We divide relations between objects, places, and robots into *spatialRelation* and *robotRelation*. *SpatialRelation* means the spatial relations between objects and places that constitute an environment. In particular, the properties, where *Object* specifies both domains and ranges, express relative position relations between objects within places. The properties can be used by robots to infer the relations between objects. Robots can also use a database which consists of *isInsideOf* and *isConnectedTo* to understand the environment similarly the way a human does. We explain the *spatialRelation* in detail with an example in Section 3.1.2. *robotRelation* is used to define the properties that are determined by the robot's current state. When the robot is in operation, it uses *isLookingAt* and *isLocatedAt* properties to comprehend the current state in real-time by associating it with objects and space stored in the database. We explain the *robotRelation* exactly in Section 3.2

We use datatype properties to define the TOSM of environmental elements; in the OWL, data properties are relations between instances of classes and RDF literals or XML Schema datatypes. The datatype property hierarchy, similar to object property hierarchy, represents the subproperty relations between datatype properties. Additionally, classes determine domains of the datatype properties, and datatypes determine ranges.

Table 3 lists definitions of the symbolic, explicit, and implicit model for *Object*. The symbolic model consists of *name* and *ID* to symbolize individuals of *Object*. For the sake of clarity, we describe ranges of *pose*, *velocity*, *size*, and *color* in the explicit model as above. The ranges of these properties are defined as an array of the datatype (e.g., float array) in the database. *isKeyObject* in the implicit model is utilized to represent crucial objects used for localization and place recognition. Furthermore, the properties, whose domains are the subclasses of *Object*, are used to define a more specific model of each subclass. In the case of *Door*, *isOpen* means whether it is open or not, and *canBeOpen* means whether it can be opened if it is closed. Robots can use this information when planning and performing tasks.

Table 3. Datatype properties for *Object*.

Datatype Property Hierarchy		Domains	Ranges
symbolicModel	name	Object	string
	ID	Object	int
explicitModel	pose	Object	(x, y, z, quaternion)
	coordinateFrame	Object	string
	velocity	Object	(μ, v, w, p, q, r)
	size	Object	(l, w, h)
	color	Object	(r, g, b)
implicitModel	isKeyObject	Object	boolean
	isMovable	Object	boolean
	isOpen	Door	boolean
	canBeOpen	Door	boolean

The definitions of the TOSM for *Place* and *Robot* using datatype properties are shown in Tables 4 and 5 respectively. The definitions of the symbolic models are the same with *Object*. We select boundary composed of polygons for the explicit model of *Place*. In the implicit model, *complexity* informs the state of *Place*. *capability* and *purpose* in the implicit model of *Robot* are the main components when making an on-demand database.

Table 4. Datatype properties for *Place*.

Datatype Property Hierarchy		Domains	Ranges
symbolicModel	name	Place	string
	ID	Place	int
explicitModel	boundary	Place	polygon
	coordinateFrame	Place	string
implicitModel	complexity	Place	float
	level	Floor	int
	purpose	Room	string
	roomNumber	Room	int

Table 5. Datatype properties for *Robot*.

Datatype Property Hierarchy		Domains	Ranges
symbolicModel	name	Robot	string
	ID	Robot	int
explicitModel	pose	Robot	(x, y, z, quaternion)
	coordinateFrame	Robot	string
	velocity	Robot	(μ, v, w, p, q, r)
	size	Robot	(l, w, h)
implicitModel	capability	Robot	string
	purpose	Robot	string

3.1.2. On-Demand Database

Figure 3 visualizes a TOSM-based environment database of a simple environment containing a multi-floor building. The database includes individuals of defined classes, data properties of the individuals, and object properties between individuals. In the figure, gray squares are individuals from subclasses of *Place*, and blue capsule-shapes are individuals from subclasses of *Object*. Unidirectional black arrows mean *isInsideOf* property between individuals; two-way red and blue arrows mean *isConnectedTo* and *isNextTo* properties, respectively, that have the symmetric characteristic. The table shown on the left side of the figure describes datatype properties for an individual of *automaticDoor*.

We generate an on-demand database based on the environment database considering the robot's hardware-dependent functionalities when it is assigned a mission. For example, assuming a robot that cannot drive on stairs is given a mission from *room1* to *sidewalk1*, a database for *building2* connected to *sidewalk1* and *staircase1* is not necessary for the robot; we can create an on-demand database by sorting out the database that the robot needs. Based on the on-demand database, the robot can realize that it is in *room*1 inside $floor2$ inside building1 using *isInsideOf* properties connected to *room1*. Besides, the paths to *sidewalk1* from *room1* can be planned, utilizing the individuals of *Place* connected to *room1* by *isConnectedTo*.

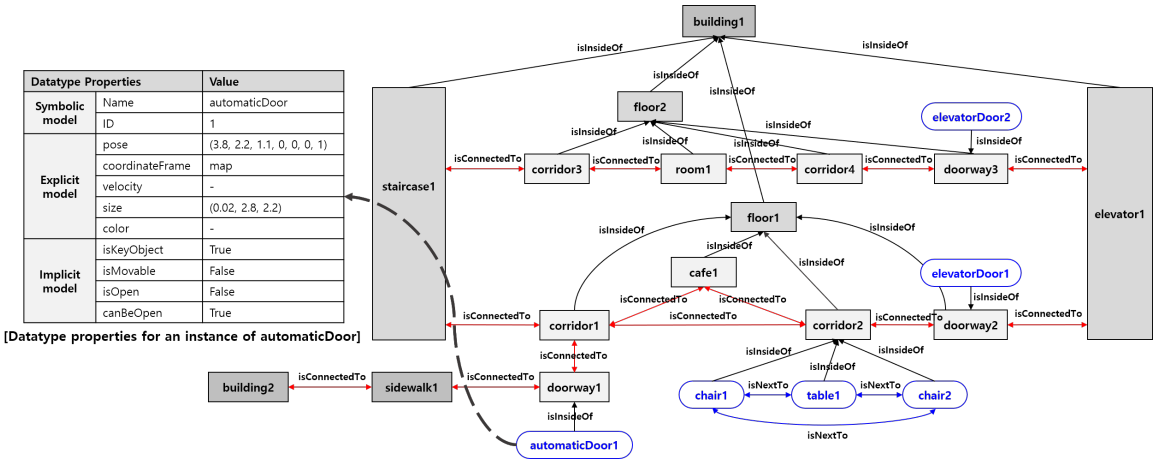


Figure 3. A simple example of the Triplet Ontological Semantic Model (TOSM)-based environment database.

3.2. Semantic Information Processing

Our Semantic Information Processing (SIP) module endows the robot with cognitive vision capability, inspired by the human ability, that tends to recognize the places by the objects present there, such as "computer lab" or "printer room," which are common terms that are used by human to recognize the places based on their objects. Similarly, humans identify outdoor places by distinguishing different features or objects, such as billboards.

Motivated by these ideas, our SIP module also uses an objects-based recognition approach to form a symbolic abstraction of the places. In the SIP module, first, the objects are recognized by a CNN model, then the inference is performed to leverage the recognized objects with their semantic information stored in the on-demand database. It also infers the robot's relationship information based on the spatial information of recognized objects in the environment. Our SIP module aims at performing recognition and localization tasks with semantically meaningful information to effectively guide the robot's behaviors to reach a goal location. To achieve this, we combine our recognition model with the on-demand database, as shown in Figure 4, and perform the inference on recognized objects against their semantic information stored in the on-demand database.

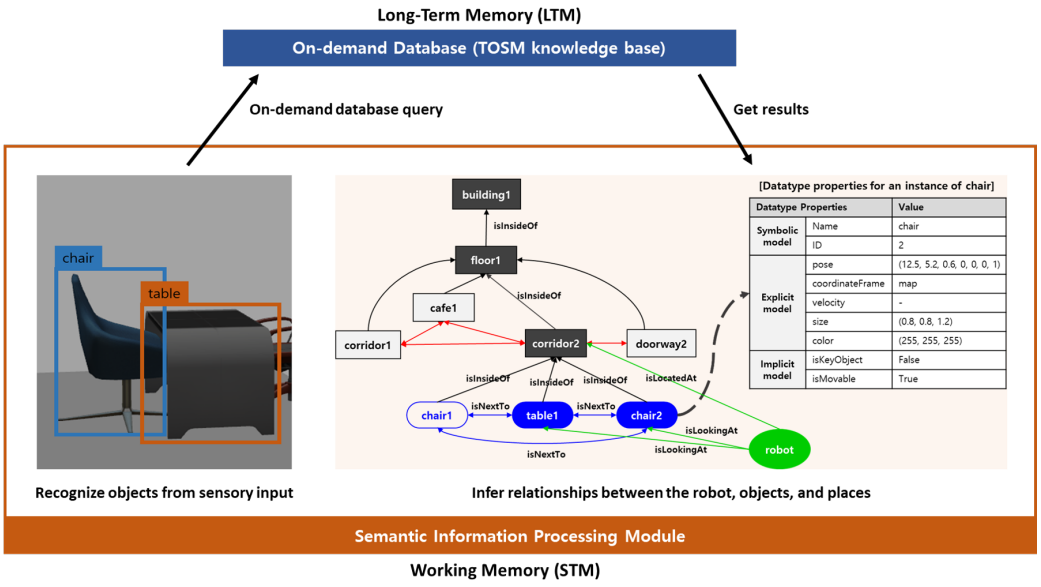


Figure 4. Semantic information processing module.

We have selected the third version of YOLO [42] for object recognition because it presents a good trade-off between execution time and accuracy, compared with other object detectors. For example, its accuracy is almost similar to RetinaNet [96]; however, it is four times faster. Moreover, it is much better than SSD variants and Darknet-53 as its backbone network architecture makes it 1.5 times faster than ResNet-101 [42].

First, we train state-of-the-art one-stage CNN based model, YOLOv3 [42] for object recognition with our own data set. When a mobile robot navigates in the environment, visual data captured by a camera is passed to the CNN model; it produces a sequence of layered activations, which generate feature vectors that have a deeply learned representation of high-level features. We perform pre-processing step to remove the noise before passing the visual data captured by the robot's camera to the CNN model. Both the raw-data and the CNN model are hosted by short-term memory (STM). Raw-data does not have access to the on-demand database, while the CNN model, which holds the recognition components, has access to the on-demand database. *Object* and *Place* models are initially stored in this database. We download these semantically enriched models of recognized objects from the on-demand database when required by the robot.

Our YOLO-based recognition model consists of two phases: the processing phase and the retrieving phase.

The processing phase is also known as an offline or training stage in which we train the YOLOv3 model for object recognition using our dataset. The training phase involves three major steps; (i) Dataset annotation: we label objects in each image using the YOLO-Mark tool. (ii) Convolution weights: we use YOLOv3 weights trained on ImageNet for convolutional layers. (iii) Configuration set-up: we prepare a model configuration file that loads 64 images in the forward pass for each iteration during the training process. The model updates the weights when backpropagating after gradients' calculation. Thus, for each iteration, our model uses 64 batch size with 16 mini-batches, and four images are loaded for each mini-batch. In our configuration file, we define filter size to 60, the maximum number of iterations 30,000 for 15 classes. The YOLOv3 has three YOLO layers; therefore, we generate nine anchors to recognize small, medium, and large size object at its three YOLO layers. Finally, when the training phase is completed, we get new weights for our dataset.

The retrieval phase is known as an online or testing stage. In the testing phase, weights that were generated during model training, are used to recognize objects in a real-world environment.

In our SIP module, after object recognition using the CNN model, inference for semantic information is performed at run-time by integrating the recognized objects with their semantic information that contains spatial relationships between objects, described in Table 2. Through these spatial relationships among recognized objects, we infer the relationship information of the robot by leveraging with the semantic objects and space stored in the on-demand database as follows:

Assume x is an object recognized in the environment by the CNN model, while X is the subclass of *EnvironmentalElement* (e.g., *Furniture*, *Door*, and *Corridor*). Then the class of object x is defined by a class definition function $IsA(x, X)$, which indicates that object x is class X. As shown in Figure 4, there are two objects in the environment: table and chair. The class definition function $IsA(table, Table)$ and $IsA(chair, Chair)$ states that in both cases, the class of the table is *Table* and the chair is *Chair*. When the class is defined, we can get the datatype properties of the object from the database based on the robot's pose and relative pose of the object. In other words, *Chair* is specified *chair2* (we call this X_n) that is an instance of *Chair* by the database query such as illustrated in the gray table inside Figure 4.

We also use a relationship inference function $SR = rel(X_n, X_m)$ to define relationships between recognized objects, other objects, and places in the database. According to the class definition function, X_n indicates an instance of class X. SR describes a spatial relationship between X_n and X_m, such that "X_n $isNextTo/isInsideOf/isInFrontOf$ X_m" (e.g., "$table1$ $isNextTo$ $chair2$".) Similarly, we can infer the robot's relationship in the environment; in that case, the X_n is an instance of *Robot*.

So, when the robot navigates in a real-world environment, and it finds two objects, table and chair, shown in Figure 4, the robot recognizes these two objects using the CNN model and infers these objects

with their semantic information that is saved in the database. As a result, we get spatial relationships such as "*table*1 *isNextTo chair*2", "*table*1 *isInsideOf corridor*2", "*robot isLookingAt chair*2", and "*robot isLocatedAt corridor*2".

We can also see that the recognition model and the TOSM-based on-demand database share human-like memory architecture when working together for visual semantic recognition. The SIP module is processed in working memory (Short Term Memory, STM), while prior semantic knowledge stored in the on-demand database is hosted by Long-Term Memory (LTM), as shown in Figure 5. When the mobile robot navigates in a real-world environment, it sends the sensory data to SIP, which passes the pre-processed data to the recognition model that is stored in working memory and gets the periodic visual updates as robot navigates.

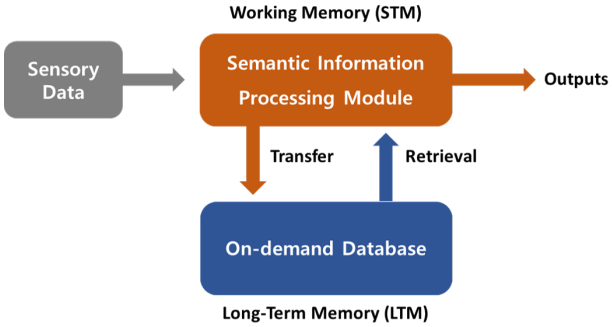


Figure 5. Human-like memory architecture.

3.3. Semantic Autonomous Navigation

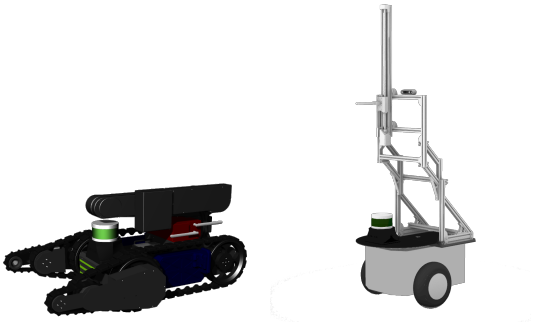
The Semantic Autonomous Navigation (SAN) module allows the robot to perform complex missions through a hybrid hierarchical planning scheme. The knowledge base, combined with a behavior database, enables autonomous navigation in a variety of environments while taking the robot capabilities into account. Throughout this section, we use a single mission example for a clearer understanding. Consider a multi-story building represented by the database displayed in Figure 3, where elevators and staircases connect each floor; a robot located on the 2nd floor inside the building; a mission given by a human that says it should retrieve a package at the sidewalk outside of the main entrance located on the first floor. The following sections demonstrate, through this mission, how the SAN module can enable robots to perform high-level multi-story navigation.

3.3.1. Task Planner

After receiving a mission, such as *robotAt place1* (navigation) and *inspected place2* (inspection), the SMF creates the robot's on-demand database based on the mission and its capabilities. For example, different robots have different capabilities: some can go up and downstairs, while others can operate an elevator autonomously, as shown in Figure 6. Considering a robot that can operate the elevator, the on-demand database does not need to contain the staircases' information.

The task (i.e., mission) planner access only to the on-demand database speeding up the queries during the plan execution. The environmental implicit information, namely *isConnectedTo* property together with the robot's location, is used to populate the ROSPLAN [11] knowledge base. The robot's implicit information is also used to provide the domain PDDL file. Thereupon, ROSPLAN is used to generate a valid behavior sequence. To generate a behavior sequence, PDDL planners require two files: domain and problem. The domain file describes the environment predicates and robot actions. As different robots can perform a distinct set of actions, a comprehensive set of domain actions, all sharing common predicates between them, is stored into the behavior database. Given the robot's implicit TOSM data, a matching domain file is generated and fed to ROSPLAN by the behavior database; the problem file is generated automatically by ROSPLAN knowledge base. It needs, however, the environment state information to do so. ROSPLAN state update services are

used to feed the map data into the knowledge base. After that, the behavior sequence is generated through the Partial Order Planning Forwards (POPF) planner [97]. Despite the current state of the behavior database not including temporal-actions, we have plans to also support them in the future. Therefore, we opted to use the POPF planner due to its ability to deal with both simple and durative actions. ROSPLAN also sends the current behavior to the Behavior Planner through the medium of the Plan Dispatcher node.



(**a**) CIR Robot (**b**) 02 Robot

Figure 6. The CIR robot is equipped with a continuous track and extension arms, enabling going up and downstairs. The 02 robot is equipped with sensors and manipulators that allow elevator operation.

In our example, the 02 robot shown in Figure 6b receives the multi-floor navigation mission. The robot then receives the on-demand database generated by the SMF based on the robot's implicit information (i.e., its capabilities). This on-demand database contains the building floors and the elevators, but no stairs, as the robot cannot transverse through them. Moreover, a domain file containing all the possible robot actions is sent to the ROSPLAN knowledge base. An example action, which sends a robot to a waypoint on the same floor, is shown on Listing 1. The predicates are general and shared between all actions stored on the database, thus making possible generating the domain file simply by querying which actions can be performed by the robot. Subsequently, the environment TOSM data is used to populate the ROSPLAN knowledge base environment state, adding connected waypoints, elevator, robot, and goal predicates. After running the POPF planner, the behavior sequence shown on Listing 2 is generated and sent by the dispatcher node to the Behavior Planner module.

Listing 1. Move to waypoint action.

```
(:action goto_waypoint
:parameters (?v - robot ?from ?to - waypoint ?fl - floor)
:precondition (and
(outside_elevator ?v)
(robot_at ?v ?from)
(waypoint_at_floor ?from ?fl)
(waypoint_at_floor ?to ?fl)
(hallway_waypoint ?from)
(hallway_waypoint ?to)
(connected_to ?from ?to)
)
:effect (and
(robot_at ?v ?to)
(not (robot_at ?v ?from))
)
)
```

Listing 2. A behavior sequence generated by ROSPLAN.

```
(goto_waypoint 02robot wp0 wp1 fl2)
(goto_waypoint 02robot wp1 wp4 fl2)
(goto_waypoint 02robot wp4 wp5 fl2)
(enter_elevator 02robot wp5 wp18 elevator2 fl2 fl1)
(exit_elevator 02robot wp18 elevator2 fl1)
(goto_waypoint 02robot wp18 wp21 fl1)
(goto_waypoint 02robot wp21 wp22 fl1)
```

3.3.2. Behavior Planner

The behavior planner can be divided into two modules: the waypoint generator and the behavior handler. After receiving the next goal from the task planner, one of those two modules takes care of breaking down this behavior into atomic actions. The waypoint generator handles motion behaviors. While complex specific activities, such as entering and exiting an elevator or going up or downstairs, are processed by the behavior handler.

The waypoint generator receives a goal point from the task planner, the current position from the localization module, and a local metric map obtained from the on-demand database and stored into the STM. It then uses the PRM algorithm proposed by [84], after being adapted for mobile robot navigation, to generate a feasible sequence of waypoints. We start by generating random samples $\mathcal{N}_i$ on the metric map free space $\mathcal{C}_{free}$. The PRM algorithm's goal is to generate a valid path between the current position $\mathcal{N}_p$ and the goal $\mathcal{N}_g$ by creating edges connecting two sample nodes $\mathcal{E}_k(\mathcal{N}_i, \mathcal{N}_j)$. Initially, it generates n random samples on the 2D metric map, eliminating any sample outside of $\mathcal{C}_{free}$ and re-generating new random samples until n valid samples are successfully generated, whereas in our approach, PRM works as a low-resolution path-planner, n can be of a smaller magnitude when compared to the overall map size, speeding up the query step. Thereon, it iterates through every sample generating valid paths connecting to the k-nearest neighbors. A path is considered valid if the two points can be connected by a straight line which is completely inside $\mathcal{C}_{free}$ and is smaller than a maximum connection distance threshold. This graph generation depends only on the metric map and can be pre-generated by the robot when idle or while executing actions that do not require a large amount of processing power. Finally, to generate a waypoint sequence, $\mathcal{N}_p$ and $\mathcal{N}_g$ are added to the graph and the *Dijkstra's algorithm* [98] is used to query the shortest path.

The behavior handler receives a specific desired behavior from the task planner and then queries the behavior database for instructions on how to execute such activity. The behavior database (BDB) can be seen as the human implicit memory [93], which saves cognitive skills learned throughout one's life. The BDB stores finite state machines (FSM) able to coordinate single behaviors, such as the one shown in Figure 7. Each state (i.e., action) has a standardized ID and a possible events list that is used to check the transition to the next state. After obtaining the FSM description, the behavior handler sends the initial action ID to the action planner and waits for the returned events. Thereon, it uses the event list to choose the next state and sends it back to the action planner. This sequence is repeated until one of the two final states is reach: success or failure.

Both the waypoint generator and the behavior handler emit a success signal to the task planner when the behavior is completed, allowing the planner dispatcher to send the next one. In case of failure, a homonymous signal is sent back instead, and the PDDL planner is able to perform re-planning starting from the current state.

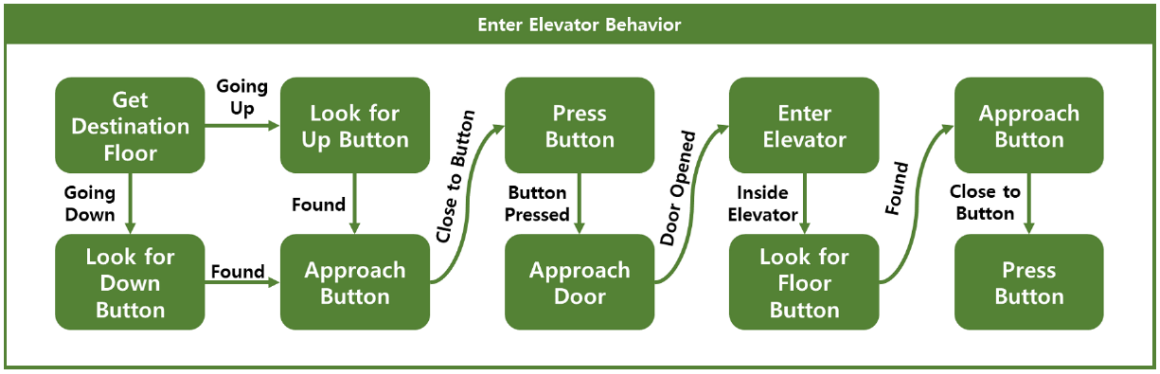


Figure 7. A Finite State Machine diagram showing a simplified version of the *Enter Elevator* behavior stored on the Behavior Database.

3.3.3. Action Planner

The action planner can receive two different kinds of actions: a waypoint from the waypoint generator or a specific action ID from the behavior handler.

When receiving an action ID, the action planner looks for the implementation of such action inside the robot's LTM. Each robot should have its implementation of the needed actions, as their execution is dependent on the robot sensors and actuators. In that case, the planner is unable to find the requested action, it returns a failure event to the behavior planner. On the other hand, if the action implementation is found, the robot executes it and returns the events related to the conclusion of such action.

If the requested action is related to following a waypoint instead, a different structure is used to handle it. A mentally simulated world was generated and used to train an autonomous navigation policy using reinforcement learning algorithms. This process is further explained on Sections 3.3.4 and 3.3.5. The action planner uses the policy returned by the mental simulation in order to navigate towards a waypoint while also avoiding close obstacles. After getting close enough to the goal point, the action handler returns the success event to the behavior planner and waits for the next instruction.

3.3.4. Mental Simulation

Mental simulation is one of the integral cognitive skills, which has been long studied by cognitive science and neuropsychology research [99]. Nonetheless, just a few works tried to implement such concepts into robotic platforms [72]. Some usages of the mental simulation are re-enacting memories, reasoning about different possible outcomes, and learning by imagination. In this work, the mental simulation module is used to perform reinforcement learning aiming for autonomous navigation. The simulation building algorithm was originally proposed on [100].

One of the main issues of performing mental simulation on robots is the need of domain experts to model the simulated world based on the robot working environment. Modeling the environment is not feasible when robots navigate through large-scale dynamic environments. To overcome such limitation, we built a middle-ware capable of generating a simulated world, including the robot itself, using only the information stored on the on-demand database. By doing so, whenever the robot updates its internal knowledge about the environment, the simulation is also updated in the same manner. Whenever the simulated environment is needed, the middle-ware queries the needed data from the on-demand database and generates Universal Robot Description Format (URDF) and Simulation Description Format (SDF) files, which are used to create the virtual world on the Gazebo Simulator. Figure 8 shows an example output of the simulation building algorithm. It uses the data stored on the on-demand database to generate the simulation environment. To ensure this simulation is robust while also maintaining its consistency with the real world, we use a library containing 3D

meshes for common objects such as doors, chairs, and tables. If the robot encounters a real-world object in which there is no 3D model available, a placeholder is automatically generated using the perceived shape and color of the object. Such point can be seen on Figure 8, where the simulated door uses a 3D mesh, while the vending-machine and the beverage storage are represented by automatically generated shapes.

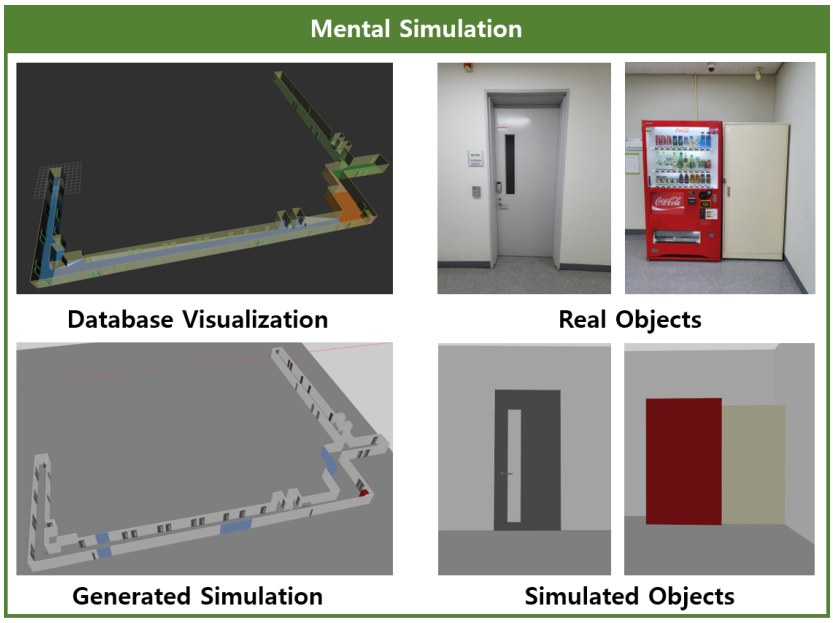


Figure 8. Example environment generated by the mental simulation building algorithm.

3.3.5. Reinforcement Learning

The main usage of the mental simulation is to generate experience tuples $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{S}', \mathcal{D})$, where $\mathcal{S}$ is the current state, $\mathcal{A}$ is the chosen action, $\mathcal{R}$ is the reward and $\mathcal{S}'$ the next state after performing $\mathcal{A}$, and $\mathcal{D}$ represents whether or not the episode has ended. Those simulated episodic memories are generated whenever the robot is idle, such as when charging. Those memories are then sampled in batches to train a policy using the DDPG algorithm [90]. The RL structure is shown on Figure 9. Three consecutive sparse laser scans and relative distances to the goal are concatenated as the algorithm input. This input goes through an actor-network which outputs two different values. A *sigmoid* function is used to encode the robot forward speed between 0 and 1, while a *tanh* function encodes the angular speed between –1 and 1. Those outputs are concatenated with the input and served to the critic-network, which outputs the Q-value (i.e., the expected future reward) for the action chosen on this given state. Finally, the environment rewards are defined as

$$\begin{cases} r_{completion} - t * r_{time}, & \text{when arriving to the goal at time t,} \\ r_{closer}, & \text{if getting closer to the goal,} \\ -r_{collision}, & \text{if too close to an obstacle,} \end{cases}$$

where $r_{completion}$, r_{time}, r_{closer} and $r_{collision}$ are positive integers defined trivially.

As the training step only needs to sample the experience tuples, it can be performed either on-board or by using a central cloud service, provided that such structure is accessible by the robot.

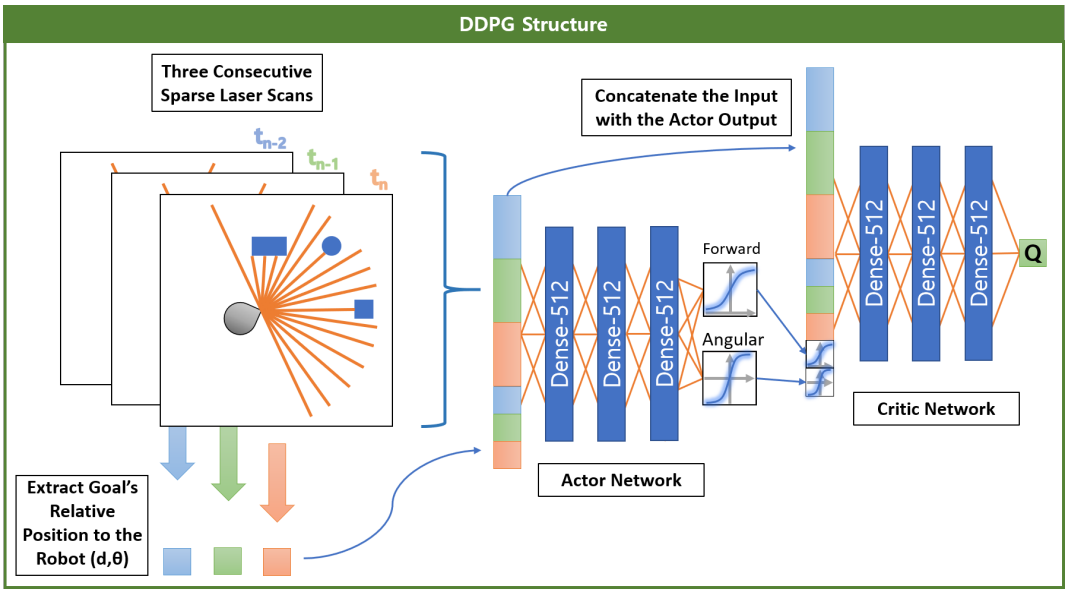


Figure 9. The Deep Deterministic Policy Gradients (DDPG) structure used to approximate an autonomous navigation policy.

4. Experiments

In this section, we present our experiments using the TOSM-based autonomous navigation framework. The objective of the experiments is to demonstrate the framework's validity and introduce a possible use case.

4.1. Experimental Environments

We conducted experiments in the convention center environment expressed in Figure 10 to apply the proposed framework. The specified working environment was a size of 5000 m^2 with dynamic pedestrians. In Figure 10a, a red circle represents the robot's home position, and a translucent red polygon represents the area where the robot performed the mission. Additionally, Figure 10b,c respectively represent the home position and the working area expressed in Figure 10a. The robot used Stereolabs' ZED camera and Velodyne's VLP16 to recognize the environment. The SIP and SAN modules of the framework were operated in a ROS Kinetic with Ubuntu 16.04 environment on Intel Zotac NUC En1070 with i5-6400T@2.2 GHz.

(**a**) Floor plan. (**b**) Home position. (**c**) Working area.

Figure 10. Experimental environment.

The environmental database, illustrated in Figure 11, for the first floor in the convention center consisted of 23 instances from three subclasses of *Object* and 16 instances from nine subclasses of *Place*. Although there were many objects on the first floor, we selected only static objects that can be worked as dominant features for recognizing environments; We defined this property as *isKeyObject* in the implicit

model for objects. In the case of objects which were not key objects (e.g., pedestrian), we considered the objects only in the short term memory for the efficiency of memory consuming. By doing so, the database consumed memory less than 0.1 MB in size, although it covered the large environments.

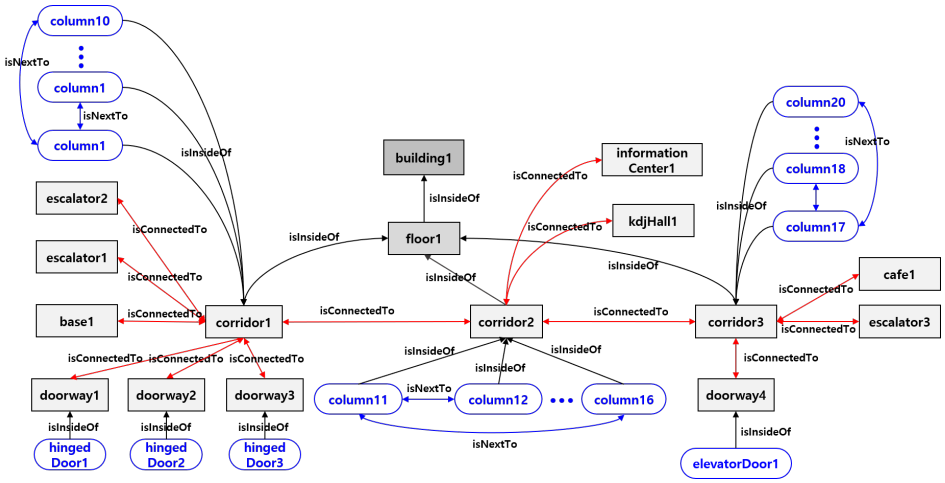


Figure 11. The TOSM-based database for experiments.

4.2. Experimental Sequences

Figure 12 illustrates how the robot performed its mission. First, the robot, located at the *corridor1*, received a mission from an administrator to inspect *kdjHall1* on the first floor of the convention center and come back *base1*. Then the SMF, shown in a blue box, generated an on-demand database that containd only the database needed in the first-floor database, considering the robot's characteristics and missions. In addition to those defined in Table 1, classes expressed in the database were used by adding object and space classes, such as *Column* and *InformationCenter*, in consideration of the convention center environment. The SMF also built and saved an occupancy map for estimating the robot's pose and generating waypoints.

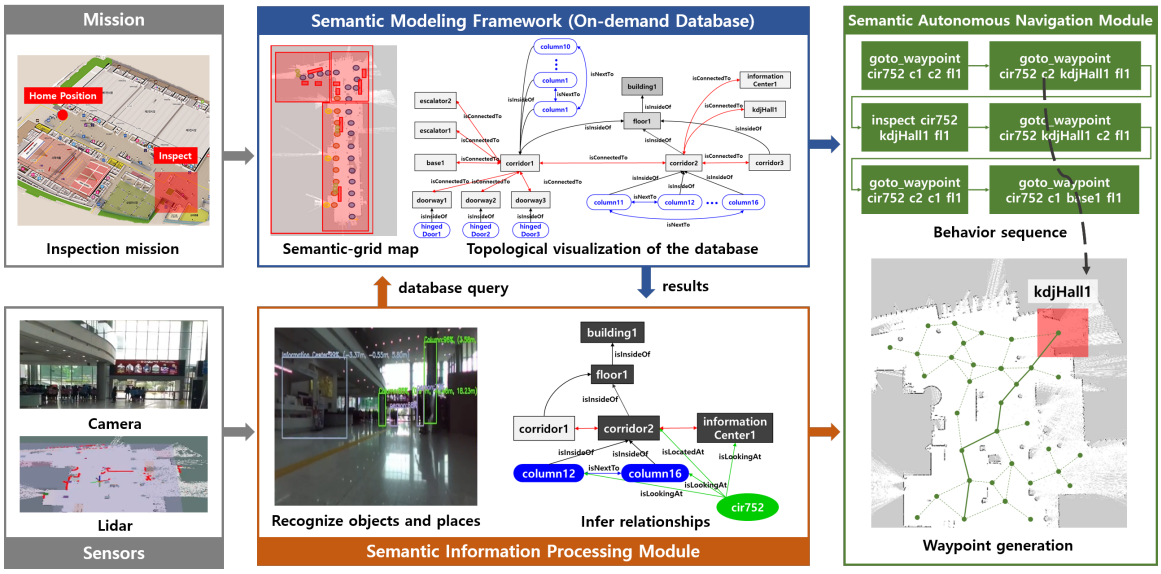


Figure 12. The TOSM-based autonomous navigation framework sequence diagram.

Second, the SAN module set goals as "inspected *kdjHall1*" and "robot_at *cir752 base1*" to divide the mission into six modular behaviors using the ROSPlan, which moved the robot to the working area, realized the inspection, and returned to the robot base; each behavior was described on Listings 1 and 3.

Each waypoint following behavior can be expanded, as shown in Figure 12, where the PRM algorithm used randomly sampled points to generate a 2D graph. This graph was traversed to generate the shortest path to the goal. Afterward, each desired point in the graph was sent sequentially to the Action planner, which then used the trained policy to navigate safely.

Listing 3. Inspect action.

```
(:action inspect
:parameters (?v - robot ?wp - waypoint ?fl - floor)
:precondition (and
(robot_at ?v ?wp)
(waypoint_at_floor ?wp ?fl)
)
:effect (inspected ?wp)
)
```

Figure 12 describes the moment of executing second behavior "goto_waypoint *cir*752 *c*2 *kdjHall*1 *fl*1", which meant *cir*752 should go to *kdjHall*1 from *c*2 inside *floor*1. When the robot started executing the behavior, it got sensory inputs to recognize the environment. The SIP module recognized objects and places using sensory inputs based on the CNN model with semantic knowledge in the on-demand database. In this case, it could infer "*cir*752 *isLookingAt column*12, *column*16, and *informationCenter*1", "*cir*752 *isLocatedAt corridor*2 that *isInsideOf floor*1", and "*column*12 *isNextTo column*16". In addition, it got datatype properties of each instance. Finally, if the robot executed the final behavior successfully, which was goto_waypoint *base*1, it accomplished the mission.

4.3. Experimental Results

In this section, we evaluate the performances of the PRM-based behavior planner and the action planner to demonstrate the validation of our framework.

4.3.1. Behavior Planner

One of the key hyper-parameters of the PRM algorithm is the number of nodes n. We conducted experiments aiming to show the influence of this value on the algorithm performance. Initially, four differently sized places were selected, as shown in Table 6. Using each of the places' grid map, we calculated the map total area by multiplying the map's height by the map's width in meters. To calculate the *free space ratio* value, we divided the amount of pixels representing a free space on the map image by the total amount of pixels. Finally, we multiplied the free space ratio by the total map area to generate the *free space area*.

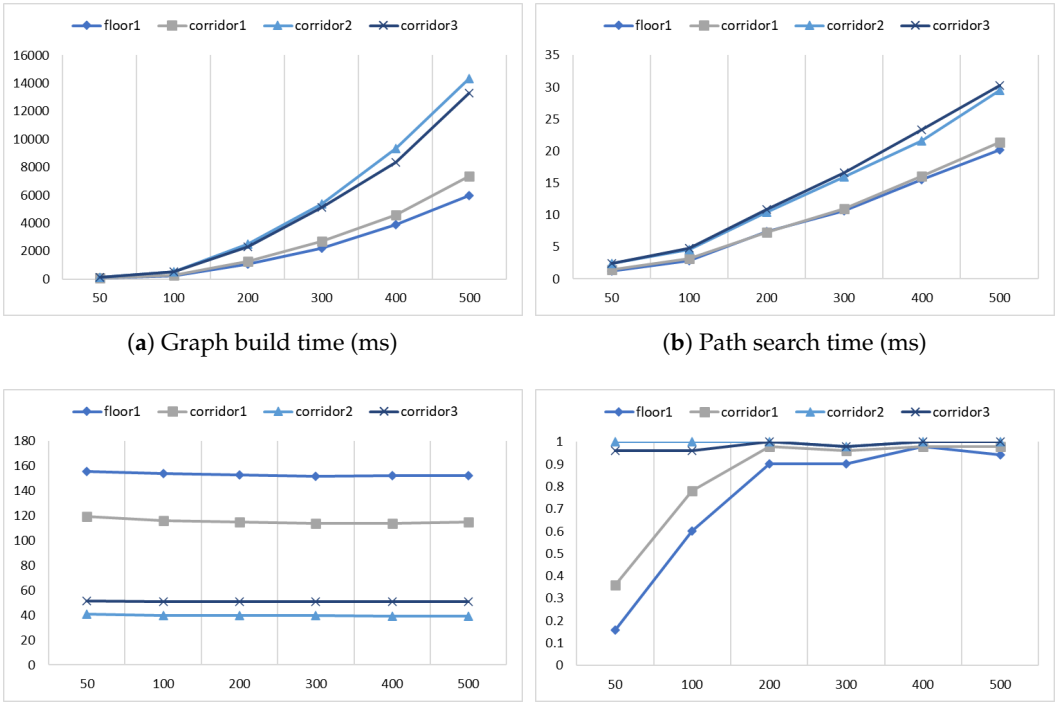
Table 6. Map information.

Place Name	Place Total Mapped Area (m^2)	Free Space Ratio	Free Space Area (m^2)
floor1	39,759.4	0.123	4890.41
corridor1	13,632.0	0.258	3517.06
corridor2	4672.0	0.391	1826.75
corridor3	3124.0	0.271	846.60

We chose a set of the number of samples $\{50, 100, 200, 300, 400, 500\}$ heuristically, and ran our PRM algorithm 50 times for each n value on each map. The most important performance information for our application was the time taken to build the PRM graph, the time taken to find the optimal path, the length of the final path, and whether or not the algorithm was able to find a path between $\mathcal{N}_p$ and $\mathcal{N}_g$. The average values for each of those measurements are shown in Figure 13. While the time taken to build the PRM graph grew quadratically with the number of samples, the time taken to

find the optimal path only grew linearly. As our approach generated the graph only when receiving a new map, the effect of building the PRM graph can be mitigated. Smaller places showed higher times for both, graph building and path finding, when using the same amount of samples as larger places, because it became progressively harder for the algorithm to find free spaces when the map was nearly filled with samples. In addition, graphs with a high density of nodes per square meter tended to have a higher variety of possible paths, which could slow down the path search.

Additionally, we were able to show that the number of samples barely influenced the final path length. However, it had a considerable influence on the ability to find a valid path. When the amount of nodes was too small while comparing to the map size, some nodes could not find valid neighbors, which led to disconnected graphs. This situation is not desirable when performing a mission, as it would require the robot to re-generate the PRM graph whenever encountering such an issue. This can lead to unexpected delays and even dead-locks.



(**a**) Graph build time (ms)

(**b**) Path search time (ms)

(**c**) Path length (m)

(**d**) Success rate

Figure 13. Influence of the number of samples (x-axis) on the PRM algorithm.

Aiming to select an optimal number of nodes based on the free space information, we performed a linear regression while trying to minimize n with two constraints: the path search time should take no more than 15 ms; the success rate should be above 90%. We obtained the following coefficients:

$$NumberOfSamples = ceil(0.057 * FreeSpaceArea + 30), \quad (1)$$

where *ceil* represents the rounding up operation. By using Equation (1), the number of samples needed by each one of the aforementioned maps is displayed in Table 7. We ran the algorithm 50 more times, this time using the optimal n values. It has been shown that by using the proposed formula, both constraints were satisfied.

Table 7. Optimized the number of samples.

Place Name	Number of Samples	Graph Build Time (ms)	Path Search Time (ms)	Path Length (m)	Success Rate
floor1	309	2489.4	11.24	152.63	92%
corridor1	231	1757.75	8.77	114.36	92%
corridor2	135	1128.89	7.09	39.42	100%
corridor3	79	375.62	4.39	50.92	92%

4.3.2. Action Planner

The policy training described in Section 3.3.5 was performed using the simulation shown in Figure 8 generated by the mental simulation building algorithm. We used an experience database with a maximum size of 1 million tuples and learned on batches of 512. The learning rate used was 10^{-6}, and the critic network decay value was 0.95. Finally, the rewards were chosen as follows: $r_{completion} = 1000$, $r_{time} = 1$, $r_{collision} = 500$, and r_{closer} were rewarded sparsely when the robot travelled 25%, 50% and 75% of the path, giving a reward proportional to $r_{completion}$. We trained the robot for 30,000 episodes (approximately 1.3 million steps). The first 5000 episodes used a purely random policy in order to generate a large variety of experiences. The rewards obtained on each episode can be seen in Figure 14. It can be seen that until episode 15,000, the robot was stagnated, then learned a better policy and started exploiting these actions, which lead to higher rewards. After 22,000 steps, the leaning plateaued again.

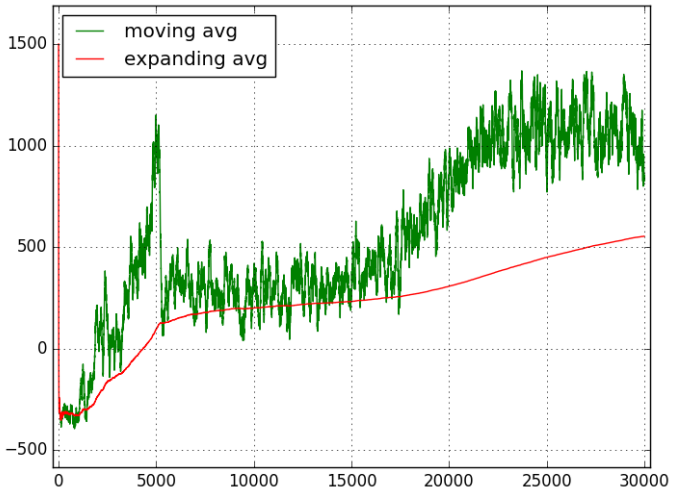
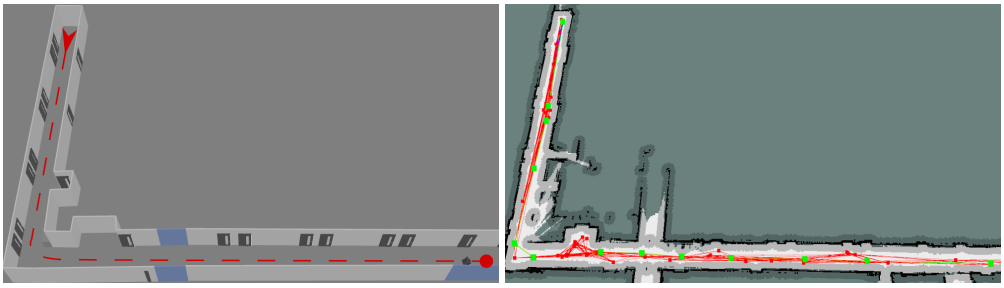


Figure 14. Reward obtained by the robot on each training episode. The line in green represents a moving average using a 100 episodes window. The line in red shows the cumulative average until the current episode.

To demonstrate the advantage of our hybrid approach when compared to pure reinforcement learning navigation methods, we performed the experiment shown in Figure 15. The robot started on the red arrow, and its goal was to navigate until the red circle by avoiding collision with any obstacle within 10 min. The simulated environment was generated by the mental simulation building algorithm. It was based on a narrow corridor sequence inside a university building. The total path size was 65.6 m. We performed two different experiments ten times each. The first one was using only the policy learned to navigate the robot directly to the goal. The second one used the PRM algorithm to generate a waypoint sequence as the one displayed in Figure 15b, which would be used as goals for the navigation policy. The results of this experiment are summarized in Table 8. It can be seen that the hybrid approach arrived at the goal 40% faster than the navigation done purely by using the learned policy. When the goal was too far away, the robot kept turning itself to the goal point,

going in the direction of a wall, then turning back to avoid a collision. This behavior slowed down the navigation and was less predictable than the path followed by the PRM-RL approach. Both methods had two failures. The RL method timed out after going back and forth in the first corridor. This failure happened depending on how close the robot was to the trashcan when approaching the corner. If the robot was too close, the laser scanner would not be able to see the second corridor's entrance, believe it was a dead-end, and turn back. In the case of the PRM-RL approach, the failures were caused because the PRM algorithm created a sample too close to trashcan, which led to a collision. We believe this could be solved by using a more robust sampling algorithm than the one currently used.



(**a**) Navigation Goal. (**b**) Generated Waypoint Sequence.

Figure 15. Autonomous navigation experiment.

Table 8. Comparison between RL and PRM-RL navigation.

Navigation Method	Average Time to Complete	Success Rate
RL	8m47s	80%
PRM-RL	5m20s	80%

The experimental results show that our hybrid approach added an extra layer of robustness to autonomous navigation based on reinforcement learning policies. It was able to perform a navigation task 40% faster than the baseline while maintaining the success rate.

5. Conclusions

In this paper, we have introduced our cognitive science inspired navigation framework and its three composing parts: semantic modeling framework (SMF), semantic information processing (SIP) module, and semantic autonomous navigation (SAN) module. We also showed how we modeled the environment employing the triplet ontological semantic model (TOSM), which allowed the three modules mentioned above to utilize the same modeled data. The on-demand database allowed different robots to share a common database by selectively querying the environment data based on each robot's intrinsic capabilities. The SIP module, performed object detection and localization using the deep learning-based approach, while, inferred the robot's relationship information based on the semantic properties of recognized objects derived from the on-demand database. Moreover, the SAN structure showed that a hybrid planning approach, which combines classical and sampling-based planners with reinforcement learning algorithms, is able to mitigate each planner's shortcomings leading to a more robust planning algorithm. The performed experiment showed that our framework can be applied to real-world scenarios, and navigation tasks on large dynamic environments can be performed seamlessly with the aid of TOSM data.

Immediate future work is focused on expanding the framework to other environments, more specifically outdoor environments. A modeled outdoor environment can also be used on different platforms, such as self-driving vehicles. Moreover, a large variety of TOSM semantic information shall be used by the SIP module when performing object recognition and localization. In addition, we are

planning to add a large variety of possible behaviors to enable higher-level missions to be performed. A more robust sampling algorithm can also be used to improve both the graph build time and the success rate of navigation tasks. Finally, another line of research is using the mental simulation to check the plan validity before its execution. For example, the feasibility of following a sequence of points generated by the PRM algorithm can be tested on simulation. In case of failure, a new sampled graph can be generated.

Author Contributions: Conceptualization, S.-H.J., S.M., Y.G.R., and T.-Y.K.; methodology, S.-H.J., S.M., and Y.G.R.; software, S.-H.J., S.M., and Y.G.R.; validation, S.-H.J.; formal analysis, S.-H.J.; investigation, S.-H.B. and K.-H.L.; data curation, S.-H.B. and K.-H.L.; writing—original draft preparation, S.-H.J., S.M., and Y.G.R.; writing—review and editing, S.-H.J., S.M., and M.K.; visualization, S.-H.J. and S.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Korea Evaluation Institute of Industrial Technology (KEIT) funded by the Ministry of Trade, Industry and Energy (MOTIE) (No. 1415162366).

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

References

1. Lowry, S.; Sünderhauf, N.; Newman, P.; Leonard, J.J.; Cox, D.; Corke, P.; Milford, M.J. Visual place recognition: A survey. *IEEE Trans. Robot.* **2015**, *32*, 1–19. [CrossRef]
2. Alterovitz, R.; Koenig, S.; Likhachev, M. Robot planning in the real world: Research challenges and opportunities. *Ai Mag.* **2016**, *37*, 76–84. [CrossRef]
3. Pendleton, S.D.; Andersen, H.; Du, X.; Shen, X.; Meghjani, M.; Eng, Y.H.; Rus, D.; Ang, M.H. Perception, planning, control, and coordination for autonomous vehicles. *Machines* **2017**, *5*, 6. [CrossRef]
4. Gayathri, R.; Uma, V. Ontology based knowledge representation technique, domain modeling languages and planners for robotic path planning: A survey. *ICT Express* **2018**, *4*, 69–74.
5. Nüchter, A.; Hertzberg, J. Towards semantic maps for mobile robots. *Robot. Auton. Syst.* **2008**, *56*, 915–926. [CrossRef]
6. Ruiz-Sarmiento, J.R.; Galindo, C.; Gonzalez-Jimenez, J. Exploiting semantic knowledge for robot object recognition. *Knowl.-Based Syst.* **2015**, *86*, 131–142. [CrossRef]
7. Galindo, C.; Fernández-Madrigal, J.A.; González, J.; Saffiotti, A. Using semantic information for improving efficiency of robot task planning. In Proceedings of the ICRA Workshop: Semantic Information in Robotics, Rome, Italy, 10 April 2007.
8. Abdi, L.; Meddeb, A. Semantic recognition: Unified framework for joint object detection and semantic segmentation. In Proceedings of the Symposium on Applied Computing, Marrakech, Morocco, 3–7 April 2017; pp. 83–88.
9. Horikawa, T.; Kamitani, Y. Generic decoding of seen and imagined objects using hierarchical visual features. *Nat. Commun.* **2017**, *8*, 1–15. [CrossRef]
10. Fox, M.; Long, D. PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *J. Artif. Intell. Res.* **2003**, *20*, 61–124. [CrossRef]
11. Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtos, N.; Carreras, M. Rosplan: Planning in the robot operating system. In Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, Jerusalem, Israel, 7–11 June 2015.
12. 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 Syst. Hum.* **2010**, *41*, 492–509. [CrossRef]
13. Weibel, M.; Beetz, M.; DAndrea, R.; Janssen, R.; Tenorth, M.; Civera, J.; Elfring, J.; Gávez-Lopez, D.; Häussermann, K.; Montiel, J.; et al. RoboEarth-A world wide web for robots. *Robot. Autom. Mag.* **2011**, *18*, 69–82.
14. Joo, S.H.; Manzoor, S.; Rocha, Y.G.; Lee, H.U.; Kuc, T.Y. A realtime autonomous robot navigation framework for human like high-level interaction and task planning in global dynamic environment. *arXiv* **2019**, arXiv:1905.12942.
15. Schlenoff, C.; Prestes, E.; Madhavan, R.; Goncalves, P.; Li, H.; Balakirsky, S.; Kramer, T.; Miguelanez, E. An IEEE standard ontology for robotics and automation. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, 7–12 October 2012; pp. 1337–1342.

16. Suh, I.H.; Lim, G.H.; Hwang, W.; Suh, H.; Choi, J.H.; Park, Y.T. Ontology-based multi-layered robot knowledge framework (OMRKF) for robot intelligence. In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007; pp. 429–436.
17. Bratko, I. *Prolog Programming for Artificial Intelligence*; Pearson Education: Toronto, ON, Canada, 2011.
18. Tenorth, M.; Perzylo, A.C.; Lafrenz, R.; Beetz, M. Representation and exchange of knowledge about actions, objects, and environments in the roboearth framework. *IEEE Trans. Autom. Sci. Eng.* **2013**, *10*, 643–651. [CrossRef]
19. Tenorth, M.; Beetz, M. KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *Int. J. Robot. Res.* **2013**, *32*, 566–590. [CrossRef]
20. Thrun, S.; Montemerlo, M. The graph SLAM algorithm with applications to large-scale mapping of urban structures. *Int. J. Robot. Res.* **2006**, *25*, 403–429. [CrossRef]
21. Grisetti, G.; Stachniss, C.; Burgard, W. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Trans. Robot.* **2007**, *23*, 34–46. [CrossRef]
22. Murphy, K.; Russell, S. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Sequential Monte Carlo Methods in Practice*; Springer: New York, NY, USA, 2001; pp. 499–515.
23. Zhang, J.; Singh, S. LOAM: Lidar Odometry and Mapping in Real-time. In Proceedings of the Robotics: Science and Systems Conference, Berkeley, CA, USA, 12–16 July 2014.
24. Klein, G.; Murray, D. Parallel tracking and mapping for small AR workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 13–16 November 2007; pp. 225–234.
25. Mur-Artal, R.; Montiel, J.M.M.; Tardos, J.D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
26. Pronobis, A.; Jensfelt, P. Large-scale semantic mapping and reasoning with heterogeneous modalities. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 3515–3522.
27. Kostavelis, I.; Gasteratos, A. Semantic mapping for mobile robotics tasks: A survey. *Robot. Auton. Syst.* **2015**, *66*, 86–103. [CrossRef]
28. McCormac, J.; Handa, A.; Davison, A.; Leutenegger, S. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In Proceedings of the 2017 IEEE International Conference on Robotics and automation (ICRA), Singapore, Singapore, 29 May–3 June 2017; pp. 4628–4635.
29. Yang, S.; Huang, Y.; Scherer, S. Semantic 3D occupancy mapping through efficient high order CRFs. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 590–597.
30. Nakajima, Y.; Tateno, K.; Tombari, F.; Saito, H. Fast and accurate semantic mapping through geometric-based incremental segmentation. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 385–392.
31. Xiao, L.; Wang, J.; Qiu, X.; Rong, Z.; Zou, X. Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment. *Robot. Auton. Syst.* **2019**, *117*, 1–16. [CrossRef]
32. Riazuelo, L.; Tenorth, M.; Di Marco, D.; Salas, M.; Gálvez-López, D.; Mösenlechner, L.; Kunze, L.; Beetz, M.; Tardós, J.D.; Montano, L.; et al. RoboEarth semantic mapping: A cloud enabled knowledge-based approach. *IEEE Trans. Autom. Sci. Eng.* **2015**, *12*, 432–443. [CrossRef]
33. Tenorth, M.; Perzylo, A.C.; Lafrenz, R.; Beetz, M. The roboearth language: Representing and exchanging knowledge about actions, objects, and environments. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 1284–1289.
34. Johnston, B.; Yang, F.; Mendoza, R.; Chen, X.; Williams, M.A. Ontology based object categorization for robots. In *International Conference on Practical Aspects of Knowledge Management*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 219–231.
35. 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.
36. Crespo, J.; Barber, R.; Mozos, O. Relational model for robotic semantic navigation in indoor environments. *J. Intell. Robot. Syst.* **2017**, *86*, 617–639. [CrossRef]

37. Jiao, L.; Zhang, F.; Liu, F.; Yang, S.; Li, L.; Feng, Z.; Qu, R. A Survey of Deep Learning-Based Object Detection. *IEEE Access* **2019**, *7*, 128837–128868. [CrossRef]
38. Zou, Z.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *arXiv* **2019**, arXiv:1905.05055.
39. Roldan, S.M. Object recognition in mental representations: Directions for exploring diagnostic features through visual mental imagery. *Front. Psychol.* **2017**, *8*, 833. [CrossRef] [PubMed]
40. Luo, Y.; Gao, Y.; Liu, L.; Huang, X. A novel object recognition system for cognitive robot. In Proceedings of the 2012 IEEE International Conference on Information and Automation, Shenyang, China, 6–8 June 2012; pp. 680–685.
41. Cichy, R.M.; Kaiser, D. Deep neural networks as scientific models. *Trends Cogn. Sci.* **2019**, *23*, 305–317. [CrossRef]
42. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
43. Weiss, I. A Dual-hierarchy Semantic Graph for Robust Object Recognition. *arXiv* **2019**, arXiv:1909.06867.
44. Zhang, H.; Xu, T.; Elhoseiny, M.; Huang, X.; Zhang, S.; Elgammal, A.; Metaxas, D. Spda-cnn: Unifying semantic part detection and abstraction for fine-grained recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 1143–1152.
45. Wang, J.; Xie, C.; Zhang, Z.; Zhu, J.; Xie, L.; Yuille, A. Detecting semantic parts on partially occluded objects. *arXiv* **2017**, arXiv:1707.07819.
46. Choi, J.H.; Park, Y.T.; Lim, G.H.; Lee, S. Ontology-Based Semantic Context Modeling for Object Recognition of Intelligent Mobile Robots. In *Recent Progress in Robotics: Viable Robotic Service to Human*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 399–408.
47. Maillot, N.E.; Thonnat, M. Ontology based complex object recognition. *Image Vis. Comput.* **2008**, *26*, 102–113. [CrossRef]
48. Allani, O.; Zghal, H.B.; Mellouli, N.; Akdag, H. A knowledge-based image retrieval system integrating semantic and visual features. *Procedia Comput. Sci.* **2016**, *96*, 1428–1436. [CrossRef]
49. Lee, S.; Naguib, A.M.; Islam, N.U. 3D deep object recognition and semantic understanding for visually-guided robotic service. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 903–910.
50. Chen, Y.; Gan, W.; Zhang, L.; Liu, C.; Wang, X. A Survey on Visual Place Recognition for Mobile Robots Localization. In Proceedings of the 2017 14th Web Information Systems and Applications Conference (WISA), Liuzhou, China, 11–12 November 2017; pp. 187–192.
51. Schönberger, J.L.; Pollefeys, M.; Geiger, A.; Sattler, T. Semantic visual localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 6896–6906.
52. Garg, S.; Suenderhauf, N.; Milford, M. Lost? appearance-invariant place recognition for opposite viewpoints using visual semantics. *arXiv* **2018**, arXiv:1804.05526.
53. Chen, Z.; Jacobson, A.; Sünderhauf, N.; Upcroft, B.; Liu, L.; Shen, C.; Reid, I.; Milford, M. Deep learning features at scale for visual place recognition. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, Singapore, 29 May–3 June 2017; pp. 3223–3230.
54. Gomez-Ojeda, R.; Lopez-Antequera, M.; Petkov, N.; Gonzalez-Jimenez, J. Training a convolutional neural network for appearance-invariant place recognition. *arXiv* **2015**, arXiv:1505.07428.
55. Mitsuhashi, M.; Kuroda, Y. Mobile robot localization using place recognition in outdoor environments with similar scenes. In Proceedings of the 2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)E, Budapest, Hungary, 3–7 July 2011; pp. 930–935.
56. McManus, C.; Upcroft, B.; Newman, P. Learning place-dependant features for long-term vision-based localisation. *Auton. Robot.* **2015**, *39*, 363–387. [CrossRef]
57. Zhu, J.; Li, Q.; Cao, R.; Sun, K.; Liu, T.; Garibaldi, J.M.; Li, Q.; Liu, B.; Qiu, G. Indoor topological localization using a visual landmark sequence. *Remote. Sens.* **2019**, *11*, 73. [CrossRef]
58. Zhong, X.; Zhou, Y.; Liu, H. Design and recognition of artificial landmarks for reliable indoor self-localization of mobile robots. *Int. J. Adv. Robot. Syst.* **2017**, *14*, 1729881417693489. [CrossRef]
59. Simmons, R.; Goodwin, R.; Haigh, K.Z.; Koenig, S.; O'Sullivan, J. A layered architecture for office delivery robots. In Proceedings of the first international conference on Autonomous Agents, Marina del Rey, CA, USA, 5–8 February 1997; pp. 245–252.

60. Choset, H.M.; Hutchinson, S.; Lynch, K.M.; Kantor, G.; Burgard, W.; Kavraki, L.E.; Thrun, S. *Principles of Robot Motion: Theory, Algorithms, and Implementation*; MIT Press: Boston, MA, USA, 2005.
61. Buehler, M.; Iagnemma, K.; Singh, S. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*; Springer: New York, NY, USA, 2009; Volume 56.
62. Mac, T.T.; Copot, C.; Tran, D.T.; De Keyser, R. Heuristic approaches in robot path planning: A survey. *Robot. Auton. Syst.* **2016**, *86*, 13–28. [CrossRef]
63. Muñoz, P.; R-Moreno, M.D.; Barrero, D.F. Unified framework for path-planning and task-planning for autonomous robots. *Robot. Auton. Syst.* **2016**, *82*, 1–14. [CrossRef]
64. Fikes, R.E.; Nilsson, N.J. STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.* **1971**, *2*, 189–208. [CrossRef]
65. Nilsson, N.J. *Shakey the Robot*; Technical Report; SRI International: Menlo Park, CA, USA, 1984.
66. Estivill-Castro, V.; Ferrer-Mestres, J. Path-finding in dynamic environments with PDDL-planners. In Proceedings of the 2013 16th International Conference on Advanced Robotics, ICAR 2013, Montevideo, Uruguay, 25–29 November 2013; pp. 1–7. [CrossRef]
67. Dvorak, F.; Bit-Monnot, A.; Ingrand, F.; Ghallab, M. *A Flexible ANML Actor and Planner in Robotics*; In Proceedings of the Planning and Robotics (PlanRob) Workshop (ICAPS), Portsmouth, NH, USA, 21–26 June 2014.
68. Smith, D.E.; Frank, J.; Cushing, W. The ANML language. In Proceedings of the ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS), Sydney, Australia, 14–18 September 2008.
69. Boutilier, C.; Dearden, R. Using abstractions for decision-theoretic planning with time constraints. In Proceedings of the AAAI, Seattle, WA, USA, 31 July–4 August 1994; pp. 1016–1022.
70. Dean, T.L.; Kaelbling, L.P.; Kirman, J.; Nicholson, A.E. Planning With Deadlines in Stochastic Domains. In Proceedings of the AAAI, Washington, DC, USA, 11–15 July 1993; Volume 93; pp. 574–579.
71. Galindo, C.; Fernández-Madrigal, J.A.; González, J.; Saffiotti, A. Robot task planning using semantic maps. *Robot. Auton. Syst.* **2008**, *56*, 955–966. [CrossRef]
72. 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, Australia, 21–25 May 2018; pp. 512–519.
73. Beßler, D.; Pomarlan, M.; Beetz, M. Owl-enabled assembly planning for robotic agents. In Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, Stockholm, Sweden, 10–15 July 2018; Volume 3, pp. 1684–1692.
74. Lau, M.; Kuffner, J.J. Behavior planning for character animation. In Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Los Angeles, CA, USA, 29–31 July 2005; pp. 271–280.
75. Koo, T.; Hoffmann, F.; Shim, H.; Sinopoli, B.; Sastry, S. Hybrid control of an autonomous helicopter. In Proceedings of the IFAC Workshop on Motion Control, Grenoble, France, 21–23 September 1998; pp. 285–290.
76. Shi, X.; Wang, F.Y.; Lever, P. Experimental results of robotic excavation using fuzzy behavior control. *Control. Eng. Pract.* **1996**, *4*, 145–152. [CrossRef]
77. Wei, J.; Snider, J.M.; Gu, T.; Dolan, J.M.; Litkouhi, B. A behavioral planning framework for autonomous driving. In Proceedings of the 2014 IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, USA, 8–11 June 2014; pp. 458–464.
78. Arney, T. An efficient solution to autonomous path planning by approximate cell decomposition. In Proceedings of the 2007 Third International Conference on Information and Automation for Sustainability, Melbourne, Australia, 4–6 December 2007; pp. 88–93.
79. Lingelbach, F. Path planning using probabilistic cell decomposition. In Proceedings of the IEEE International Conference on Robotics and Automation, 2004. Proceedings, ICRA'04, New Orleans, LA, USA, 26 April–1 May 2004; Volume 1, pp. 467–472.
80. Šeda, M. Roadmap methods vs. cell decomposition in robot motion planning. In Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation, Corfu Island, Greece, 16–19 February 2007; pp. 127–132.

81. Pradhan, S.K.; Parhi, D.R.; Panda, A.K.; Behera, R.K. Potential field method to navigate several mobile robots. *Appl. Intell.* **2006**, *25*, 321–333. [CrossRef]
82. Kang, Y.H.; Lee, M.C.; Kim, C.Y.; Yoon, S.M.; Noh, C.B. A study of cluster robots line formatted navigation using potential field method. In Proceedings of the 2011 IEEE International Conference on Mechatronics and Automation, Beijing, China, 7–10 August 2011; pp. 1723–1728.
83. Lee, J.; Kwon, O.; Zhang, L.; Yoon, S.E. A selective retraction-based RRT planner for various environments. *IEEE Trans. Robot.* **2014**, *30*, 1002–1011. [CrossRef]
84. Kavraki, L.; Latombe, J.C. Randomized preprocessing of configuration for fast path planning. In Proceedings of the 1994 IEEE International Conference on Robotics and Automation, San Diego, CA, USA, 8–13 May 1994; pp. 2138–2145.
85. LaValle, S.M. *Rapidly-Exploring Random trees: A New Tool for Path Planning*; Research Report 98-11; Department of Computer Science, Iowa State University: Ames, IA, USA, 1998.
86. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894. [CrossRef]
87. Elbanhawi, M.; Simic, M. Sampling-based robot motion planning: A review. *IEEE Access* **2014**, *2*, 56–77. [CrossRef]
88. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
89. Shah, P.; Fiser, M.; Faust, A.; Kew, J.C.; Hakkani-Tur, D. Followmet: Robot navigation by following natural language directions with deep reinforcement learning. *arXiv* **2018**, arXiv:1805.06150.
90. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
91. Tai, L.; Paolo, G.; Liu, M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 31–36.
92. Faust, A.; Oslund, K.; Ramirez, O.; Francis, A.; Tapia, L.; Fiser, M.; Davidson, J. PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 5113–5120.
93. Gabrieli, J.D. Cognitive neuroscience of human memory. *Annu. Rev. Psychol.* **1998**, *49*, 87–115. [CrossRef]
94. 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**, *10.02*.
95. Rocha, Y.G.; Joo, S.H.; Kim, E.J.; Kuc, T.Y. Automatic Generation of a Simulated Robot from an Ontology-Based Semantic Description. In Proceedings of the 2019 19th International Conference on Control, Automation and Systems (ICCAS), Jeju, Korea, 15–18 October 2019; pp. 1340–1343. [CrossRef]
96. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
97. Coles, A.J.; Coles, A.I.; Fox, M.; Long, D. Forward-chaining partial-order planning. In Proceedings of the Twentieth International Conference on Automated Planning and Scheduling, Toronto, ON, Canada, 12–16 May 2010.
98. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271. [CrossRef]
99. Kahneman, D.; Tversky, A. *The Simulation Heuristic*; Technical Report; Department of Psychology, Stanford University: Stanford, CA, USA, 1981.
100. Rocha, Y.G.; Kuc, T.Y. Mental simulation for autonomous learning and planning based on triplet ontological semantic model. *CEUR Workshop Proc.* **2019**, *2487*, 65–73.