THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (Toulouse INP)

Discipline ou spécialité :

Robotique et Informatique

Présentée et soutenue par :

M. YINGSHEN ZHAO

le lundi 17 juin 2019

Titre :

An ontology-based approach towards coupling task and path planning for the simulation of manipulation tasks

---

Ecole doctorale :

Systèmes (EDSYS)

Unité de recherche :

Laboratoire de Génie de Productions de l'ENIT (E.N.I.T-L.G.P.)

Directeur(s) de Thèse :

M. BERNARD ARCHIMEDE

M. PHILIPPE FILLATREAU

Rapporteurs :

M. ALBERTO IZAGUIRRE ALTUNA, UNIVERSITE DE MONDRAGON

Mme NADA MATTA, UNIVERSITE DE TECHNOLOGIE DE TROYES

Membre(s) du jury :

M. YACINE OUZROUT, UNIVERSITE LYON 2, Président

M. BERNARD ARCHIMEDE, ECOLE NATIONALE D'INGENIEUR DE TARBES, Membre

Mme VERONIQUE PERDEREAU, UNIVERSITE SORBONNE, Membre

M. MOHAMED HEDI KARRAY, ECOLE NATIONALE D'INGENIEUR DE TARBES, Membre

M. PHILIPPE FILLATREAU, ECOLE NATIONALE D'INGENIEUR DE TARBES, Membre

Acknowledgement

First of all, I would like to express my sincere gratitude and appreciation to my supervisors: Mr. Bernard ARCHIMEDE, Mr. Philippe FILLATREAU, and Mr. Mohamed Hedi KARRAY. Their excellent patient guidance supports me all along with my research work. I have learned a lot during these three-years from their professional knowledge and their rigorous attitude in scientific research. I was impressed by the rigorous logic mind of Mr. Philippe FILLATREAU towards the clear scientific research work. I learned from him the way to be curious and innovative to solve scientific issues. The passion and the diligence of Mr. Mohamed Hedi KARRY influenced and touched me a lot. The wisdom and the humor of Mr. Bernard ARCHIMEDE taught me to be optimistic not only during my research work but also during my whole life. I really appreciated them to give me the opportunity of having such an impressive and extraordinary experience.

Secondly, I would like to thank my wife: MING Lu. We met in the laboratory where I do my Ph.D. research work. Her optimism encourages me to be brave and to be enterprising. I love her deep in my heart. Moreover, I would also want to appreciate my family. My father and my mother gave me incalculable love and support during my whole life. Their encouragement has always been the source of my strength and faith.

Last but not least, I would like to appreciate all the mates I met during my Ph.D. career: LIU Quan, WANG Qirong, XU Da, Linda ELMHADBI, and Lucky etc. I am so grateful to make friends with these amazing people and to have beautiful stories and memories with them that could be my precious wealth in the rest of my life. It is my honor to work with them.

Abstract

This work deals with the simulation and the validation of complex manipulation tasks under strong geometric constraints in virtual environments. The targeted applications relate to the industry 4.0 framework; as up-to-date products are more and more integrated and the economic competition increases, industrial companies express the need to validate, from design stage on, not only the static CAD models of their products but also the tasks (e.g., assembly or maintenance) related to their Product Lifecycle Management (PLM).

The scientific community looked at this issue from two points of view:

- Task planning decomposes a manipulation task to be realized into a sequence of primitive actions (i.e., a task plan)
- Path planning computes collision-free trajectories, notably for the manipulated objects. It traditionally uses purely geometric data, which leads to classical limitations (possible high computational processing times, low relevance of the proposed trajectory concerning the task to be performed, or failure); recent works have shown the interest of using higher abstraction level data.

Joint task and path planning approaches found in the literature usually perform a classical task planning step, and then check out the feasibility of path planning requests associated with the primitive actions of this task plan. The link between task and path planning has to be improved, notably because of the lack of loopback between the path planning level and the task planning level:

- The path planning information used to question the task plan is usually limited to the motion feasibility where richer information such as the relevance or the complexity of the proposed path would be needed;
- path planning queries traditionally use purely geometric data and/or “blind” path planning methods (e.g., RRT), and no task-related information is used at the path planning level

Our work focuses on using task level information at the path planning level. The path planning algorithm considered is RRT; we chose such a probabilistic algorithm because we consider path planning for the simulation and the validation of complex tasks under strong geometric constraints. We propose an ontology-based approach to use task level information to specify path planning queries for the primitive actions of a task plan.

First, we propose an ontology to conceptualize the knowledge about the 3D environment in which the simulated task takes place. The environment where the simulated task takes place is considered as a closed part of 3D Cartesian space cluttered with mobile/fixed obstacles (considered as rigid bodies). It is represented by a digital model relying on a multilayer architecture involving semantic, topologic and geometric data. The originality of the proposed ontology lies in the fact that it conceptualizes heterogeneous knowledge about both the obstacles and the free space models.

Second, we exploit this ontology to automatically generate a path planning query associated to each given primitive action of a task plan. Through a reasoning process involving the primitive actions instantiated in the ontology, we are able to infer the start and the goal configurations, as well as task-related geometric constraints. Finally, a multi-level path planner is called to generate the corresponding trajectory.

The contributions of this work have been validated by full simulation of several manipulation tasks under strong geometric constraints. The results obtained demonstrate that using task-related information allows better control on the RRT path planning algorithm involved to check the motion feasibility for the primitive actions of a task plan, leading to lower computational time and more relevant trajectories for primitive actions.

Résumé

Ce travail traite de la simulation et de la validation de tâches de manipulation complexes sous de fortes contraintes géométriques dans des environnements virtuels. Les applications visées sont liées au Framework industriel 4.0 ; à mesure que les produits s’intègrent de plus en plus et que la concurrence économique s'intensifie, les industriels expriment le besoin de valider, dès la conception, non seulement les modèles CAO statiques de leurs produits mais aussi les tâches (ex. : assemblage ou maintenance) liées à leur Product Lifecycle Management (PLM).

La communauté scientifique s'est penchée sur cette question sous deux angles :

- La planification des tâches décompose une tâche de manipulation à réaliser en une séquence d'actions primitives (c.-à-d. un plan de tâches).
- La planification de trajectoire calcule des trajectoires sans collision, notamment pour les objets manipulés. Elle utilise traditionnellement des données purement géométriques, ce qui conduit à des limitations classiques (temps de calcul élevé possible, faible pertinence de la trajectoire proposée par rapport à la tâche à exécuter, ou échec) ; des travaux récents ont montré l'intérêt d'utiliser des données de plus haut niveau d'abstraction.

Les approches conjointes de planification des tâches et des trajectoires que l'on trouve dans la littérature effectuent habituellement une étape classique de planification des tâches, puis vérifient la faisabilité des demandes de planification de trajectoire associées aux actions primitives de ce plan de tâches. Le lien entre la planification des tâches et la planification des trajectoires doit être amélioré, notamment en raison de l'absence de bouclage entre le niveau de planification des trajectoires et le niveau de planification des tâches :

- L'information de planification de trajectoire utilisée pour remettre en question le plan de tâches se limite habituellement à la faisabilité du mouvement lorsqu'une information plus riche telle que la pertinence ou la complexité de la trajectoire proposée serait nécessaire ;
- Les requêtes de planification de trajectoire utilisent traditionnellement des données purement géométriques et/ou des méthodes de planification de trajectoire "aveugles" (par exemple, RRT), et aucune information liée aux tâches n'est utilisée au niveau de la planification de trajectoire

Notre travail se concentre sur l'utilisation de l'information au niveau des tâches au niveau de la planification de la trajectoire. L'algorithme de planification de trajectoire RRT est considéré ;

nous avons choisi un tel algorithme probabiliste parce que nous considérons la planification de trajectoire pour la simulation et la validation de tâches complexes sous fortes contraintes géométriques. Nous proposons une approche basée sur l'ontologie pour utiliser les informations au niveau des tâches afin de spécifier les requêtes de planification de trajectoire pour les actions primitives d'un plan de tâches.

Tout d'abord, nous proposons une ontologie pour conceptualiser les connaissances sur l'environnement 3D dans lequel la tâche simulée se déroule. L'environnement dans lequel se déroule la tâche simulée est considéré comme une partie fermée de l'espace cartésien 3D encombré d'obstacles mobiles/fixes (considérés comme des corps rigides). Il est représenté par un modèle numérique s'appuyant sur une architecture multicouche de données sémantiques, topologiques et géométriques. L'originalité de l'ontologie proposée réside dans le fait qu'elle conceptualise des connaissances hétérogènes tant sur les obstacles que sur les modèles d'espace libre.

Deuxièmement, nous exploitons cette ontologie pour générer automatiquement une requête de planification de trajectoire associée à chaque action primitive donnée d'un plan de tâches. Grâce à un processus de raisonnement impliquant les actions primitives instanciées dans l'ontologie, nous sommes capables de déduire les configurations de départ et d'objectif, ainsi que les contraintes géométriques liées aux tâches. Enfin, un planificateur de trajet multi-niveaux est appelé pour générer la trajectoire correspondante.

Les contributions de ce travail ont été validées par la simulation complète de plusieurs tâches de manipulation sous de fortes contraintes géométriques. Les résultats obtenus démontrent que l'utilisation de l'information liée aux tâches permet un meilleur contrôle sur l'algorithme de planification de trajectoire RRT impliqué pour vérifier la faisabilité du mouvement des actions primitives d'un plan de tâches, ce qui entraîne une réduction du temps de calcul et des trajectoires plus pertinentes pour les actions primitives.

Table of content

List of figures

List of tables

Acronymes

2D	2-Dimension.
3D	3-Dimension.
BB	Building Block
BG	Behavioral Graph
B-Rep	Boundary Representation
B-spline	Basic Spline
CAD	Computer-aided Design
CSG	Constructive Solid Geometry
CSP	Constraint Satisfaction Problem
DL	Description Logics
FCL	Flexible Collision Library
FF	Fast-Forward
G	Geometry
GO	Geometry and Ontology
GT	Geometry and Topology
GTO	Geometry, Topology and Ontology
GTS	Geometry, Topology and Semantics
HTN	Hierarchical Task Network
IRI	Internationalized Resource Identifier
LGP	Laboratoire Génie de Production
NURBS	Non-uniform rational B-spline
OMRKF	Ontology-based multi-layered robot knowledge framework
OOP	Object-oriented programming
OWL	Web Ontology Language
OWLAPI	Web Ontology Language – Application Protocol Interface
PA	Primitive action
PAS	Primitive action Specification
PDDL	Planning Domain Description Language
PLM	Product Lifecycle Management

PP	Path planning
PPQD	Path planning query description
PRM	Probabilistic roadmap
RRT	Rapid exploring random tree
SIFT	Scale-invariant feature transform
SPARQL	SPARQL Protocol and RDF Query Language
STEP	STandard for Exchange of Product model data
SWRL	Semantic Web Rule Language
VR	Virtual Reality

Chapter 1 General Introduction

The increasingly competitive market and economic competition pushes harder and harder on industrial companies to reduce the time and the cost of new product development, whereas the up-to-date industrial products become more complex and integrated. Industry companies express the strong need to validate, from the design stage on, not only the static models of their products but also all tasks related to their Product Lifecycle Management (PLM). With the help of digital prototypes (e.g., Computer-aided design models), complex industrial operations under strong geometric constraints (e.g., assembly, disassembly or maintenance) can be simulated and validated before entering into the real product manufacturing. The abnormalities of the product design can be earlier detected before building physical prototypes.

In this dissertation, we are particularly interested in simulating and validating manipulation tasks under strong geometric constraints. The scientific community looked at this issue from two points of view:

- task planning decomposes a manipulation task to be realized into a feasible sequence of primitive actions (i.e., a task plan);
- path planning demonstrates the feasibility of manipulations by computing collision-free trajectories, notably for the manipulated objects. It traditionally uses purely geometric data, which leads to classical limitations (possibly high processing time, low path relevance regarding the task to be performed, or failure)

Task planning cannot verify the motion feasibility (i.e., finding feasible trajectories) of primitive actions of a task plan and path planning does not allow to reason at the task level. Therefore, solving a complex manipulation task requires to consider task planning and path planning jointly. Joint task and path planning approaches found in the literature usually perform a classical task planning step and check out the feasibility of path planning requests associated with the primitive actions of this task plan. However, the link between task planning and path planning has to be improved, notably because of the lack of loopback between task planning level and path planning level:

- path planning queries associated with primitive actions of a task plan traditionally use purely geometric data; the task level information has not yet been considered at the path planning level;

- the path planning related information used to question a task plan is usually limited to the motion feasibility of primitive actions of the task plan; richer path planning information would be needed to compute an optimal task plan, such as the relevance or the complexity of the proposed path.

This thesis work focuses on path planning for a primitive action of a task plan. It is motivated by the idea that using task-related information at the path planning level can lead to better path planning results (i.e., lower processing time, better path relevance regarding the task to be performed). Therefore, better simulations for the assistance of manipulations can be obtained. In order to generate a path planning query for a primitive action of a task plan, different kinds of task-related information should be considered:

- The environment information of a manipulation task to be performed: This information allows us to determine the start and the goal location of the path planning query.
- The information associated to a primitive action to be performed, for example, the spatial constraints given by a user restricting relative location between two objects involved in the primitive action: This information further allows us to determine the geometric constraints (associated to the path planning query) that restricts the movement of a manipulation object.

Indeed, the idea of using task-related information at the path planning level is difficult to be implemented, because the modeling of task-related information varies from one application to another. In order to build a path planning system capable of dealing with different applications, it is necessary to have an evolvable knowledge model of task-related information. Different levels of abstraction should be considered. Domain-independent information is commonly shared among applications, whereas application-specific information is updated regarding the task to be performed. Moreover, using such a knowledge model, the path planning system should be able to infer new information so that a relevant path planning query can be generated for a primitive action, e.g., such as finding the right goal location to reach.

This requirement drives us to explore the knowledge modeling tools in the literature. We choose ontology as our knowledge modeling tool in this dissertation, because of its knowledge querying and reasoning capability. Moreover, ontologies allow separating information at different levels of abstraction by constructing modules.

In this dissertation, we propose an ontology-based approach to use task level information to specify path planning queries for the primitive actions of a task plan. The work presented consists of three main parts:

- We propose an ontology to conceptualize the knowledge about the 3D environment in which the simulated task takes place. The environment is represented by a digital model relying on a multilayer architecture involving semantic, topologic and geometric data. The originality of this ontology lies in the fact that it conceptualizes heterogeneous knowledge about both the obstacles and the free space models.
- We propose an ontology of action-specific knowledge. It models the spatial constraints to be taken into account in the specification of a primitive action and the geometric constraints to be obeyed in the description of a path planning query. Their definitions rely on the ontology of the 3D environment in which the simulation takes place.
- We propose an ontology-based method to generate a path planning query for a primitive action of a task plan. Through a reasoning process involving a primitive action instantiated in the ontology of action-specific knowledge, we are able to infer the start and the goal configurations, as well as task-related geometric constraints by exploring an instantiated ontology of 3D environment.

The contributions of this thesis work will be validated by full simulations of several manipulation tasks under strong geometric constraints. The path planning results will be compared. The thesis work is organized in the following chapters:

In Chapter 2, we will introduce the industrial and the scientific context of this thesis work. We focus on simulating and validating manipulation tasks under strong geometric constraints (i.e., assembly, disassembly and maintenance). The scientific community discussed this issue from the task planning point of view and the path planning point of view. When task planning and path planning are used jointly, the link between them still has to be improved, notably because of the lack of loopback between them. This thesis work particularly focuses on using task-related information at the path planning level. Although different kinds of task-related information can be found at the task level, they usually rely on the task to be solved, and they cannot be easily reused or updated from one application to another. In the meanwhile, ontology, as a knowledge modeling tool, allows to build a modular and evolvable knowledge model. Therefore, building ontologies of task-related information draws our attention. In our research, we presented an ontology-based approach to use task-related information to specify a path planning query of a primitive action of a task plan.

Chapter 3 gives a detailed description of an ontology of 3D environment where a simulated manipulation task takes place. A waterfall-like engineering method is applied. In this chapter, we discuss why different levels of environment information should be considered (i.e., semantics, topology and geometry) jointly. We apply a modular architecture in the proposed ontology. We can easily see that a concept might have different meanings at different levels.

Chapter 4 discusses an ontology-based method to generate a path planning query for a primitive action of a task plan. Firstly, an ontology of action-specific knowledge is proposed. This ontology is modeled based on the ontology of 3D environment (Chapter 3). Through a reasoning process involving a primitive action instantiated in the ontology of action-specific knowledge, we demonstrate how a corresponding path planning query is generated. In particular, we show how geometric constraints are inferred from spatial constraints.

Chapter 5 performs ontology validation and simulation experiment for the contributions proposed in Chapter 3 and Chapter 4. The path planning strategies to be compared are introduced at the beginning of this chapter. Then, two simulation scenarios are presented. In the end of this chapter, the evaluation on each of the two scenarios are presented and discussed.

In Chapter 6, we make a synthesis of our proposed contributions in section 6.1. Section 6.2 discusses the benefits of our proposed ontology-based approach to use task-related informatin in the path planning of a primitive action. After we have discussed the benefits, we also present the analysis of the limitation of the proposed approach. At the end of this chapter, we will discuss several key perspectives in which we might interest in the future.

Chapter 2 Motivation and Context of the work

This chapter describes the motivation and the context of our work. The first part (section 2.1) points out the industrial context at which this work is targeted. Facing the economic competition, PLM has been applied in the industrial production to reduce the time and the cost of the product development (section 2.1.1). Virtual prototypes have been used along with the V-Cycle model of PLM to pre-validate the design of industrial products. Simulations performed on virtual prototypes allow to validate industrial operations under strong geometric constraints (i.e., assembly, disassembly, maintenance) before real product manufacturing (section 2.1.2). The recently emerged VR technique provides the user a great realism and the capability of interacting with simulations so that industrial operations can be better verified (section 2.1.3). The second part (section 2.2) illustrates the concentration of this thesis work on simulating and validating manipulation tasks under strong geometric constraints (assembly, disassembly, and maintenance). This issue begins with a discussion of task planning (section 2.2.1) and path planning (section 2.2.2). Solving a complex manipulation task requires to consider them jointly. However, the loopback between task planning and path planning remains poor (section 2.2.3). In this thesis work, we are interested in using task-related information at the path planning level. The modeling of task-related information is discussed (section 2.2.4) and the necessity of building a knowledge model of task-related information is expressed (section 2.2.5). In order to properly generate a path planning query of a primitive action, action-specific knowledge should also be carefully considered (section 2.2.6). Section 2.2.7 analyzes the locks of these works, introduces the positioning of our work and discusses the challenges. In the final section (section 2.3), the principle ideas and contributions of our work are presented.

2.1 INDUSTRIAL CONTEXT

Nowadays, industrial companies are required to reduce the time and the cost of the product development, whereas industrial products become more and more integrated, and quality and normative standards are increasingly demanding. In this context, since the beginning of the 21st century, the Product lifecycle management (PLM) has emerged as a new paradigm, from the design stage on, of managing all aspects (e.g., resource, people, data, method) of a product during its lifecycle.

The information technology has been put into service to manage a product's data along its lifecycle consistently. In particular, the design of new products has been given foremost support to apply new ideas. Indeed, the process and resource management, collaboration and communication among people or organizations, and project management can also be consolidated using the information technology [Neubert et al. 2004, Sureephong et al. 2008]. Such systematic management of the product data makes it easier to develop, to manufacture, to use and to dispose of a product.

Moreover, in order to take full advantage of what PLM offers, the implementation of PLM has to be adapted into the company's business process. Recently, the implementations of PLM in the aerospace industry have been reviewed [Cantamessa et al. 2012]. The benefits of using PLM tools have been mainly categorized into three levels:

1) Operational effects and benefits: PLM assists in identifying and reducing ineffective/inefficient/unnecessary activities. It helps the maximum reuse of the past design information (e.g., through comparison for the best match), and then reduces the possibility of the design error and enables the design optimization. It can also help the individual workers better understand the product and the product's architecture (e.g., through the product's portfolio).
2) Organizational effects and benefits: PLM enables data transparency and interoperability along a product's lifecycle. It reduces the information redundancy, integration efforts, and conflicts in the collaboration during the product's lifecycle. It results in a more rigorous product development process, to reduce the product change impact, to get a higher design quality and to early detect abnormal in the development process.
3) Strategic effects and benefits: PLM aims at assisting a company in achieving its business objectives, through improving the firm's performance. The improvements include getting a higher innovation level of a product, minimizing the time and cost of the product development process, minimizing the time of a product to the market, and maximizing the satisfaction of the customers.

2.1.1 Industry and Product Lifecycle Management (PLM)

Product lifecycle management is not a particular method or a specific tool. From the product development point of view [Karniel et al. 2011], it is a systematic way to manage all possible information related to a product to be developed. In particular, the collaborative knowledge embedded in collaborative activities of PLM allows to enhance traceability in decision-making within PLM process [Matta et al. 2011, Matta et al. 2013]. From the business point of view [Javvadi May 2011], PLM is a business process to minimize the development time and cost during the product's lifecycle, speed up the launch of the product on the market, maximum the product's value for money and customer's expectation. Last but not least, from the company's management point of view [Stark 2015], PLM is an integrated and joint-up product management strategy and a cornerstone of the company's success.

The traditional product development process in PLM is driven by the customers' needs to meet their expectation with some targeted functionalities. Each of these needs is carefully analyzed and specified as the basic functions of a product. These basic functions are then defined, developed and validated individually before integrated into the final expected product. It allows the company to adopt a commonly-used and rigorous development lifecycle model during product development, called the V-cycle model [Hirschberg 2000]. The V-Cycle model, initially developed in German [Plögert 1996] and United States [Forsberg et al. 1991] in the late 1980s, gives a rigorous and formal development step description of a product (see *Figure 1*):

1) Definition and Design: The conceptualization of a product follows a top-down approach. The product's portfolio is determined, through a list of sequential steps, from the general system description to the detailed function implementation.
2) Test and Integration: The test and integration of a product follow a bottom-up approach. Each unitary function is validated and verified firstly, and it is then integrated into a higher level system module until the whole system is assembled and checked.

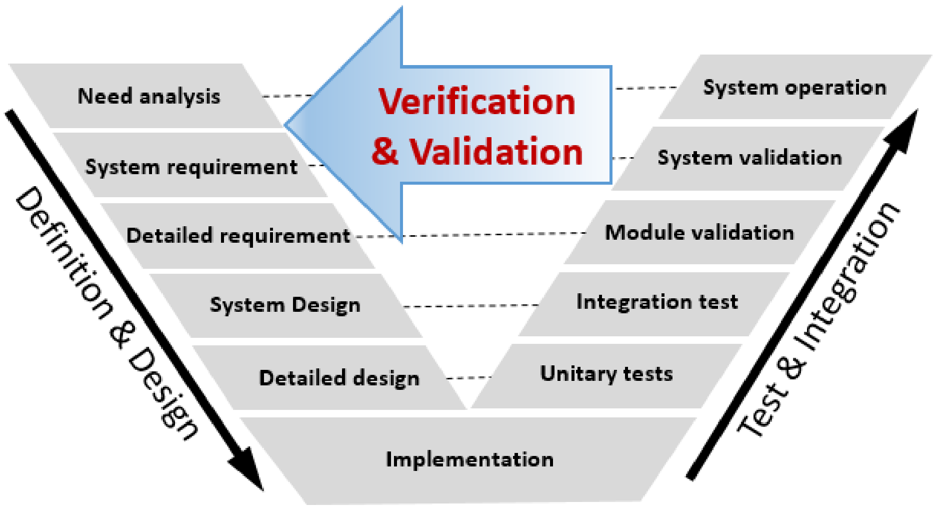


Figure 1 V-Cycle Model for the product development process

The increasingly complex products require a large number of tests to verify and to validate their design, their functionalities and their associated development procedures (e.g., assembly).

However, building physical prototypes can be too time and cost consuming to meet the deadline and the budget, and it is sometimes impossible for complex products, e.g., an airplane.

2.1.2 Using virtual prototypes along the PLM

Responding to time and cost constraints of the product development, industrial companies are seeking for a new, higher controllable and low-cost prototyping tool, from the design stage on,

- Not only to validate the static models of their industrial products
- But also to validate all possible tasks associated with their PLM, such as assembly, disassembly, and maintenance.

Answering to these requirements, 3D Computer-Aided Design (CAD) modeling, starting from the mid of the 1960s, uses the power of computers to support the creation, modification, analysis, and optimization of a product using virtual prototypes [Sarcar et al. 2008]. Functional and integration tests performed on virtual prototypes reduce the number of physical models required, and they allow to eliminate abnormities early at a significantly reduced cost. However, these tests are conducted with inaccuracies using the approximate product models. The virtual prototypes may not assist an exclusive validation and verification of products. As a result, the physical prototypes are often interleaved with the virtual prototypes in the V-cycle product development process proposed in Figure 2 [Cailhol 2015]. Virtual prototypes pre-validate a product at its conceptualization stage, and physical models allow the real-world tests before the product is manufactured.

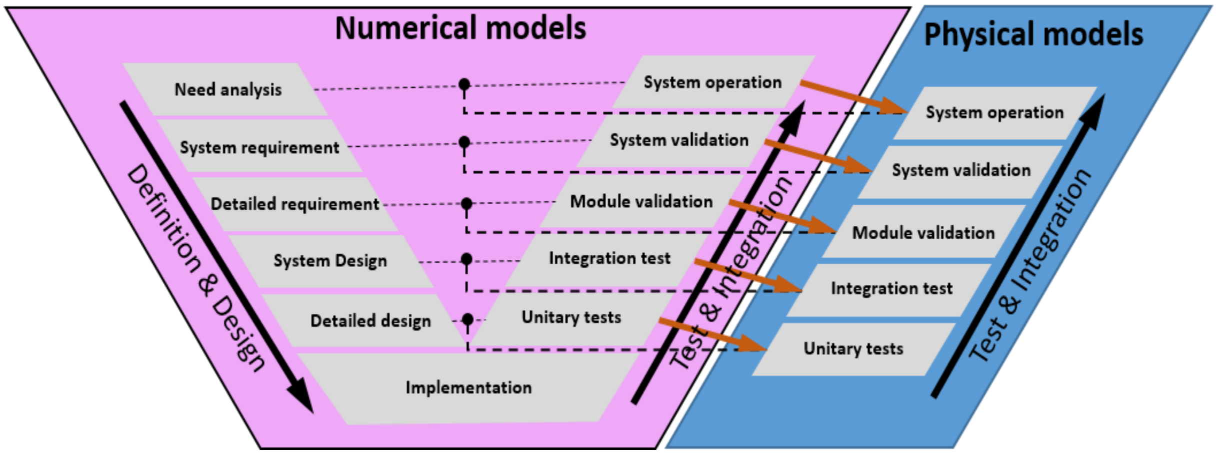


Figure 2 The V-cycle product development process in PLM using numerical and physical models [Cailhol 2015]

Besides foremost support to the design of products (i.e., the static CAD models), virtual prototypes have also been widely used for the simulation purpose. Different simulation software, e.g., DMU kinematics for CATIA® [Systemes 2014], Motion modules for Solidworks®

[Systemes 2014], carries out the kinematic analysis of a system and verifies whether the system can function correctly. Other tools concern the simulation of the assembly or the dismantling of products, and the feasibility of integrating the components of a product can then be verified. However, for a highly integrated product, these tools encounter difficulties of finding the right assembly paths.

2.1.3 PLM with the Virtual Reality (VR) technique

Recently, thanks to the progress made by sensorimotor interfaces and their coupling with 3D content, the emergence of VR techniques allow immersive and interactive task simulations while considering the human operator in the loop. The presence of human operators offers the possibility to modify the scenario interactively, and its immersion in the virtual world allows greater realism of the simulation.

One main VR application is to simulate and to validate the assembly tasks of products at the design stage before entering into real manufacturing and production. We noted that a highly integrated industrial product makes it difficult to perceive geometric constraints and to define reasonable assembly sequences and trajectories respecting user's immersion in the virtual simulation. These simulations can be then enhanced by tools to assist users.

- The feasible sequence of assembly and disassembly can be automatically generated by identifying the geometric constraints between the virtual assembly parts. These constraints can either be directly extracted from the assembly tree of the CAD software or through analyzing the contacts between the virtual parts. Once they have been identified, visual guidance compliant to the constraints can be provided to users through a haptic device to facilitate the assembly of the components [Tching et al. 2010]. This work progressively uses geometric visual fixtures (i.e., software generated forces and position signals [Abbott et al. 2007]) as guidance until the final assembly position of a component is reached.
- The feasible (possibly collision-free) trajectories for the assembly of virtual parts of a product can be automatically calculated using path planning tools [Ladeveze et al. 2009]. The trajectories are then used to guide users through a 'haptic force' until the final location is reached. Once the manipulated object is forced to move away from this trajectory, the user's manipulations trigger the re-calculation of a new path.

2.1.4 Synthesis

As up-to-date industrial products are more and more integrated and the economic competition increases, industrial companies express the need to validate, from the design stage on, not only the static CAD models of their products but also the tasks (e.g., assembly or maintenance) related to their Product Lifecycle Management (PLM). Virtual prototypes are used along with the V-cycle model of the product development to pre-validate the design and the integration of products at the conceptualization stage, whereas physical prototypes run the real world tests to eliminate the inaccuracies brought by the approximate virtual product models. Simulations running on the virtual prototypes allow further to verify and to validate the tasks associated with PLM, e.g., the assembly of industrial products. The recently emerged VR techniques allow interactive and immersive simulations to provide the involved human operator a great realism and the capability to alter the scene.

The simulations of complex tasks related to PLM possibly under strong geometric constraints, i.e., assembly and maintenance, check the behaviors of products and their associated parts during assembly. The abnormalities of the design of products can be early and better detected. The simulations also possibly provide the manipulation assistance to the involved users, so that the inexperienced users can be better trained.

2.2 STATE OF THE ART: MANIPULATION TASK SIMULATION

This thesis work deals with the simulation and the validation of complex manipulation tasks under strong geometric constraints (i.e., assembly, disassembly and maintenance) in virtual environments. The scientific community looked at this issue from two points of view.

- Task planning, reasoning at the task level, decomposes a manipulation task to be realized into a sequence of primitive actions (i.e., a task plan)
- Path planning computes collision-free trajectories, notably for the manipulated objects. It traditionally uses purely geometric data, which leads to the classical limitations (i.e., high processing time, low path relevance regarding the task to be performed or failure).

Joint task and path planning approaches perform a classical task planning step and checks out the feasibility of path planning requests associated with primitive actions of a task plan. However, the link between task planning level and path planning level has to be improved because there lacks loopback between these two levels:

- Path planning queries of primitive actions of a task plan use purely geometric data; no task-related information has been considered to control the path planning process.

- The path planning information used to question a task plan is usually limited to the motion feasibility of primitive actions of the task plan; richer path planning information would be needed, such as the relevance or the complexity of the proposed path.

This thesis work focuses on using task-related information to control the path planning for a primitive action of a task plan. Different modalities of task-related information should be considered; the method of using task-related information in the path planning process of a primitive action should be discussed.

The following sub-sections start with discussing the related work of task planning (section 2.2.1) and path planning (section 2.2.2), following the necessity to consider them jointly in solving a complex manipulation task (section 2.2.3). Section 2.2.4 explores the modeling of task-related information. Section 2.2.5 discusses the necessity of building a knowledge model of task-related information. Moreover, in order to properly generate a path planning query for a primitive action, task-related information should also take action-specific knowledge into consideration (section 2.2.6). In the end, through analyzing the state of the art concerning manipulation task simulations, their locks are identified, the objectives of this thesis work are specified and the corresponding challenges are listed (section 2.2.7).

2.2.1 Task planning

One issue of simulating complex manipulation tasks is to find a feasible sequence of primitive actions, e.g., to re-arrange the objects to be manipulated. Such an issue belongs to the task planning problem, which generally considered as symbolic planning in the artificial intelligence domain [Russell et al. 2003]. Task planning maintains a symbolic representation of the environment where the simulated task performs. In such a representation, objects and locations are categorized regarding the task context, such as objects like ‘Book’, ‘Table’, locations like ‘Office’, ‘Corridor’. Some basic concepts of task planning are given in Table 1 before discussing different task planning strategies.

Table 1 The basic concept of task planning

Concept	Definition
Relations	A list of predicates describing how two or more things are related to each other. For example, On (x, y) defines a relation that x is on y, where x and y are variables.
Environment State	A list of propositions describing the state of an environment [Bidot et al. 2017], e.g., where objects are, how they are related to each other, what is the state of a location, such as In (CoffeeMachine, Office), On (Book, Table), OnFire (Office). A proposition sometimes can be considered as a relation with grounded values.
Initial State	Initial State: The state that an environment initially holds

Goal State	Goal State: The state that an environment must reach
Primitive action	A primitive action manipulates the state transition. It describes how a state changes when it is applied [Tzafestas 2013]. It is propositionally defined by preconditions and effects [Srivastava et al. 2014]. - Precondition: A list of propositions that an environment state must hold to trigger the primitive action - Effect: A list of propositions, to which an environment state will transit, after the execution of the primitive action.
State Space	The set of all states reachable from the initial state by a sequence of primitive actions [Russell et al. 2003]

2.2.1.1 *Task planning strategies*

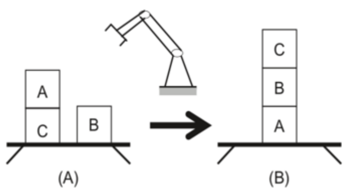
The objective of task planning is to find a feasible sequence of primitive actions (a task plan) to reach a given goal environment state. Different planning strategies can be found in the literature, as shown in Table 2.

Table 2 The task planning strategies

Planning strategies		**Description**
The state-action approaches	Forward-searching	Starting from a given initial state, this strategy progressively explores applicable primitive actions until the goal state is reached. It can either search the entire state space until it finds a feasible path (i.e. breath-first algorithm [Giunchiglia et al. 1999]) or uses heuristic-based functions to get an optimal task plan [Bonet et al. 1999].
	Backward-searching	Starting from a given goal state, this strategy recursively decomposes goal into a list of sub-goals until it solves conflicts and holds the initial states. [Kaelbling et al. 2011]
Hierarchical task network (HTN)		Unlike classical “goals-to-be” or “state of the world” approaches, the HTN approaches focuses on solving “abstract tasks” or “goals-to-do” [Nau et al. 2003, Kemke et al. 2006, de Silva et al. 2013]. It maintains a finite set of methods (*M*) and a finite set of primitive tasks (*PTs*). We note that the definition of *PT* is equivalent to the definition of primitive action. Each method (*m*) has - a name (*name*) - an abstract task (*abtask*) that it is used to solve - a sequence of sub-tasks (*subtask*) to solve the abstract task (a sub-task is either an abstract task or a primitive task) - a list of constraints to be obeyed (*constraints*) Given a sequence of (primitive or abstract) tasks (*d*) to be solved, the HTN planning process recursively searches for the applicable methods in *M* until *d* contains only primitive tasks (*PTs*).

Currently, we focus on classical state-action approaches. Figure 3 presented an example of re-arranging object A, B, C on the table [Tzafestas 2013]. The initial state of the environment (Figure 3-A) is “*On (C, Table), On (B, Table), On (A, C)*”, and the goal state to reach (Figure

3-B) is "*On (A, Table), On (B, A), On (C, B)*". A feasible sequence of actions to re-arrange them is "*unstack (A, C), put_down (A), pick_up (B), stack (B, A), pick_up (C), stack (C, B)*"



e.g., unstack (A, C), put_down (A), pick_up (B), stack (B, A), pick_up (C), stack (C, B)

Figure 3 An example of task planning

2.2.1.2 Synthesis

However, task planning alone cannot geometrically verify and implement primitive actions of a task plan. It assumes that all primitive actions can be performed if an environment state satisfies their pre-conditions. For example, a '*pick_up*' action can be performed if a robot's hand is free. However, the geometric motions of this action may be not realizable due to the obstacles between the robot and the targeted object.

2.2.2 Path planning

Another issue of simulating complex manipulation tasks is to generate and to validate motions for the manipulated objects. This issue has already been discussed by the robotics community since 1980, as a path planning problem [LaValle 2006]. It computes a collision-free trajectory, notably for a manipulated object, to demonstrate the feasibility of the manipulation.

2.2.2.1 Automated path planning strategies

Until now, the solutions for path planning problems appear as a technical library book [Choset et al. 2005]. [Cailhol et al. 2015] categorizes them, regarding whether a global environment model is built (global and local approaches) and how the path is constructed (deterministic and probabilistic methods), into the four-quadrants.

Table 3 The categorizations of path planning works [Cailhol 2015]

	Global approach	**Local approach**
Deterministic methods	Cell decomposition Roadmap	Potential field
Probabilistic methods	Probabilistic roadmap	RRT

We will now see the most representative works in each quadrant in Table 3.

1) The global approach with the deterministic method: Two main groups exist in this quadrant.

- Cell decomposition: This technique decomposes an environment into elementary geometric cells. A cell is classified as free or occupied, depending on whether this cell is occupied by obstacles. The connectivity of free cells may be used in the path search. Different kinds of works can be found in this group, according to how the cells are formed, such as Regular cell decomposition [Samet 1984], Octree/Quadtree [Meagher 1982], Triangular decomposition [Brooks et al. 1982].
- Roadmap: This technique uses samples of interest of an environment and their interconnectivity to synthesize an environment's connectivity in a graph. The Voronoï diagram [Voronoi 1908] construct the graph with the maximum distance to environmental obstacles. The visibility graph [Alt et al. 1988] follows a way how a human sees things in an environment. It connects all the peaks of obstacles and remains those who do not intersect with obstacles.

2) The global approach with the probabilistic method: The techniques in this quadrant obtain random samples of an environment and uses their interconnectivity to construct a global environment model as a connected graph. Then this graph guides the search for collision-free paths. The probabilistic roadmap (PRM) [Kavraki et al. 1996] is the most well-known method, and it aims at solving high-dimensional path planning problems.
3) The local approach with the deterministic method: The techniques in this quadrant search for a collision-free path progressively and deterministically. The potential field method [Khatib 1986] computes an artificial potential for a robot, depending on the robot's location. The potential is a superposition of the attractive field of the objective G (low potential) and repellent fields of obstacles (high potential). The robot follows the maximum gradient of the potentials to reach G. In a static and known environment, the gradient in any position is deterministic.
4) The local approach with the probabilistic method: The techniques in this quadrant search for a collision-free path progressively using random samples. Rapid exploring random tree (RRT) method [LaValle 1998] has been proposed to deal with high-dimensional planning problem in a complex, possibly unknown or dynamic environment. It progressively explores the environment by placing random samples until a valid path can be found between the start and the goal configuration.

However, these path planning techniques heavily rely on their geometric models and use purely geometric data. In a highly geometrically constrained manipulation environment, they may fail, consume a large computational time or generate a path with little relevance regarding the task to be performed (i.e., classical limitations of path planning techniques).

2.2.2.2 The use of high abstraction level environment information

A human operator often performs a manipulation task more relevantly and finds solutions in a reasonable time. This is due to his cognitive capability and trade-oriented knowledge [Cailhol et al. 2019]. The human's understanding of an environment and of objects are not limited to their geometric models but include higher abstraction level environment information, such as their topological properties and their semantics possibly related to a task to be performed (e.g., complexity). Research works of the robotic localization and navigation in an indoor environment [Hu et al. 2004, Tsetsos et al. 2006, Galindo et al. 2008, Bastianelli et al. 2013] has explored the interest of using different natures of environment data (i.e., semantics, topology, geometry) for path planning. In the meanwhile, our research work focuses on manipulation task simulations. Fewer works in this domain have discussed the benefits of using high abstraction level environment information for path planning. Two main kinds can be found in the literature.

A. The semantic 3D object map

[Rusu 2010] presented a semantic 3D object map to annotate environment objects and their surfaces with semantic labels. It consists of a 3D point cloud perceived from robot perception (e.g., vision [Dandois et al. 2013], touch [Vásquez et al. 2017]), polygonal models of objects constructed from clustering and segmentation of the point cloud, and a semantic interpretation of objects and their surfaces. It serves as semantic resources to determine the final grasp or placement position for a manipulation. For example, in an indoor kitchen environment, it allows a robot to locate the hinge of a drawer when a robot is given a high-level command to open the drawer. In the meanwhile, [Tenorth et al. 2010] presented a richer semantic description of environment objects, called 'Knowrob-Map', by using knowledge from different resources, such as encyclopedic knowledge, common-sense knowledge, and action-related knowledge. However, the modeling of semantic data is limited to the indoor household environment. Except for finding the goal location, the semantic information of this map has not yet been considered to control the path planning process, e.g., to restrict the pose of a manipulated object.

B. The multi-level environment modeling and path planning

Recently, researchers from LGP proposed a multi-level environment modeling and path planning architecture [Cailhol et al. 2015] to consider higher abstraction level information (i.e., semantics and topology) rather than purely geometric data traditionally used in simulating manipulation tasks. It consists of a multi-level environment model and a two-step path planning strategy.

1) Multi-level environment model

The 3D simulation environment for a manipulation task is considered as a closed part of 3D Cartesian space cluttered with static or mobile obstacles considered as rigid bodies. The proposed model consists of a free space model and a rigid body model, as illustrated in Figure 4.

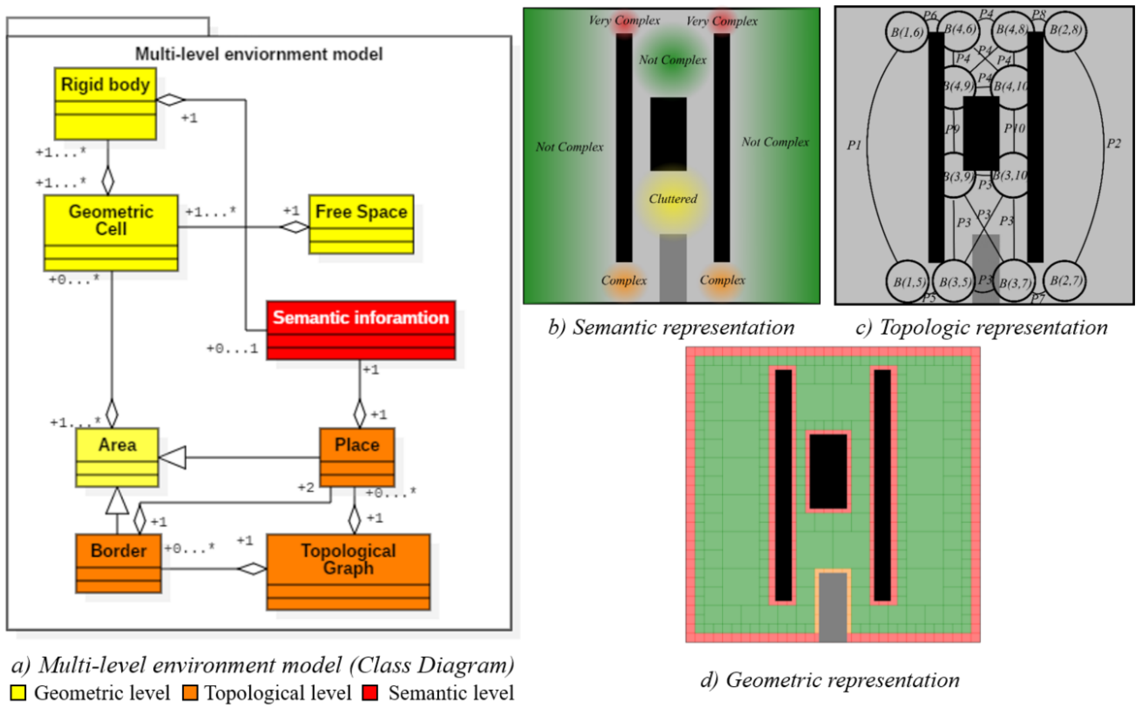

Multi-level enviornment model
Rigid body
Geometric Cell
Free Space
Semantic inforamtion
Area
Place
Border
Topological Graph
+1
+1...*
+0...*
+0...1
+2
Very Complex
Not Complex
Cluttered
Complex
P1
P2
B(1,6)
B(4,6)
B(4,8)
B(2,8)
B(1,5)
B(3,5)
B(3,7)
B(2,7)
b) Semantic representation
c) Topologic representation
d) Geometric representation
a) Multi-level environment model (Class Diagram)
🟨 Geometric level 🟧 Topological level 🟥 Semantic level

Figure 4 The multi-level environment model [Cailhol et al. 2015]

A rigid bodies model

- Geometric layer: The geometry of a rigid bodies model is represented by a classical polygonal model (i.e., vertices, edges and faces from Delaunay triangulated mesh). It is represented by a single class ‘*Rigid Body*’ in the class diagram (Figure 4).

- Semantic layer: The semantics of a rigid bodies model consists of attributes attached to rigid bodies (see the association link between ‘*Rigid body*’ and ‘*Semantic Information*’ in Figure 4-a). The form of a rigid body is specified by a shape attribute. The mobility attribute defines whether a rigid body is movable or fixed.

A free space model

- Geometric layer: The geometry of a free space model is described using a cell decomposition based on Octree (Figure 4-d). Each cell is marked as free, intersected or blocked depending on whether or how it intersects with rigid bodies.
- Topological layer: The topology of a free space model uses a topological graph to describe the connectivity between places and borders (Figure 4-c). A *Place* is noted as *P (x)*, and a *Border* is noted as *B (x, y)*. In the topological graph, a node represents a border and an edge corresponds to a place. Geometrically, places and borders consist of a collection of free cells (see the association link between '*Area*' and '*Geometric Cells*' in Figure 4-a).
- Semantic layer: The semantics of a free space model is made of attributes attached to places and borders (Figure 4-b). For example, the complexity attached to a place describes the difficulty of crossing this place. The existence of mobile obstacles implies the cluttering of a place.

2) Two-step path planning strategy

A two-step path planning strategy is used along with the proposed multi-level environment model [Cailhol et al. 2019], as in Figure 5.

- A coarse planning process computes a *Topological Path* on the *Topological Graph* (Figure 5-a). It consists of *Topological Steps*. Each step consists of a place to across (e.g., P1) and a border to reach (e.g., B (1,5)). The choice of which place to across relies on the semantic attributes attached to places and the manipulated rigid body (e.g., the complexity of a place).
- A fine planning process aims at refining each *Topological Step* into a concrete geometric trajectory to guide the movement of the manipulated *Rigid Body*. Depending on the complexity of a place, different local path planning strategies can be used. For example, a direct link between the current location of the manipulated *Rigid Body* and the milestone to reach is used when the place is free, whereas RRT method is preferred when it is narrow or congested.

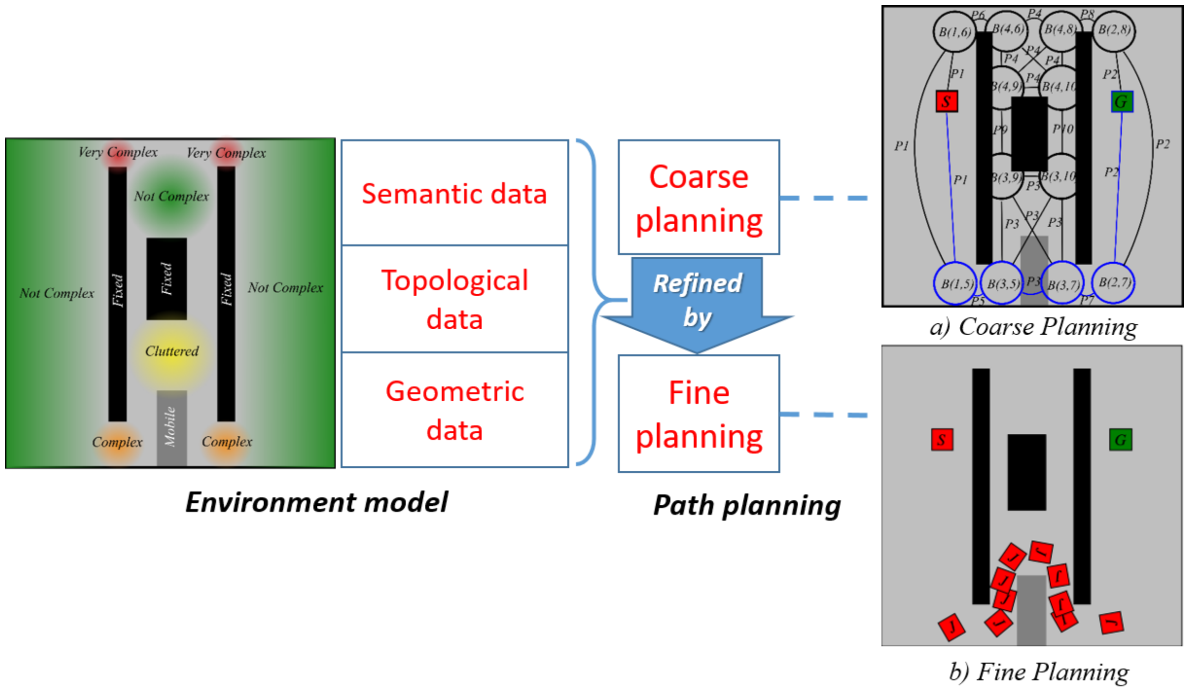


Figure 5 The multi-level path planning

This work uses a case study inspired from a 'shaped game' for babies. The experimental results have demonstrated that, using higher abstraction level information (i.e., semantics and topology), the path planning results have been improved comparing to the state of the art automated path planning techniques.

- From the qualitative aspect: The computed path is of higher relevance, thanks to the semantic information attached to places and rigid bodies.

- From the quantitative aspect: The number of needed random samples to define a trajectory is largely reduced by restricting the search space to the targeted places and using the semantic control to avoid the narrow and complex regions. Thus, it results in less processing time for path computation.

However, its semantic model is still preliminary with no task-related information considered. The semantic and topologic data have not yet taken into account in the geometric path computation. The RRT algorithm used in narrow places still suffers the classical limitation of automated path planning techniques.

2.2.2.3 Synthesis

A key issue of simulating manipulation tasks is to find collision-free trajectories for manipulated objects to demonstrate the feasibility of manipulations. We begin the discussion by reviewing the related works of automated path planning techniques. These works traditionally use purely geometric data and they encounter classical limitations. Using higher abstraction level environment information (e.g., semantics, topology) rather than purely geometric data in path planning can possibly lead to better path planning results. In the domain of manipulation task simulations in which we are interested, two main kinds of works have been presented. Although they have demonstrated the benefits of using high abstraction level environment information (e.g., semantics, topology) in path planning, they provide limited control on the path planning process and task-related information has not yet been fully considered. Moreover, path planning cannot reason at the task level to find a feasible task plan.

2.2.3 The joint usage of task planning and path planning

Solving a complex manipulation task problem requires to consider task planning and path planning jointly [Siméon et al. 2004]. The common type of joint task and path planning approaches consists of two significant processes [Lagriffoul et al. 2014], as shown in Figure 6:

- At the task planning level, a task planner decomposes a manipulation task to be realized into a feasible sequence of primitive actions (i.e., a task plan) so that a goal state, e.g., where objects A, B, C are arranged properly, can be reached.

- At the path planning level, a geometric path planner checks out the feasibility of path planning requests associated with primitive actions of a task plan. Once a primitive action is found geometrically infeasible, the path planner informs the task planner to find an alternative task plan.

Therefore, the search space of joint task and path planning [Lagriffoul et al. 2016] is the cross product of the state space (task planning) and the configuration space (path planning).

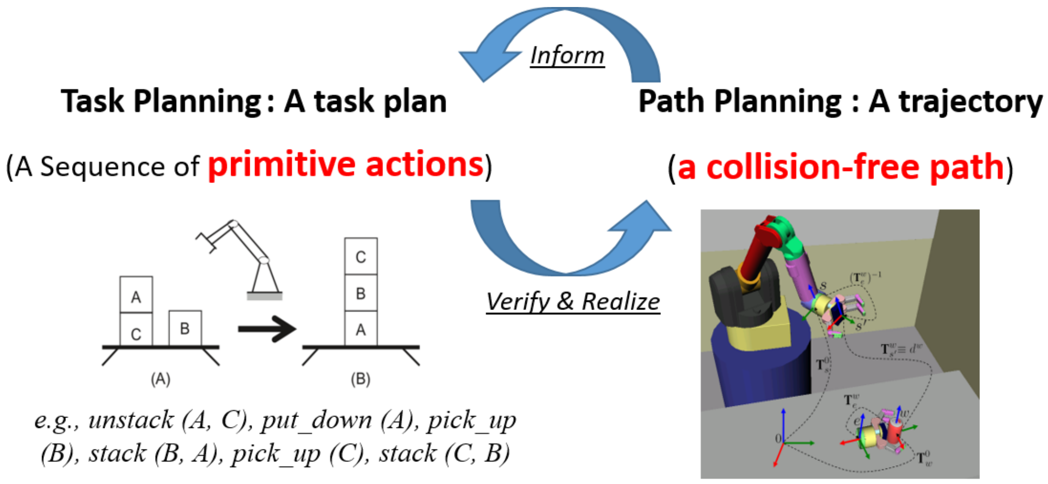

Inform
Task Planning : A task plan
(A Sequence of primitive actions)
Path Planning : A trajectory
(a collision-free path)
A
C
B
(A)
C
B
A
(B)
Verify & Realize
e.g., unstack (A, C), put_down (A), pick_up (B), stack (B, A), pick_up (C), stack (C, B)

Figure 6 The joint usage of task and path planning

2.2.3.1 *Planning strategy*

There exist in the literature different joint task and path planning techniques from the robotic domain allowing to solve complex task problems. In terms of when the geometric feasibility of a primitive action in the task plan is verified, we category these techniques in two main classes (

Table 4).

Table 4 The planning strategies of the classical joint task and path planning

Planning strategy	**Definition**
Task planning, then path planning, iteratively	The key point of this planning strategy is that the feasibility of primitive actions of a task plan is verified after the complete task plan is constructed. - If all primitive actions of a task plan are geometrically feasible, the task plan is verified. - Otherwise, the logical constraints representing the failures will be added into the task planning problem description and then the task planner looks for an alternative task plan.
Using path planning during task planning	The key point of this planning strategy is that the feasibility of primitive actions of a task plan is verified during the construction of a task plan. At each leaf candidate state of the search tree, the feasibility of each applicable candidate primitive action is instantly verified by a path planner. - If the candidate primitive action under verification is feasible, the corresponding branch will be further explored. - Otherwise, the corresponding branch will be abandoned.

1) Task planning, then path planning, iteratively

Different research works implement their own task and path planning process. [Erdem et al. 2011] proposed to invokes a RRT path planner to compute continuous trajectories for a robot once a task plan is constructed. [Dearden et al. 2013] discussed a probability model, built from a learning process, to generate the valid geometric pose of manipulated objects or of the robot for the symbolic states involved in a generated task plan. The geometric description is then used to compute the continuous trajectory for the robot. More recently, [Srivastava et al. 2014] proposed to use arbitrary off-the-shelf task planners (e.g., CCALC [McCain et al. 1997]) instead of developing a new variation. The approach addresses the problem of continuous refinement of an initial task plan by correcting inaccurate task description with constraints gained from geometric reasoning (path planning). Once an initial task plan is generated, the task plan is progressively refined from where the failure occurs (i.e., no feasible trajectory for a primitive action). The process continues until a geometrically feasible task plan is found or the resource limit (e.g., time) is reached. However, the above works consider the geometric reasoning of primitive actions in a task plan separately and unrelated. The former geometric decision can fail the latter one. Thus, it could result in a long or even intractable processing time. [Lozano-Pérez et al. 2014] defines several plan skeletons (partial task plan with unparameterized constraints associated with each primitive action) and formalizes them as Constraint Satisfaction Problem (CSP) [Dechter et al. 2003]. The work searches in a given task plan for the skeletons, and quickly verifies and implements the matched part as a whole by solving the corresponding CSP.

2) Using path planning during task planning

[Caldiran et al. 2009, Cambon et al. 2009, Dornhege et al. 2009, Guitton et al. 2009, Eyerich et al. 2010, Wolfe et al. 2010, Kaelbling et al. 2011] presented implementations of this kind. Specific logical predicates are introduced in the precondition and/or effects of a primitive action (see the primitive action definition in Chapter 2). They allow the task planner to invoke calls to external functions, e.g., collision checker and path planner. For example, a specific logic predicate "can_move(?x, ?y, ?o)" invokes an external path planner (e.g., implemented in C++) to generate and to validate motions for a manipulated object or for a robot. The initial and the final configuration of the object or the robot are extracted from the symbolic state 'x' and 'y'. These predicates are commonly recognized as "external predicate" [Caldiran et al. 2009] or "semantic attachment" [Eyerich et al. 2010]. The "external predicate" is initially implemented in the action domain description C+ [Giunchiglia et al. 2004] using undocumented features of CCALC, and the "semantic attachment" is developed as an extension of PDDL (Planning Domain Description Language, [Fox et al. 2003]) used along with a modified the FF [Hoffmann et al. 2001] task planner. Later on, [Kaelbling et al. 2011] presented a hierarchical task and path planning approach using goal-regression strategy. It regressively decomposes the top-level goal into sub-goals and forms a hierarchical level of abstraction. The task planner solves sub-goal problems at each level and finally summarizes a final task plan. A set of geometric generators act as the interface between the task planning and geometric reasoning. In the given washing example, they assist in determining the obstacles along the swept volume of paths of the robot and where they should be placed.

2.2.3.2 The nature of the exchanged information

A key issue of joint task and path planning concerns the nature of information exchanged between task planning level and path planning level and how the information has been used. In this section, we are going to review, in the joint task and path planning, the path planning level information used at the task planning level (section A) and the task planning level information used at the path planning level (section B).

A. *Path planning level information used at the task planning level*

The motion feasibility of primitive actions of a task plan: The most commonly used path planning information feedback to the task planning level is the motion feasibility of primitive actions of a task plan [Tzafestas 2013, Lagriffoul et al. 2014]. If a path planner fails to verify the motion feasibility of a candidate primitive action, the joint task and path planning process successively performs the following steps:

- For current candidate primitive action, it generates an alternative path planning query with different start and goal position.
- If the number of path planning query attempts exceeds a given threshold, it finds an alternative primitive action in the search tree or even looks for an alternative task plan.

Reachability / Visibility of the objects to be manipulated: Some other research works [de Silva et al. 2013, Garrett et al. 2014, Bidot et al. 2017] define the reachability or the visibility of the object to be manipulated as the pre-conditions of primitive actions. During the task planning process, these pre-conditions allows to prune out those branches in the search tree that the objects to be manipulated can be reached or be visible to the involved robot.

B. *Task planning level information used at the path planning level*

In the literature of the joint task and path planning, two main kinds of task-related information are used at the path planning level.

Task-related symbolic environmental information: This kind of environmental information often comes along with the task description [Galindo et al. 2008, Pronobis 2011]. It defines not only about the types of objects and locations, e.g., refrigerator, cupboard and their inner space in a kitchen environment, but also their properties, e.g., shape and functionality of refrigerator and cupboard. Using such information allows to interpret primitive actions of a task plan, e.g., Move (book, storage), and to determine the start and the goal position of the associated path planning queries. It is achieved by a proper semantic mapping [Nüchter et al. 2008] between the symbolic environment representation and geometric models.

Geometric constraints related to a task problem: Some other works have also discussed the interest of using the task-related geometric constraints, either to determine the final placement of a manipulated object [Bidot et al. 2017] or to search for the final pose of the assembly objects [Perzylo et al. 2015].

2.2.3.3 Synthesis

Although task planning and path planning have been respectively well studied by the artificial intelligence community and the robotics domain, the link between them should be improved due to the lack of loopback between these two levels.

- The path planning information used to question the task plan is usually limited to the motion feasibility where richer information such as the relevance or the complexity of the proposed path would be needed;
- Path planning queries traditionally use purely geometric data and/or "blind" path planning methods (e.g., RRT), and no task-related information has been considered at the path planning level to control the motions of a manipulated object.

Currently, this thesis work concentrates on modeling task-related information and using it in the path planning of a primitive action to control the motions of a manipulated object.

2.2.4 The modeling of task-related information

As we have mentioned in section 2.2.3.2, two different kinds of information can be found at the task level: 1) the 3D environment information related to a simulated manipulation task; 2) task-related geometric constraints restricting the final pose of a manipulated object. In this section, we will discuss respectively the modeling of these two kinds of information in details.

2.2.4.1 The modeling of environment information

In this section, we will first introduce some concepts definitions about the 3D environment where a simulated manipulation task is performed, before entering into the detail discussion.

A. Concept definition for the environment where a manipulation task is performed

3-Dimensional (3D) Cartesian Space (3D Space): The space where each point is expressed by a vector of 3 Cartesian coordinates. With the definition of an origin and a world reference frame (i.e., three orthogonal reference lines), the position of a point is determined by its projection onto each of these three reference lines. It is noted as R_3.

6D Configuration: A free-flyer, no-kinematics-link object in 3D Cartesian space has not only position (the projections to the reference lines), but also its orientation relative to the reference frame. The orientation is expressed by a vector of 3 angles. Each angle describes an independent object rotation to a reference line. We usually describe it as a 6D representation. Each point in such representation presents a 6D configuration of the free-flyer, consisting of a position and an orientation.

Configuration Space: The 6D representation is a specification of configuration space. Each 6D configuration describes that an object has six-degree-of-freedom, where the object can move freely on the x-, y- and z-axis and can rotate freely around them. When considering a list of objects constrained by holonomic links, the situation of one object in relation to another is characterized by the links connecting them. This issue has already studied by the robotic domain as the n-links kinematic chain problem. Each connected link presents a parameter. A vector grouping all the parameters along the links forms a configuration of the chained object and describes its degree-of-freedom. Thus, studying the movement of the chained object is to find collision-free points (n-dimensional vector) in an n-dimensional configuration space (noted as R_n) [Lozano-Perez 1990].

Free Space: In the considered space, the obstacle-free component or free space is generally noted as E_{free}. In the R_3, it includes all the points that are not belonging to any obstacles (noted as E_{free}). In the configuration space (R_n), the definition of free space is more complicated. The set of the kinematic chain configurations that are in collision with obstacles is noted as $C_{obstacle}$. Therefore, the free space (noted as C_{free}) can be regarded as complementary of $C_{obstacle}$, and its definition depends on the part of R_3 occupied by obstacles and the shape of the rigid body.

B. The nature of the environment information

The environment where a simulated manipulation task takes place is considered as a closed part of 3D Cartesian space cluttered with mobile/fixed obstacles (regarded as rigid body). These rigid bodies are built on CAD models. Recently, the proposed environment model [Cailhol et al. 2019] consists of a rigid bodies model and a free-space model. Both of them involves different levels based on semantic, topologic and geometric information. This thesis work is especially interested in such a multi-level environment model, and it studies the nature of environment information at each level.

Modeling of rigid bodies

Two main classes of information associated with rigid bodies have been considered in the CAD and the robotics literature (Table 5).

- Geometric information: different geometric models of rigid bodies can be implemented using infographic tools from CAD. They have been widely used in the numerical design of a CAD product and the modeling of the environments where robotic applications perform.

- Semantic information: the semantics of rigid bodies must be adapted to the tasks proceed by applications. Different kinds of semantics can be modeled according to the task needs, including the categorization of rigid bodies, their functionalities, hierarchical part structure.

Table 5 The modeling of rigid bodies

Geometric modeling	Two major representative schemata are used to model the geometries of rigid bodies based on CAD models. **1) Volume Representation** In the CAD part design, the volume representations of rigid bodies are mainly obtained using the constructive solid geometry technique (CSG). A 3D object is built upon some standard primitives using regulated Boolean operations [Requicha 1980]. - *The standard primitives of CSG* are some simple regular shaped objects, such as parallelepiped (block), the triangular prism, the sphere, the cylinder, the cone, and the torus. - *Boolean operations* are regularized union, regularized difference, regularized intersection. The standard primitives are generic with a parametric definition in the sense that they represent shaped objects that must be instantiated by users to determine the variables. **2) Surface Representation** Boundary representation is the main technique to represent the surface boundaries of 3D objects in CAD [Hoffmann 1989]. This representation consists of two parts: - A topological description of connectivity and orientations of vertices, edges, and faces. - A geometric description of points, lines, and surfaces, regarding vertices, edges and faces. In BREP, the surface description can be polygonal [Botsch et al. 2007] or parametric [Böhm et al. 1984]. In particular, our work concerns more the polygonal representation. In such representation, the surface is characterized by meshes consisting of vertices, edges, faces. It can be then regarded as a specific implementation of BREP. The accuracy of this representation heavily relies on the size of faces and the sampling resolution on the modeled surface. Moreover, different polygon face shape can be used, but the Delaunay triangulation is the most well-known in the polygonal mesh implementation.
Semantic modeling	***Semantic (3D) object map***: 'Semantic (3D) object map' has been proposed [Rusu 2010] and developed to associate/map the semantic information (e.g., the taxonomy of objects, such as ‘Container’, ‘Handle’, and their properties) with the geometric description of 3D object models. The semantic information of rigid bodies varies among applications. Its modeling must be adapted to the tasks proceed by applications.

	- Certain works are only interested in the kitchen environment [Marton et al. 2008, Blodow et al. 2011] - Some others [Mozos et al. 2007, Zender et al. 2008] consider the whole indoor household environment - [Tenorth et al. 2009] uses encyclopedic and common-sense knowledge as two major semantic sources.. Such semantic information is rarely used independently from the geometries of rigid bodies. The resultant model allows to find the geometric description for a given symbolic term, e.g., the position of the handle of 'Container A'.

Modeling of the free space

The human's understanding of an environment has different levels of abstraction. Indeed, a free space model of an environment should consider all the geometric, topologic, and semantic information (Table 6).

Table 6 The modeling of the free space

Geometric modeling	Two main techniques have been used to synthesize the geometries of a free space model. - Cell decomposition decomposes a closed-part of 3D Cartesian space into smaller geometric cells. The shape of the geometric cells depends on the techniques to use. - Roadmap models the points of interest and interconnects them as a graph to describe the connectivity of the free space. ***Cell decomposition*** Currently, we focus on the cell decomposition technique. This technique decomposes an environment into elementary geometric cells. A cell is classified as free, intersected or blocked, depending on whether this cell is intersected or fully occupied by obstacles. The free space of an environment is a group of connected free cells. According to how the cells are formed, [Cailhol 2015] classifies the related works in mainly 4 categories: Exact cell [Demyen et al. 2006], Rectangular Cells [Brooks et al. 1982], Regular Cells [Samet 1984], Unbalanced Tree (quadtree in 2D space [Samet 1984] and octree in 3D space [Meagher 1982]). This work focuses on the octree decomposition to build the free space model of a 3D environment, because it allows to reduce the size of the free space model by ignoring the free and blocked cells.
Topologic modeling	The topology of the free space model synthesis the connectivity of an environment, such as reachability between different locations. The graph, as a common tool for describing the connections between a set of nodes, is used to specify this level of information. A node in the graph represents a unique place and an arc connects two different places. We should note that the 'place' here is generic and its definition might have different meanings. Exploring different method of building a 'place', we can categorize them in two main kinds:

	1) A view perceived by the mobile robot A view for the mobile robot has been defined as a combination of sensory inputs (e.g., laser, visual) parameterized by a set of variables. A view represents a constant state when the variations of its sensory inputs do not exceed a given threshold during a robot's environment exploration. Otherwise, a new view representing a new state will be created. Thus, the arc connecting two views represents the state transition of the environment that the robot 'sees'. [Kortenkamp et al. 1994] [Dedeoglu et al. 1999] [Kuipers et al. 2004] proposed the construction of a topological map using distinct views as nodes and their transitions as arcs. **2) A region representing a location of the environment** A region is a unique location of the environment, representing a distinct area in 2D plane or a closed-volume of a location in 3D space. The distinct regions of an environment are either manually defined [Hirtle et al. 1985, McNamara 1986], or automated generated. - [Mozos et al. 2007] proposed to construct spatial regions by detecting doorways. - [Cailhol et al. 2019] defines a place as a continuous set of free geometric cells, where the free space model is built upon the octree decomposition. This thesis work focuses on the second 'place' definition, since it deals with simulations with no robot involved.
Semantic modeling	***Semantic map*** 'Semantic map' in the literature [Galindo et al. 2008, Nüchter et al. 2008, Zender et al. 2008] captures the human's point-of-view of the environment where tasks are performed. It associates the semantic information (the taxonomy of locations, like 'room', 'corridor', and their properties) with the places constructed at the topologic level and also their geometric description at the geometric level. Similar to rigid bodies, the semantic information of the free space model varies among applications and it must be adapted to the tasks proceed by applications. For example, for indoor robotic applications, both common-sense and encyclopedic knowledge [Tenorth et al. 2009] can be used to annotate different household locations, such as kitchen, corridor, bedroom. In the robotic applications, the semantic information uses together with the geometric and the topologic levels of the free space model. The resultant model allows a path planning system, e.g., to find a goal location to reach, such as 'move the book to the storage room'

2.2.4.2 The modeling of geometric constraints

Another kind of information, often used in manipulation task simulations, is the geometric constraints, notably to restrict the movement of a manipulated object. The geometric constraints are formulated as the equality or the inequality mathematical functions [Guitton et al. 2009], and they must be adapted to the tasks processed by applications.

The distance and the reference angle constraints proposed in [Guitton et al. 2009] are used to compute the destination pose of a robot that has to reach. In [Tenorth et al. 2014], the geometric constraint is specified as a kind of relation between a part of a tool and a part of an object in the world, '*left of*', '*right of*', '*above of*', '*below of*', '*in front of*' and '*behind*' are used.

In [Bidot et al. 2017], a placement constraint is used to determine how the water cup should be put on the table. It represents the relatively stable position of a manipulated object at a given location, and it is formalized as a list of variables: the local reference frame R_{loc} of the table is located at the bottom corner of the table, and the table has a size of X-size (the width), Y-size (the long) and Z-size (the height) referencing to R_{loc}. The local reference frame of the cup (R_{loc_cup}) is assumed to be at the bottom of the cup. Thus, the final location of the water cup can be determined by the transformation matrix between R_{loc} and R_{loc_cup}, as illustrated in Figure 7. Moreover, the grasp constraint determines how an object can be grasped and the kinematic constraint determines a robot's kinematic capability [Zacharias et al. 2011]. In our work, these two constraints are out of consideration and the manipulated objects are considered as free-flyers.

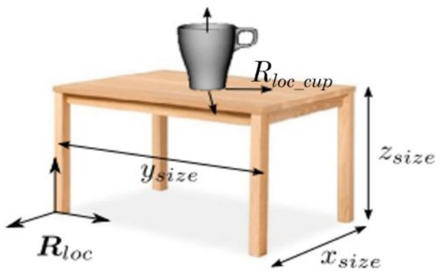


Figure 7 Placement Constraint [Bidot et al. 2017]

However, the definitions of geometric constraints are specific to applications. Whenever an environment changes (e.g., the shape of the cup, the table or the robot hand), these definitions should be reconsidered.

2.2.4.3 Synthesis

In section 2.2.4, we explored the modeling of two types of task-related information:

- The 3D environment information related to a simulated manipulation task
- The geometric constraints related to a task to be performed

However, the modeling of task-related information relies on the task to be performed. It is often locally defined and not evolvable. Each time when the application changes, the modeling of task-related information concerning this application should be reconsidered.

2.2.5 The knowledge modeling of task-related information

In this thesis work, we aim at constructing an evolvable knowledge model of task-related information. Using such a kind of knowledge model, a path planning system (e.g., robot) is able to deal with different applications by simply updating the contextual semantics related to the targeted application. The proper knowledge modeling of task-related information is critical for a path planning system to act intelligently [Vassev et al. 2012]. Although several knowledge modeling and processing frameworks have already been developd [Suh et al. 2007, Tenorth et al. 2009, Diab et al. 2019] and have been used [Akbari et al. 2015, Diab et al. 2017, Akbari et al. 2019] in the robotics domain, they rarely consider in-depth how to use task-related information to control motions of a manipulation object or a robot in path computation.

In manipulation task simulations, a proper knowledge model of task-related information allows a path planning system to possibly automatically generate a path planning query for a primitive action of a task plan. The start and the goal configurations, as well as task-related geometric constraints, can be generated. Such information could be then used to control the path planning of this prmitive action.

In this sub-section, before exploring the existing ontologies of task-related information, we will firstly introduce the cognitive tools of knowledge modeling and discuss why we choice ontology as our knowledge modeling schema.

2.2.5.1 The cognitive tools of knowledge modeling

In order to properly and formally model task-related information, three typical kinds of knowledge formalism techniques can be then identified in the literature. Before introducing these techniques, we will firstly review some important terms used in knowledge modeling.

- Concept: Words and examples to describe ‘things’ from a given domain in order to reach precise verbal definitions. The ‘things’ can anything we would like to represent, such as physical objects, events.
- Relation: A relation is a connection between two concepts. It is a set of words to describe correlativity, such as dependence, ownership, father-child.
- Instance: An implementation of a concept. Part of or all attributes of this concept have to be grounded with real values.

- Taxonomy: A classification of things/concepts represented in a hierarchic form

A. *Semantic network*

Semantic Network is a representation schema of encoding knowledge in a directed graphical depiction [Woods 1975]. In such graphical representation, nodes can represent both concepts and instances of objects, attributes, events, values, etc., whereas the directed arcs/edges describe the directed relationships between two nodes.

- Node: The concepts can be extracted from the taxonomy of a given domain. Such conceptualization is often goal-oriented depending on tasks proceed by applications. Besides, the node can also be an instance of a concept. Thus, many semantic networks distinguish the nodes representing concepts and those representing instances. The instantiation thus becomes an overlay of its conceptual schema.
- Arc: The arc in the graph is directed, representing a directed relationship between two nodes. However, there is not a standardized set of relations designed for the semantic network. Here, we noted some commonly used relations in this representation (Table 7).

Semantic network was considered as a very powerful tool to formalize the domain concepts and their relations, since the graphical representation is easily readable by a human being and can be directly used by a robot for knowledge query and reasoning. However, the semantic network approach lacks formal semantics (precise semantic characterization) despite the nodes only have semantics in its name.

Table 7 Some commonly used relations in the semantic network

Relation	Definition
is-a relation	it defines the inheritance relation between two concepts. For example, 'A is-a B' means 'A' is a subtype of a more generic concept 'B'.
has-a relation	it defines the composition relation between two concepts. For example, 'A has-a B' means 'B' is a composition property of 'A'.
instance-of relation	it defines an implementation of a concept. For example, 'Miracle is an instance of Person' means that 'Miracle' instantiates part of or all the properties of the 'Person' concept.

B. *Frame-based model*

Frame-based Model has emerged in the 70s and 80s [Minsky 1974]. Such a model consists of a set of frames organized in the hierarchical structure, inspired from the class inheritance hierarchy of object-oriented programming (OOP). The following paragraphs give some concept definitions of the frame-based model:

- Frame: Referencing to the class of OOP, a frame is a data structure that includes all the knowledge of a particular concept. Frames are organized as a hierarchical structure such that one frame is a stereotype or a subtype of another. Like a class having a set of attributes, a frame has a set of slots as its properties.
- Slot: Referencing to attributes in a class of OOP, a slot represents either a relation to another frame or an attribute to a (numerical, literal) value. A slot can have multiple facets restricting the relations or the values of a slot.
- Facet: A facet is a restriction put on the slot value or relation. For example, the seasons of a year can only have spring, summer, fall, winter; the color of the tree leaves can only be green or yellow.

Having so much similarity to the class-attribute definition founded in object-oriented programming [Cox 1986], we can regard this model as an application of the object-oriented approach in knowledge modeling. Moreover, such a frame-based model is often used along with other representation formalism. For example, using together with the semantic network, the frame-based model adds a dimension by allowing nodes to have structures to centralize all the knowledge of concepts that nodes represent.

The frame-based formalism can provide a clear and structured representation of knowledge representation in a natural manner, where all the attributes of a concept (e.g., a complex object, a task problem, the robot definition, entire situation) is encompassed in a single frame and frames are organized in a hierarchical structure according to the inheritance relations between them. However, such representation is so complex that the reasoning on such kind of formalism is difficult with limited expressiveness.

C. Ontology

Ontology emerges more recently. As a word brought up from philosophy, ontology is a conceptualization of the world with a specific point of view that we would like to represent. Ontology has been considered as a representation designed or made to be about something of the real-world [Arp et al. 2015]. It is a systematic consideration of being for existence. An ontology model [Gruber 1993] consists of two proper parts: a taxonomy whose representation is some combination of concept, and certain relations between them.

- Taxonomy: The concepts in the taxonomy make use of 'term' as linguistic expressions to represent the real world. The concept can be anything we want to represent, such as physical objects in the real world, situations that we might encounter, events might happen or already happened. The corpus of the concepts is a controlled vocabulary of terms extracted from the terminologies in a given discipline and determined by the purpose the ontology is made to address. Moreover, the taxonomy of an ontology is often structured in a way to represent a hierarchy of generality, where the top-level concepts are stereotypes of lower-level concepts. Therefore, such hierarchical structure allows organizing the corpus in different levels of abstraction. The top-level concepts are abstract and domain-independent and the lower level concepts are precise and domain-specific or even related to a task proceed by an application.

- Relation: A relation in an ontology establishes a binary connection between two entities (concept or instance). It is very familiar to us in our commonsense knowledge, such as a woman has a son, a book has a cover, the table is 7 inches high. [Arp et al. 2015] classifies three basic binary relationships used in the ontology (Table 8).

Table 8 Some commonly used relations in the ontology

Relation	Definition
Concept-Concept relation	A typical concept-concept relation is the 'is a subtype of' relation. For example, Woman is a sub-type of Person. Such 'subtype' relations hold the concepts in a hierarchy of generality, where the 'child' concept is a specification of the 'parent' one. More concept-concept relations are also possible, such as Engine is a part of Car.
Concept-Instance relation	A typical concept-instance relation is the instantiation of a concept. For example, Mary is a woman; the object is a bottle. Using this kind of relations, a robot maps environment objects with symbol concept, e.g., to assist intuitive human-robot interaction.
Instance-Instance relation	An example of the instance-instance relation is 'part-of' relation. For example, Mary's arms are part of Mary. The instance-instance relations allows to describe an instance of a concept in detail, e.g., Mary is the wife of Jone and she is the professor at the MIT.

Description Logics

Currently, the ontology model is formalized in a computer interpretable/readable format using Web Ontology Language also noted as 'OWL' [McGuinness et al. 2004]. This language is based on the description logics (DL) regarded as 'OWL-DL'. The description logics are a set of logical statements (regarded as axioms) to describe the concept, instances, and roles (properties of concepts or instances) of an application domain [Baader 2003]. TBox and ABox are two basic components.

- TBox: introduces the terminology, i.e. the vocabulary of an application domain
- ABox: introduces assertions about named individuals in terms of vocabulary and their related properties.

These statements/axioms of knowledge provide a logical formalism of ontologies describing the formal semantics for concepts, instances, and roles. Using them as current beliefs, the DL-based inferences allow reasoning about new knowledge, such as the model's consistency in the theorem proving (whether there exist contradictions), or new facts about the world. The decidability of the description logics makes such reasoning on the taxonomic knowledge powerful enough to discover implicit and hidden relationships between concepts, although its expressiveness power is restricted due to the objectives of reducing computational complexity and guaranteeing its decidability. Moreover, the DLs can be further extended with non-monotonic reasoning to invalidate previous beliefs. Such capability allows extending the use of ontology to more applications, such as cleaning and fixing the knowledge of a mobile robot in a dynamic and unknown environment.

D. Synthesis

Comparing to the semantic network and the frame-based model, the ontology models based on DLs can provide a precise and easy way to model concepts of a given task (or a domain) and their relations using formal semantics. It also has an ‘import’ mechanism allowing to reuse ontologies from existing resources. Furthermore, the reasoning capability of DLs allows to infer new knowledge from a given ontology model easily.

Therefore, in this thesis work, we choose ontology formalized in OWL-DL to construct our knowledge model of task-related information. The Protégé$^{®}$ developed by Stanford University [Protégé 2017] is then used as the viewing and editing software of ontology models. It is widely used in the scientific community and it is also freely available for scientific researches.

2.2.5.2 The ontology models of environment information

As mentioned in section 2.2.4.1, this thesis work is interested in modeling the 3D environment information involving different levels based on semantic, topologic and geometric information. In this section, we respectively explore ontology models of conceptualizing semantic and topological information used in robotic applications, and geometric information from CAD models, as shown in Table 9.

Table 9 The ontology models for conceptualizing environment information

Level	The existing ontology models
Semantic information	The semantic information of an environment is made of the types of objects and locations and also their properties. This kind of information has already been discussed in knowledge representations for robots, mainly the indoor household environment. ***Semantics in KnowRob***: KnowRob [Tenorth et al. 2009], developed in Technical University of Munich, is a Prolog-based knowledge processing system, capable of accessing OWL ontologies. The environment knowledge model makes use of - the encyclopedic knowledge to describes the types and the properties of objects, e.g., refrigerator, drawer, micro-oven in a kitchen. - the commonsense knowledge to describe what the objects can be used for ***Semantics in OMRKF***: OMRKF is called an ontology-based multi-layered robot knowledge framework [Suh et al. 2007]. The environment knowledge model for the robot has three different levels: - object features level describes the visual features (e.g., color, texture and SIFT) used to recognize an object - object level describes the taxonomy of objects and their properties; - space level describes the taxonomy of locations, e.g., living room, bedroom In either approach, the semantic information is restricted to the environment where a task performs. Therefore, the vocabulary of terms used to describe the types (objects and locations) and its taxonomic structure vary among applications.
Topologic information	As mentioned in section 2.2.4.1, the topological information of an environment describes the connectivity between places of the free space model. Such information has also been considered in KnowRob and OMKRF: ***Topology in KnowRob*** [Tenorth et al. 2009]: The KnowRob model defines - a 'Place' concept to represent a relevant location in an environment - and also a 'Map' concept as an abstract symbol for, e.g., a topological map of places in an environment. ***Topology in OMRKF*** [Suh et al. 2007]: The OMRKF model describes - The taxonomy of different locations of an environment - the topological map to describe the connectivity among the locations. However, in both case, the structure of the topological map is not clarified. It relies heavily on how the place representing a location is identified.
Geometric information	The synergy effects of leveraging the geometric information to the conceptual level are obvious, such as, fast query for localizing the top face of a table so that a robot can put down a bottle, understanding a constraint of holding a cup upwards though inferencing the corresponding geometric constraint.

We can discover several efforts of conceptualizing the geometries of rigid bodies based on CAD models in the literature, whereas the conceptualizing the geometries of the free space model is still lacking.

1) OntoSTEP

The STEP schema is called STandard for Exchange of Product model data [Pratt 2001]. It is developed by ISO organization (referenced as ISO 10303) to meet the needs of modern industry to facilitate the exchange of product data (including the CAD models) among different phases of product's development or different organizations. It consists of many parts, called application protocols (APs). Each AP is responsible for modeling one or several aspects of product modeling. AP203 (Configuration-controlled 3D design of mechanical parts and assemblies) is the most widely used AP for the CAD data exchange and it mainly focuses on representing geometric information of the CAD model [Clark et al. 1995].

OntoSTEP is an approach to translate the STEP schema into an ontology model formalized in OWL [Barbau et al. 2012]. An implementation of a particular product thus can be instantiated in the defined ontology model. It takes the AP203 as an example to transform the geometric schema in STEP as an ontology model. It discussed the benefits of having a common controlled knowledge resource in the product development process to share the product data, of having a scalable knowledge model to integrate different aspects of a product during its development, and of having the reasoning ability to, e.g., check the consistency of a product model.

However, the automated extracted taxonomy of AP203 is meaningless as an ontology model with no clear taxonomy and concept definition.

2) OntoBREP

[Perzylo et al. 2015] introduces an approach of leveraging geometric descriptions of CAD model to the knowledge level. It is achieved by constructing an ontology model defining boundary representations (BREP) of objects, that is a BREP ontology. This ontology consists of

- a topological part, illustrating the topological connectivity and orientations of vertices, edges, and faces,
- and of a geometric part, describing the geometric primitives relating to the topological part, i.e., points, curves, surfaces.

It defines an open and application independent format of BREP-based CAD data and it can be regarded as a part of common knowledge resource to be shared and reused during product development.

Yet, it is still limited to the boundary representation of CAD objects, whereas other models (such as CSG) are also possible to illustrate the geometries of CAD objects.

2.2.5.3 The ontology model of geometric constraints

Few works have discussed the modeling of geometric constraints using ontology. In [Perzylo et al. 2015], after constructing an ontology model based on the BREP of CAD models, a rich set of geometric constraints is introduced and they are formalized in an ontology model (see the taxonomy of geometric constraints in Figure 8). The geometric constraint is recognized as an interrelation between two geometric entities (i.e., points, curves and surfaces of objects).

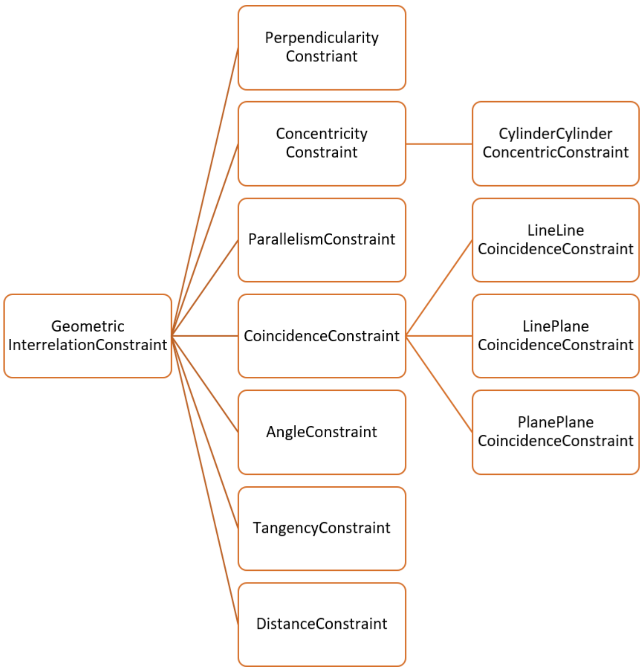

Geometric InterrelationConstraint
Perpendicularity Constriant
Concentricity Constraint
ParallelismConstraint
CoincidenceConstraint
AngleConstraint
TangencyConstraint
DistanceConstraint
CylinderCylinder ConcentricConstraint
LineLine CoincidenceConstraint
LinePlane CoincidenceConstraint
PlanePlane CoincidenceConstraint

Figure 8 A taxonomy of geometric constraints [Perzylo et al. 2015]

Figure 9 shows an assembly task between two objects. Two different constraints have been specified: "*CylinderCylinderConcentricConstraint(CylinderA, CylinderB)*" means that the AxisA and the AxisB should be aligned. "*PlanePlaneCoincidentConstraint (PlaneA, PlaneB)*" means that the distance between two planes should be zero and their normal should be against each other. They are used to determine the final assembly pose of the manipulated object.

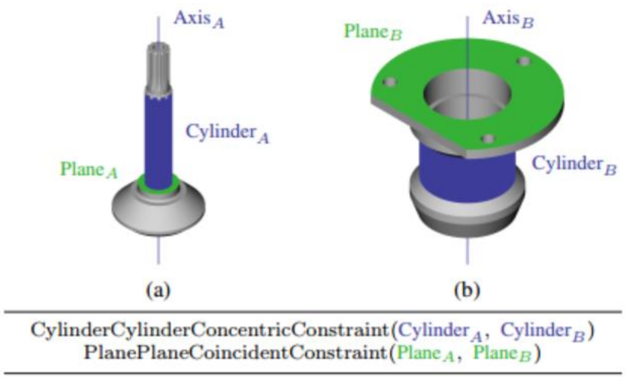

Axis$_A$
Cylinder$_A$
Plane$_A$
(a)
Plane$_B$
Axis$_B$
Cylinder$_B$
(b)
CylinderCylinderConcentricConstraint(Cylinder$_A$, Cylinder$_B$)
PlanePlaneCoincidentConstraint(Plane$_A$, Plane$_B$)

Figure 9 The assembly task between two objects [Perzylo et al. 2015]

This approach demonstrates a more flexible combination of geometric constraints defined for an assembly task using the ontology model. However, the assignment of geometric constraints can be tedious and not intuitive enough to users.

2.2.5.4 Synthesis

The section firstly explores different knowledge modeling tools and discusses the reason for using ontology (section 2.2.5.1). The existing ontologies of task-related information (i.e., 3D environment information and geometric constraints) are then discussed (section 2.2.5.2 and 2.2.5.3). Although plenty of research works have done their efforts to conceptualize different kinds of task-related information, no ontology has considered all these kinds of task-related information jointly:

- No ontologies have considered the conceptulization of the heterogeneous environment information (i.e., semantics, topology, geometry) for both the rigid bodies and the free space models in an evolvable ontology; This kind of ontology allows to adapt into different applications by updating the contextual semantics of 3D environment. It also allows to fast query any environment information and possibly to infer new knowledge.
- More importantly, rather than manually assigning the tedious and long geometric constraints to a primitive action, they can be automatically generated referring to the information related a primitive action to be performed.

In order to properly generate task-related geometric constraints to a path planning query associated with a primitive action, it requires us to study the modeling of action-specific knowledge in section 2.2.6.

2.2.6 Action-specific knowledge

Not many research works talked about action-specific knowledge. In order to correctly execute a primitive action for manipulating objects (such as "Put the bottle on the table”), the path planning system should be able to understand and has to process action-specific knowledge [Dang-Vu et al. 2016]. The action-specific knowledge is used not only to identify current and goal locations of a manipulated object but also to provide specific constraints on the way that the object should be manipulated.

The definitions of geometric constraints to a primitive action are difficult because these constraints are often hard to be understood by common users. In order to make the constraints definition of a primitive action more intuitive, this thesis work is interested in taking advantages of human’s spatial knowledge of a 3D environment where a simulated manipulation task takes place. The spatial knowledge provide a task/path planning system valueable spatial information on where primitive actions take place [Aztiria et al. 2008].

In order to restrict motions of a manipulated object regarding a reference object, we focus on the spatial knowledge describing a qualitative representation describing how two objects are relatively located to each other in a 2D/3D environment, noted as "Spatial Relation"[Hernandez 1994]. Considering the relative position and orientation between two objects, spatial relations consist of three main categories [Borrmann et al. 2009].

- Distance relation: It relies on the Euclidean metric distance. By giving a distance threshold, the distance relation between two objects can be qualified, such as "*Far*" and "*Close*".
- Topological relation: It describes how the boundaries and the interiors of two objects relate, based on Set Theory. *Disjoint*, *Touch*, *Overlap*, *Inside* and *Equal* are the mutual exclusive topological predicates based on the 9-intersection model [Strobl 2017].
- Orientation / directional relation: It describes how two geometries (object or area) or two geometric elements are placed relative to one another. Typical orientation relations are *Top*, *Down*, *Front*, *Back*, *Left*, *Right*,

Using such spatial relations, [Borrmann et al. 2009] considers a '*Spatial Constraint'* as a constraint that restricts a spatial relation between two building elements. More widely speaking, such spatial constraint definition coulod also be applied in manipulation task simulations, such as "Object A should be pointed to Object B". Comparing to geometric constraints, spatial constraints are easier to be understood by common users. Moreover, when a primitive action and its associated spatial constraints are defined in an ontology, the task-related geometric constraints can be easily inferred from the spatial constraints using some pre-defined inference rules. The rules can be easily updated according to the task to be performed.

2.2.7 Locks, Challenges

As a conclusion of the state of the art, this thesis work is interested in path planning for a primitive action of a task plan to study:

- The modeling of task-related information,
- The use of task-related information to control path planning of a primitive action

Instead of complex and application-specific geometric constraints, this thesis work considers spatial constraints in the modeling of task-related information.

Facing the limitations of existing ontologies of task-related information, there lacks an evolvable ontology of task-related information that can be easily updated according to the task to be performed. The use of such an ontology in the path planning of a primitive action is also out of reach. Different challenges have been identified:

- **An ontology of 3D environment where a simulated manipulation task takes place.** Different natures of environment information (i.e., semantics, topology and geometry) of the rigid bodies and the free space models should be considered. The necessity of modeling the heterogeneous environment information in an ontology is to respond to the requirement of interpreting a primitive action, so that, e.g., the final location of a manipulated object can be determined, task-related geometric constraints can be inferred. Different expertise should be considered in different task. In our case, the semantics of an environment must be adapted to the task processed by the application. Therefore, this modeling should carefully consider and distinguish the information related to a domain and those related to an application context.
- **An ontology of action-specific knowledge.** The modeling of a primitive action and its associated spatial constraints have to be defined. The modeling of a path planning query and its associated geometric constraints should also be defined. The modeling of a primitive action of a task plan should be application-independent so that different primitive actions can be instantiated in the ontology.
- **The strategy of using ontologies of task-related information in the path planning of a primitive action of a task plan.** A path planning query should be generated from a primitive action of a task plan. The start and the goal locations, as well as task-related geometric constraints, have to be generated.

2.3 CONTRIBUTIONS

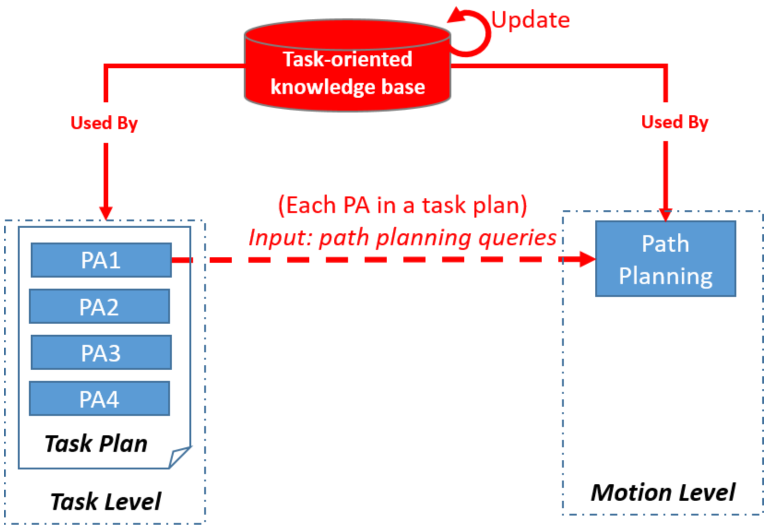


Figure 10 Contributions

Facing these challenges, we proposed an ontology-based approach to use task level information to specify a path planning query for a primitive action of a task plan. The evolvable ontologies of task-related information should be built. The method of using these ontologies to generate a path planning query for a primitive action of a task plan has to be studied (Figure 10). Therefore, the contributions of the work are based on two main aspects:

1) An ontology of 3D environment where a simulated manipulation task takes place (Chapter 3)

This ontology centralizes the heterogeneous environment data (i.e., semantics, topology and geometry) for the rigid bodies and the free space model. This ontology is made of:

a) The conceptualization at the semantic level: The knowledge at the semantic level describes not only the types of rigid bodies, places and borders but also their properties. Thanks to different levels of abstractions, it is scalable and can be enriched using domain- or application-related concepts according to manipulation tasks to be performed.

b) The conceptualization at the topological level: It models places and borders of an environment and their connectivity, defined by [Cailhol 2015].

c) The conceptualization at the geometric level: Different geometric modeling techniques of CAD models are considered, since rigid bodies are built on CAD models. Cell decomposition technique is used to model the geometry of the free space model.

2) An ontology-based approach to generate a path planning query for a primitive action of a task plan (Chapter 4)

This approach concentrates on the path planning of a primitive action of a task plan. A path planning query should be generated from the primitive action, using ontologies of task-related information. The start and the goal location, as well as the task-related geometric constraints, should be determined.

This thesis work proposes to model the primitive action and its associated path planning query in an ontology. In order to facilitate the constraint definition more intuitively, this thesis work uses '*Spatial Constraint*' to restrict the relative position and orientation between two rigid bodies or a rigid body and a place, and models it in the specification of a primitive action.

By exploring the instantiated ontology of 3D environment and the instantiated primitive action, this thesis works studies the strategy to

- to generate the start and the goal locations of the path planning query from the primitive action.
- to generate geometric constraints associated with the path planning request of the primitive action from the '*Spatial Constraints*' of the primitive action.

The results of this thesis work are presented in Chapter 5. Two different scenarios are used. In either scenario, firstly, the verification and the validation of ontology models will be presented; the generation of a path planning query for a primitive action of a task plan is discussed; finally, taking this path planning query as input, the path planning results are compared.

Chapter 3 An ontology of 3D environment where a simulated manipulation task takes place (ENVOn)

3.1 INTRODUCTION

A key part of the task-related information (useful in the path planning of a primitive action) concerns the 3D environment where a simulated manipulation task takes place. As mentioned in section 2.2.4.1, the 3D environment information consists of different levels of abstraction (i.e., semantics, topology and geometry) concerning both the rigid bodies and the free space models.

The interest of modeling and using such heterogeneous environment information in the path planning have been explored by the robotics community (section 2.2.2.1). However, the modeling of the heterogeneous environment information is considered separately in each level. Such discrete modeling architecture does not allow fast queries of any environment information, such as "what is the central axis of *Object A* or *Hole B*?". Also, these models are not semantically defined in a formal way, and thus they lack inference support to determine, such as "whether *Object A* fits *Hole B*".

Considering the heterogeneous environment information in a unified knowledge model is vital to process a primitive action of manipulating an object, such as finding the start and the goal locations and inferring potential geometric constraints. Although ontology models have been developed to conceptualize different levels of 3D environment information (section 2.2.5.2), no work has considered them jointly.

To address such problem, in this chapter, we propose an ontological knowledge base composed of the semantic, topologic and geometric information of both the rigid bodies and the free space models. This environment ontology is inspired from the multi-level environment model [Cailhol et al. 2019]. Currently, this chapter is organized in the following sections:

Firstly, in section 3.2, the requirements that the ontology of 3D environment must answer is presented, following a set of competency questions in order to verify the correctness of the ontology and to validate whether it can answer to the need of querying any environment information. In the next step, the implementation of the ontology model of 3D environment is presented (section 3.3). Different levels of environment information of both the rigid bodies and the free space models are respectively considered. The validation and verification of the proposed ontology of 3D environment will be discussed using two use-case scenarios in Chapter 5.

3.2 REQUIREMENT ANALYSIS AND COMPETENCY QUESTIONS

Firstly, let us have a brief discussion to explore the benefits of considering the semantic, topologic and geometric information jointly in an ontology to simulate manipulation tasks.

Imaging a scenario composed of a "*Cylindrical*" object (*cylinder_obj*) with a radius (*radius_obj*), a Panel (*panel*), two "*Cylindrical*" holes (*hole1, hole2*) with different radius (i.e., *radius_hole1, radius_hole2*) and a "*TriangularPrism*" hole (*hole3*) on the Panel (*radius_hole1 > radius_obj; radius_hole2 < radius_obj*).

In order to correctly process a primitive action of "*Insert (cylinder_obj, panel)*", the path planning system must be able to answer to the questions of

- querying the holes in the panel, i.e., *hole1, hole2, hole3*; they belong to "*Place*" constructed at the topologic level.
- determining the holes with the right shape, i.e., *hole1, hole2*; the shape of *cylinder_obj* should match the shape of holes, i.e., *Cylinder* rather than *TriangularPrism*; such information belongs to the semantics associated to object and holes.
- determining the hole with the right size, i.e., *hole1*; *cylinder_obj* cannot be inserted into *hole2* because *radius_hole2 < radius_obj*; the geometric information is mandatory in solving such issue.

The existing environment models based on different layers consider the semantic, topologic and geometric information so separately that additional (often hard-coded) functions have to be developed in the path planning system to process these questions. These functions have to be adopted into applications and sometimes are difficult to be changed.

An ontology of 3D environment allows to tightly connect different layers of 3D environment information, so that a path planning system can use the knowledge querying and reasoning on this ontology to develop the desired functions without additional code (e.g., Java). The inference rules realizing the functions can be easily changed and adapted to the targeted applications. This loosely-coupled architecture between the knowledge model and the path planning system enhance the reusability of the planning algorithm (task or path) in different applications or tasks. Last but not least, in this thesis work, such an ontology opens the possibility to study the use of the task-related information (e.g., finding the goal to reach, inferring the geometric constraints to be obeyed) in the path planning of a given primitive action of a task plan.

A. *The objective of the ontology*

The ENVOn should be able to allow:

- Fast query of any environment information, for example, "where is *Object A*?", "Does *Object A* is inside of *Place B*?", "What is the central axis of *Object A*?", "What is the sweeping surface of *Object A*'s volume?", "Does *Object A*'s volume is *Cylinder*?", "What is the shape of *Object A*?", "What is the top face of *Object A*?", etc.
- Inferring implicit information, such as "Whether *Object A* fits *Place B*?", "Which hole should *Object A* be inserted?"

B. *The requirements of the ontology*

This ontology should capture the core notions and relations related to a 3D environment where a simulated manipulated task takes place, i.e., semantics, topology, and geometry of both the rigid bodies and the free space models. This ontology should reuse and extend the concepts already defined in the multi-level environment model [Cailhol et al. 2019]. This ontology should also reuse or map to the terminology defined in existing standards or ontologies related to the modeling of a manipulation environment, such as the geometries of CAD models defined in the STEP standard [Clark et al. 1995].

C. *Competency Questions*

Since 1995, competency questions (CQs) have been recognized as a common technique of defining the specification of an ontology [Grüninger et al. 1995]. The CQs consist of a set of questions that an ontology must answer [Karray et al. 2019]. Regarding the objective and the requirement of the ontology to be developed, we expect this ontology can answer the competency questions (CQs) like in Table 10:

Table 10 Competency Questions

Querying geometric details	CQ1	What is the central axis of (X) or (Y)?
	CQ2	What is the opening direction of (X) or (Y)?
	CQ3	What is the pointing direction of (X) or (Y)?
	CQ4	What is the volume of (X) or (Y)?
	CQ5	What is the sweeping plane of (X) or (Y)'s volume?

	CQ6	What is the symmetric vector of the sweeping plane of (X) or (Y)'s volume?
	CQ7	What is the sweeping direction of (X) or (Y)'s volume?
	CQ8	What is the central axis of (X) or (Y)?
	CQ9	What is the origin of (X) or (Y)?
	……	……
Localization	CQ10	Where is (X) in the 3D environment?
	CQ11	Which places (Y) are inside a rigid body(X), such as "*Panel*"?
	……	……
Navigation	CQ12	Does (Y) is the right hole to insert (X)?
	CQ13	Does (X) fit the hole (Y)?
	CQ14	Which places (Y) are the least complex to across?
	……	……

(X: Rigid Body, Y: Place)

3.3 IMPLEMENTATION: ENVOn

This section describes in detail the proposed ontology of 3D environment. Section 3.3.1 demonstrate the general architecture of the proposed ontology. The key concepts at each level of the simulation environment and their relations are discussed. Afterward, section 3.3.2, 3.3.3 and 3.3.4 respectively discuss the conceptualization of geometric, topologic and semantic information of a 3D environment.

3.3.1 The general architecture of the ontology model

As mentioned at the beginning of this chapter, different levels of environment information should be considered in the knowledge modeling of 3D environment where a simulated manipulation task takes place.

Here we should firstly make a difference between the multi-level environment model [Cailhol et al. 2019] and the proposed ontology of 3D environment. Rather than the simple semantic level of the multi-level environment model, we enrich it by conceptualizing all levels of environment information (semantics, topology and geometry) using formal semantics. We still reuse the notion 'semantic level'. Rather than simple textual attributes, the semantic level of our proposed ontology describes the contextual semantics of a 3D environment.

Figure 11 shows the general architecture of the proposed ontology with some key concepts and relations put on each level. A key objective of building the ontology of 3D environment is to have a common vocabulary to be reused by different applications concerning manipulation tasks. The knowledge at each level should be easily extracted, updated and reused by other domain ontologies. Therefore, we consider a modular architecture of the ontology model, where each level (Figure 11) represents an individual module. In our design, three different modules are proposed:

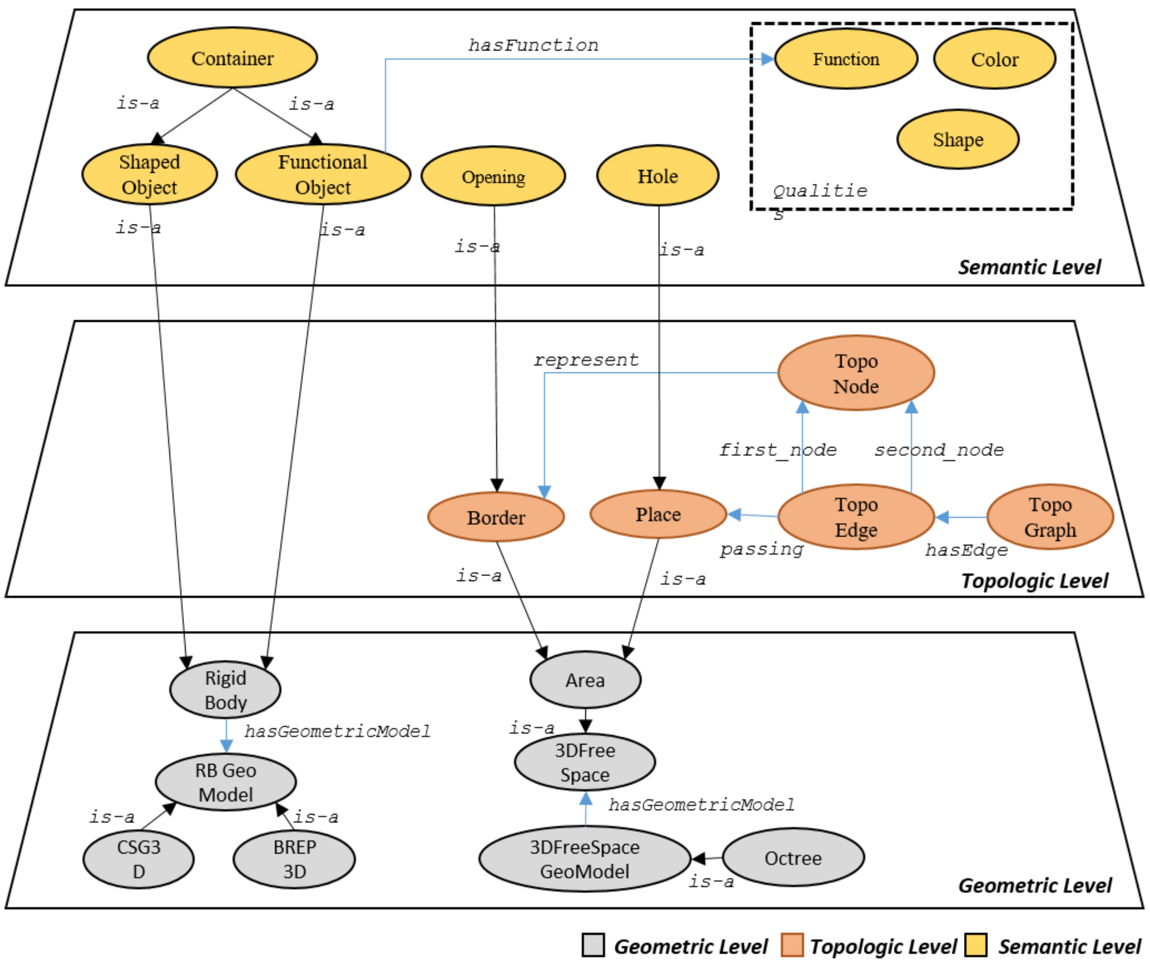


Figure 11 The general architecture of the ontology of 3D environment

- The geometry description module: This module groups the concepts and relations related to the geometries of the rigid bodies and the free space module. '*RB Geo Model*' consists of two possible geometric models (i.e., CSG and BREP) of rigid bodies (*Rigid Body*) based on CAD. '*3D Space Geo Model*' concerns the cell decomposition (*Cell decomposition 3D*) of the free space model (*3D free space*). '*Area*' represents a bounded volume of '*3D free space'*.

- The topology description module: This module describes places (*Place*) and borders (*Border*) identified in the 3D simulation environment. It also illustrates their connectivity by constructing a topological graph (*TopoGraph*).
- The semantic description module: This module provides the semantic description of rigid bodies (*Rigid body*), places (*Place*) and borders (*Border*). Such a kind of description includes the potential taxonomy (e.g., *Container, Opening, Hole*) and the related properties (*Function, Color, Shape*). We must note that this level of information heavily relies on the application (i.e., manipulation task to be performed).

Although the modular structure of the proposed ontology can facilitate the reusability by other domain ontologies, the modules rely so heavily on one another that the reused concepts and relations should be carefully considered. Please note that Figure 11 is only an example of the proposed ontology and more detailed concepts will be given in the following sections. Before entering into details, we firstly introduce some basic concepts in ontology (Table 11).

Table 11 Basic Concepts in Ontology

Concept	Definition
"EquivalentTo" Class Axiom	Class A *EquivalentTo* Class B means that there is an equivalent relation between Class A and Class B. Thus, it is a two directional relations that should satisfy the following conditions: - ∀ a ∈ A, a ∈ B: any instance 'a' of Class A is the instance of Class B - ∀ b ∈ B, b ∈ A: any instance 'b' of Class B is the instance of Class A For example, "Driver *EquivalentTo* (Person and drives Car)": "A driver is a person who drives a car", and then "A person who drives a car is a driver"
"SubClassOf" Class Axiom	Class A *SubClassOf* Class B means that there is an inheritance relation between Class A and Class B. Thus, it is a one directional relation that should satisfy the following condition: - ∀ a ∈ A, a ∈ B: any instance 'a' of Class A is the instance of Class B However, this relation is not enough to clarify that "any instance 'b' of Class B is the instance of Class A". For example, Person is subclass of Animal. We can say that any person is Animal, but we cannot say that any animal is Person.

3.3.2 The geometric description module

The precision and the correctness of the geometric description module of the proposed ontology is essential to query unambiguously geometric information of an environment. Therefore, this module should be able to answer to the CQs that is designed for querying geometric details (Table 10). In this dissertation, we define that an environment where a simulated task takes place is considered as a closed part of 3D Cartesian space cluttered with mobile/fixed obstacles (regarded as rigid bodies). Therefore, the geometric description module should consist of two basic components: the rigid bodies model and the free space model.

Before separately discussing these two components, we will first introduce some common concepts and relations in their geometric description.

3.3.2.1 The Common Concepts and Relations in the geometric description module

A. *The basic mathematical concepts for the geometric description*

The first fundamental part that we need to consider is some mathematical concepts that are the basis for the description of geometric information, for example, the coordinate of *Point*, the direction of *Axis*, the local reference frame of *Rigid bodies*. Figure 12, a conceptual map illustrates the main concepts defined in this part.

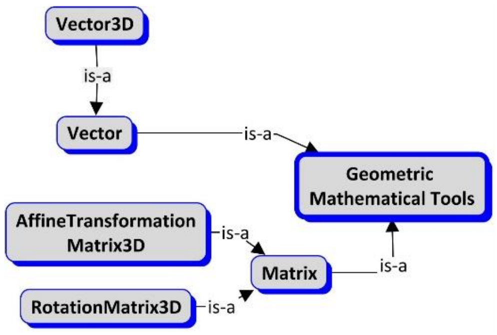


Figure 12 The basic mathematical concepts for the geometric description

The *Vector* concept is formally defined as an element in the commonly encountered Euclidean n-space R_n, given by n-coordinates and specified as (a_1, a_2, …, a_n). An n-dimensional vector is often called n-vector. The *Matrix* concept was introduced in 1851 by [Sylvester 1851] to represent an array of concerned determinants of a system, with m lines and n columns. In linear algebra, it is a very useful tool to represent a linear transformation. In the geometric description of a 3D environment, the *Vector* and *Matrix* concepts are further classified so that, e.g., the coordinate of Points and the rotation or the transformation between two reference frame can be described. We will first provide some description before providing formal definitions in Table 12.

- *Vector3D*: A *Vector3D* is defined in the 3-dimensional Cartesian space, given by the “x, y, z” coordinates and specified as (x, y, z).

- *RotationMatrix3D*: A *RotationMatrix3D* represents a rotation between two frames of reference in 3-dimensional Cartesian space (R_3), any rotation can be given by a composition of rotations of the “x, y, z” axis, given in the form as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

- *AffineTransformationMatrix3D*: An affine transformation matrix is a combination of rotation, translation, and scale. It preserves the collinearity (points on a line remains collinear after transformation) and the proportions on the lines (the midpoint is still midpoint after transformation). Suppose A1 and A2 are two n-dimensional affine spaces R^n, An affine transformation matrix represents a mapping A1 → A2 in a form:

  $$F\ (vector_{a2}) = M \cdot vector_{a1} + b,$$

  where M is the linear transformation from A1→A2 and b is a translation in A2.

  A *AffineTransformationMatrix3D* combines a *RotationMatrix3D* with a 3D translation vector (*Vector3D*) using homogeneous coordinates, in a form:

  $$Aff_{Matrix} = \begin{bmatrix} RotationMatrix3D & Vector3D_{translation} \\ 0 & 1 \end{bmatrix},$$

  with an extended form:

  $$Aff_{Matrix} = \begin{bmatrix} a11 & a12 & a13 & a14 \\ a21 & a22 & a23 & a24 \\ a31 & a32 & a33 & a34 \\ a41 & a42 & a43 & a44 \end{bmatrix},$$

  where a41, a42, a43 = 0, a44=1, b = (a14, a24, a34)

Table 12 Axioms – Basic mathematical concepts

Concept	Axiom
Vector3D	**EquivalentTo:** Vector and (x exactly 1 xsd:double) and (y exactly 1 xsd:double) and (z exactly 1 xsd:double)
RotationMatrix3D	**EquivalentTo:** Matrix and (a11 exactly 1 xsd:double) and (a12 exactly 1 xsd:double) and (a13 exactly 1 xsd:double) and (a21 exactly 1 xsd:double) and (a22 exactly 1 xsd:double) and (a23 exactly 1 xsd:double) and (a31 exactly 1 xsd:double) and (a32 exactly 1 xsd:double) and (a33 exactly 1 xsd:double)
AffineTransformation Matrix3D	**EquivalentTo:** Matrix and (a11 exactly 1 xsd:double) and (a12 exactly 1 xsd:double) and (a13 exactly 1 xsd:double) and (a14 exactly 1 xsd:double) and (a21 exactly 1 xsd:double) and (a22 exactly 1 xsd:double) and (a23 exactly 1 xsd:double) and (a24 exactly 1 xsd:double) and (a31 exactly 1 xsd:double) and (a32 exactly 1 xsd:double) and (a33 exactly 1 xsd:double) and (a34 exactly 1 xsd:double) and (a41 exactly 1 xsd:double) and (a42 exactly 1 xsd:double) and (a43 exactly 1 xsd:double) and (a44 exactly 1 xsd:double)

B. *The conceptualization of geometric primitives*

Geometric primitives were considered as a set of elementary geometric objects [Hoffmann 1989], the combination of which represented the geometry of a solid design (e.g., CAD parts). Irrelevant to any geometric modeling technique, a common set of geometric primitives includes point, curve, and surface [Roth et al. 1993]. Inspired by [Standardization 2014, Kaiser et al. 2018], in order to further separate the geometric data from geometric modeling techniques, 'volume' should also be considered in the set of geometric primitives. In such a representation, 'volume' not only can be used to describe the primitive shapes in Constructive Solid Geometry (CSG) but also can be used to describe the geometry of any closed part of 3D Cartesian space. Figure 13 is the representation of geometric primitives using a conceptual map:

- *Point3D*: A *Point3D* represents a position in a 3D Cartesian coordinate space. A *Vector3D* describes its position.
- *Curve3D*: A *Curve3D* represents a path of a *Point3D* moving through a 3D Cartesian coordinate space.
- *Surface3D*: A *Surface3D* represents a 2D sub-space of a 3D Cartesian space.
- *Volume3D*: A *Volume3D* represents the bounded volume by surface patches. A *Volume3D* might be formed by sweeping a certain *Surface3D* following a certain curve.

Curve3D, *Surface3D*, and *Volume3D* are abstract concepts. They can be further given some free parameters, depending on the kind of curve, surface, and volume that they describe. Before introducing examples of curves, surface or volume, we firstly present '*AxisPlacement3D*' so that we are able to describe where a symmetrical curve, surface or volume is located in 3D Cartesian space. It belongs to '*SupplementaryGeometricPrimitive'* (introduced later).

- *AxisPlacement3D* identifies a reference frame in 3D Cartesian space with a location point (the origin) and three orthogonal axes (i.e., x-, y-, z-axis). The 'Location' attribute describes the position of a symmetrical curve, a surface, and a volume, the 'RefZAxis' attribute describes their reference axis (i.e., their orientation).

A '*CircularCurve3D*' is a *Curve3D* that is located at an *AxisPlacement3D* and has a radius; a '*CircularPlane3D*' is a *Surface3D* that is located at an *AxisPlacement3D*, has a radius and is bounded by a '*CircularCurve3D*'curve; a '*CylindricalVolume*' is a *Volume3D* that has a sweeping plane (*CircularPlane3D*), a sweeping direction (*Vector3D*) and a sweeping length. The formal definitions of these concepts are given in

Table 13. In the process of finding the feasible set of geometric primitives, we realized that they could be used to describe the geometries for both the rigid bodies model and the free space model.

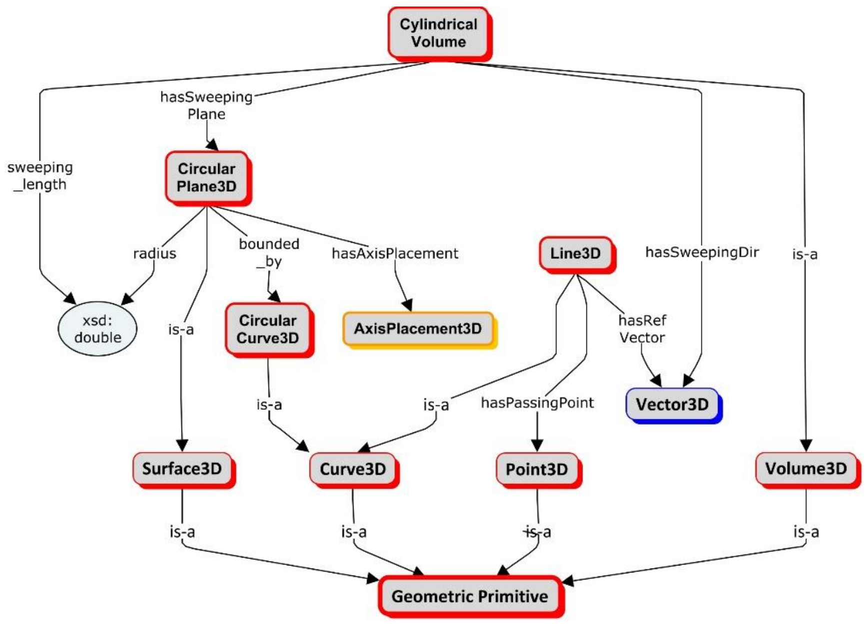


Figure 13 The representation of geometric primitives

Table 13 Some examples of axioms – Geometric Primitives

Concept	Axiom
Point3D	**`EquivalentTo:`** `GeometricPrimitive and (hasPosition exactly 1 Vector3D)`
Curve3D *Surface3D* *Volume3D*	**`SubClassOf`** `GeometricPrimitive`
Line3D	**`EquivalentTo:`** `Curve3D and (hasRefVector exactly 1 Vector3D) and (hasPassingPoint exactly 1 Point3D)`
CircularCurve3D	**`EquivalentTo:`** `Curve3D and (hasAxisPlacement3D exactly 1 AxisPlacement3D) and (radius exactly 1 xsd:double)`
CircularPlane3D	**`EquivalentTo:`** `BoundedPlane3D and (hasAxisPlacement3D exactly 1 AxisPlacement3D) and (radius exactly 1 xsd:double)and (bounded_by exactly 1 CircularCurve3D)`
CylindricalVolume	**`EquivalentTo:`** `Volume3D and (hasSweepingPlane exactly 1 CircularPlane3D) and (hasSweepingDir exactly 1 Vector)and (sweeping_length exactly 1 xsd:double)`

C. The conceptualization of supplementary geometric primitives

Besides the geometric information about their composition, the rigid bodies model and the free space model also possibly contain other geometric properties, such as the central axis, the oriented bounding box. The supplementary geometric primitives present a list of geometric elements that do not composite the geometry of the rigid bodies model or the free space model but assist in describing them. Figure 14 presents 4 different concepts in a conceptual map, where the formal definitions are provided in

Table 14. '*AxisPlacement3D*' is not further presented, since it has been introduced when geometric primitive is being discussed.

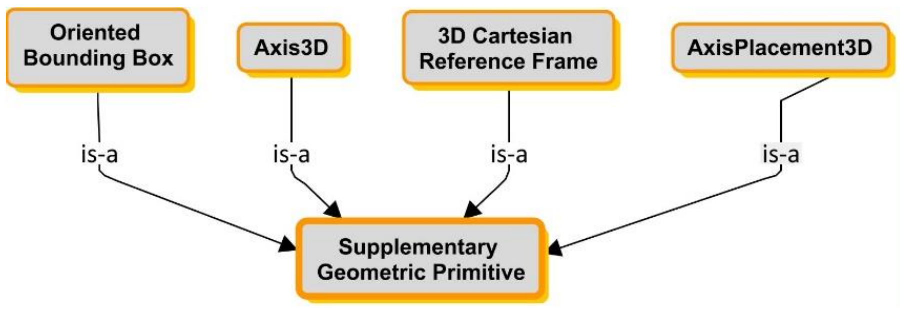

Oriented Bounding Box
Axis3D
3D Cartesian Reference Frame
AxisPlacement3D
is-a
is-a
is-a
is-a
Supplementary Geometric Primitive

Figure 14 The representation of supplementary geometric primitives

Table 14 Some examples of axioms – Supplementary geometric primitives

Concept	Axiom
Axis3D	***EquivalentTo***: `SupplementaryGeometricPrimitive and (hasPassingPoint exactly 1 Point3D) and (hasRefVector exactly 1 Vector3D)`
AxisPlacement3D	***EquivalentTo***: `SupplementaryGeometricPrimitive and (hasOrigin exactly 1 Point3D) and (hasRefXVector exactly 1 Vector3D) and (hasRefYVector exactly 1 Vector3D) and (hasRefZVector exactly 1 Vector3D)`
3D Cartesian ReferenceFrame	***EquivalentTo***: `SupplementaryGeometricPrimitive and (hasAffineMatrix exactly 1 AffineTransformationMatrix3D or hasAxisPlacement3D exactly 1 AxisPlacement3D)`
Oriented BoundingBox	***EquivalentTo***: `SupplementaryGeometricPrimitive and (hasLocalReferenceFrame exactly 1 3DCartesianReferenceFrame) and (hasMinPoint exactly 1 AxisPlacement3D) and (hasMaxPoint exactly 1 AxisPlacement3D)`

- *3DCartesianReferenceFrame*: It is a framework to perform measurements on, such as location, distance, and angle, precisely and mathematically in 3D Cartesian space. It is specified either by an '*AxisPlacement3D*' (an origin point, three orthogonal x, y, z axes), or an affine transformation regarding a world reference frame in 3D Cartesian space.

- *Axis3D*: An *Axis3D* is a *Line3D* to which a point, a curve, a surface or a rigid body is measured, rotated, etc. For example, a symmetry axis of a surface indicates that each side of the axis is a mirror image.
- *OrientedBoundingBox*: An oriented bounding box is the minimum enclosing box for a point set (such as all points of a rigid body), regarding the local reference frame. It is defined by a minimum and a maximum point in the local reference frame.

3.3.2.2 The geometric representation of the rigid bodies model

The rigid bodies model is built upon CAD models. In this dissertation, we regard the geometric models of rigid bodies as the geometry of CAD models. Rather than semantically meaningless (Delaunay triangulated) polygonal meshes, the geometry of CAD models is usually formally defined. By exploring the formalization of CAD models, we can find in the literature two main representations in describing the geometry of CAD models: 1) the surface representation; 2) the volume representation. As we have discussed in the state of the art of Chapter 2, Constructive Solid Geometry (CSG) and Boundary Representation (BREP) are two main modeling techniques respectively in formalizing the volume representation and the surface representation of CAD models.

Indeed, the STEP (Standard for the Exchange of Product model data) format defines a rigorous geometric data formalism for the CAD model, and it has been widely used for the CAD data exchange and storage. Comparing to STEP standard, conceptualizing the geometry models of rigid bodies (i.e., the geometry of CAD models) in an ontology provides the following benefits:

- A rigorous interconnected information model: It allows fast query for the desired geometric information. The implicit geometric information can also be obtained through an inference process. The data consistency can be verified thanks to description logic.
- Linking with other kinds of knowledge: For example, through a proper knowledge mapping, a correlation can be established between geometric models of rigid bodies with other geometric properties, and even with the contextual semantics of rigid bodies.

Indeed, geometric models of rigid bodies in the proposed ontology consider only simple geometries at the moment. For example, only simple kinds of *Surface3D* are used, whereas the *NURBS (Non-uniform rational B-spline) surface* [Piegl et al. 1987] has not yet been considered. Moreover, besides the geometric models of rigid bodies, we also introduce some common geometric properties related to rigid bodies, such as the central axis, the oriented bounding box, the origin of a rigid body. Figure 15 shows the main concepts and relations involved in the geometric representation of the rigid bodies model. The formal definitions are given in Table 15. We should notice that the concepts with yellow, red and blue background color have already been introduced in section 3.3.2.1. They are used here so that geometric properties of rigid bodies can be properly defined.

- *RigidBodyGeometricModel*: A *RigidBodyGeometricModel* describes how a rigid body is geometrically composed. This thesis work uses CSG and BREP as two main modeling techniques to describe the geometric models of rigid bodies, since the rigid bodies model is based upon CAD models.
    - *BREP3D*: It concentrates on the boundary description of a rigid body ('bounded_by' attribute). '*Solid_Boundary*' describes the whole boundary of a rigid body. '*BREP_Face*', topologically, represents an oriented 2D-manifold in 3D Cartesian space on the '*Solid_Boundary*' of a rigid body, and it is geometrically described by a *Surface3D*.
    - *CSG3D*: It describes the volume representation of a rigid body. The *CSG3D* of a rigid body is constructed using a set of standard primitives (*CSG_Primitive*) and Boolean operations among them. A *CSG3D* representation contains the top-level root *CSG_composite* ('root_composite' attribute).

RigidBody: A *RigidBody* represent any (fixed or mobile) obstacles in a 3D Cartesian space, with no deformation allowed. It contains one or more *RigidBodyGeoemtric Models* to describes its geometric composition. A *RigidBody* has a *Point3D* as its origin to identify its position in the world reference frame, and a *3DCartesianReferenceFrame* describes the local reference frame of the *RigidBody*; some geometric properties of the RigidBody are described in the local reference frame, such as the central axis (*Axis3D*), the oriented bounding box (*OrientedBoundingBox*), the geometric models (*RigidBodyGeometricModel*) and the standard form. We consider the standard form of a *RigidBody* as the volume (*Volume3D*) bounded by the '*Solid_Boundary*' of the *RigidBody*.

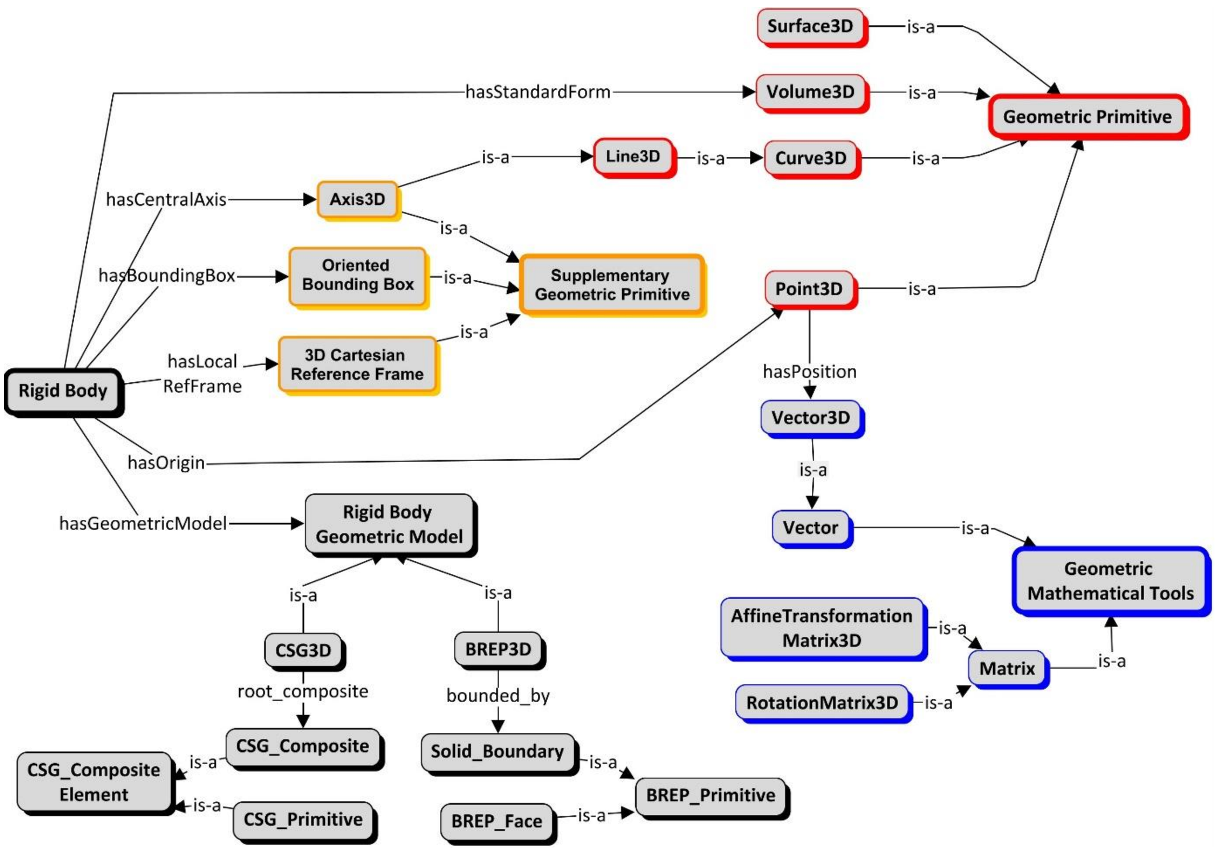


Figure 15 The geometric representation for the rigid bodies model

Table 15 Some examples of axioms – The rigid bodies model

Concept	**Axiom**
CSG3D	**`EquivalentTo:`** `RigidBodyGeometricModel and (root_composite exactly 1 CSG_Composite)`
BREP3D	**`EquivalentTo:`** `RigidBodyGeometricModel and (bounded_by exactly 1 Solid_Boundary)`
Rigid Body	**`SubClassOf:`** `(hasStandardForm exactly 1 Volume3D) and (hasCentralAxis exactly 1 Axis3D) and (hasBoundingBox exactly 1 OrientedBoundingBox) and (hasLocalReferanceFrame exactly 1 3DCartesianReferenceFrame) and (hasGeometricModel exactly 1 RigidBodyGeometricModel) and (hasOrigin exactly 1 Point3D)`

3.3.2.3 The geometric representation of the free space model

The geometric representation of the free space model is mainly discussed in the robotic literature and it has been used in different robotic applications, such as navigating a robot in an indoor household environment or manipulating an object to be assembled without collisions. As we have discussed in the state of the art of Chapter 2 (section 2.2.4.1), this thesis work focuses on decomposing the free space model into a set of smaller geometric cells using cell decomposition techniques. Such a representation allows to characterize the geometric volume of the free space model, or even more, to easily find the part of the free space model (volume) in which a simulated task is interested, such as the part that belongs to a hole where a screw should be inserted. Similar to the rigid bodies model, the geometric representation of the free space model also contains other geometric properties besides its geometric model.

Figure 16 shows the main concepts and relations involved in the geometric representation of the free space model. The formal definitions are given in

Table 16.

- *3DFreeSpaceGeometricModel*: It describes how a closed-part of 3D Cartesian space is geometrically composed. In this thesis work, we use a cell decomposition technique (Octree) so that the geometric volume of different locations can be identified, since we focus on simulating tasks of manipulating objects into the desired goal location.
- *CellDecomposition3D*: The *CellDecomposition3D* is a method to decompose a closed part of 3D Cartesian space into several smaller geometric cells.
- *Octree*: *Octree* is a well-known cell decomposition method to subdivide a cuboid-formed, closed part of 3D Cartesian space into smaller cuboids until the predefined depth is reached. Since *Octree* only divides those geometric cells overlapped by obstacles, it is an unbalanced tree.
- *OctreeNode*: Each cell in the *Octree* is called an *OctreeNode*. Geometrically, it has a cuboid volume.
- *3DFreeSpace*: It represents any part of the 3D Cartesian space that is free. Similar to the *RigidBody* geometric representation, a *3DFreeSpace* might have an origin (*Point3D*) to describe its position in the world reference frame, and a local reference frame (*3DCartesianReferenceFrame*) at this origin so that some geometric properties can be locally described, such as an oriented bounding box, a central axis, and a standard geometric form.
- *Area*: An area represents a continuous closed part of *3DFreeSpace* with a collection of common properties. Semantically, it can be further classified, such as kitchen, corridor.

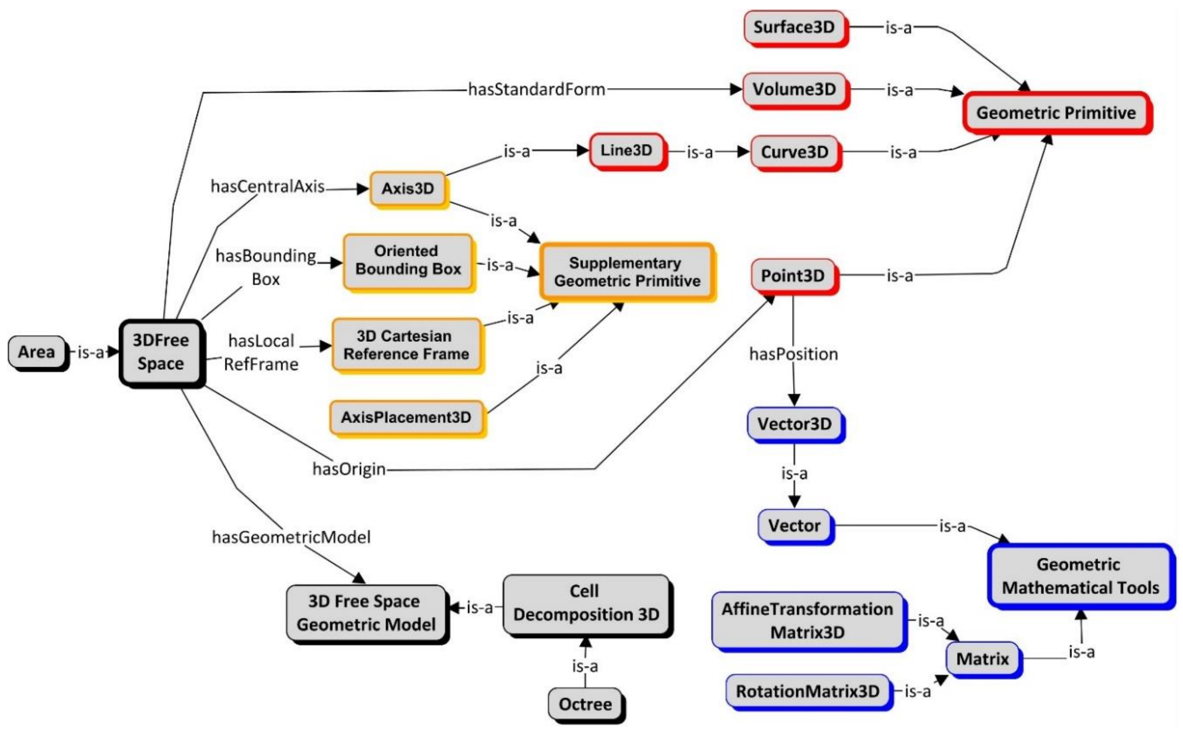


Figure 16 The geometric representation for the Free Space model

Table 16 Some examples of axioms – The free space model

Concept	Axiom
Octree	**EquivalentTo:** CellDecomposition3D and (hasRoot exactly 1 OctreeNode)
3DFreeSpace	**SubClassOf:** (hasStandardForm exactly 1 Volume3D) and (hasCentralAxis exactly 1 Axis3D) and (hasBoundingBox exactly 1 OrientedBoundingBox) and (hasLocalReferanceFrame exactly 1 3DCartesianReferenceFrame) and (hasGeometricModel exactly 1 RigidBodyGeometricModel) and (hasOrigin exactly 1 Point3D)
Area	**EquivalentTo:** 3DFreeSpace and (isCloseBounded exactly 1 true)

3.3.3 The topological description module

Reviewing what we have discussed in the requirement analysis, the ontology of 3D environment should be able to answer some competency questions related to localization and navigation, that are CQ11 to CQ14, such as “Does a *Place* Y is the right hole to insert a *RigidBody* X?”, “Does a *RigidBody* X can reach a *Place* Y from its current location?”. In order to answer such questions, *Places* representing different locations of an environment should be firstly constructed. As we have discussed in the state of the art of Chapter 2 (section 2.2.4.1), the definition of ‘Place’ varies among applications. In the targeted applications of simulating manipulation tasks, the multi-level environment model proposed in [Cailhol et al. 2019] provides a well-structured topological description of an environment where the connectivity between *Places* are illustrated thanks to their *Borders*. A *Border* represents the overlapped *Area* between two *Places*. The adjacency between the identified *Borders* allows to construct a *TopologicalGraph* that describes the connectivity of an environment (*Places*) where a simulated manipulation task takes place. Currently, the topological description module concerns the free space model. The topological description of the rigid bodies model (i.e., the connectivity of surfaces of a rigid body) is currently out of our scope. Figure 17 shows the main concepts and relations involving in the topological description module. Some formal definitions are given in Table 17.

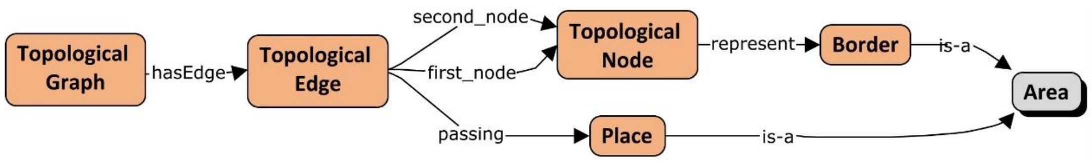


Figure 17 The topological description module

- *Place* and *Border*: A *Place* represents a location in the environment. It is further classified regarding some concrete properties. For example, bedroom, kitchen and bathroom are different *Places* specified by their functionality (i.e., sleeping, cooking, bathing respectively). A *Border* is the overlapped *Area* between two *Places*, such as the entrance between a room and a corridor.
- *TopologicalNode*: It is a basic element of the *Topological Graph* to represent a *Border*.
- *TopologicalEdge*: A *TopologicalEdge* connects two *TopologicalNodes* passing through a certain *Place*.
- *TopologicalGraph*: It is a concept describing the general connectivity of the free space model. It contains a number of *TopologicalEdges* to describe connections between different *Borders*.

Table 17 Some examples of axioms – The topological description module

Concept	**Axiom**
Place *Border*	`SubClassOf: Area`
Topological Node	`SubClassOf: (represent exactly 1 Border)`

Topological Edge	`SubClassOf:`
	`(first_node exactly 1 TopologicalNode) and (second_node exactly 1 TopologicalNode) and (passing exactly 1 Place)`

TopologicalGraph	`SubClassOf: (hasEdge min 0 TopologicalEdge)`

3.3.4 The semantic description module

In manipulation task simulations, an involved human operator rarely sees a simulation environment from the geometric point of view. Rather than millions of faces, he/she considers

RigidBodies, *Places* and *Borders* with contextual semantics. For example, in an indoor household environment, the involving *RigidBodies* can be a table, a door or a booklet; the identified *Places* can be bedrooms or corridors; their *Borders* can be the entrances of bedrooms. In the construction of the semantic description module, we realized that the contextual semantics of *RigidBodies, Places* and *Borders* heavily relies on the application to which a simulated manipulation task targets.

In the knowledge modeling literature, a distinction between ontologies and contexts have already been discussed in various kinds of research works [Bouquet et al. 2003, Firat 2003, Stuckenschmidt et al. 2004, Benslimane et al. 2006, Challam et al. 2007, Gursel et al. 2009]. In the research work of [Xu 2017], such distinction has been formalized as "ontologies are shared models of some domain that encodes a common view of different parties, whereas contexts are local and non-shared models that encode a party's view on a particular domain".

Following the idea of separating an ontology from its context, the contextual description module of the ontology of 3D environment consists of two major parts: context-independent semantics and context-specific semantics. Table 18 provides some formal definitions of the concepts involving in the semantic description module.

<u>The context-independent semantics</u>

The context-independent semantics consists of *Qualities* and '*realizable entities*' description of *RigidBodies*, *Places* and *Borders*, as shown in the red dashed box of Figure 18. In the research work of [Arp et al. 2015] that constructs a top-level ontology, a *Quality* of an entity should be fully realized and manifested in the entity without any extra realization process and a *'realizable entity'* of an entity can only be exhibited through certain realization process.

1) Quality

- *Shape*: The *Shape* quality is used to describe the geometric form of *RigidBodies*, *Places* and *Borders*. Two subgroups, which is '*RegularShapeQuality*' and '*IrregularShapeQuality*', are further obtained depending on whether their *Shapes* are regular or not. Typical '*RegularShapeQualities*' are '*RSQ_Cylinder*', '*RSQ_TriangularPrism*', '*RSQ_Cuboid*'.
- *Form Convexity*: If a *RigidBody* is convex, the line segment between any two points (in the interior or on the boundary of the *RigidBody*) should not go outside of the *RigidBody*. Otherwise, it is concavely formed.

- *Object Mobility*: A *RigidBody* can be fixed to the ground and cannot be moved during the whole simulation. Otherwise, it is a mobile obstacle that can be moved.
- *Environment Congestion*: It determines whether a *Place* or a *Border* is narrow regarding the size of the manipulated *RigidBody*. '*Wide*' and '*Narrow*' are two instances.
- *Environment Complexity*: It determines whether a *Place* or a *Border* is complex, e.g., filled with moving obstacles as a dynamic environment. 'Complex' and 'Not complex' are two instances.
- *Presence of Mobile Obstacle*: It is designed to specify whether a *Place* or a *Border* contains moving obstacles. '*Free*' means that no moving obstacle is inside of an *Area*, '*Intersected*' means that one or several moving obstacles is inside of an *Area*, '*Blocked*' means that an *Area* is completely covered by a moving obstacle.

2) 'realizable entity'

- *Color*: A color of a *RigidBody* can only be exhibited through an optical lighting process. Classical colors are '*Black*', '*White*', '*Red*', '*Blue*', '*Green*', '*Orange*', '*Yellow*'.
- *Function*: The desired function of a *RigidBody* is determined at the very beginning of the product design stage. However, it is not an intrinsic property of a *RigidBody*, and it can only be realized during a certain process. For example, the '*Fasten*' function of a screw can only be sensed in an assembly process of a product.

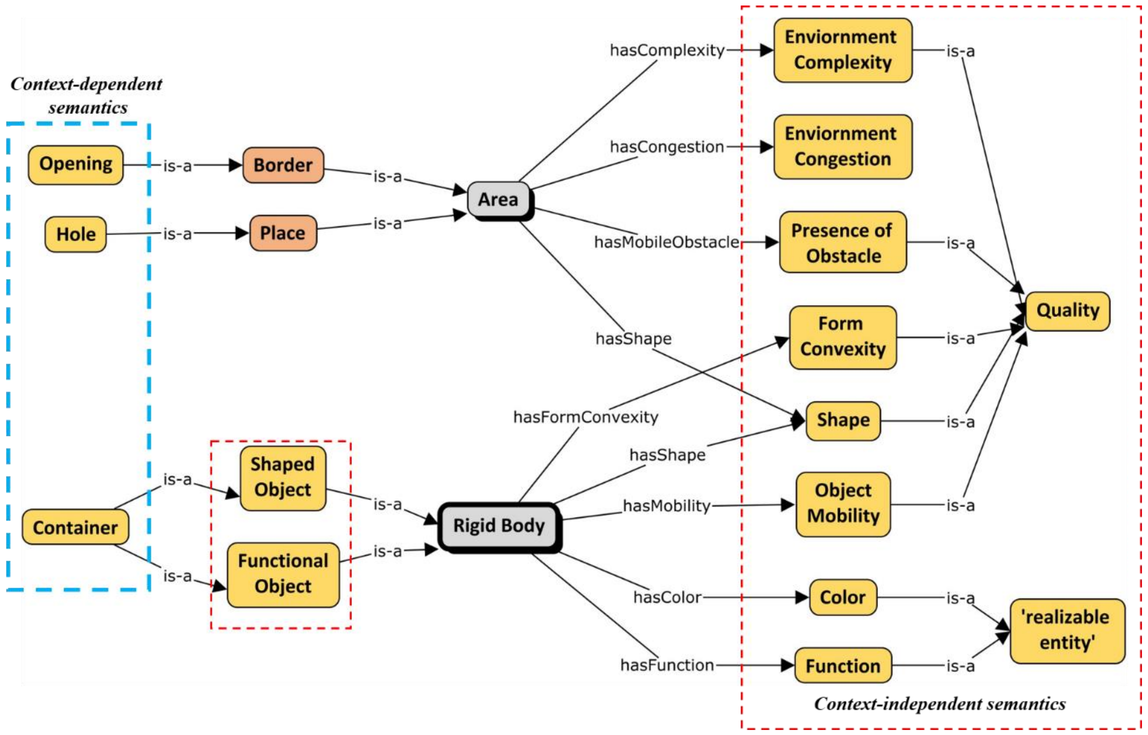


Figure 18 The semantic description module

3) ***The semantic representation of Rigid bodies and Areas***

- *RigidBody*: The *Shape, Color* and *FormConvexity* define how a *RigidBody* looks like. The '*Object Mobility'* defines whether a *RigidBody* is movable. The *Function* defines what a *RigidBody* can do, e.g., cooking, heating.
    - Some context-independent categories of *RigidBodies* can be defined, such as *ShapedObject, FunctionalObject.* These categories can also be further defined according to the domain of simulations. For example, in the construction domain, [Chitkara 1998] categorized the construction projects in three groups: '*Building construction*', '*Infrastructure construction*' and '*Industrial Construction*'.
- *Area*: The *Shape* defines how an *Area* looks like. The *Presence of Mobile Obstacle* defines whether an *Area* contains mobile obstacles (i.e., *Free, Intersected, or Blocked*). The *EnvironmentComplexity* and *EnvironmentCongestion* describe whether an *Area* is difficult to across.

<u>The context-dependent semantics</u>

The context-dependent semantics of *RigidBodies*, *Places* and *Borders* relies heavily on the application that a simulated task handles. The concepts and relations are locally defined. The modeling of the context-dependent semantics is a difficult activity due to that it varies among applications. Currently, the context-dependent semantics is not the focus of this thesis work. We only introduce '*Hole*', '*Opening*' and '*Container*' as local concepts (the blue dashed box of Figure 18), so that the environment information of two scenarios in this dissertation can be instantiated in the ontology.

Table 18 Some examples of axioms – The semantic description module

Concept	**Axiom**

Hole	**`SubClassOf:`**
	`Place and (hasCentralAxis exactly 1 Axis3D)`

Opening	**`SubClassOf:`**
	`Border and (hasOpeningDirection exactly 1 Vector3D)`

ShapedObject	**`SubClassOf:`**
	`RigidBody and (hasShape exactly 1 Shape)`

FunctionalObject	**`SubClassOf:`**
	`RigidBody and (hasFunction exactly 1 Function)`

Container	**`SubClassOf:`**
	`ShapedObject and FunctionalObject and (hasSpaceInContainer exactly 1 Hole) and (hasOpening exactly 1 Opening)`

Area	**`SubClassOf:`**

	`(hasMobileObstacle exactly 1 PresenceOfObstacle) and (hasComplexity exactly 1 EnvironmentComplexity) and (hasCongestion exactly 1 EnvironmentCongestion) and (hasShape exactly 1 Shape)`
Rigid Body	**`SubClassOf:`** `(hasFormConvexity exactly 1 FormConvexity) and (hasColor exactly 1 Color) and (hasFunction exactly 1 Function) and (hasMobility exactly 1 ObjectMobility) and (hasShape exactly 1 Shape)`

3.4 SUMMARY

In this chapter, an ontology of 3D environment where a simulated manipulation task takes place is proposed. We have followed a waterfall-like approach to build our ontology from scratch to evaluation. The evaluation of the proposed ontology will be discussed at the results analysis part of Chapter 5 using two manipulation task scenarios. The proposed ontology integrates the environment information of the rigid bodies and the free space models. Different levels of environment information have been considered, i.e., semantics, topology and geometry. A separation of environment information between these levels allows a better modularization of the proposed ontology. The modules representing different levels can be easily extracted and reused by other domain ontologies; the semantic description module can be easily updated or changed according to the target applications. Indeed, these modules are linked together so that a complete representation of the environment can be obtained, such as:

- A *Place* identified at the topological level consists of a list of octree nodes as its geometric description. Up to the semantic level, a *Place* can be further defined with a specific type and additional semantic attributes (properties).
- A *RigidBody* can contain both geometric models and semantic attributes.

One thing that draws our attention, during the ontology development, is that not all environment information is suitable to be instantiated in an ontology, for example, points and lines of the polygonal meshes of rigid bodies. The number of these points and lines might be large and they are sometimes semantically meaningless. Saving such geometric information in the ontology makes the knowledge base so overstaffed that the knowledge querying and reasoning can be slow, and even sometime impossible. Indeed, such a kind of issue does not only happen in our ontology development. Rather, it is a common issue in the scientific community to build proper ontologies. Moreover, we expect that our ontology of 3D environment can serve as a belief for the planning system so that the ontology can be updated whenever the belief changes: somethings are added, and somethings are deleted. With respect to OWL and SWRL that is monotonic in terms of logics, it is difficult to modify the already constructed ontology.

Currently, our ontology hasn’t taken the iterative environment update into account, and it only concerns the environment state at the moment when a primitive action of manipulating an object takes place. Last but not least, the calculation support using OWL and SWRL is limited [Sánchez-Macián et al. 2007]. In our research, only simple numerical comparisons are used, e.g., to find out whether a ‘cylinder’ object can fit the ‘cylinder’ hole by comparing their radius. Complex mathematical computations are not suitable to be modeled by logics but rather defined by external functions (e.g., Java), however, OWL and SWRL lack of the mechanism to link the predicates with external functions (except simple built-in functions of SWRL [Consortium 2004]).

This is the first contribution of our thesis work. The generation a path planning query of a primitive action discussed in Chapter 4 must be built on this environment ontology. The verification and the validation of the proposed ontology of 3D environment will be performed in Chapter 5.

Chapter 4 An ontology-based approach to generate a path planning query for a primitive action of a task plan

4.1 INTRODUCTION

In Chapter 3, we have presented a domain related ontology that captures the 3D environment information related to a manipulation task. However, in order to generate a path planning query for a primitive action of a task plan, the proposed environment ontology alone is not enough. As mentioned in section 2.2.5.4, we have also to consider the information related to a primitive action to be performed. Such a consideration drives us to develop an ontology of action-specific knowledge, which consists of:

- the specification of a primitive action and its associated spatial constraints
- the description of a path planning query and its associated geometric constraints.

The ontology of action-specific knowledge relies on the ontology of 3D environment where a simulated manipulation task takes place. Using these two domain ontologies as the basis, the presented work in this chapter then discusses an ontology-based method to generate a path planning query for a primitive action of a task plan. In particular, through a reasoning process involving a primitive action instantiated in the ontology of action-specific knowledge, the start and the goal locations, as well as the task-related geometric constraints, can be generated.

This chapter firstly presents a discussion of ontology reasoning based on OWL-DL and SWRL rules (section 4.2). Section 4.3 discusses our ontology-based approach to generate path planning query of a primitive action of a task plan. The ontology of action-specific knowledge is firstly presented, and the generation of a path planning query for a primitive action is discussed later. Section 4.4 presents the implementation of our proposed approach. In the final section (section 4.5), a use case is used to demonstrate how a path planning query is generated from a primitive action.

4.2 A DISCUSSION OF ONTOLOGY REASONING

4.2.1 Inference Process based on OWL-DL

Table 19 presents some necessary notions before discussing OWL-DL in detail.

Table 19 Some necessary notions

Notion	Definition
Σ	An ontology model formalized in OWL-DL
C	Concept C
D	Concept D
a	Instance
C (a)	a is an instance of concept C
⊆	Inclusion, e.g., C ⊆ D, if every instance of C belongs to D
∈	'Belong to', e.g., a ∈ C if 'a' is an instance of C

OWL (Web Ontology Language) is a language designed to formalize the ontology model (i.e. the taxonomy of concepts and relations between them). The OWL-DL is the most widely used one based on description logics (DL). Using logical statements formalized in DL, the DL-based system can provide several reasoning services:

- Consistency checking: It is a problem of checking the model Σ is satisfiable, i.e., whether the model has contradictions in concept definitions. For example, a 'Teacher' cannot be both 'Man' and 'Woman' because 'Man' and 'Woman' are disjoint
- Instance checking: It is a problem of checking C(a) is satisfiable in the model Σ, i.e. whether the instance definition has contradictions in the model Σ. For example, Mary cannot be both 'Man' and 'Woman' because 'Man' and 'Woman' are disjoint.
- Subsumption: It is a problem of checking whether C ⊆ D is satisfiable in the model Σ, such as checking whether 'Four' can be subsumed under 'CuboidObject'
- Instance Classification: It is a problem of checking whether 'a ∈ C' D is satisfiable in the model Σ. For example, 'a' can be classified as an instance of 'Four' if it has cuboid shape and can heat food using 'radiant heat'.

4.2.2 Inference Process based on SWRL Rules

SWRL is called Semantic Web Rule Language. The use of SWRL is the complementary of OWL to provide an expended reasoning capability of knowledge allowing to infer additional information from an ontology model. A key reason for using SWRL rules is to solve the problem that some inferences are not expressible using OWL language. For example, in the inference of the transitive relation noted below:

$$hasFather\ (A, B)\ and\ hasFather\ (B, C) \rightarrow hasGrandFather\ (A, C)$$

OWL is linguistic inexpressiveness about the inference of such transitive relation, mainly because the reasoning based on description logics do not generate new relations.

SWRL language has been formalized in the documentation of the W3C consortium and it is a combination of OWL and RuleML (Rule Markup Language) languages. Similar to the logical axioms in OWL like 'subClassOf' and 'equivalentClassOf', SWRL extends OWL with the rule axioms, noted as:

$$Axiom \coloneqq Rule$$

In its formalization, a rule axiom consists of an antecedent and a consequence. Both antecedent and consequence consist of a (possibly empty) set of atoms. Thus, they can be described as the following abstract syntax.

$$Rule \coloneqq implies\ (antecedent, consequence)$$

$$antecedent \coloneqq \{atoms\}$$

$$consequence \coloneqq \{atoms\}$$

A rule axiom can be informally interpreted as "if antecedent holds true, then consequence holds true". Non-empty antecedent and consequence holding true require that every atom in them holds true. Empty antecedent trivially holds true, whereas empty consequence trivially holds false. Thus, rules with empty antecedent are treated as unconditional facts, whereas such unconditional facts are better to be expressed using OWL itself rather than rule axiom.

Before introducing the typical atoms used in the rule axiom, we will first illustrate some concepts useful in defining atoms.

$$i - object \coloneqq i - variable \mid instanceID$$

$$d - object \coloneqq d - variable \mid dataLiteral$$

Atoms may refer to the instances, data literals (values), instance variables or data variables. Variables are universally quantified possibly with restrictions on them, such as maxCardinality. Such variables are not limited to a singular instance or data literal but a kind of them. The restrictions appearing in the antecedent must also appear in the consequence. For example,

$$Rule \coloneqq \Big(hasBrother\ (a, b)\ and\ restriction\ \big(hasSon\ maxCardinality\ (3)\big)(a)),$$
$$restriction\ \big(hasNephew\ maxCardinality\ (3)\big)(b)\Big)$$

An interpretation of such example is that “a has b as brother, and a has maximum of 3 sons, b, therefore, has maximum of 3 nephews.”. This example includes several people who have the same relations rather than a single individual.

The symbols in Table 20 are used in the description of atoms.

Table 20 Symbols for the atom definition

Symbols	Definition
x, y	Variables representing either variables, OWL individuals or data values.
C	OWL Concept description
P	OWL Property description

The typical atoms used in the rule axiom are listed in Table 21 [Consortium 2004].

Table 21 Typical Atoms in SWRL

Atom	Definition	Form
description (i-object)	i-object is an instance of an OWL Concept	C (x)
dataRange (d-object)	d-object is a data value	C (x)
individualvaluedPropertyID (i-object1, i-object2)	i-object1 has a property relation linked to i-object2	P (x, y)
datavaluedPropertyID (i-object, d-object)	i-object has a data value property relation linked to d-object	P (x, y)
sameAs (i-object1, i-object2)	i-object1 is the same object as i-object2	sameAs (x, y)
differentFrom (i-object1, i-obejct2)	i-object1 and i-objects2 are different objects	differentFrom (x, y)
builtIn (relation (d-objects))	the builtIn relation holds on the interpration of every argument.	builtIn (r, x, ….)

Using the above atoms, a human-readable syntax of SWRL rule can be regarded in the following for

$$Rule := antecedent => consequence$$

, where the antecedent and consequence are both conjunction of atoms a1 ^ a2 ^ a3 ^ …. ^ an and the variables in the atoms are prefixed with a question mark (e.g., ?x).

Therefore, the SWRL rule asserting the 'Grand Father' relation at the beginning of this section can be formalized as:

$$hasFather\ (?\,x, ?\,y)\ and\ hasFather\ (?\,y, ?\,z) \rightarrow hasGrandFather\ (?\,x, ?\,z)$$

Inference Process using SWRL rules extends the reasoning capability of the OWL-DL with rule definitions. All the rule axioms are expressed in terms of OWL description (concepts, instances, properties, data values, etc.). Using reasoners like Pellet and HermiT, SWRL rules provide a much powerful reasoning process than OWL-DL alone, such as:

- Instance reclassification: *Person (?x) ^ Working (?x, 'University') ^ hasWork (?x, 'Teaching') -> Teacher (?x)*
- Property value assignment: *hasFather (?x,?y) and hasFather (?y,?z) → hasGrand Father (?x,?z)*

We should realize that the inference process using SWRL rules is monotonic that:

- Negative atoms are not permitted.
- The retraction or modification of the knowledge is not allowed.

4.3 THE PROPOSED ONTOLOGY-BASED APPROACH

In this part, we will present the general architecture of our proposed ontology-based approach. As shown in Figure 19, our proposed approach consists of two different parts.

- The *Knowledge Base* manages the ontologies of task-related information (i.e., the ontology of 3D environment, the ontology of action-specific knowledge). It also manages all the 3D environment information and the primitive action to be performed that are instantiated in ontologies.
- The *Path Planning* part takes a primitive action request as input through a path planning manager. This primitive action request is then instantiated in the ontology of action-specific knowledge. We should note that spatial constraints are assigned to the specification of a primitive action through an external user interface. A reasoner is invoked to perform the reasoning process of generating a path planning query for this primitive action request. The start and the goal locations, as well as geometric

constraints, are generated. Collision-free trajectories are computed concerning this path planning query.

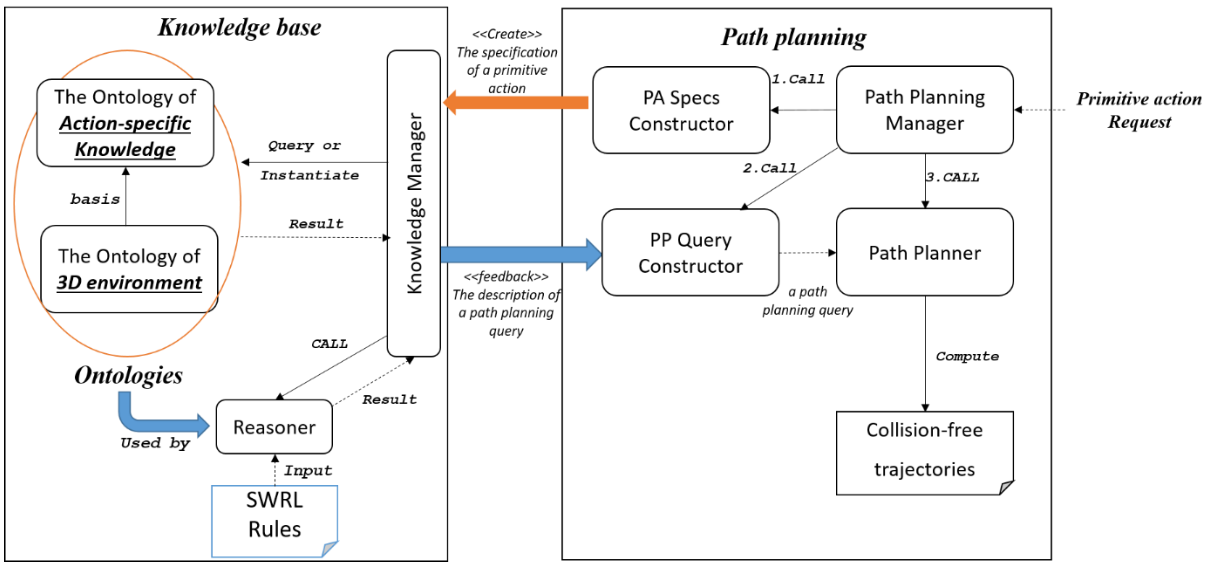


Figure 19 The general architecture of the proposed ontology-based approach

In this section, an ontology model of action specific knowledge for a primitive action of a task plan is presented firstly (section 4.3.1). In order to instantiate a primitive action in the ontology of action-specific knowledge, the operating environment has to be firstly instantiated in the ontology of 3D environment (section 4.3.2). Afterward, section 4.3.3 discusses the generation of a parameterized path planning query from the specification of the instantiated primitive action. In particular, geometric constraints are inferred from spatial constraints associated with this primitive action using some pre-defined SWRL rules. These geometric constraints are then associated with a path planning query. Finally, the multi-level path planner is invoked to generate the corresponding trajectory for the manipulated object, respecting the geometric constraints (section 4.3.4).

4.3.1 An ontology of action specific knowledge

A. Requirement analysis

In the generation of a path planning query for a primitive action of a task plan, it is necessary for us to consider the information related to a primitive action to be performed. As we have discussed in section 2.2.6 and 2.2.7, this thesis work is particularly interested in defining spatial constraints rather geometric constraints with a primitive action to facilitate the constraint definition by common users. Therefore, in the construction of an ontology for action-specific knowledge, we should consider the modeling of the *Spatial Constraints* to be applied in the *Primitive action Specification*.

In this thesis work, *Spatial Constraints* restrict a *Spatial Relation* between two *RigidBodies* or between a *RigidBody* and a *Place*. Through a reasoning process involving primitive action instantiated in the ontology, the start and the goal locations, as well as the *Geometric Constraints*, are generated. Figure 20 shows a general view of the proposed ontology of action-specific knowledge.

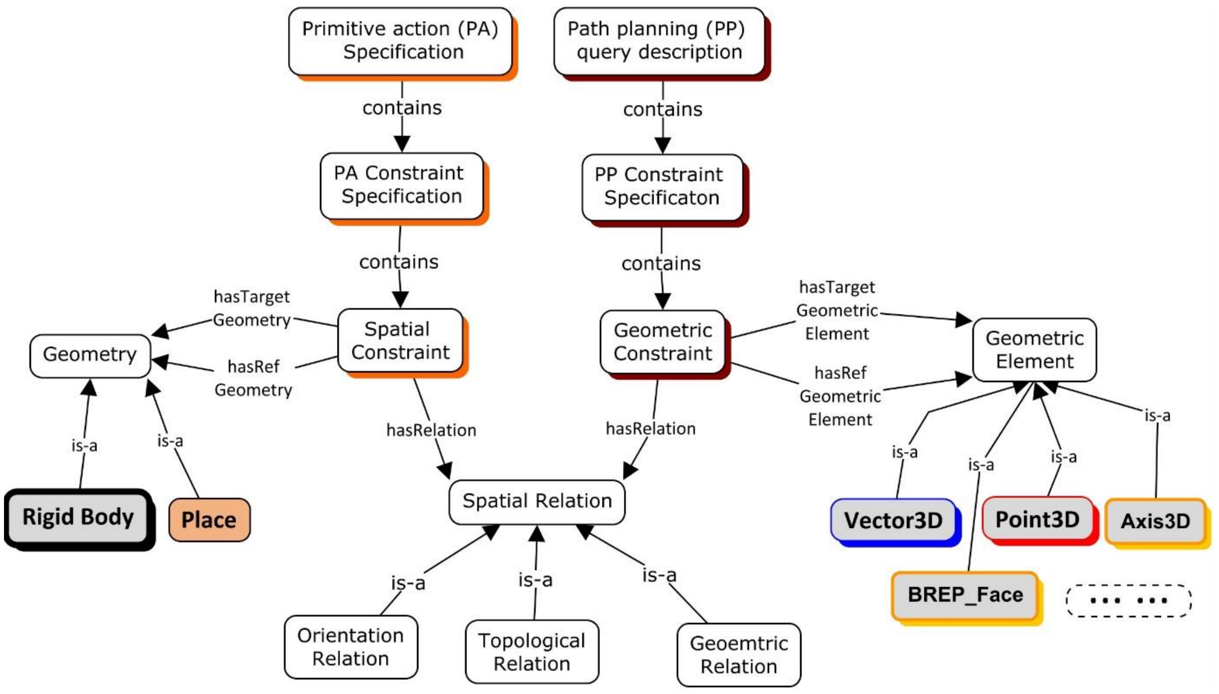


Figure 20 The conceptual map of the ontology of action-specific knowledge

B. Spatial relation

In this thesis work, *Spatial relation* describes the relative location between two *Geometries* (*RigidBody* or *Place*) or between two *Geometries Elements*. Inspired by [Hernandez 1994, Belouaer et al. 2011], we consider *Spatial Relation* into three categories (Figure 20).

Topological Relation describes how the boundaries and the interiors of two *Geometries* (*RigidBody* or *Place*) relate, based on Set Theory. *Disjoint*, *Touch*, *Overlap*, *Inside* and *Equal* are all instances of this concept.

Orientation Relation describes how two *Geometries* (*RigidBody* or *Place*) or two *Geometries Elements* are placed relative to one another. *Top*, *Down*, *Front*, *Back*, *Left*, *Right*, *PointTo*, *ShapeAligned* are all instances of this concept. This relation is described according to a frame of reference.

Geometric Relation describes how two *Geometries Elements* (e.g., *Vector3D*, *Point3D*, *Area3D*, *BREP_Face*) relate to each other. *Coplanar*, *Perpendicular*, *Offset*, *Collinear*, *Against*, *Parallel*, *SameDirection* are all instances of this concept.

C. Spatial and geometric constraint

A *Spatial Constraint* here specifies a *Spatial Relation* between two *Geometries* (*RigidBody* or *Place*). Currently, that is between two *RigidBodies* or between a *RigidBody* and a *Place*. It consists of three parts: 1). A reference *Geometry*, 2). A target *Geometry*, 3). A related *Spatial Relation*

A *Geometry Constraint* here specifies a *Spatial Relation* between two *Geometric Elements*, for example, *Left* (Point1, Point2), *Parallel* (Line1, Line2). It consists of three parts: 1). A reference *Geometric Element*, 2). A target *Geometric Element*, 3). A related *Spatial Relation*.

D. Primitive action specification and Path planning query description

In order to generate a path planning query for a primitive action in a task plan, it is necessary to model a primitive action and its related path planning query appropriately. Figure 20 shows the modeling of a *Primitive action Specification (PAS)* and a *Path planning query description (PPQD)*.

Primitive action (PA) specification (PAS)**:** It consists of four parts:

- A primitive action identity (I): to identify the kind of action that a system is executing, e.g., insert, put.
- Manipulated object (MO): the *RigidBody* that a system manipulates.
- Reference Geometry (RG): a *RigidBody* or a *Place* to which this primitive action references.
- PA Constraint Specification: The *Spatial Constraints* for the primitive action.

Path planning (PP) query description (PPQD)**:** It consists of three parts:

- Manipulated object: the *RigidBody* that a system manipulates.
- Goal Configuration: The configuration to reach
- PP Constraint Specification: The *Geometric Constraints* in a path planning query

4.3.2 The instantiation of a 3D simulation environment

Before a simulation begins, the related environment is instantiated in the knowledge base (Figure 21). The objective of such an instantiation allows not only to specify a primitive action and its associated *Spatial Constraints* but also to support the inference of *Geometric Constraints* for the corresponding path planning query. The instantiated environment data includes:

- The contextual semantics of *RigidBodies*, *Places*, and *Borders* (e.g., the top face of a table, the opening direction of a container).
- The topological representation of the 3D simulation environment (*Places*, *Borders*, *TopologicalGraph*)
- The geometric information of *RigidBodies*, *Places* and *Borders* (e.g., the standard form of a *RigidBody*, the central axis of *Place*)

Instantiating these kinds of environment data in the knowledge base is mandatory so that a *Geometric Constraint* between two *Geometric Elements* can be inferred from a *Spatial Constraint* between two *Geometries* (*RigidBody* or *Place*).

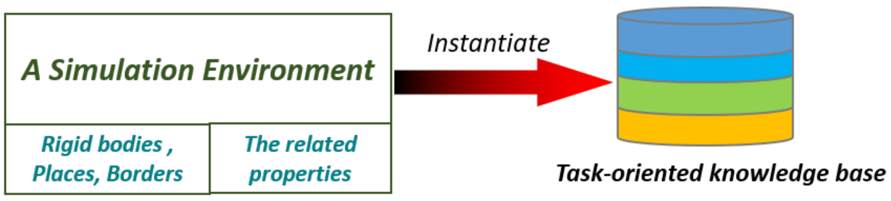


Figure 21 The instantiation of the simulation environment

4.3.3 The generation of a path planning query of a primitive action

As the first step of the generation processs of a path planning query, a *Primitive action Specification (PAS)* is constructed according to the targeted primitive action to be executed. The primitive action identity, the manipulated object and the reference geometry are automatically extracted from the primitive action to be performed. A list of *Spatial Constraints* is manually assigned by human operator and is associated with the *PAS* (Figure 22).

Then, the path planning query constructor (*PP Query Constructor*) takes the *PAS* as input and generates a related *Path planning query description (PPQD)*. The manipulated object in this *PPQD* is directly extracted from the *PAS*. The goal configuration is randomly sampled in the goal region. This goal configuration usually should be compliant to *Geometric Constraints* (that are inferred from the *Spatial Constraints* of *PAS*).

In particular, a *Reasoner* is invoked to generate the related *Geometric Constraints* from the *Spatial Constraints*, using the pre-defined inference rules. We define this constraint generation process as "Spatial-Geometric Constraint Mapping", see from Figure 22. Because inference rules vary among different applications, the next section discusses the constraint generation process in detail, using a specific inference rule as an example in a pen-penbox use case.

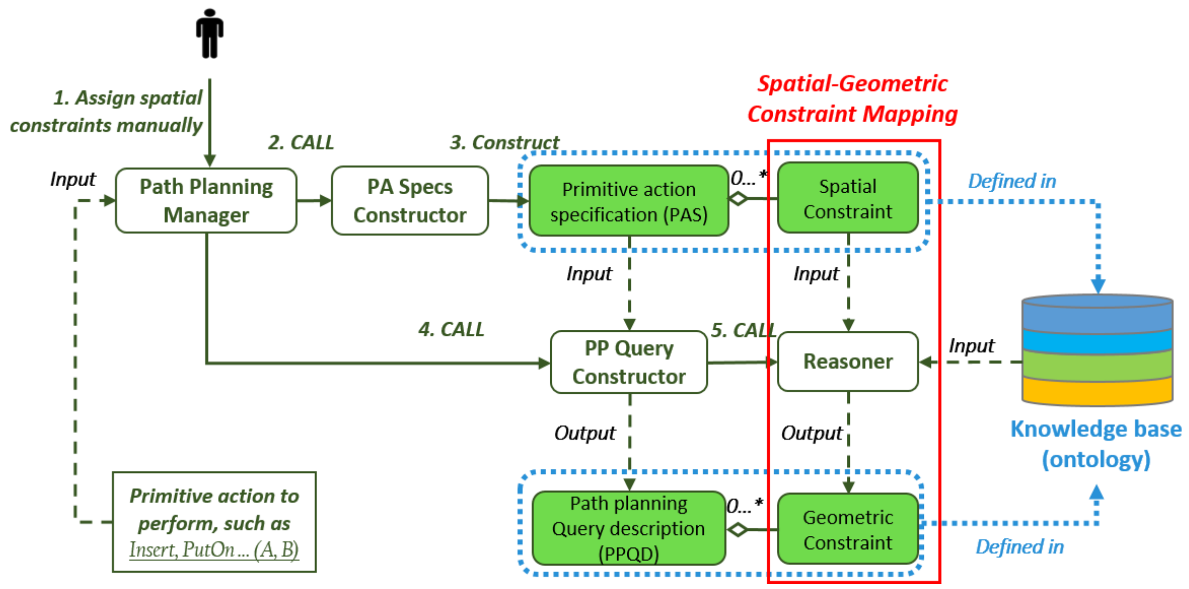

1. Assign spatial constraints manually
Input
Path Planning Manager
2. CALL
PA Specs Constructor
3. Construct
Primitive action specification (PAS)
0...*
Spatial Constraint
Spatial-Geometric Constraint Mapping
Defined in
Input
Input
4. CALL
PP Query Constructor
5. CALL
Reasoner
Input
Knowledge base (ontology)
Output
Output
Primitive action to perform, such as Insert, PutOn... (A, B)
Path planning Query description (PPQD)
0...*
Geometric Constraint
Defined in

Figure 22 The process to construct a PAS and to generate a related PPQD

4.3.4 Interfacing with the multi-level path planning process

The feasibility of a primitive action needs to be validated by a path planner through generating a collision-free trajectory. This thesis work interfaces with the multi-level path planning architecture [Cailhol et al. 2019].

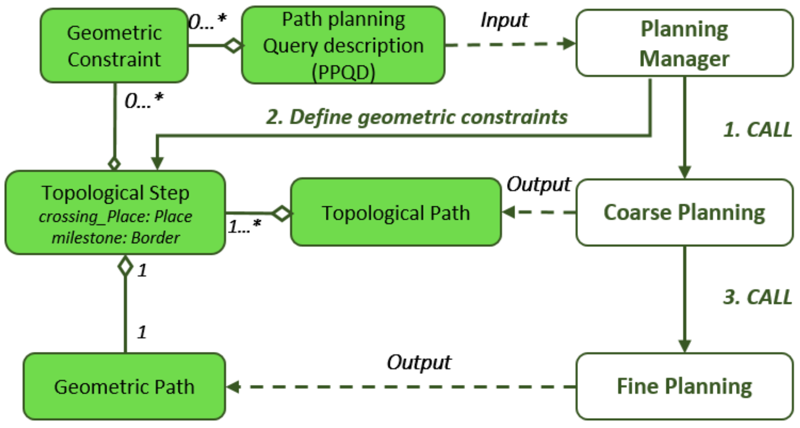


Figure 23 Interfacing with the multi-level path planning architecture

Taking the parameterized path planning query (section 4.3.3) as input (Figure 23), the multi-level path planning process firstly constructs a *Topological Path* (based on the *Topological Graph* defining connectivity of places and borders) to avoid complex and narrow places. A *Topological Path* consists of a set of *Topological Steps* and each *Topological Step* crosses a *Place* and has a *Border* as a milestone to reach. In each *Topological Step*, it then uses a local path planner (i.e., RRT path planner) to construct a collision-free *Geometric Path* in order to control the movement of the manipulated *Rigid Body*.

The geometric constraints are applied differently to a local path planner depending on the crossing place and obstacles within/surround this place.

4.4 IMPLEMENTATION

The ontology-based approach towards coupling task and path planning has been implemented in the dedicated libraries using the object-oriented programming languages JAVA and C++.

This section firstly introduces the principle classes used in our approach through UML (Unified Modeling Language) class diagram. It demonstrates how the knowledge base is linked with the path planning system and presents how they have been used in the path planning process of a primitive action. We will then present certain principle processes of our approach using the UML sequential diagrams.

Open-source libraries have been used for linking the knowledge base and the path planning system [OWLAPI 2018] and for collision detection [FCL 2014].

4.4.1 The static model: Classes and Responsibility

The UML class diagram in Figure 24 presents the principle classes involved in our approach to generate a path planning query for a primitive action.

The multi-level environment model (Blue box in Figure 24): This part reuses the multi-level environment model proposed in [Cailhol et al. 2019].

- The *Rigid body* model describes different 3D scene objects in a simulated environment and the definition of the polygonal models of *Rigid bodies* are realized using the CGAL library.
- The *Free Space* model is represented by a cell decomposition consisting of *Geometric Cells*. In this work, the octree decomposition proposed by [Ladeveze 2010] is used to present the dynamic aspect of the environment (the distinction between cells occupied by the fixed and the mobile *Rigid bodies*).
- The class *Area* aggregates the connected *Geometric Cells* and defines a closed formed of the 3D space that might have some common properties. It has been considered as the geometric anchoring with the related *Geometric Cells* for *Places* and *Borders*.
- The class *Place*, inherited from the class *Area*, represents a portion of 3D space of the simulated environment that has some common properties, such as hole and chamber.
- The class *Border*, also inherited from the class *Area*, represents the transition area between two *Places*.
- The collection of *Places*, *Borders* and their connectivity constitutes a *Topological Graph* of the simulated environment.

Environment Constructor: This class manages the construction of the multi-level environment model. The constructed *Rigid bodies, Places, Borders* and *Topological Graph* are stored in this class, and they will be further used in assisting the multi-level path planning process. Moreover, the *Environment Constructor* is also responsible for the instantiation of the *Rigid bodies, Places, Border* and *Topological Graph* in the knowledge base through the *Knowledge Manager*

Knowledge Manager: This class presents a loosely coupled structure between the knowledge base and the path planning system. It allows adapting the path planning system in various kinds of applications through simply updating the knowledge model without re-developing the path planning system. It encapsulates the methods not only to conserve data in the knowledge base (i.e., Instantiation process) but also to query the stored data from the knowledge base. It also includes the method to call an external reasoner to infer new information from the knowledge base. In this thesis work, the *Knowledge Manager* allows to instantiate the multi-level environment model, it allows to define the properties of *Rigid bodies, Places* and *Borders* through a user interface, and it also allows the *Path Planning Query Constructor* to determine the corresponding *Geometric Constraints* from the *Spatial Constraints* of a primitive action through an inference process.

Path Planning Manager: This class coordinates different stages of the path planning process of a primitive action. It successively starts the construction of a *Primitive Action Specification* from the user's action command, the generation of the corresponding *Path planning query* from the specification, and path computation for the manipulated object regarding the generated *path planning query*.

Primitive action specification constructor: This class manages the construction of a *Primitive Action Specification* regarding the user's action command. The constructed specification aggregates several *Spatial Constraints*. Each *Spatial Constraint* is defined in the high abstraction level symbolic manner to facilitate the interaction with the user. The ensemble of the *Spatial Constraints* aims at controlling the motions of the manipulated *Rigid body*. The *Knowledge Manager* is invoked to instantiate the constructed *Primitive Action Specification* in the knowledge base.

Path planning query constructor: This class generates the corresponding path planning query from a given Primitive Action Specification. It invokes the Knowledge Manager to start the inference process of the knowledge base. In the inference process, Spatial Constraints of a primitive action are mapped into the corresponding Geometric Constraints so that they can be used to control the path computation of the primitive action. As a result, the generated path planning query aggregates several Geometric Constraints.

The multi-level path planning (Blue box in Figure 24): This part also reuses the multi-level path planning architecture proposed in [Cailhol et al. 2019].

- *Semantic Path Planner*: The *Semantic Path Planner* accesses the semantic information of *Places* and *Rigid bodies* through the *Semantic Interpreter*. The information allows to define the cost of *Topological Graph* and the *Planning Strategy*. It coordinates different path planners in the path planning process.
- *Topological Path Planner*: The *Topological Path Planner* explores the *Topological Graph* of the simulated environment and defines a *Topological Path*. The path consists of several *Topological Steps*. In each *Topological Step*, a *Planning Strategy* defines the local path planning method (i.e., RRT) and the *Geometric Constraints* for the geometric path computation. These constraints are extracted from the *path planning query* regarding the crossing *Place* and *Rigid bodies* surrounding/within the *Place*.
- *Geometric Path Planner*: The *Geometric Path Planner* uses the geometry of the *Free Space* model (octree decomposition) and computes a “tunnel” (free *Geometric Cells*) allowing to explorer the *Topological Step*. The explored *Geometric Cells* are limited to the *Place* of the step.
- *RRT Path Planner*: The *RRT Path Planner* is then used to compute a 6D trajectory (*RRT Path*) free of collisions with *Rigid bodies* of the simulated environment. The geometric portion of the *Free Space* model considered by the *RRT Path Planner* is limited to either the *Place* where the *Topological Step* crosses or the “tunnel” exploring the *Topological Step*. The generated collision-free geometric path has to obey the *Geometric Constraints* associated to the *Planning Strategy* of the *Topological Step*. Finally, the ensemble of the geometric paths of all *Topological Steps* belonging to a *Topological Path* constitutes a global geometric trajectory for the manipulated *Rigid body*.

4.4.2 The dynamic model: Principle processes

This section presents how the generation of the path planning query for a primitive action has been implemented regarding the domain UML class diagram in Figure 24. Four different principle processes have been identified.

A. The instantiation of the environment model in the knowledge base

The implementation of the environment model instantiation in the knowledge base is presented using the UML sequential diagram in Figure 25. We can see that the instantiation process is initialized by the construction of the multi-level environment model using *Environment Constructor*. The process consists of two different steps:

Firstly, once the multi-level environment model is constructed, the *Rigid Bodies*, *Places* and *Borders* of the simulated environment and their geometric models are instantiated in the knowledge base through the *Knowledge Manager*.

Secondly, thanks to an intuitive user interface, different properties of *Rigid Bodies*, *Places* and *Borders* can be defined and conserved in the knowledge base. In this step, when the user opens the interface, the conserved *Rigid Bodies*, *Places* and *Borders* and the applicable properties to them are queried from the knowledge base. They are then displayed as the closable lists on the interface through the *User Interface Manager*. The user can choose any one of them (*Rigid Bodies*, *Places*, *Borders* and the associated properties), and the *User Interface Manager* then invokes the *Display Manager* to display it in the virtual environment. The user can also assign properties to a *Rigid Body*, a *Place* or a *Border*, and the *User Interface Manager* calls the *Knowledge Manager* to update them in the ontology.

The instantiation of the environment model in the knowledge base is vital for the definition of the *Path Planning Query* for a primitive action, in particular for generating the related *Geometric Constraints* for the path computation.

B. The construction of a primitive action specification

After the initialization of the environment model (i.e., construction and instantiation), a primitive action command is given by a user. Figure 26 presents the UML sequential diagram of constructing the specification of this primitive action.

The user sets the primitive action to be performed and also the *Spatial Constraints* to be obeyed through an intuitive user interface. Then, with the input data, the *User Interface Manager* informs the Path Planning Manager to start the construction of the *Primitive Action Specification*. Afterward, the *Path Planning Manager* calls the *Primitive Action Specification Constructor* to build the specification. A *Primitive Action Specification* aggregates three different variables: The *Rigid Body* to be manipulated, the reference geometry (*Rigid Body* or *Place*), and a list of *Spatial Constraints*. Finally, the *Primitive Action Specification* is conserved in the ontology through the *Knowledge Manager*.

The instantiation of a *Primitive Action Specification* is prerequisite for generating the corresponding path planning query.

C. The generation of a parameterized path planning query

A key issue of our approach is to generate and to validate motions for a manipulated rigid body involved in a primitive action, by computing collision-free trajectories. Firstly, a *Path Planning Query* has to be generated from the instantiated *Primitive Action Specification*. Figure 27 presents the UML sequential diagram about the generation of the *Path Planning Query* of a primitive action.

Once the *Primitive Action Specification* is constructed, the *Path Planning Manager* calls the *Path planning query constructor* to generate a *Path Planning Query*. In particular, the constructor invokes the *Knowledge Manager* to map the *Spatial Constraints* of a primitive action into the *Geometric Constraints*. Such a mapping process is performed using an inference process calling an external reasoner involving several pre-defined SWRL Rules. Due to the loosely-coupled architecture between the knowledge base and the path planning system, the SWRL rules can be easily updated according to applications. As a result, the generated *Path Planning Query* aggregates a manipulated Rigid Body (in order to find the start configuration), a goal configuration and a list of *Geometric Constraints*. Finally, the generated *Path Planning Query* will be conserved in the ontology.

D. The multi-level path planning process of a primitive action

The implementation of the multi-level path planning process using a parameterized *Path Planning Query* is illustrated in the UML sequential diagram Figure 28. Once a parameterized *Path Planning Query* is generated, the *Path Planning Manager* takes it as input and initializes the path planning process for the primitive action.

By transmitting the *Path Planning Query* to the *Semantic Path Planner*, it updates the *Topological Graph* with the nodes corresponding to the initial and the final configurations of the *Path Planning Query*. Afterward, the cost of nodes and arcs of the *Topological Graph* is determined by evaluating the semantic information associated with *Borders* (node) and *Places* (arc). Once the graph update is complete, the *Topological Path Planner* constructs a *Topological Path*. This *Topological Path* represents the least difficulty (cost) of reaching the goal through exploring the *Topological Graph* using Dijkstra algorithm. The result of this algorithm constructs the *Topological Steps*, and each *Topological Step* aggregates a *Place* to cross (arc) and a *Border* as a milestone to reach (node).

For each *topological step* of a *topological path*, the *Semantic Path Planner* interrogates the *Semantic Interpreter* to define a *Planning Strategy*. This strategy defines not only the choice of *local path planner* but also the applicable *Geometric Constraints*. Firstly, a valid collision-free geometric configuration is necessary to be found at the milestone (*Border*) so that the manipulated *Rigid body* can reach. When the initial and the goal configuration is defined, the *Planning Strategy* is then executed to compute the collision-free geometric path for the manipulated *Rigid body* while compliant to the associated *Geometric Constraints*. In particular, our approach concentrates on the *RRT Path Planner* for the geometric path construction. The found geometric path (*RRT Path*) will then be returned and memorized in the *Topological Step*. Finally, the ensemble of the geometric paths of the *Topological Steps* forms the global geometric path (*RRT Path*) for the *Topological Path*, and it will then be displayed in the Virtools to show the feasibility of the manipulation.

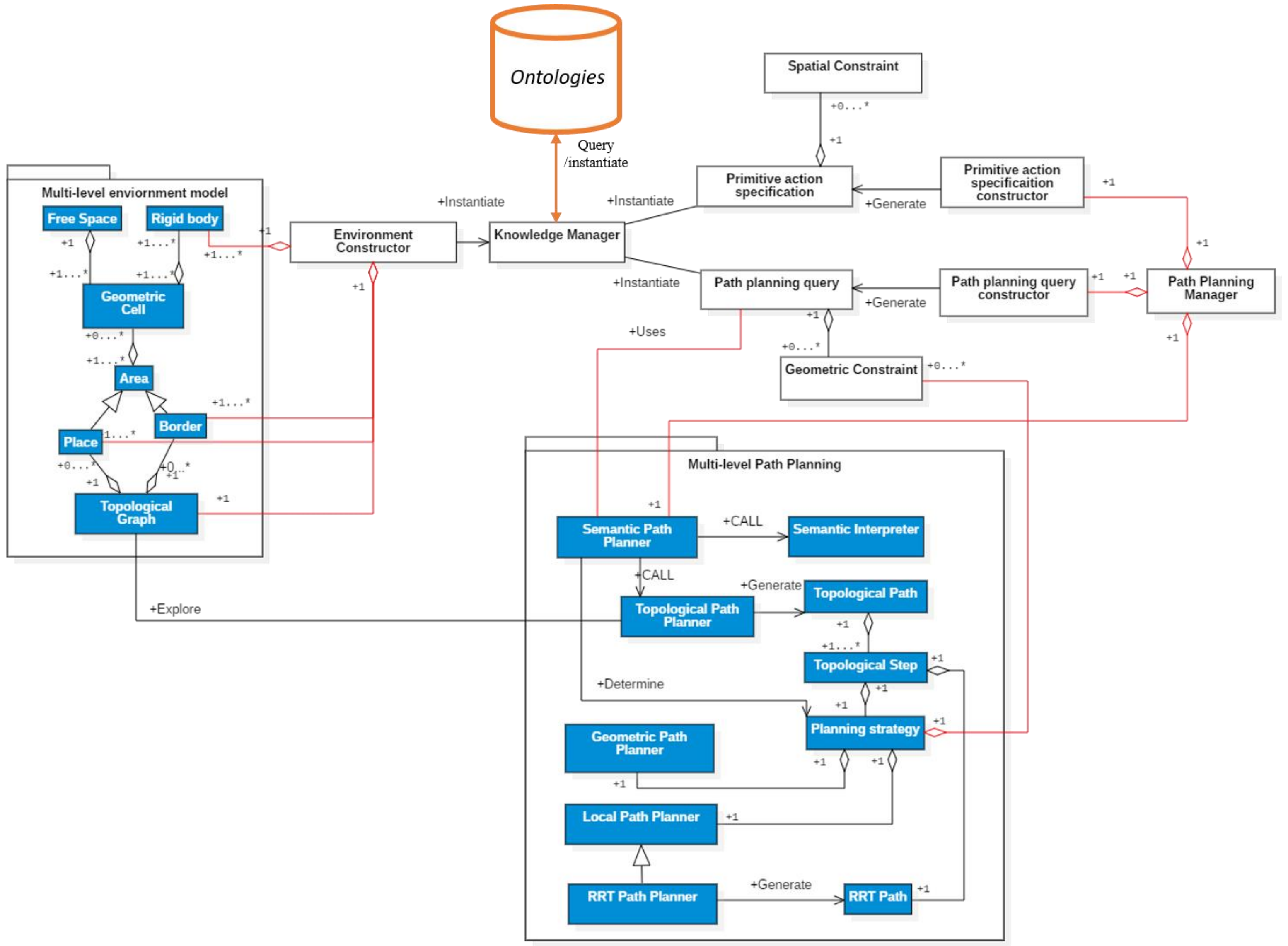


Figure 24 The domain UML class diagram of the ontology-based approach towards coupling task and path planning

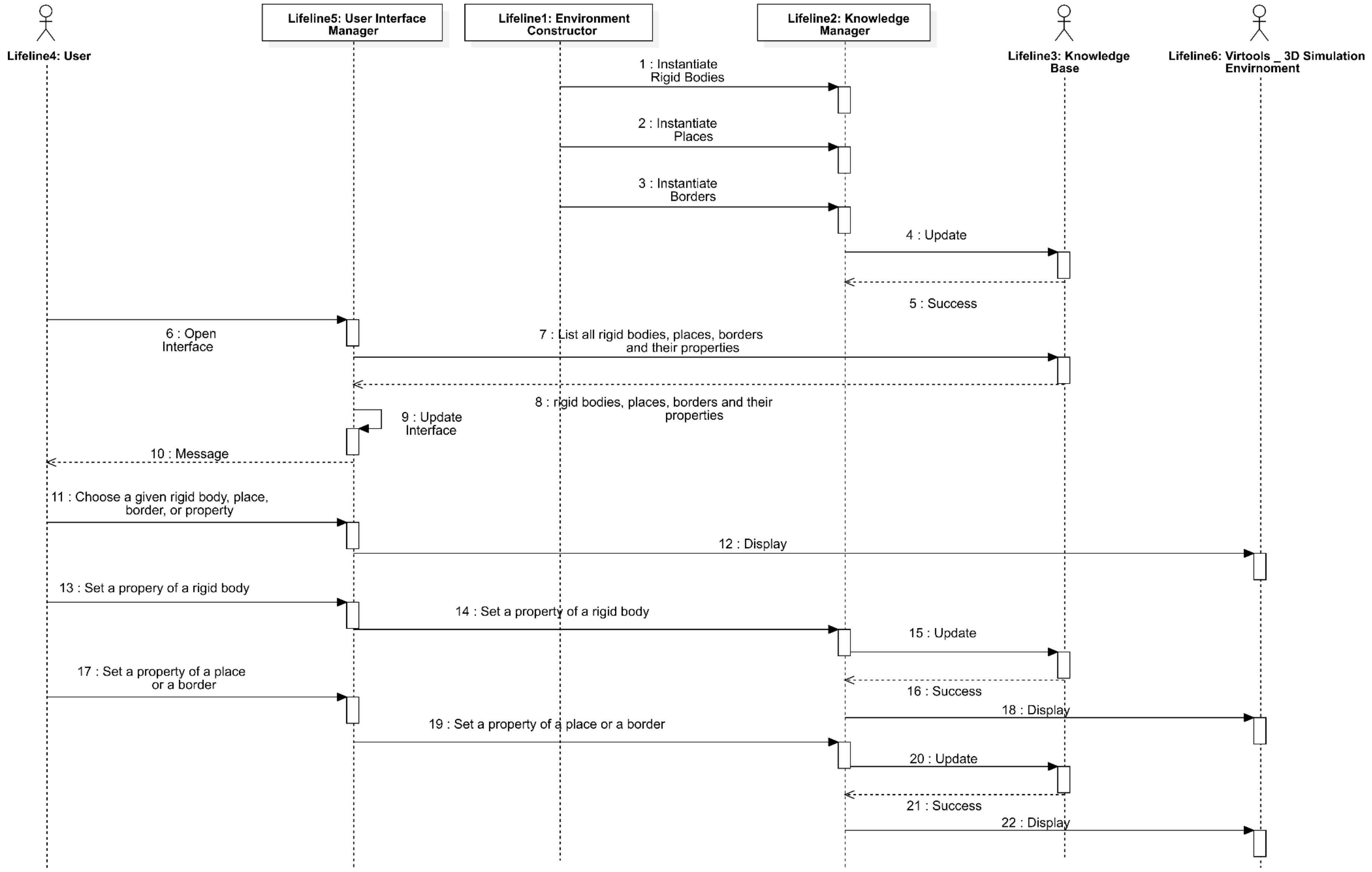

Lifeline4: User
Lifeline5: User Interface Manager
Lifeline1: Environment Constructor
Lifeline2: Knowledge Manager
Lifeline3: Knowledge Base
Lifeline6: Virtools _ 3D Simulation Envirnoment
1 : Instantiate Rigid Bodies
2 : Instantiate Places
3 : Instantiate Borders
4 : Update
5 : Success
6 : Open Interface
7 : List all rigid bodies, places, borders and their properties
8 : rigid bodies, places, borders and their properties
9 : Update Interface
10 : Message
11 : Choose a given rigid body, place, border, or property
12 : Display
13 : Set a propery of a rigid body
14 : Set a property of a rigid body
15 : Update
16 : Success
17 : Set a property of a place or a border
18 : Display
19 : Set a property of a place or a border
20 : Update
21 : Success
22 : Display

Figure 25 The environment instantiation in the knowledge base

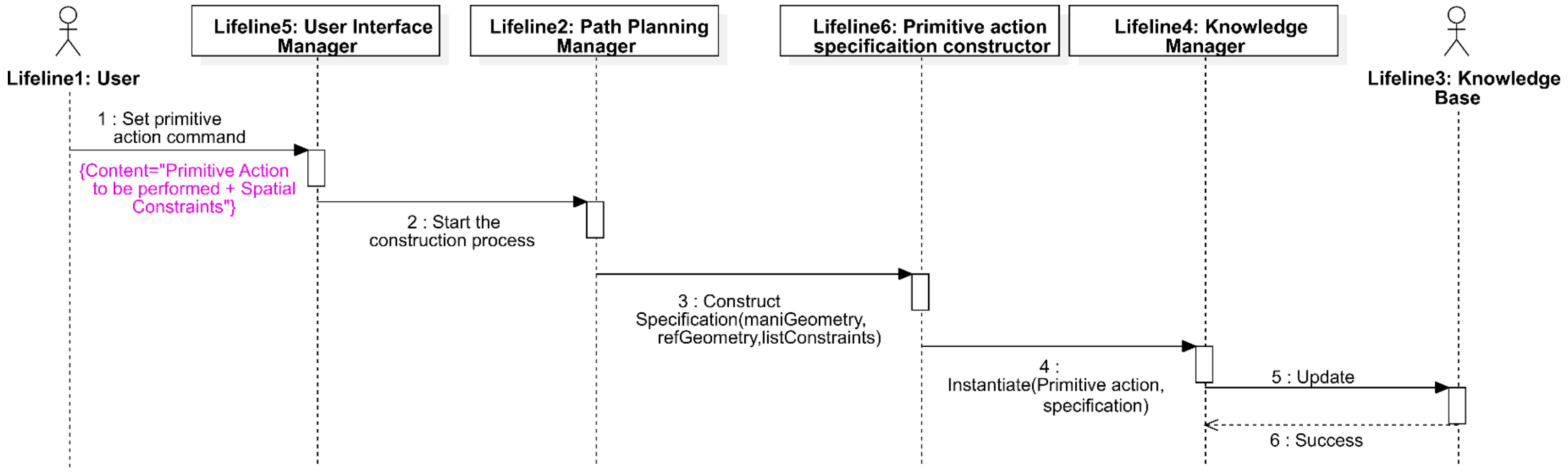

Lifeline1: User
Lifeline5: User Interface Manager
Lifeline2: Path Planning Manager
Lifeline6: Primitive action specificaition constructor
Lifeline4: Knowledge Manager
Lifeline3: Knowledge Base
1 : Set primitive action command
{Content="Primitive Action to be performed + Spatial Constraints"}
2 : Start the construction process
3 : Construct Specification(maniGeometry, refGeometry,listConstraints)
4 : Instantiate(Primitive action, specification)
5 : Update
6 : Success

Figure 26 The construction of a primitive action specification

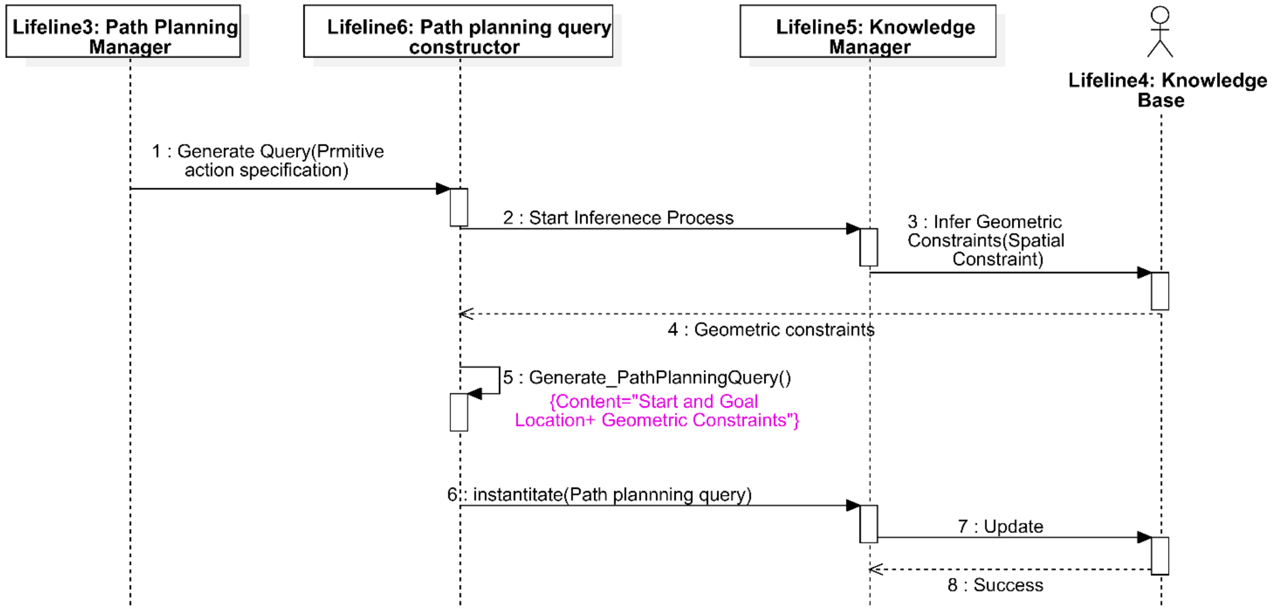

Lifeline3: Path Planning Manager
Lifeline6: Path planning query constructor
Lifeline5: Knowledge Manager
Lifeline4: Knowledge Base
1 : Generate Query(Prmitive action specification)
2 : Start Inferenece Process
3 : Infer Geometric Constraints(Spatial Constraint)
4 : Geometric constraints
5 : Generate_PathPlanningQuery()
{Content="Start and Goal Location+ Geometric Constraints"}
6 : instantitate(Path plannning query)
7 : Update
8 : Success

Figure 27 The generation of the path planning query of a primitive action

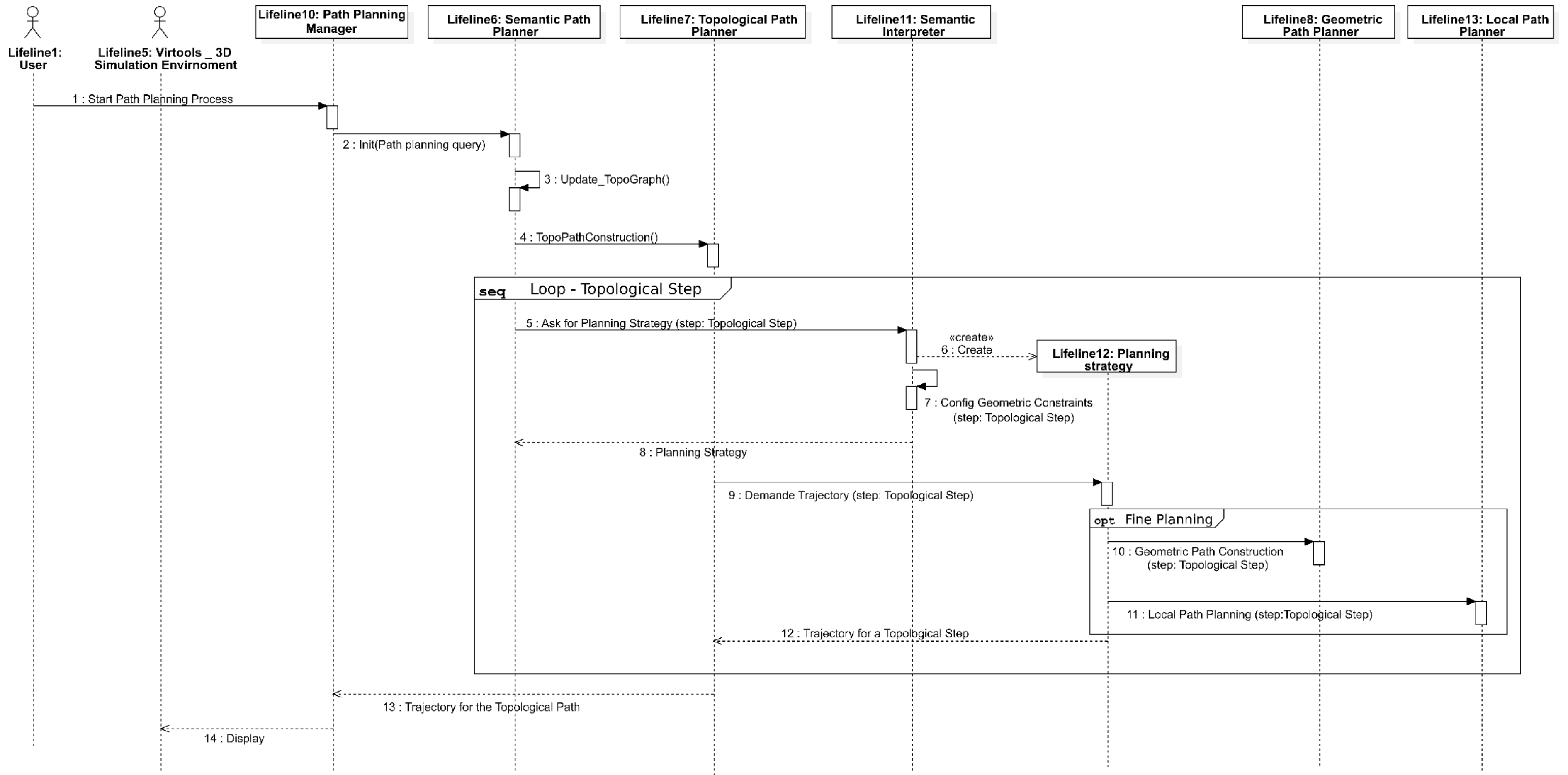

Lifeline1: User
Lifeline5: Virtools _ 3D Simulation Envirnoment
Lifeline10: Path Planning Manager
Lifeline6: Semantic Path Planner
Lifeline7: Topological Path Planner
Lifeline11: Semantic Interpreter
Lifeline8: Geometric Path Planner
Lifeline13: Local Path Planner
1 : Start Path Planning Process
2 : Init(Path planning query)
3 : Update_TopoGraph()
4 : TopoPathConstruction()
seq Loop - Topological Step
5 : Ask for Planning Strategy (step: Topological Step)
«create» 6 : Create
Lifeline12: Planning strategy
7 : Config Geometric Constraints (step: Topological Step)
8 : Planning Strategy
9 : Demande Trajectory (step: Topological Step)
opt Fine Planning
10 : Geometric Path Construction (step: Topological Step)
11 : Local Path Planning (step:Topological Step)
12 : Trajectory for a Topological Step
13 : Trajectory for the Topological Path
14 : Display

Figure 28 The multi-level path planning process with a parameterized path planning query

4.5 AN USE CASE: PEN-PENBOX INSERTION SCENARIO

4.5.1 A brief introduction of the scenario

We use a pen-penbox insertion scenario as an example throughout our discussion. The objective is to insert a pen (mobile) into a penbox (fixed) to a final placement "pen_goal", from Figure 29-(a) to Figure 29-(b). The reason of using this use case is that 1) this simple example is complex enough for path planning, and 2) it can provide enough task-related information (e.g., constraints to be obeyed during a manipulation) to study the benefits of using this information in the path planning process of an insertion action.

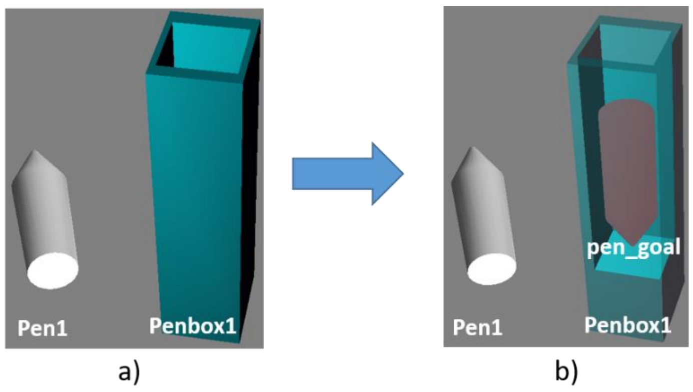


Figure 29 A pen-penbox insertion use case

4.5.2 The instantiation of the environment in the ontology of 3D environment

Different levels of environment information should be instantiated in the knowledge base. It includes geometric models of *Rigid Bodies* and *Free Space* (geometric level), the constructed *Places* and *Borders* (topological level), and the taxonomy of *Rigid Bodies*, *Places*, and *Borders* (semantic level).

Figure 30 shows an instantiation of a pen in the environment model knowledge. "*Pen1*" is an instance of the concept *Pen*, it has an origin (*Point3D)* "*Origin_pen_1*", a central axis (*Axis3D*) "*axis1*" and a pointing direction (*Vector3D*) "*Vector1*". The instantiation of 3D environment information could be either manually assignment or automatically generated. Although it could be long, the instantiated 3D environment information can be saved as a configuration file and be reused in the next time.

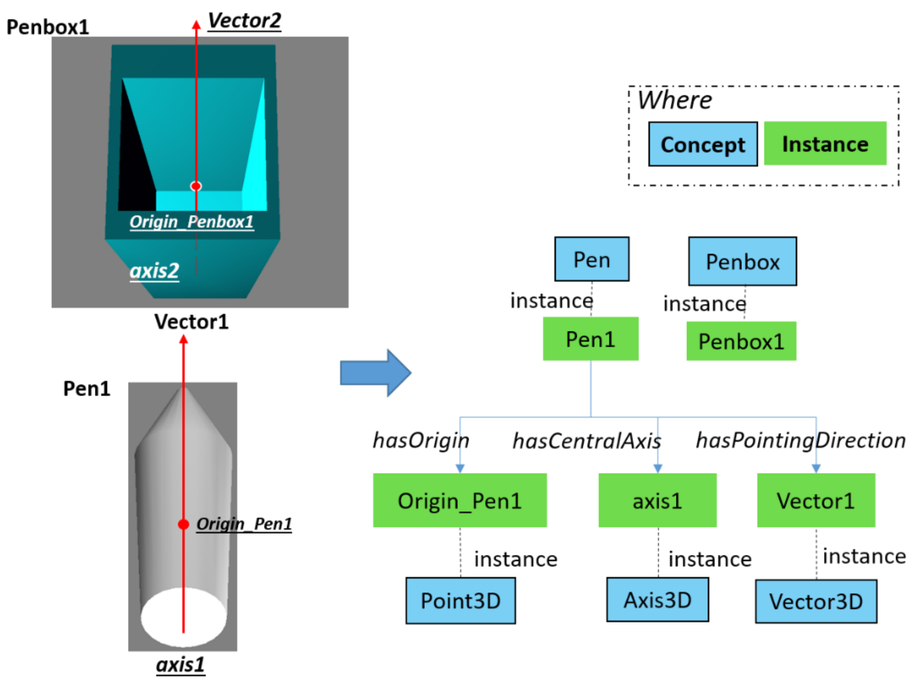


Figure 30 An instantiation of a pen in the environment model knowledge

4.5.3 The definition of the primitive action: ‘*Insert (Pen1, Penbox1)*’

A *Primitive action Specification* is constructed from an insertion action between "*Pen1*" and "*Penbox1*" (Figure 31). The primitive action identity ("*Insert*"), the manipulated object ("*Pen1*") and the reference geometry ("*Penbox1*") are directly extracted from this insertion action.

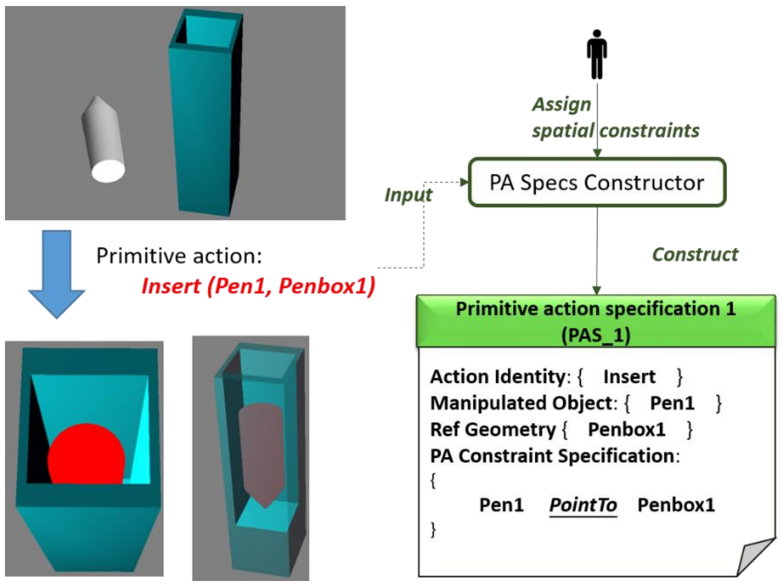


Figure 31 The construction of a primitive action specification

In the meanwhile, different ways of insertions are possible to insert a pen into a penbox. In our example, a human operator manually assigns *Spatial Constraints* to the insertion action between "*Pen1*" and "*Penbox1*", and then they are associated to the specification of this insertion action. The *Spatial Constraint* “Pen1 *PointTo* Penbox1” is used here.

4.5.4 The generation of the corresponding path planning query

In the next step, a path planning query constructor (*PP Query Constructor*) takes the specified primitive action specification ("*PAS_1*") as input and generates a related path planning query description ("*PPQD_1*"), as shown in Figure 32. The manipulated object ("*Pen1*") in "*PPQD_1*" is directly extracted from "*PAS_1*". The goal configuration ("*pen_goal*") is obtained by random sampling in the goal region, while satisfying the *Geometric Constraints* attached to "*PPQD_1*".

More specifically, we are interested in how to generate *Geometric Constraints* from *Spatial Constraints*, specified in section 4.5.3, through an inference process using the pre-defined inference rules (Figure 32).

Pen1 *PointTo* Penbox1: For simplicity, we use a *Spatial Constraint* "Pen1 *PointTo* Penbox1" as an example to show how a *Geometric Constraint*, "Vector1 *Against* Vector2", is inferred or generated using an inference rule. The SWRL rule below assists this inference process.

SWRL Rule Explanation: A SWRL rule contains an antecedent (IF) and a consequent (THEN). If the terms in antecedent hold, then the terms in consequent must hold.

1) **Antecedent** (line 1-13): 1) Environment specification (line 2-6): "object a" is an instance of *RigidBody* (line 2), "object b" is an instance of *RigidBody* or *Place* (line 3), "vector_1" and "vector_2" are both instances of *Vector* (line 4), and are respectively pointing direction of "object a" (line 5) and opening direction of "object b" (line 6). 2) Constraint specification (line 7-12): "sconstraint" is an instance of *SpatialConstraint* (line 7), and it specifies a spatial relation "PointTo" (line 8) between two geometries "object a" (line 10) and "object b" (line 9). "gconstraint" is an instance of *GeometricConstraint* (line 11), and it has a related spatial constraint "sconstraint" (line 12).
2) **Consequent** (line 14-18): If the terms in the antecedent holds, then "gconstraint" specifies a new spatial relation "Against" (line 17) between two geometric elements "vector_1" (line 16) and "vector_2" (line 15), which means that the scalar product of these two vectors should be negative.

A SWRL Rule for spatial-geometric constraint mapping

```
IF {
 a instance_of RigidBody and
 b instance_of RigidBody or Place and
 vector_1, vector_2 instance_of Vector and
 a hasPointingDirection vector_1 and
 b hasOpeningDirection  vector_2 and
 sconstraint instance_of SpatialConstraint and
    sconstraint  hasRelation "PointTo" and
    sconstraint  hasRefGeometry "b" and
    sconstraint  hasTargetGeometry "a" and
gconstraint instance_of GeometricConstraint and
    gconstraint hasRelatedSC "sconstraint"
}
THEN {
 gconstraint hasRefGeoElement "vector_2" and
 gconstraint hasTargetGeoElement "vector_1" and
 gconstraint hasRelation "Against"
}
```

Figure 33 visually shows why the "Pen1 *PointTo* Penbox1" results in "Vector1 *Against* Vector2". In our pen-penbox insertion example, the corresponding geometric constraints (see in Figure 32), generated from the spatial constraints specified in section 4.5.3, are

Vector1 *Against* Vector2

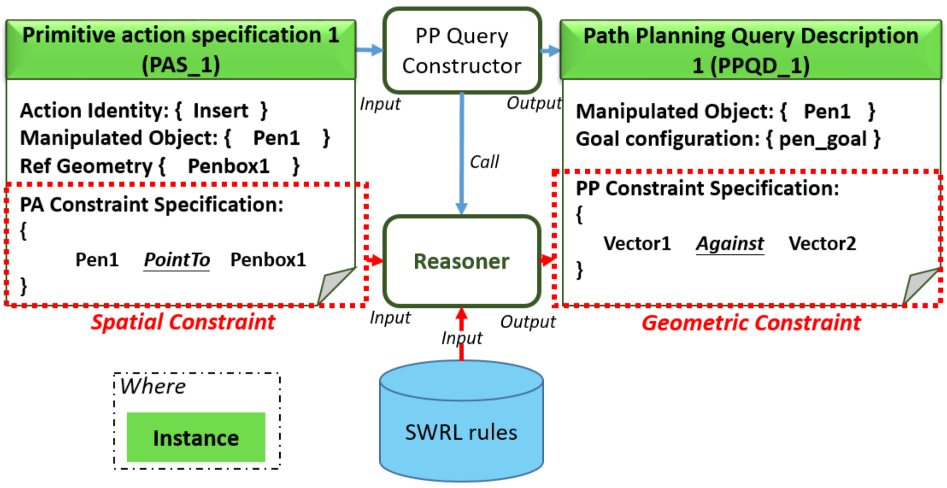

Primitive action specification 1 (PAS_1)
Action Identity: { Insert }
Manipulated Object: { Pen1 }
Ref Geometry { Penbox1 }
PA Constraint Specification:
{ Pen1 PointTo Penbox1 }
Spatial Constraint
PP Query Constructor
Input
Output
Call
Reasoner
Input
Output
Input
SWRL rules
Path Planning Query Description 1 (PPQD_1)
Manipulated Object: { Pen1 }
Goal configuration: { pen_goal }
PP Constraint Specification:
{ Vector1 Against Vector2 }
Geometric Constraint
Where
Instance

Figure 32 The generation of a path planning query description

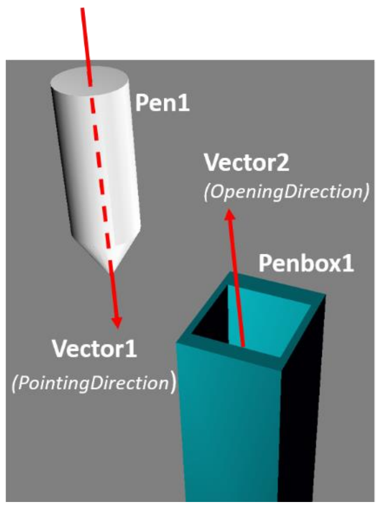


Figure 33 A graphical representation of the SWRL rule

4.5.5 The multi-level path planning process

When all *Geometric Constraints* are inferred and are generated from the *Spatial Constraints*, they are specified in a *Path planning query description (PPQD_1)*. Using this parameterized path planning query description as an input, the multi-level path planning process is then invoked to computes a relevant trajectory/path for this pen-penbox insertion action while respecting the *Geometric Constraints*.

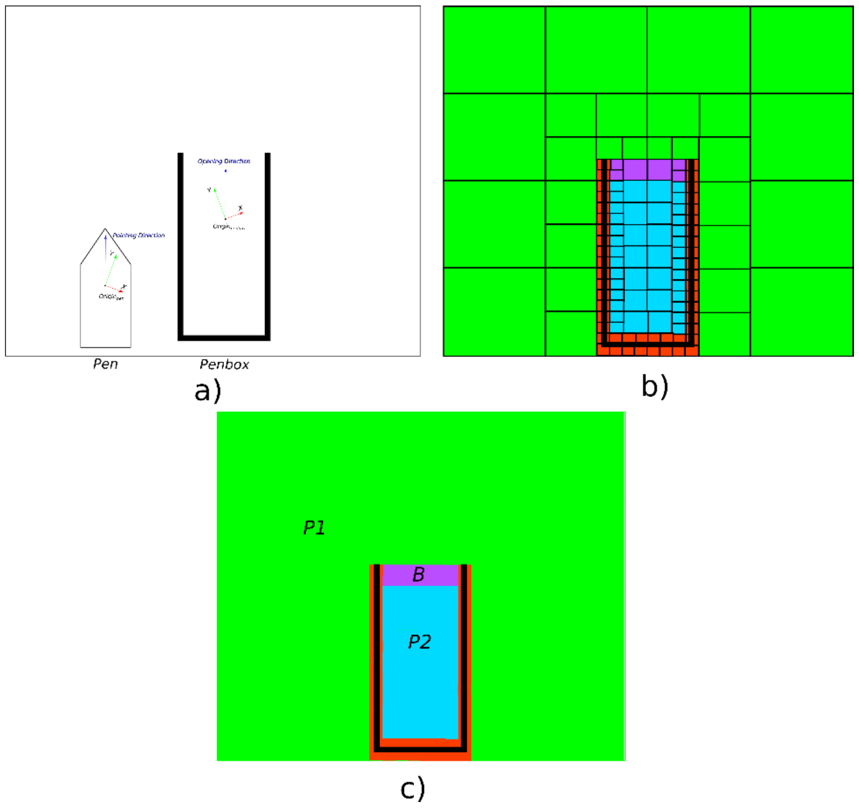


Figure 34 Environment Construction: a) Initial Environment b) Cell decomposition c) Places and Borders

Figure 34 visually demonstrates the multi-level environment model construction of the pen-penbox insertion scenario in the 2D view. The workspace of the simulated environment is a squared and closed-formed bounded 3D volume (Figure 34-a). The penbox where the pen is inserted is placed in the bottom-center of the workspace. The pen is put aside the penbox. Figure 34-b presents the octree decomposition of the workspace, and Figure 34-c shows the *Places* (P1, P2) and *Borders* (B) constructed from the octree decomposition of the geometric level. Since only one *Border* is identified, the *Topological Graph* of the simulated environment contains a single node representing the *Border* without any arc.

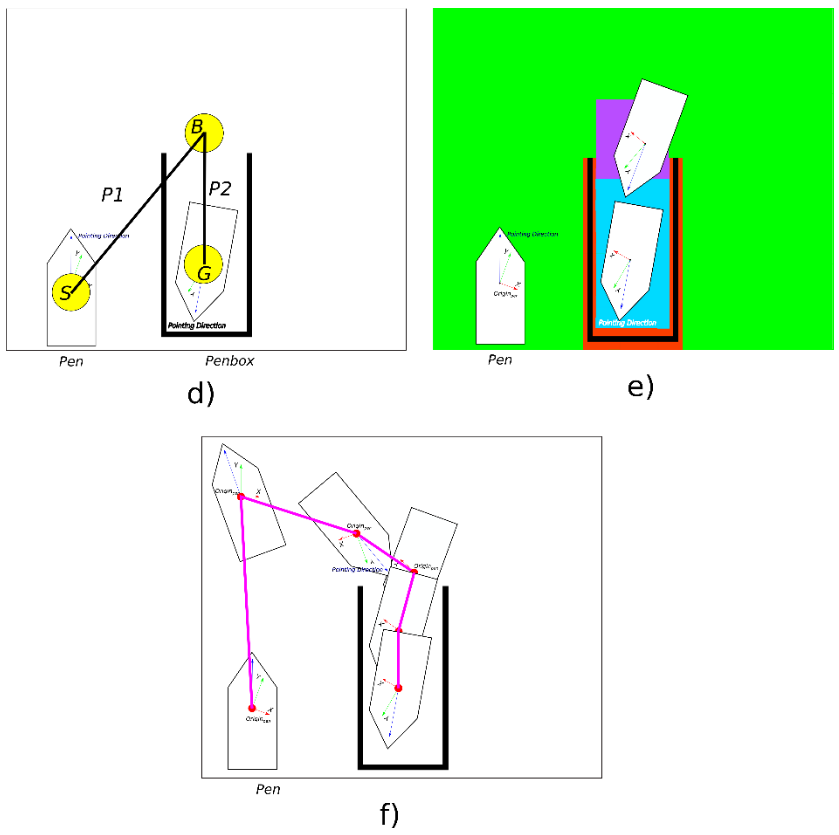


Figure 35 Multi-level path planning: d) Topological Path e) Initial, Final and Border Configuration f) the generated collision-free geometric path (RRT Path)

After the multi-level environment model is constructed, Figure 35 presents the multi-level path planning process regarding the parameterized path planning query (i.e., *PPQD_1*). Firstly, the *Topological Graph* is updated with two nodes respectively representing the initial and the final configuration (extracted from the path planning query). The two nodes are added into the *Topological Graph* by adding an arc to the neighborhood *Border* node (Figure 35-d). Secondly, the cost of node and arcs are updated using the semantic information associated with *Places*, *Borders* and the manipulated *Rigid Body*. Finally, a *Topological Path* is constructed and it represents the least cost to reach the goal by exploring the *Topological Graph* (Figure 35-d). In each *Topological Step* of *Topological Path* (Figure 35-e), a key geometric configuration (geometric landmark) has to be found at the milestone (*Border B or G*) of each *Topological Step*. The geometric landmark at each *Topological Step* is vital for the geometric path computation for this *Topological Step*. Figure 35-f demonstrates the computed collision-free *Geometric Path* (RRT Path) for the pen-penbox use case (i.e., insertion action).

4.6 SUMMARY

The proposed ontology-based approach aims at generating a path planning query for a primitive action. Not only the start and the goal configuration should be determined, but also *Geometric Constraints* can be generated from the *Spatial Constraints,* which are often assigned by the user and are associated to the task to be performed. The variety of the *Spatial Constraints* allows to restrict the movement of a manipulated *Rigid Body*. Therefore, using such information in the path planning of a primitive action can improve the performance of the path planning system and the relevance of the computed path.

The implementation of the approach develops a loosely-coupled architecture between the knowledge base and the path planning system. It allows adapting the path planning system into different applications and answering to different primitive action commands from users by updating the knowledge base. Moreover, the task-related symbolic environment representation allows an intuitive interaction between the user and the path planning system to define the primitive action to be performed and the spatial constraints to be obeyed.

This chapter uses a pen-penbox insertion use case to illustrate the strategy to generate the path planning query for a given "*Insert*" primitive action, and to interface with the multi-level path planning architecture. The *Geometric Constraints* are applied differently to different *Places*. The detailed result analysis will be performed in the next chapter, and also a more complex use case will be given. The path planning results will be compared with the original multi-level path planning approach.

Chapter 5 Ontology Validation, Simulation Experiments and Results

5.1 INTRODUCTION

In this chapter, the two proposed contributions of this thesis work (i.e., Chapter 3, Chapter 4) will be implemented and evaluated at the LGP (Laboratoire Genie de Production). The evaluation of the proposed work consists of three aspects:

- The verification and the validation of the ontology of 3D environment will be performed by answering to certain competency questions and verifying the returned answers.
- The ontology-based approach to generate path planning query of a primitive action of a task plan will be verified by checking whether a feasible path planning query is generated.
- Taking the generated path planning query as input, path planning experiments will be performed in the non-interactive simulation and compare the results of different path planning strategies.

Two increasingly complex simulations conduct the evaluation process. The first simulation concerns a pen-penbox insertion scenario that has already been introduced in Chapter 4. The second simulation, inspired from the “shaped embedding game” for babies, aims at inserting an object into a hole with the right shape.

The chapter is organized in the following sub-sections. Section 5.2 introduces some necessary materials for performing simulations, including the simulation software and the path planning strategies to be compared. Section 5.3 discusses the software architecture of our proposed work. Section 5.4 presents two simulation scenarios. Section 5.5 and section 5.6 present the evaluation of this thesis work in the two proposed simulations. The ontology of 3D environment will be evaluated and the path planning results will be analyzed. At last, the conclusion of this chapter is given in section 5.7.

5.2 THE MATERIALS FOR TASK SIMULATIONS

5.2.1 The simulation software

Two different kinds of software have been used in this thesis work:

- ***Virtools*** is the 3D simulation software to simulate manipulation tasks.

- ***Protégé*** is the ontology editor to create and to manage ontology models

Moreover, additional open-source libraries are also presented in this thesis work, such as to allow checking collisions and to interface with ontologies.

A. *Virtools* ®

The simulations are realized and implemented using Virools Dev. Virtools was firstly developed at 1993 and was used for creating video games. In 2005, Virtools was acquired by Dassault Systèmes [Systèmes 2006]. Since then, Virtools has been used to create 3D real-time simulations. Recently, it has been widely used to create industrial virtual reality (VR) applications, since an add-on library (i.e., VR Park) for Virtools has been developed to allow 3D visualization and interfacing with different VR peripherals.

The interface of Virtools consists of three different windows (Figure 36):

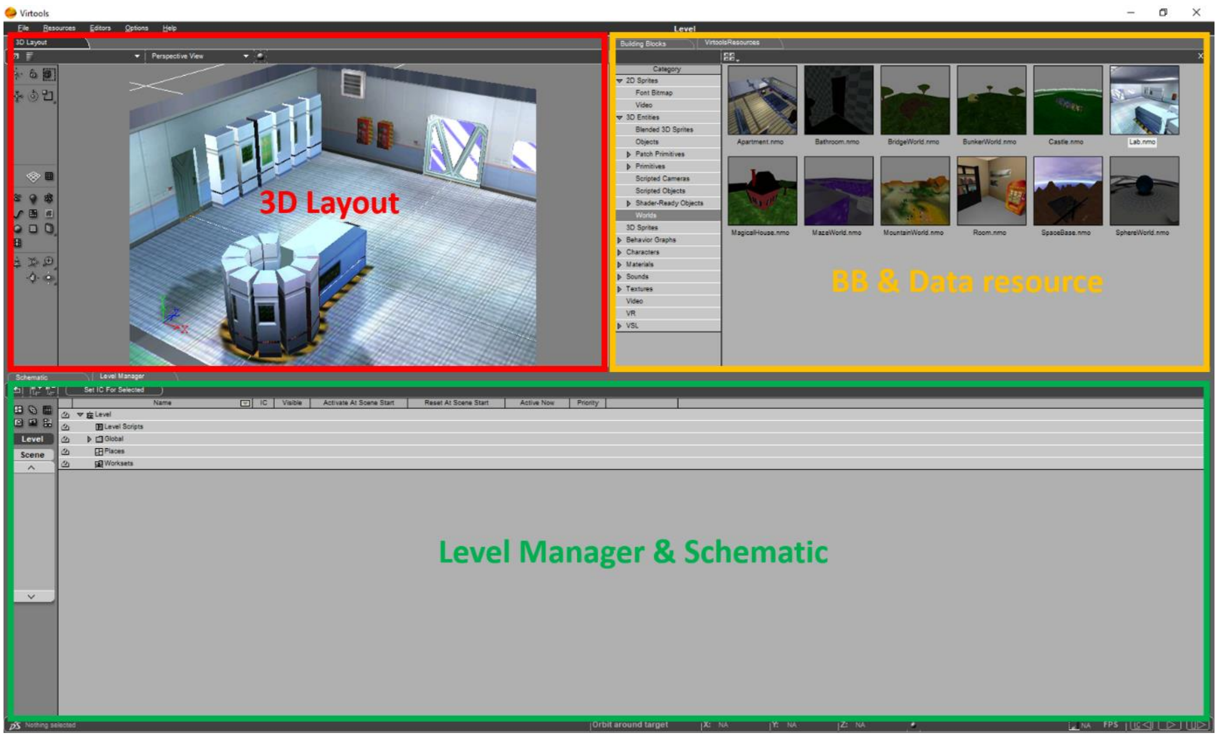


Figure 36 Virtools Interface

- Top-left window: It is the 3D layout window to visualize the virtual environment of an application in real-time.
- Top-right window: It groups all the resources (i.e., building blocks and data resources) necessary for developing an application.
- Bottom window: It allows to manage the virtual environment using a level manager, and it also allows to create scripts defining the virtual objects' behavior in the "Schematic" tab.

To provide necessary functionalities for creating 3D simulations, the architecture of Virtools consists of three different layers:

- ***The graphical layer*** allows the user to see and to define the virtual environment of an application.
- ***The script layer*** allows the user to define the behaviors of the virtual objects. A script constitutes several interconnected "building blocks (BB)", each of which implements an elementary behavior. A BB (Figure 37) has some behavioral inputs/outputs (horizontally) and some parametric inputs and outputs (vertically). The set of BBs, aggregated in a Behavioral Graph (BG), defines a composite behavior. The simplicity of combining BBs to implement the virtual object's behavior facilitates the creation of an application for uninitiated users.
- ***The programming layer*** allows the user to develop customized BBs. It is realized by using classes and functionalities of the Software Development Kit (SDK) provided by Virtools. This layer offers much more powerful behavior customization but less friendly to the uninitiated users.

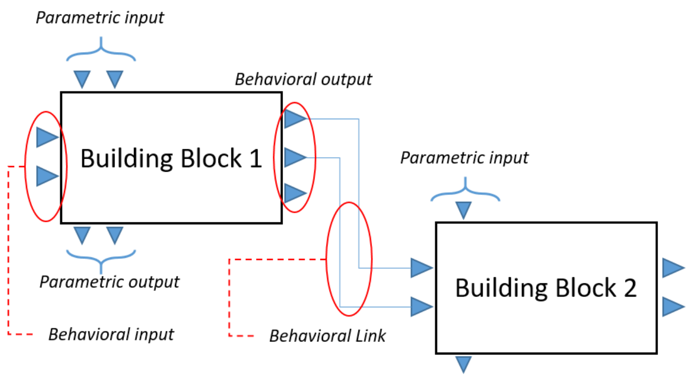


Figure 37 The script for defining the behavior using BB

B. *Protégé*

Protégé is an open-source ontology editor (under BSD 2-Clause license) allowing to create and to manage ontology knowledge models. It was firstly developed in 1999 by Stanford University using the JAVA language. Until now, it has been updated to the fifth version.

The software provides the user with a graphical interface (Figure 38) to manage an ontology model conveniently. An ontology model and its related content (i.e., concepts, instances, object properties, and data properties) are all uniquely specified by the Internationalized Resource Identifier (IRI). Once the ontology model is published on the Internet, the IRIs allows a remote agent to access the model's content unambiguously. The red box of Figure 38 shows the specification of an ontology's IRI. The blue box of Figure 38 gathers the statistics/metrics of an ontology model, such as the number of classes and instances. Furthermore, an ontology model can possibly reuse the content of other ontologies, e.g., concepts. Such an idea originates from the consideration of the modularity and the generality of ontologies. It is implemented by the ontology import, as shown in the yellow box of Figure 38.

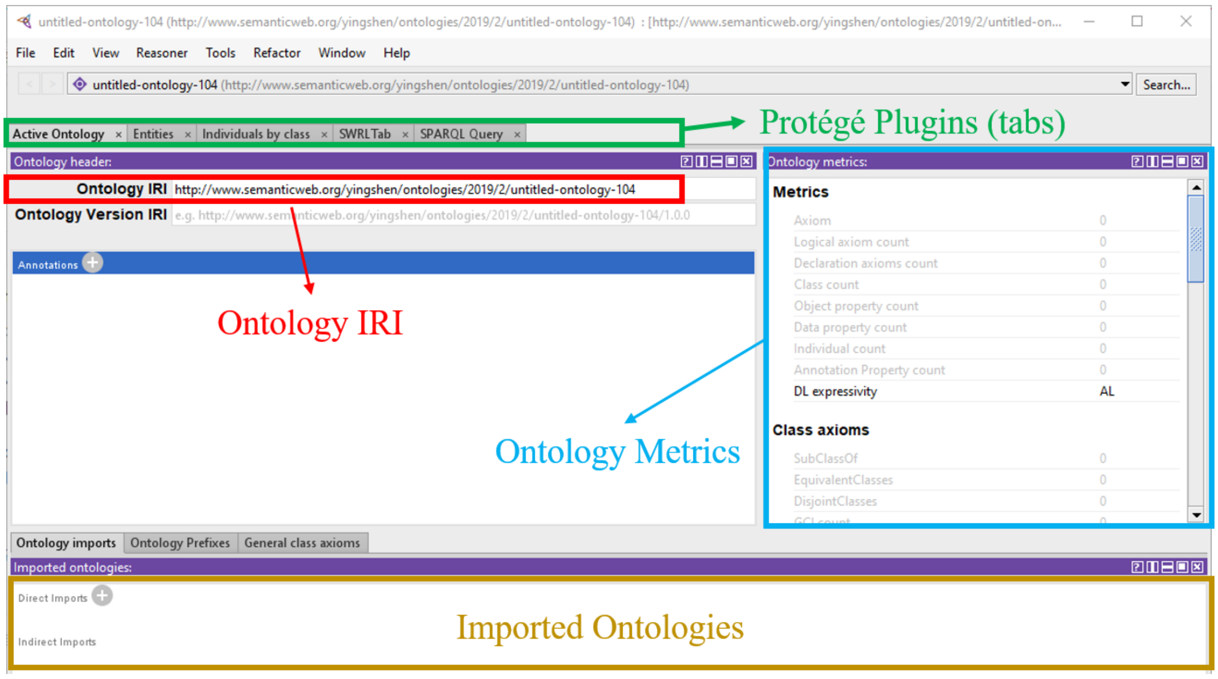

untitled-ontology-104 (http://www.semanticweb.org/yingshen/ontologies/2019/2/untitled-ontology-104) : [http://www.semanticweb.org/yingshen/ontologies/2019/2/untitled-on...
File Edit View Reasoner Tools Refactor Window Help
untitled-ontology-104 (http://www.semanticweb.org/yingshen/ontologies/2019/2/untitled-ontology-104)
Search...
Active Ontology
Entities
Individuals by class
SWRLTab
SPARQL Query
Protégé Plugins (tabs)
Ontology header:
Ontology IRI http://www.semanticweb.org/yingshen/ontologies/2019/2/untitled-ontology-104
Ontology Version IRI e.g. http://www.semanticweb.org/yingshen/ontologies/2019/2/untitled-ontology-104/1.0.0
Annotations
Ontology IRI
Ontology Metrics
Ontology metrics:
Metrics
Axiom 0
Logical axiom count 0
Declaration axioms count 0
Class count 0
Object property count 0
Data property count 0
Individual count 0
Annotation Property count 0
DL expressivity AL
Class axioms
SubClassOf 0
EquivalentClasses 0
DisjointClasses 0
Ontology imports
Ontology Prefixes
General class axioms
Imported ontologies:
Direct Imports
Indirect Imports
Imported Ontologies

Figure 38 The graphical Interface of Protégé

The functionalities of Protégé are realized by using different plugins. They are often displayed as tabs in the Protégé (the green box of Figure 38). For example, the "Entity" tab allows the user to manage concepts, instances, object properties and data properties of an ontology model; the "SPARQL Query" tab allows the user to acquire particular information from an ontology using SPARQL (***S***PARQL ***P***rotocol ***A***nd ***R***DF ***Q***uery ***L***anguage) queries; the "SWRL Tab" tab allows the user to create/delete/edit the SWRL rules associated with an ontology.

Different kinds of reasoners (e.g., Pellet, HeimiT in Figure 39) have also been implemented as the Protégé's plugins. They perform the inference of an ontology, 1) to check the consistency of the model, or 2) to infer new knowledge from the model's asserted knowledge. The asserted and the inferred classes of an ontology are displayed in the "Class hierarchy" of the "Entity" tab. The inconsistency of the class definitions will be highlighted in red Figure 40.

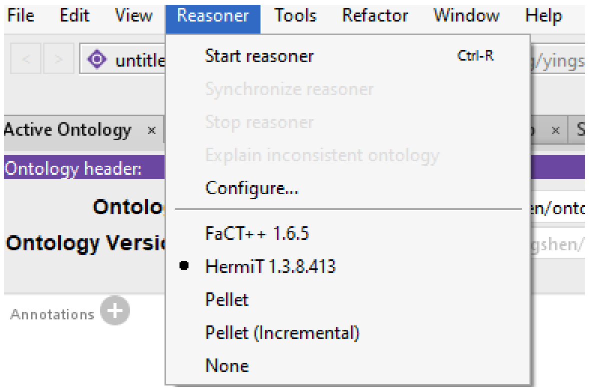


Figure 39 The types of the available reasoners

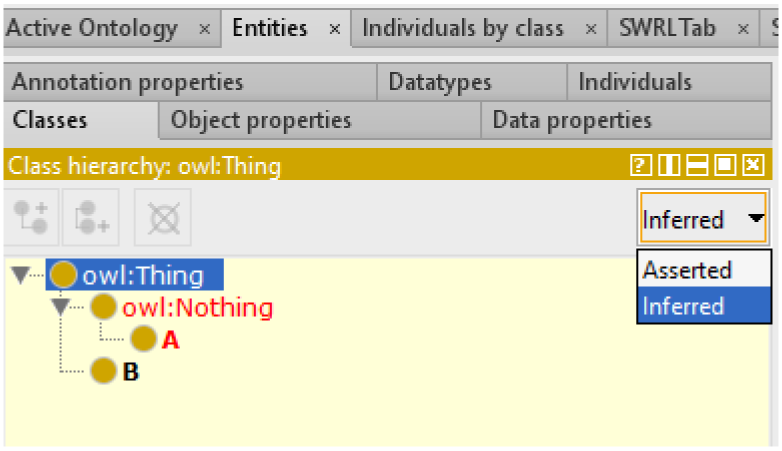


Figure 40 The asserted/inferred classes of an ontology

C. *Open source libraries*

This sub-section presents two open-source libraries used in this thesis work.

FCL (Flexible Collision Library) is an open-source library for performing the proximity queries between two triangulated geometric meshes. Three types of proximity queries are supported:

1) *Collision checking* detects whether the two meshes overlap. It can be performed either between the bounding boxes or between all the triangles of the two meshes.
2) *Distance computation* computes the minimum distance between the two meshes, i.e., the distance between the pair of closest points.
3) *Tolerance verification* checks whether two meshes are closer or farther than a given tolerance distance.

In this thesis work of path planning for a manipulated object, the RRT (Rapid Exploring Random Tree) algorithm uses the collision-checking feature of FCL to obtain collision-free random samples, and further to compute the collision-free trajectory for a manipulated object.

OWLAPI (Web Ontology Language – Application Program Interface) is an open-source library to provide Java APIs for manipulating OWL ontologies. It consists of a list of classes and functions 1) to create an ontology, 2) to load an ontology from the OWL (Web ontology language) or RDF (Resource description framework) files, 3) to define axioms, 4) to interface with reasoners, and 4) to save an ontology in an OWL/RDF file.

In this thesis work, OWLAPI allows the path planning system (i.e., path planning for a primitive action in a task plan) to access the knowledge base (i.e., ontologies), so that, e.g.,

- the simulation environment can be instantiated in the ontology,
- the reasoner can be invoked to infer geometric constraints from the spatial constraints of a primitive action.

5.2.2 Path Planning Strategies

In [Cailhol 2015], three path planning strategies have been defined to explore the benefits of using different natures of environment data (i.e., semantics, topology and geometry) in the path planning process. These strategies consist of:

- purely geometric path planning (G strategy)
- two-step path planning with topological and geometric data (GT strategy)
- two-step path planning with semantic, topological and geometric data (GTS strategy)

This thesis work further explores the benefits of using the task-related information (formulated in an ontology) in the path planning process. The inferred geometric constraints can be applied not only to the G strategy but also each topological step of the GT/GTS strategy. Therefore, this thesis work studies two more path planning strategies.

- geometric path planning using information in the ontology (GO strategy)
- two-step path planning with topological and geometric data and using information in the ontology (GTO strategy)

A. G Strategy

This path planning strategy explores no more than geometric environment data. It is implemented by a bidirectional RRT algorithm [Cailhol 2015]. It samples uniformly in the 6D configuration space limited by the 3D space of the scene (line cubes in the two scenarios). Giving a start (q_{init}) and a goal (q_{final}) configuration, the strategy constructs a start tree (T_{init}, where q_{init} as the root node, blue part of Figure 41) and a goal tree ($T_{final,}$ where q_{final} as the root node, red part of Figure 41). In each iteration, a collision-free random sample is obtained and the strategy tries to add it into T_{start} and T_{goal}. The iteration continues until a random sample (α_4) can be both added to T_{start} and T_{goal} (i.e., the two trees are connected). Thus, a trajectory is found by searching the path between q_{init} and q_{final} in the connected trees. In order to limit the time of defining a trajectory, we have limited the allowed iterative number of RRT algorithm to 10^6.

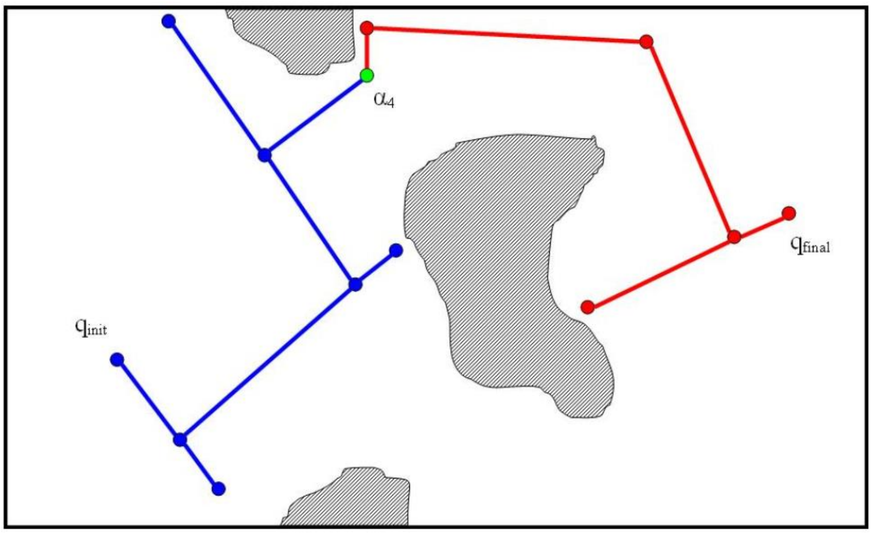
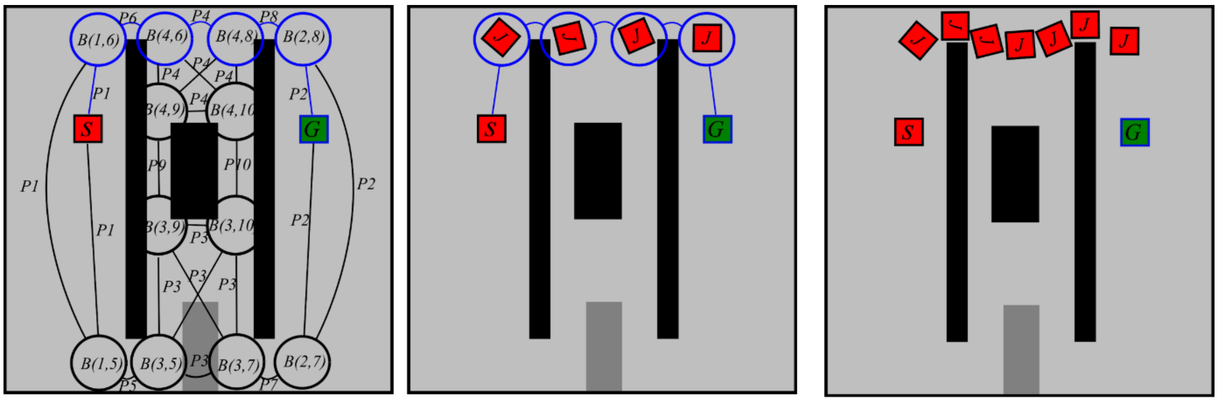


Figure 41 The bidirectional RRT algorithm [Cailhol 2015]

B. GT Strategy

This path planning strategy explores the use of both topological and geometric data in the two-step path planning process (i.e., coarse and fine planning) presented in Chapter 2 (Section 2.2.2.2-B). In this case, the cost of the topological graph (constructed at the topological level) is brought by the arcs of the graph. It is proportional to the distance between the two borders (B) of an arc.



a) Topological Path b) Milestones et each border c) RRT Path

Figure 42 Path planning using GT strategy [Cailhol 2015]

By exploring the weighted topological graph, the coarse planning searches for the topological path with the least cost (Figure 42-a). Once the topological path is found, the geometric landmarks in the borders along the topological path are determined by 6D uniformed sampling in the 3D space of each border (Figure 42-b). For the fine planning of each topological step, the

RRT algorithm is performed to find a collision-free geometric path in the relevant place (i.e., P1, P2, P6, P4, P8 in Figure 42-a). The final RRT path is shown in Figure 42-c.

In order to limit the time of defining a trajectory, we have fixed some limitations for path planning:

- Coarse planning: the number of random samples for defining the geometric landmarks at the borders is limited 2000. If no collision-free random sample can be found after 2000 attempts, a sample located at the center of a border is used without considering collisions with environment obstacles.
- Fine planning: the number of random samples for the RRT algorithm is limited to 10^5, since the geometric path planning is limited to the 3D space traversed by a topological step.

C. GTS Strategy

This path planning strategy explores the use of semantic, topological and geometric data in the two-step path planning process (i.e., coarse and fine planning) presented in Chapter 2 (Section 2.2.2.2-B). Comparing to GT strategy, the cost of the topological graph is brought from different modalities of semantic information. The information is associated with different places of the scene's environment. Two kinds of semantic information are considered:

- ***Complexity*** identifies the difficulty to across a place ($C_{complex}$), where different cost values are used: not complex (0.5), not very complex (1), complex (2) and very complex (5).
- ***Geometric form*** defines the shape of the manipulated object and the places (C_{shape}). If their shapes are identical, a cost value of 0 will be given. If not, a cost value of 0.5 will be given if the object shape can fit the place shape, otherwise, a cost value of 5 will be given.

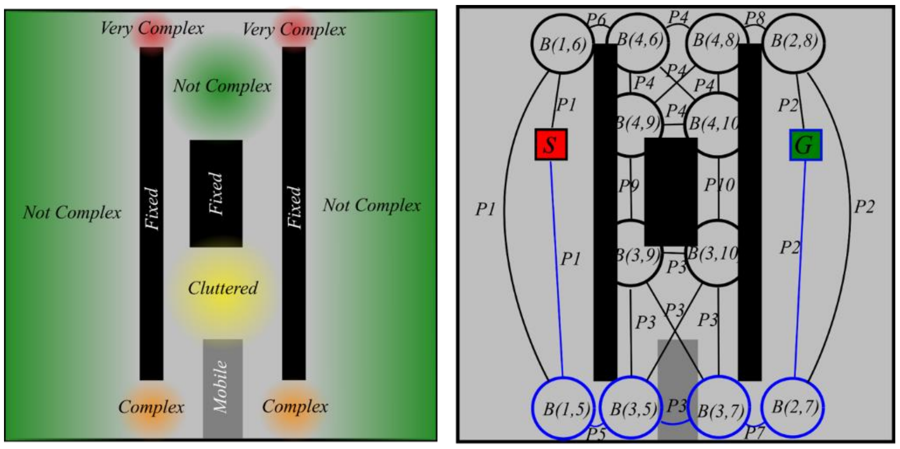
The cost of crossing a place 'k' ($C_{semantic_k}$) is determined by the sum of the $C_{complex}$ and C_{shape}. Therefore, in the topological graph, the cost of a node 'n' ($C_{n_{i,j}}$) is determined by the cost of adjacent places i,j, and the cost of an arc 'arc' (C_{arc_m}) is determined by the cost of the place 'm' that the arc crosses. The equation 16 indicates these cost definitions.

$$C_{semantic_k} = C_{complex_k} + C_{shape_k}$$

$$C_{n_{ij}} = \frac{C_{semantic_i} + C_{semantic_j}}{2} \tag{16}$$

$$C_{arc_m} = C_{semantic_m}$$

As shown the Figure 43, P6 and P8 are more difficult to cross than P5 and P7 (because P6 and P8 have higher complexity). Using such semantic information, the computed topological path will go through P5 and P7 rather than P6 and P8.



a) Semantic Information (Complexity) b) The topological path regarding semantics

Figure 43 The topological path construction using GTS strategy [Cailhol 2015]

D. GO Strategy

This path planning strategy extends the G strategy to use task-related information (formulated in ontologies) in the geometric path planning process (i.e., RRT algorithm). The presented work focuses on using the task-related geometric constraints to control the sampling process of RRT instead of random sampling.

In order to find a sample compliant to a geometric constraint i (GC_i), this thesis work specifies the following conditions (equation 17) about when it will take effects.

In each iteration of RRT, the position of a sample ($Sample_i$) for the manipulated rigid body is firstly randomly chosen (Pos_Sample_i) . For GC_i, the path planning system determines the distance ($Dis_{pos\_sample_i Rf_{origin}}$) between Pos_Sample_i and the origin of the reference geometry (Rf_{origin}). If $Dis_{pos\_sample_i Rf_{origin}}$ is larger than a given threshold ($Threhold_{dis}$), GC_i cannot be used. Otherwise, it is used to constraint the possible orientation of $Sample_i$.

$$The\ application\ of\ GC_i \begin{cases} Dis_{pos\_sample_i Rf_{origin}} > Threhold_{dis}, & OK \\ Dis_{pos\_sample_i Rf_{origin}} \leq Threhold_{dis}, & NO \end{cases} \qquad (17)$$

The $Threhold_{dis}$ forms a circular (2D space) or a spherical effective space located at the Rf_{origin}. If we take the ‘pen-penbox insertion scenario’ (introduced in section 4.5) for example, the effective space of the geometric constraint GC “Vector1 *Against* Vector2” is a spherical part of 3D Cartesian space located at the origin of ‘Penbox1’ with a radius ‘Threshold$_{dis}$’. In the

Figure 44, we can see that only sample1 can be applied with the GC to constraint their orientation.

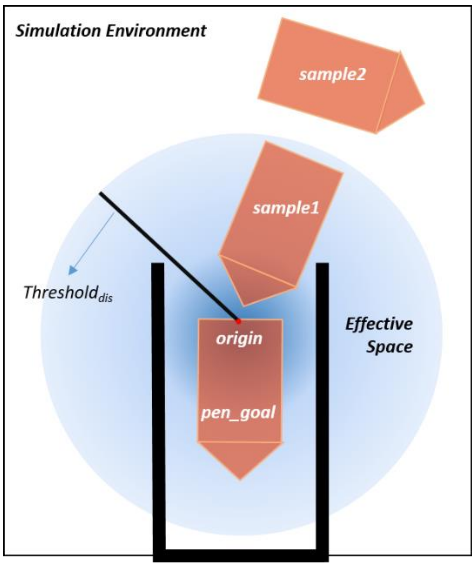


Figure 44 The application of geometric constraints in GO strategy

In order to limit the time of defining a trajectory, the allowed number of iterations for RRT algorithm is limited to 10^6.

E. GTO Strategy

This path planning strategy extends the GT strategy to use task-related information in the two-step path planning process (i.e., coarse and fine planning) presented in Chapter 2 (Section 2.2.2.2-B). The inferred geometric constraints are used accordingly to the place crossed by each topological step.

In the two-step path planning process, the coarse planning computes a topological path ($TopoPath$) using the same method of GT strategy (i.e., the cost of a topological graph is determined by the arc's length). At the $Border_{milestone}$ of $TopoStep_j$, the geometric landmark should be compliant to the geometric constraints found in both places ($Place_i \ and \ Place_k$) overlapped at $Border_{milestone}$, as illustrated in Equation 18.

$$GCs_{Border_{milestone}} \subseteq GCs_{place_i} \cup GCs_{place_k},$$
$$where\ overlaps\ (place_i, place_k)\ at\ Border_{milestone} \quad (18)$$

In the fine planning of $TopoStep_i$, a set of geometric constraints (GCs) is defined regarding to the corresponding place ($Place_{topostep_i}$). The GCs will then be used to control the sampling process of RRT algorithm in the $Place_{topostep_i}$. The fine planning process is similar to the GO strategy.

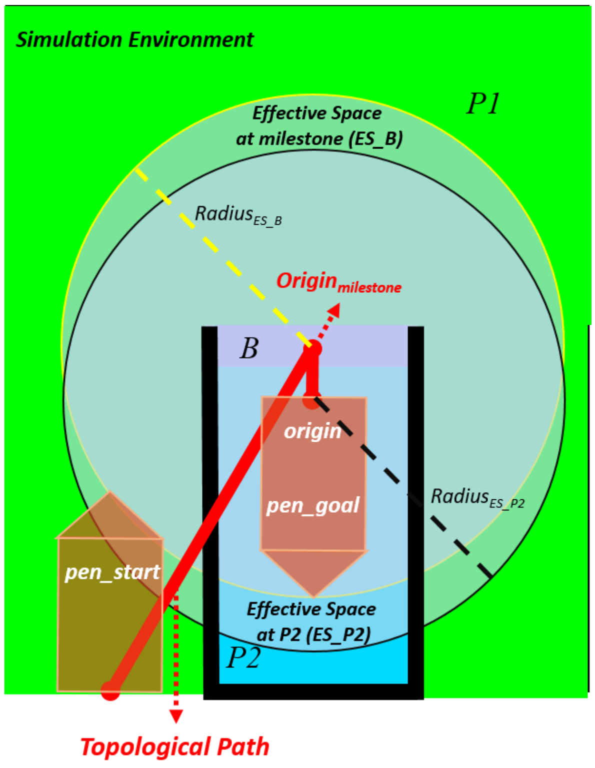


Figure 45 The application of geometric constraints in GO strategy

Here, we still take the ‘pen-penbox insertion scenario’ (introduced in section 4.5) for example. Two *Places* (P1 and P2) and one *Border* (B) have been identified. The inferred geometric constraint ‘Vector1 *Against* Vector2’ (GC) is applied in P2 rather than P1, because P2 is inside of Penbox1 whereas P1 is not. This GC is also applied to the *Border* (B) since geometric configurations at B has to compliant to geometric constraints of P1 and P2. Therefore, along the computed *Topological Path* (in Figure 45),

- the fine planning of a *Topological Step* (*pen_start*, B) has to compliant to the GC. The effective space of the GC in this step is a spherical part of 3D Cartesian space located at the origin of the milestone ($Origin_{milestone}$) with a radius $Radius_{ES\_B}$.
- the fine planning of a *Topological Step* (B, *pen_goal*) has to compliant to the GC. The effective space of the GC in this step is a spherical part of 3D Cartesian space located at the origin of P2 with a radius $Radius_{ES\_P2}$.

In order to limit the time of defining a trajectory, we fix some limitations for path planning.

- Coarse planning: the number of random samples for defining the geometric landmarks is limited to 10000. Comparing to GT strategy, the increased number is due to the necessity of finding collision-free landmarks for the fine planning process.
- Fine planning: the number of random samples for the RRT algorithm is limited to 10^5, the same as the GT strategy.

5.3 THE PROPOSED SOFTWARE ARCHITECTURE

In order to generate the path planning query for a primitive action of a task plan, the software architecture of this thesis work consists of three different modules (Figure 46).

- The knowledge base module manages the task-related information in ontologies.
- The Java GUI module provides a user interface to facilitate the task-related information definition in ontologies
- The planning module generates the path planning query for a primitive action using task-related information (formalized in ontologies) and compute the relevant trajectories.

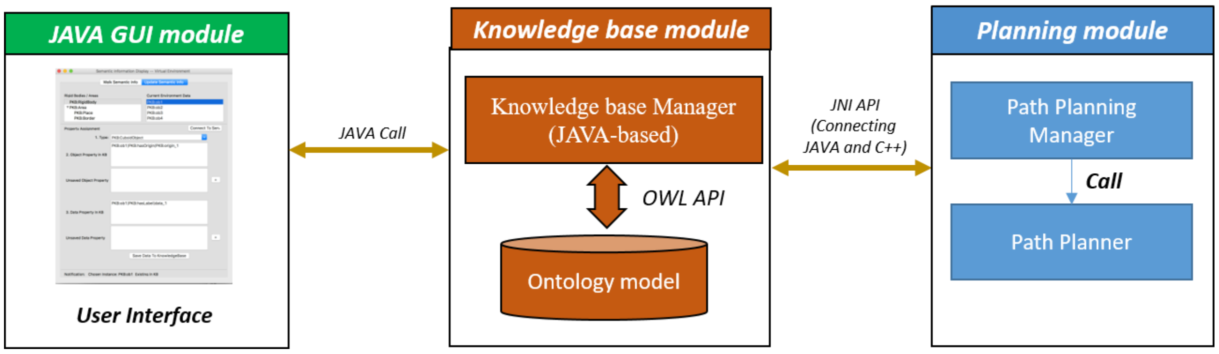


Figure 46 The architecture of the path planning system

The modules communicate with each other through different channels. In the knowledge base module, a Java-based knowledge base manager is developed to access the ontology model using classes and functions provided by OWLAPI. The JAVA GUI module calls the knowledge base manager (i.e., direct JAVA Calls) to manipulate information in the ontology model. The planning module is developed using C++ in Virtools. To access the information in the ontology model, it communicates with the knowledge base manager using JNI (Java Native Interface) API, who establishes the link between Java and C++.

The following sub-sections present how the three modules interact in the key processes of performing the path planning query for a primitive action.

A. Environment Construction and Instantiation

First of all, the 3D environment model of a task simulation should be constructed and then be instantiated in the ontology model. The environment knowledge is critical for performing a primitive action.

As we can see in Figure 47, the 3D environment of a task simulation is displayed in the 3D layout of Virtools. Taking it as input, the environment constructor builds a corresponding environment model. Then the knowledge base manager is called to instantiate the environment model in the ontology model. Moreover, the user can manually assign properties to rigid bodies, places and borders through a user interface. These properties can then be visualized in the 3D layout of Virtools by calling a display manager.

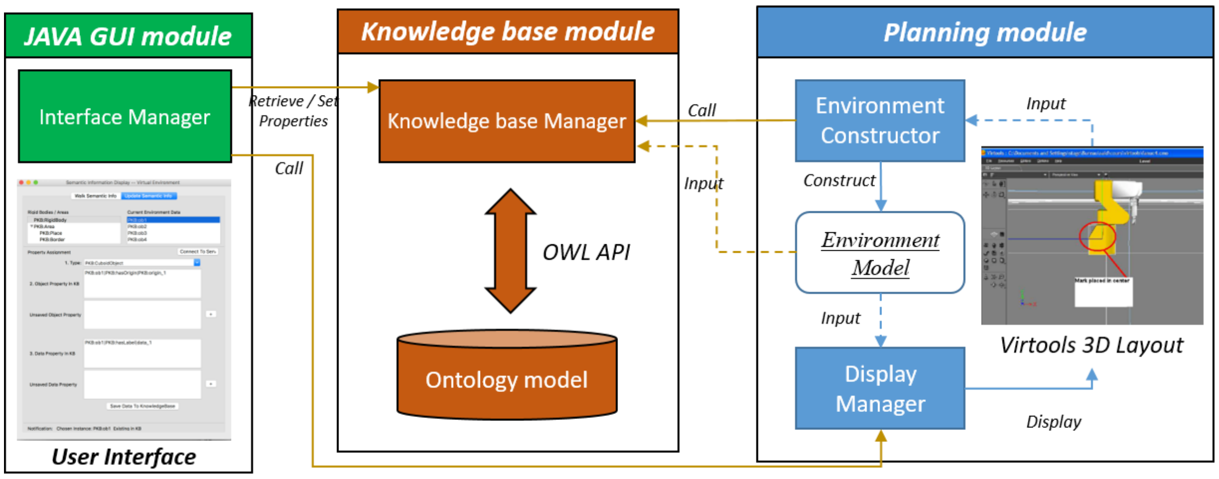


Figure 47 Environment Construction and Instantiation

B. The specification of a primitive action of a task plan

The primitive action has to be instantiated in the ontology model for the further generation of its corresponding path planning query. Currently, this thesis work focuses on path planning for a primitive action of a task plan. We do not take the generation of a task plan (i.e., task planning) into consideration. Instead, the specification of a primitive action is manually set by the user to simulate the generation process (Figure 48). Usually, spatial constraints are given by the user and are assigned along with the primitive action specification.

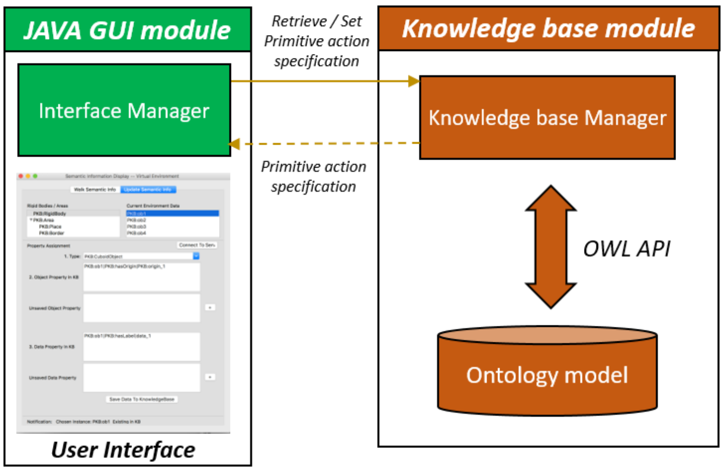


Figure 48 Primitive action specification

C. The path planning for a primitive action

The key contribution of this thesis work is to generate the path planning query for a primitive action of a task plan and to compute the corresponding trajectory for the manipulated object. As we can see from Figure 49, when the path planning manager receives the request to start path planning for a primitive action, it retrieves the parameterized path planning query from the knowledge base, including the start and the goal position and the corresponding geometric constraints. In particular, the geometric constraints are inferred from the spatial constraints, using some pre-defined inference rules by calling a reasoner. Afterward, taking the parameterized path planning query as input, a path planner is invoked to generate the corresponding trajectory, which is then displayed in the 3D layout of Virtools.

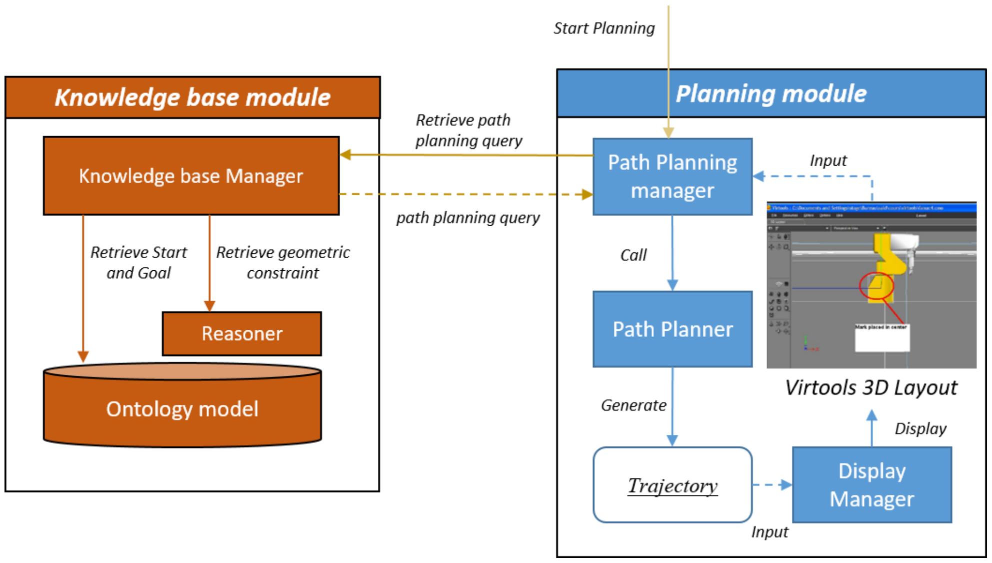
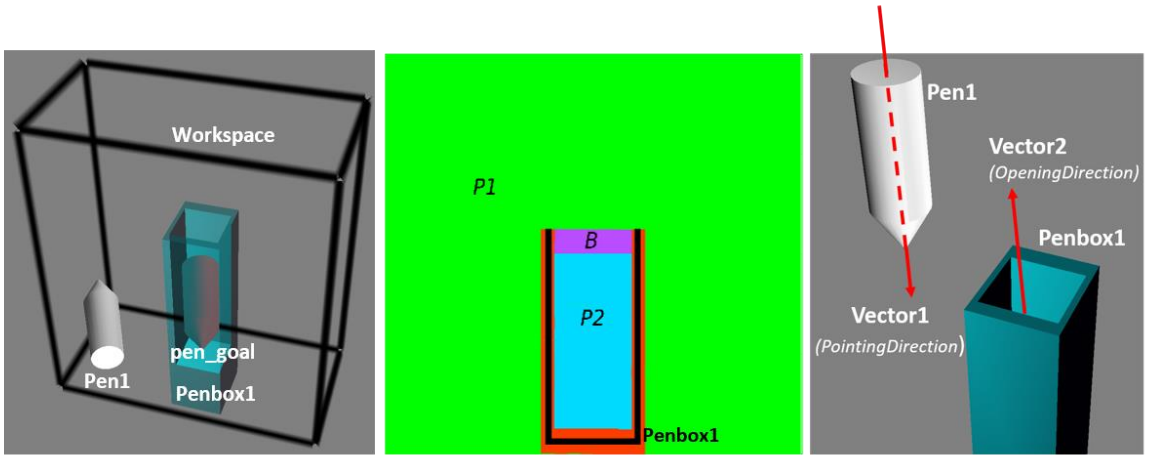


Figure 49 The path planning for a primitive action

5.4 SIMULATION SCENARIOS

The first case study concerns inserting a pen into a narrow penbox. Controlling the path planning process with geometric constraints provides a higher possibility of finding a collision-free trajectory for the insertion. The second case study introduces the “shape” attribute, which makes the insertion much harder.

5.4.1 Scenario 1: Pen-Penbox Insertion Use Case



a) b) c)

Figure 50 A pen-penbox insertion use case

The pen-penbox insertion use case (Figure 50-a) has already been discussed in section 4.5. The “workspace” of the simulation environment is the 3D Cartesian Space bounded by a line cube. In the workspace, two obstacles can be found. Among them, “Pen1” is a mobile obstacle and “Penbox1” is a fixed obstacle. The objective of the task simulation is to insert “Pen1” into “Penbox1”, where “pen_goal” is the configuration where the “Pen1” should reach. The “pen_goal” is obtained by pre-sampling within “Penbox1” (bounding box or P2).

In order to interface with the multi-level path planning architecture, the multi-level environment model should be first constructed. Figure 50-b demonstrates the construction of the topological level of the free space model (i.e., two places (P1, P2) and one border (B)) from the geometric level of the free space model (i.e., cell decomposition of the workspace). Comparing to the size of “Pen1”, P1 is enriched with the complexity attribute “Free” and P2 is “Narrow”. This allows to apply geometric constraints differently in P1 and P2. Figure 50-c demonstrated an example when “Pen1” is pointed to “Penbox1” (i.e., “Vector1” is against “Vectro2”). This constraint will be used in the path computation when “Pen1” is inserted into “Penbox1”.

5.4.2 Scenario 2: Shape Embedding Game

A more complex use case is inspired from the shape embedding game for babies. Comparing to the pen-penbox insertion use case, it requires additionally to match the shape between the hole and the manipulated object.

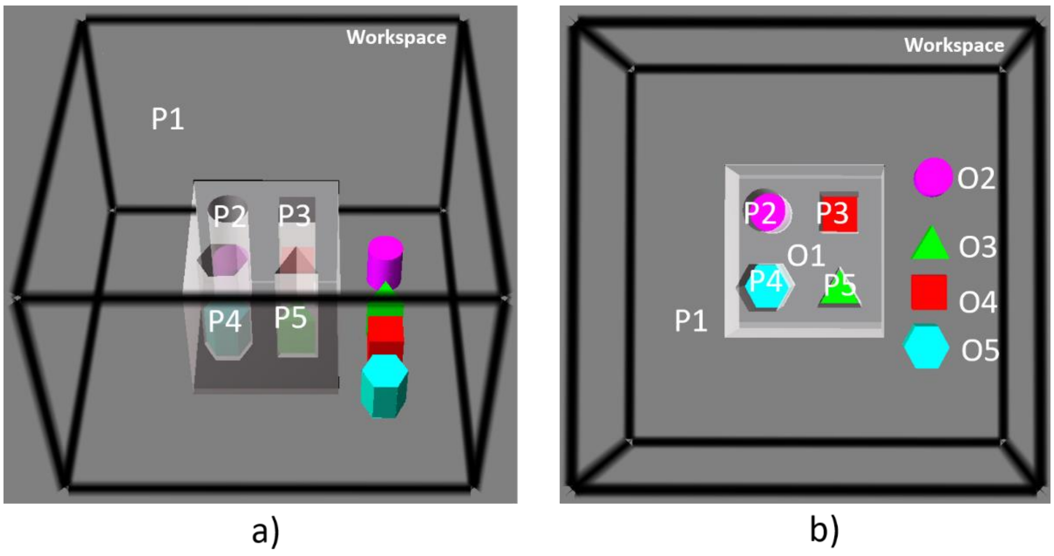


Figure 51 Shape Embedding Game

The 3D environment for the simulation (Figure 51) constitutes a cuboid workspace cluttered with 5 rigid bodies (O1 to O5). O1 is fixed and O2 to O5 are moveable. 5 different places (P1 to P5) are identified at the topological level of the 3D environment’s free space model.

Semantically, P2 to P5 are defined as O1's holes, and they respectively have the shape *(Quality:Shape)* of *RSQ_Cylinder, RSQ_Cuboid, RSQ_PentagonPrism* and *RSQ_Triangular Prism*. O2 to O5 have the shape *(Quality:Shape)* of *RSQ_Cylinder*, *RSQ_TriangularPrism*, *RSQ_Cuboid*, and *RSQ_PentagonPrism*. The objective of the task simulation is to insert O2 to O5 into holes with the same shape, i.e., O2 into P2, O3 into P5, O4 into P3, and O5 into P4. The valid goal configurations that O2 to O5 should reach are obtained by pre-sampling within P2 to P5.

5.5 VALIDATION OF SCENARIO 1

In this section, we first validate the ontology of three dimensional (3D) environment for the pen-penbox insertion scenario (section 5.5.1). Secondly, using the generated path planning query (section 5.5.2) as input, we compare the results obtained by different path planning strategies (section 5.5.3).

5.5.1 Verification and Validation of the ontology of 3D environment

The ontology verification evaluates whether an ontology is built correctly against ontology specification documents. The ontology validation evaluates whether a correct ontology is built against the intended model of the world aiming to conceptualize. In order to verify and validate the ontology of 3D environment for simulating manipulation tasks, we instantiate the ontology with the real environment data of the pen-penbox scenario and we examine whether the instantiated ontology can answer correctly to the competency questions listed in Chapter 3. In this research work, SPARQL (SPARQL Protocol and RDF Query Language) is used as the query language to retrieve data from the ontology.

Our motivation of constructing an ontology of 3D environment is to conceptualize heterogeneous environment information (i.e., semantics, topology, geometry) in a unified knowledge model, so that it can easily and effectively extract the environment data, such as "what is the central axis of Pen1?". This environment knowledge model will allow to determine the start and the goal location of a path planning query for a primitive action, or to identify the geometric elements used in the geometric constraints.

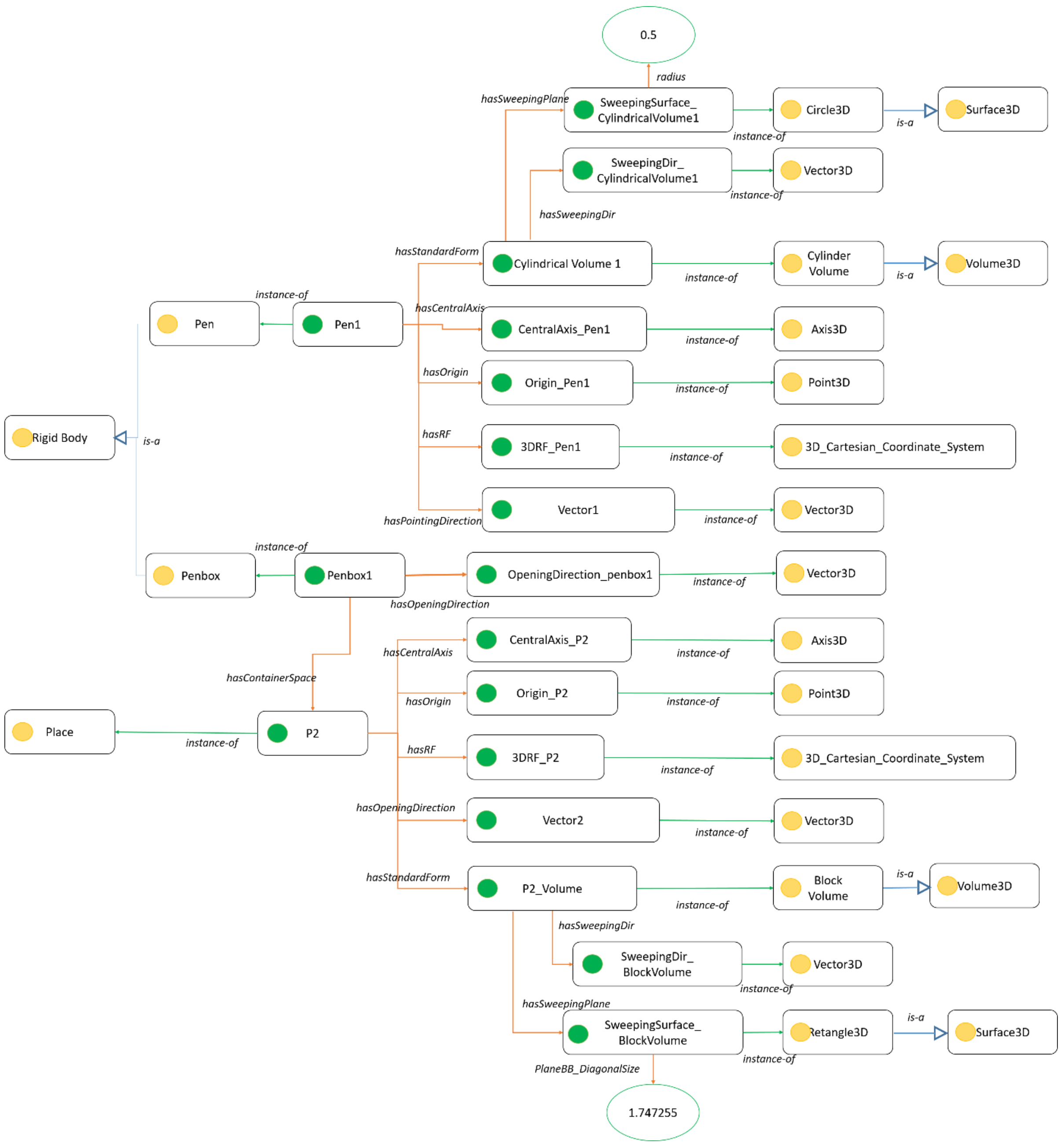


Figure 52 Instantiated Ontology for 3D Environment of the Pen-penbox insertion scenario

Firstly, the ontology of 3D environment is instantiated with the environment data of the pen-penbox insertion scenario (Figure 52). For example, Pen1 is an instance of *Pen* and thus an instance of *Rigid body*. Pen1 has different object properties, such as CentralAxis_Pen1 (*Axis3D*) as its central axis, Vector1(*Vector3D*) as its pointing direction, and CylindricalVolume1 (*CylindricalVolume*) as its standard form. SweepingDir_CylindricalVolume1 (*Vector3D*) and SweepingPlane_CylindricalVolume1 (*Circle3D*) are respectively the sweeping direction and the sweeping plane of the Pen1's *Volume3D* (CylindricalVolume1).

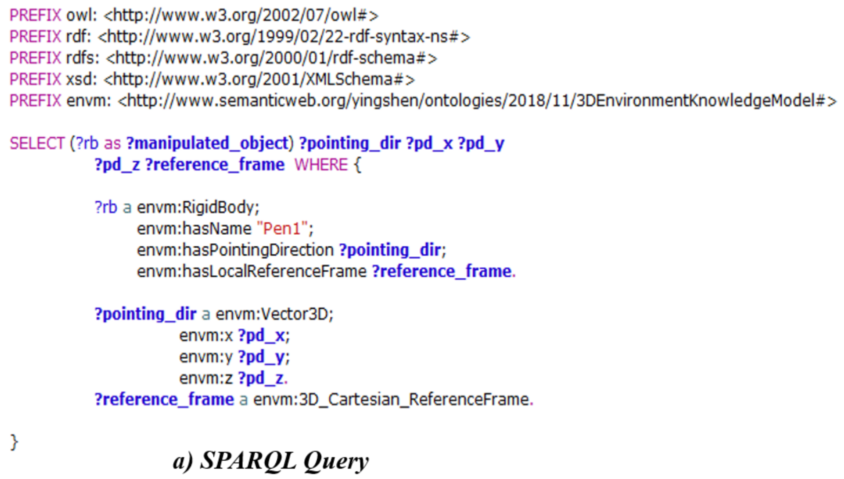
After instantiating the environment data in the ontology, we design and define some competency questions in Table 22 to validate the correctness of the proposed ontology of 3D environment. The evaluation also shows the facility of fast querying the environment data. For example,

- the competency question "What is the central axis of Pen1" is straight forward to search for the Pen1's central axis (i.e., *CentralAxis_Pen1*),
- the competency questions "What is the opening direction of P2" and "What is the pointing direction of Pen1" query the direction (*Vector3D*) where P2 opens or Pen1 points, i.e., Vector1 and Vector2.

Table 22 Competency Questions - Querying geometric details

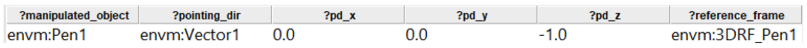
Rigid Body / Place	**Competency Questions**	**Result**
Pen1 (Type: Pen)	What is the central axis of Pen1 ?	CentralAxis_Pen1
	What are the sweeping plane and the sweeping direction of Pen1's Volume?	SweepingDir_CylindricalVolume1 SweepingSurface_CylindricalVolume1
	What is the pointing direction of Pen1?	Vector1 (see Figure 53)
Penbox1 (Type: Penbox)	What is the opening direction of Penbox1?	OpeningDirection_penbox1
P2 (Type: Place)	What is the opening direction of P2?	Vector2
	What are the sweeping plane and the sweeping direction of *P2*'s Volume?	SweepingDir_BlockVolume SweepingSurface_BlockVolume

In Figure 53-a, we demonstrate an example of SPARQL query to search for the pointing direction of Pen1 (*?pointing_dir*) and the local reference frame in which *?pointing_dir* is defined. Figure 53-b and c respectively show the obtained results and their visual display in Virtools.



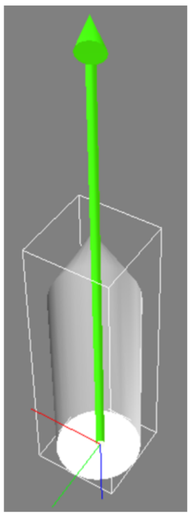
```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX envm: <http://www.semanticweb.org/yingshen/ontologies/2018/11/3DEnvironmentKnowledgeModel#>

SELECT (?rb as ?manipulated_object) ?pointing_dir ?pd_x ?pd_y
            ?pd_z ?reference_frame  WHERE {

            ?rb a envm:RigidBody;
                  envm:hasName "Pen1";
                  envm:hasPointingDirection ?pointing_dir;
                  envm:hasLocalReferenceFrame ?reference_frame.

            ?pointing_dir a envm:Vector3D;
                        envm:x ?pd_x;
                        envm:y ?pd_y;
                        envm:z ?pd_z.
            ?reference_frame a envm:3D_Cartesian_ReferenceFrame.

}
```

a) SPARQL Query



?manipulated_object	?pointing_dir	?pd_x	?pd_y	?pd_z	?reference_frame
envm:Pen1	envm:Vector1	0.0	0.0	-1.0	envm:3DRF_Pen1

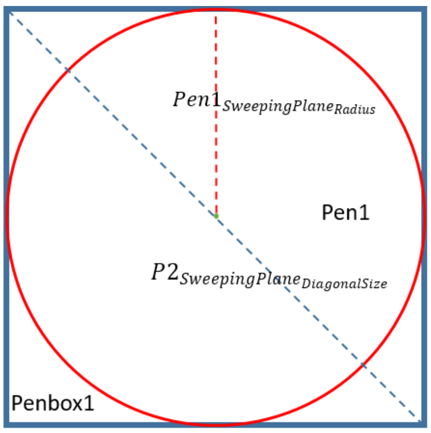
b) Query – Result



c) Visual Display

Figure 53 SPARQL Query Result - Pen1

Moreover, in order to correctly insert Pen1 into Penbox1, we have firstly to determine whether Pen1 can be inserted into Penbox1. In this scenario, it means whether Pen1 can be inserted into P2. The competency question of this issue is described in Table 23. P2 is a *Place* which has a standard form *BlockVolume*, and Pen1 is a *Rigid Body* who has a standard form *CylindricalVolume*. Both standard forms of P2 and Pen1 have the regular sweeping plane *Retangle3D* and *Circle3D*, i.e., the length of P2's *Retangle3D* equal and Pen1's *Circle3D* is round. Therefore, the condition of whether Pen1 can be inserted into P2 is shown in Figure 54.


$Pen1_{SweepingPlane_{Radius}}$
Pen1
$P2_{SweepingPlane_{DiagonalSize}}$
Penbox1

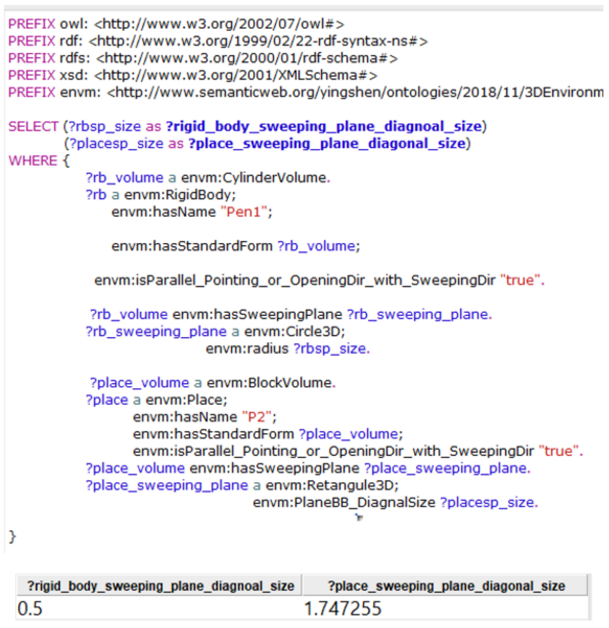
$$Pen1_{SweepingPlane_{Radius}} < 0.35355 * P2_{SweepingPlane_{DiagonalSize}}$$

Figure 54 The condition of whether Pen1 can be inserted into P2

Table 23 Competency Question – The possibility of inserting Pen1 into Penbox1

Competency Questions	**Result**
Whether Pen1 can be inserted into P2?	1 (meaning: Yes) Pen1_SweepingPlane_Radius: 0.5 P2_SweepingPlane_DiagnalSize: 1.747255

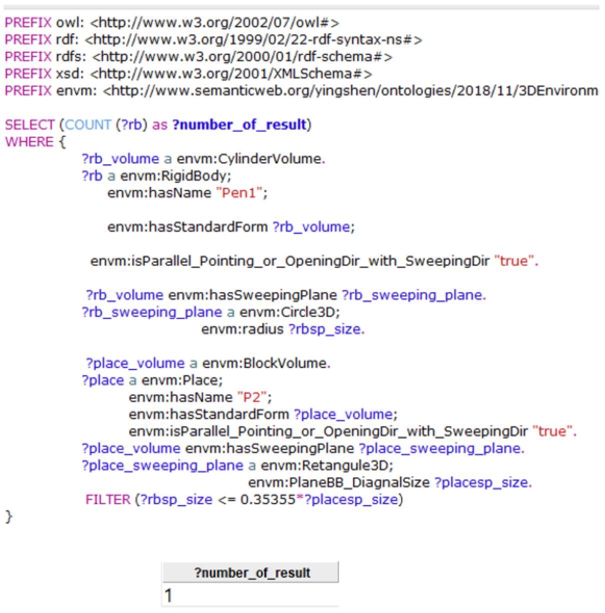
In Figure 55-a, we demonstrate an example of SPARQL query to search for the radius of the sweeping plane of Pen1 *(?rigid_body_sweeping_plane_diagonal_size*) and the diagonal size of the sweeping plane of P2 (*?place_sweeping_plane_diagonal_size*). In Figure 55-b, we demonstrate an example of SPARQL query to determine whether Pen1 can be inserted into P2. Because results can be found *(?number_of_result = 1*) so that Pen1 can be inserted into Penbox1 in this use case.



```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX envm: <http://www.semanticweb.org/yingshen/ontologies/2018/11/3DEnvironm

SELECT (?rbsp_size as ?rigid_body_sweeping_plane_diagnoal_size)
       (?placesp_size as ?place_sweeping_plane_diagonal_size)
WHERE {
          ?rb_volume a envm:CylinderVolume.
          ?rb a envm:RigidBody;
             envm:hasName "Pen1";

             envm:hasStandardForm ?rb_volume;

           envm:isParallel_Pointing_or_OpeningDir_with_SweepingDir "true".

           ?rb_volume envm:hasSweepingPlane ?rb_sweeping_plane.
          ?rb_sweeping_plane a envm:Circle3D;
                         envm:radius ?rbsp_size.

           ?place_volume a envm:BlockVolume.
          ?place a envm:Place;
                envm:hasName "P2";
                envm:hasStandardForm ?place_volume;
                envm:isParallel_Pointing_or_OpeningDir_with_SweepingDir "true".
          ?place_volume envm:hasSweepingPlane ?place_sweeping_plane.
          ?place_sweeping_plane a envm:Retangule3D;
                               envm:PlaneBB_DiagnalSize ?placesp_size.

}
```

?rigid_body_sweeping_plane_diagnoal_size	?place_sweeping_plane_diagonal_size
0.5	1.747255

a) Sweeping plane's radius or diagonal size



```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX envm: <http://www.semanticweb.org/yingshen/ontologies/2018/11/3DEnvironm

SELECT (COUNT (?rb) as ?number_of_result)
WHERE {
          ?rb_volume a envm:CylinderVolume.
          ?rb a envm:RigidBody;
             envm:hasName "Pen1";

             envm:hasStandardForm ?rb_volume;

           envm:isParallel_Pointing_or_OpeningDir_with_SweepingDir "true".

           ?rb_volume envm:hasSweepingPlane ?rb_sweeping_plane.
          ?rb_sweeping_plane a envm:Circle3D;
                         envm:radius ?rbsp_size.

           ?place_volume a envm:BlockVolume.
          ?place a envm:Place;
                envm:hasName "P2";
                envm:hasStandardForm ?place_volume;
                envm:isParallel_Pointing_or_OpeningDir_with_SweepingDir "true".
          ?place_volume envm:hasSweepingPlane ?place_sweeping_plane.
          ?place_sweeping_plane a envm:Retangule3D;
                               envm:PlaneBB_DiagnalSize ?placesp_size.
           FILTER (?rbsp_size <= 0.35355*?placesp_size)
}
```

?number_of_result
1

b) Result

Figure 55 SPARQL Query - – The possibility of inserting a pen into a penbox

5.5.2 Validation of the generation of the path planning query for a primitive action

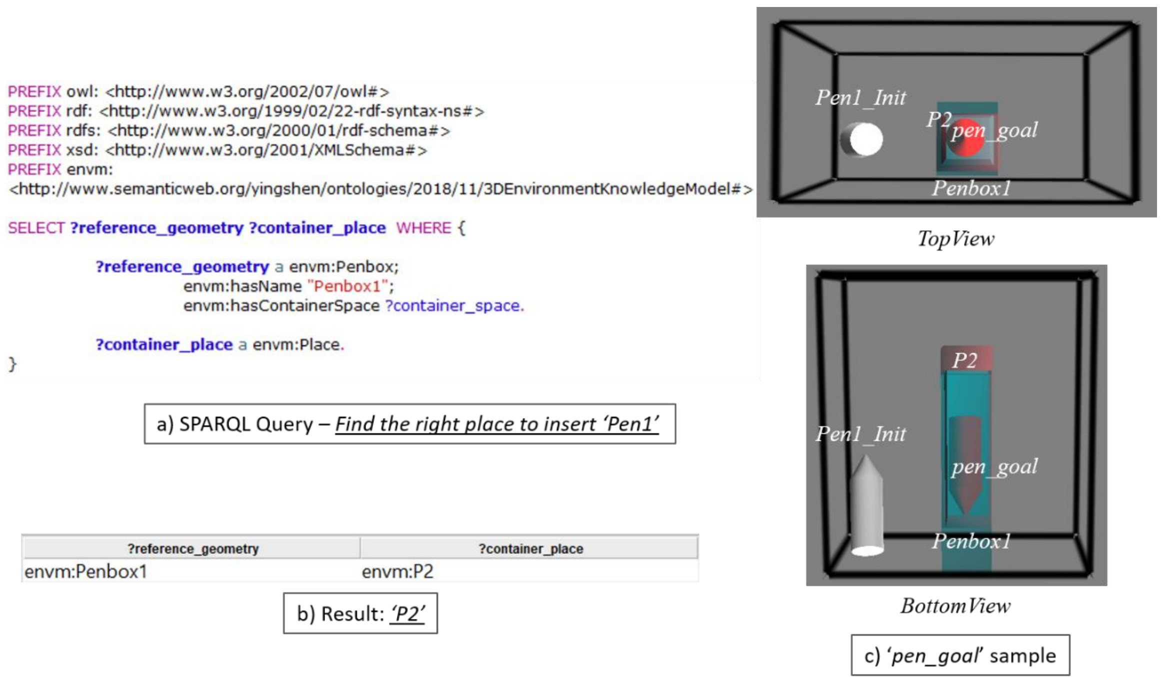
In Table 24, the performed action of this use case is presented: Pen1 should be inserted into Penbox1, along with a spatial constraint that Pen1 should point to Penbox1.

Table 24 The primitive action specification

Primitive action (PA) to perform	Insert
The manipulated object	Pen1
The reference geometry	Penbox1
PA constraints	Pen1 *PointTo* Penbox1

The corresponding path planning query is built from the specification of this primitive action.

Firstly, the goal configuration of the path planning query is obtained by random sampling in the goal place. The sampling process must obey the geometric constraints associated with the goal place. In Figure 56-a, we show an example of SPARQL query of finding the right place to insert Pen1. Figure 56-b shows the result we obtained, i.e., P2 is the goal place. Figure 56-c shows the goal configuration of Pen1 (i.e., *pen_goal*) is obtained using a random sampling within P2.



```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX envm:
<http://www.semanticweb.org/yingshen/ontologies/2018/11/3DEnvironmentKnowledgeModel#>

SELECT ?reference_geometry ?container_place  WHERE {

          ?reference_geometry a envm:Penbox;
                    envm:hasName "Penbox1";
                    envm:hasContainerSpace ?container_space.

          ?container_place a envm:Place.
}
```

a) SPARQL Query – *Find the right place to insert 'Pen1'*

?reference_geometry	?container_place
envm:Penbox1	envm:P2

b) Result: *'P2'*

Figure 56 The search of goal to reach: 'pen_goal'

Secondly, the spatial constraint is mapped into the corresponding geometric constraint, through an inference process. Finally, the description of the generated path planning query is shown in Table 25.

Table 25 The corresponding path planning query

Path planning(PP) query	
The manipulated object	Pen1
Goal configuration	pen_goal
PP constraints	Vector1 *Against* Vector2 (Figure 50-c)

This use case aims to take advantages of "Vector1 *Against* Vector2" to control the trajectory definition when Pen1 is inserted into Penbox1. Appendix 1 describes how to use this geometric constraint to control the possible orientation of Pen1.

5.5.3 Validation of non-interactive path planning for a primitive action

In secion 5.5.2, a parameterized path planning query is generated for a primitive action "*Insert (Pen1, Penbox1)*". Taking this path planning query as input, different path planning strategies (G, GT, GTS, GO and GTO strategies) are compared in Figure 57 and Figure 58.

- The GT and the GTS strategies use almost ½ computational time of the G strategy, but with a bit more random samples. The result is due to that the multi-level path planning process reduces the search space of RRT to the place where a topological step crosses.

Since two topological steps are involved in the pen-penbox insertion case, the total number of random samples can be more than the one of the G strategy.

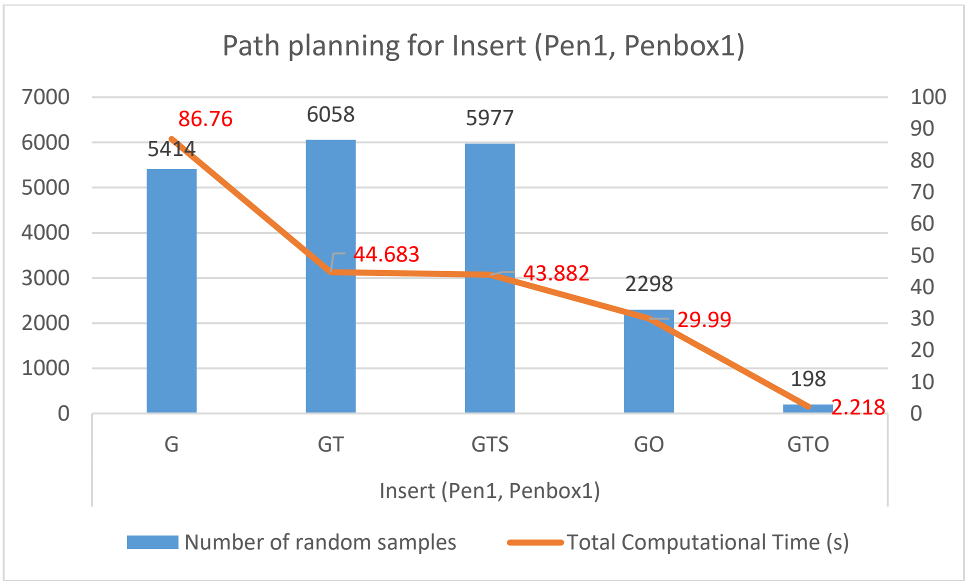


Figure 57 Path planning results for 'Insert (Pen1, Penbox1)'

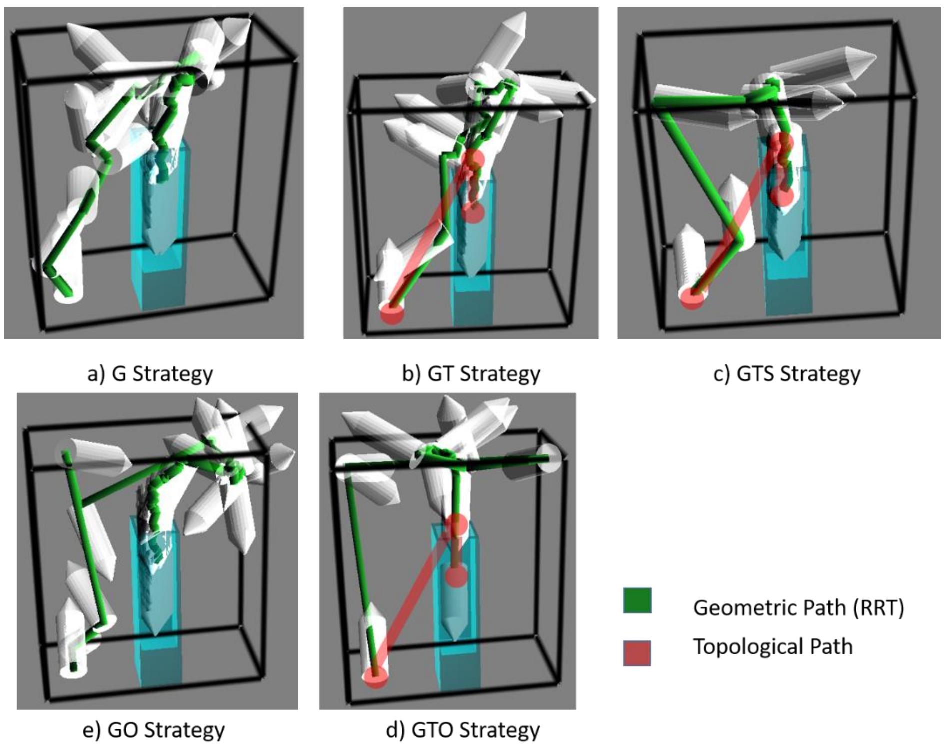


Figure 58 The computed path using 'G, GT, GTO, GO, GTS' Strategies

- The results of the GT and the GTS strategies do not have a big difference, due to that the semantic information (i.e., complexity and geometric form) do not control the motions of how Pen1 should be inserted into P2.

- The GO strategy uses less than half the number of random samples and 1/3 computational time, comparing to the G strategy. The result is caused by the inferred geometric constraints (from task-related information) to reduce the search space of RRT.
- The result of the GTO strategy is much better than the one of the GO strategy, because the GTO strategy can determine the key milestone configuration at the border (B) between P1 and P2. Once it is determined, the search space of RRT is reduced to the place where a topological step crosses.

5.5.4 Synthesis on the experimental results of pen-penbox insertion scenario

Firstly, the ontology of 3D environment is instantiated with the environment data of the pen-penbox insertion scenario. The instantiated ontology of 3D environment is able to answer to different competency questions proposed, and thus the correctness of this ontology is validated and verified, regarding to the requirement document of building such ontology.

Secondly, the path planning query of a primitive action of a task plan is generated. The generation process uses an ontology of 3D environment where the simulated task occurs to determine

- not only the start and the goal configurations,
- but also the task-related geometric constraints through an inference process.

Thirdly, using this path planning query as input, the results of non-interactive path planning based on different strategies are analyzed, and thus their advantages and disadvantages are discussed.

The G, GT, and GTS strategies reuse the ones of the multi-level path planning [Cailhol 2015].

- The G strategy (RRT) uses only the geometric level information of the environment. It has the ability of fast exploration of the environment. However, it encounters difficulties in finding collision-free trajectories in narrow passages. The results of the use case show that the G strategy uses much more computational time than others.
- The GT strategy uses the topological level information overlapped on the geometric level. It firstly computes a topological path (made of topological steps) in a reasonable time. Because the search space of RRT is restricted to the place where a topological step crosses, the GT strategy costs less time than the G strategy.
- The GTS strategy (using the semantic, topologic and geometric environment information) is able to control the path planning process by reasoning on the environment information at a higher level of abstraction (topology, semantics).

However, the semantic information (complexity and geometric form) is insufficient to restrict the samples' poses in the narrow passages. Thus, the GTS strategy has a similar result as the GT strategy in the scenario 1.

The GO and the GTO strategies take advantages of the task-related geometric constraints associated with a path planning query of a primitive action.

- The GO strategy (RRT) uses the geometric constraints to limit the poses of random samples, when they are near the goal configuration. It reduces the search space for collision-free trajectories, so that less processing time and less random samples are needed (comparing to G, GT, GTS strategies).
- The GTO strategy is similar to the GTS strategy. Instead of the text information used in the GTS strategy (i.e., complexity, geometric form), the GTO strategy applies the geometric constraints all along the computed topological path. The search space is reduced to the place where a topological step crosses; the poses of random samples are limited by the geometric constraints; the key geometric configuration is found at the border to enter the narrow passage. The result of the GTO strategy demonstrates a huge improvement comparing to others (G, GT, GTS, GO strategies).

The advantages of the GTO strategy over the others show the interests of using the task level information (e.g., the task-related geometric constraints) to restrict the poses of random samples, regarding the places where the topological path crosses. It computes a collision-free trajectory for a primitive action in a reasonable time, so that the real-time interaction between the user and the path planning system in performing manipulation tasks can be considered in the future.

5.6 VALIDATION OF SCENARIO 2

In this section, we define and use a more complex use case, inspired from the shape embedding game for babies, to demonstrate the benefits of using task-related information in the path planning for a primitive action. Similar to the discussion in section 5.5, we first validate the ontology of 3D environment for the shape embedding game by answering some competency questions (section 5.6.1). Secondly, using the generated path planning query (section 5.6.2) as input, the results of different path planning strategies are compared (section 5.6.3).

5.6.1 Verification and Validation of the ontology of 3D environment

First of all, the environment data of the shape embedding game scenario is firstly instantiated in the ontology of 3D environment, as shown in Figure 59. This data consists of the semantic, topologic and geometric information of the 3D environment where the simulated task takes place.

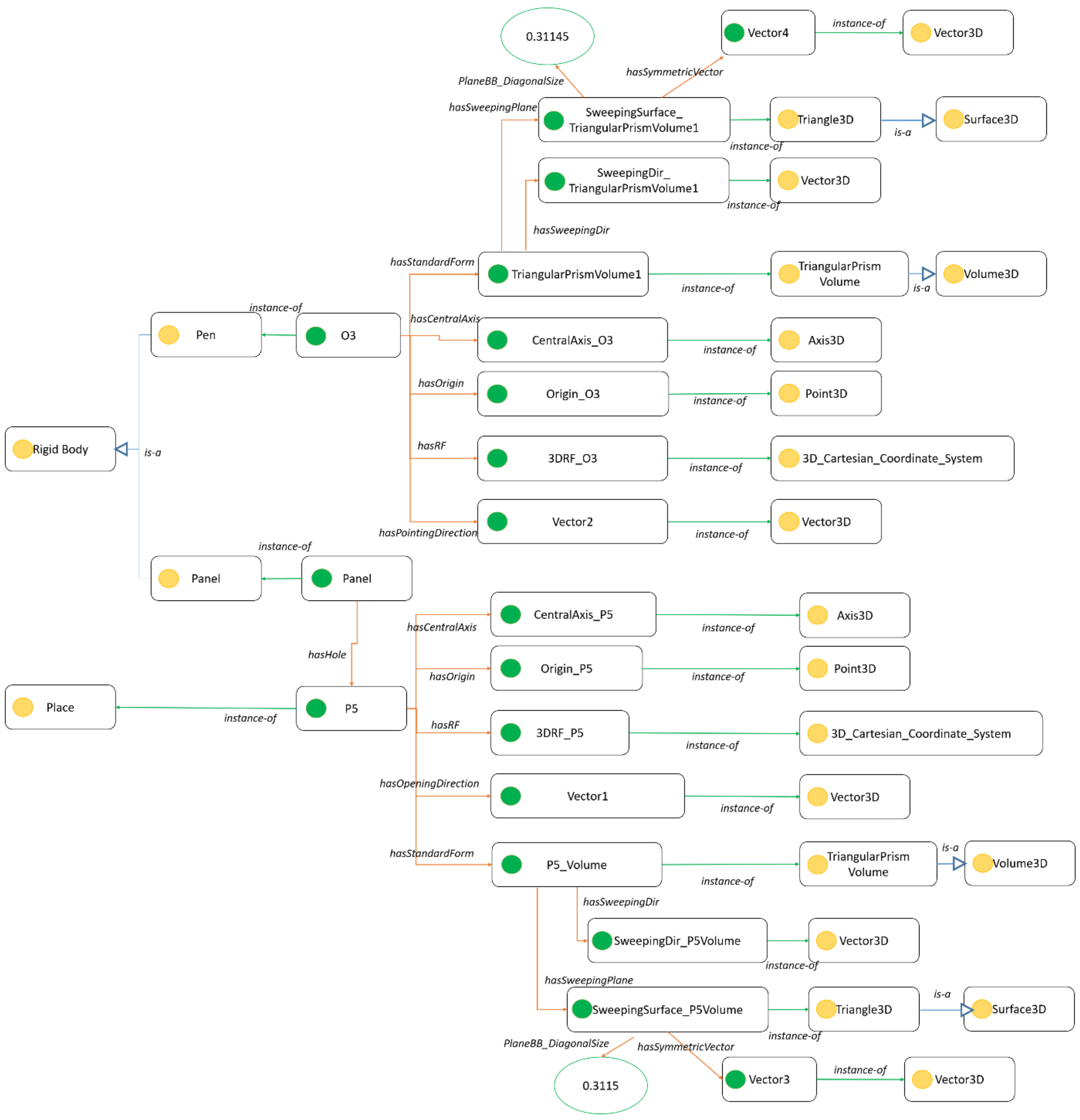


Figure 59 Instantiated Ontology for 3D Environment of the Shape Embedding Game Scenario

At the geometric level, different geometric properties of rigid bodies and places are captured. In the shaped game scenario, O3 is an instance of *RigidBody* and it has a pointing direction Vector2 (type: *Vector3D*), a local reference frame 3DRF_O3 (type: *3DCartesianReference Frame*), an origin Origin_O3 (*Point3D*), a central axis CentralAxis_O3 (*Axis3D*), and a standard form TriangularPrismVolume1 (type: *TriangularPrismVolume*). O3's standard from has a sweeping plane SweepingSurface_TriangularPrismVolume1 (type: *Triangle3D*) and has a sweeping direction SweepingDir_TriangularPrimsVolume1 (type: *Vector3D*). The sweeping plane's diagonal size is 0.31145. P5 is an instance of place; it has a standard form P5_Volume (type: *TriangularPrismVolume*) with different diagonal size 0.3115 of the sweeping plane SweepingSurface_P5Volume (type; *TrangularPrismVolume*). At the topological level, P1, P2, P3, P4, P5 are 5 different places constructed. P2-P5 has direct topological connections with P1. At the semantic level, O3 is further defined as an instance of Pen. P2-P5 are instances of *Hole* (they are narrow and have respectively shape of *RSQ_Cylinder*, *RSQ_Block*, *RSQ_TriangularPrism*, *RSQ_PentagnonPrism*).

We design and define some competency questions in Table 26 and Table 27. We can see from the obtained results that the instantiated ontology of 3D environment can correctly process these questions with the right answers returned. For example, "What is the opening direction of O3" and "What is the pointing direction of P5" correctly returns with the answers Vector2 and Vector 3.

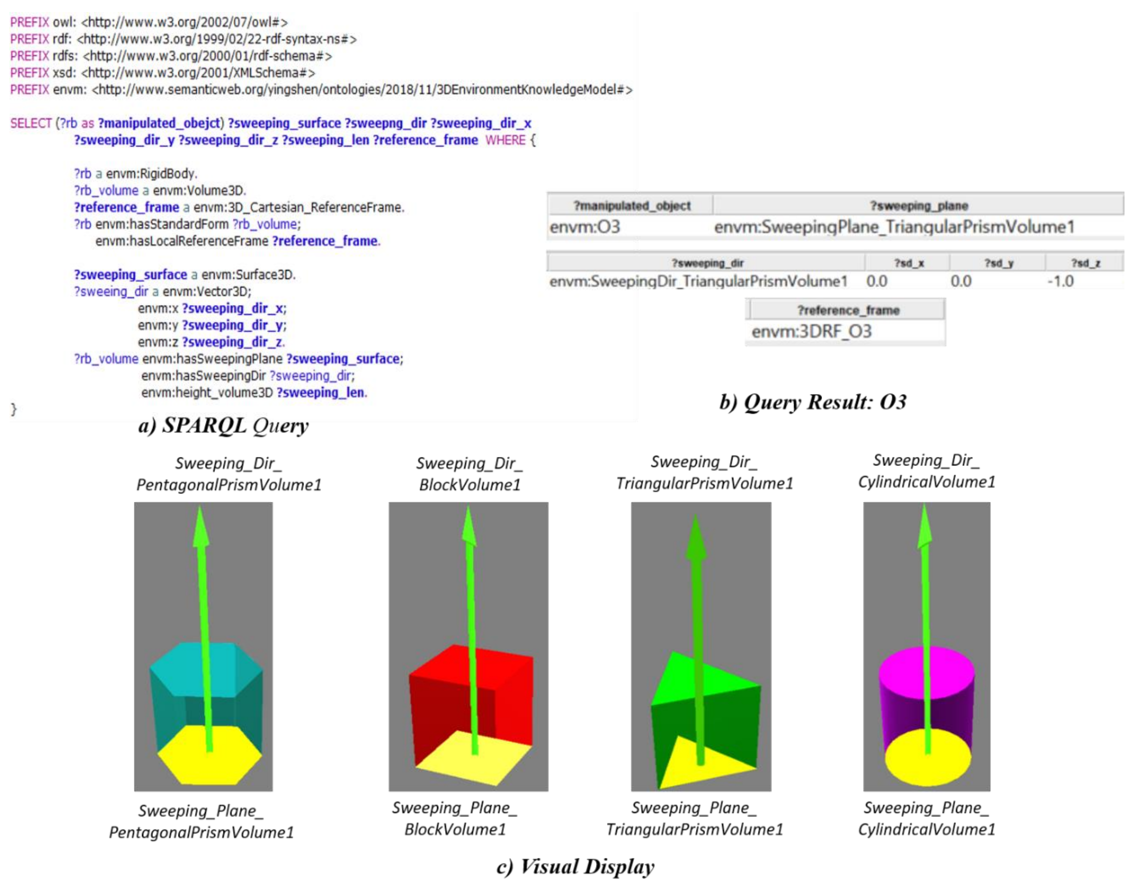
Table 26 Competency Question - Query geometric details

Rigid Body	Competency Question	Result
O3	What is the central axis of O3?	CentralAxis_O3
	What is the pointing direction of O3?	Vector2
	What are the sweeping plane and the sweeping direction of O3's Volume?	SweepingDir_TriangularPrismVolume1 SweepingSurface_TriangularPrismVolume1 (see Figure 60)
	What is the symmetric vector for the sweeping plane of P5's Volume	Vector4

Table 27 Competency Question - Query geometric details (P5)

Place	Competency Question	Result
P5	What is the central axis of P5?	CentralAxis_P5
	What is the opening direction of P5?	Vector1
	What are the sweeping plane and the sweeping direction of P5's Volume?	SweepingDir_P5Volume SweepingSurface_P5Volume (see Figure 61Figure 60)
	What is the symmetric vector for the sweeping plane of P5's Volume	Vector3

In Figure 60-a and Figure 61-a, we demonstrate two example of SPARQL query to respectively search for the sweeping plane and the sweeping direction of O3 and P5. Figure 60 and Figure 61 (b and c) respectively show the obtained results and their visual display in Virtools.



```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX envm: <http://www.semanticweb.org/yingshen/ontologies/2018/11/3DEnvironmentKnowledgeModel#>

SELECT (?rb as ?manipulated_obejct) ?sweeping_surface ?sweepng_dir ?sweeping_dir_x
          ?sweeping_dir_y ?sweeping_dir_z ?sweeping_len ?reference_frame  WHERE {

          ?rb a envm:RigidBody.
          ?rb_volume a envm:Volume3D.
          ?reference_frame a envm:3D_Cartesian_ReferenceFrame.
          ?rb envm:hasStandardForm ?rb_volume;
              envm:hasLocalReferenceFrame ?reference_frame.

          ?sweeping_surface a envm:Surface3D.
          ?sweeing_dir a envm:Vector3D;
                    envm:x ?sweeping_dir_x;
                    envm:y ?sweeping_dir_y;
                    envm:z ?sweeping_dir_z.
          ?rb_volume envm:hasSweepingPlane ?sweeping_surface;
                     envm:hasSweepingDir ?sweeping_dir;
                     envm:height_volume3D ?sweeping_len.
}
```

a) SPARQL Query

?manipulated_object	?sweeping_plane
envm:O3	envm:SweepingPlane_TriangularPrismVolume1

?sweeping_dir	?sd_x	?sd_y	?sd_z
envm:SweepingDir_TriangularPrismVolume1	0.0	0.0	-1.0

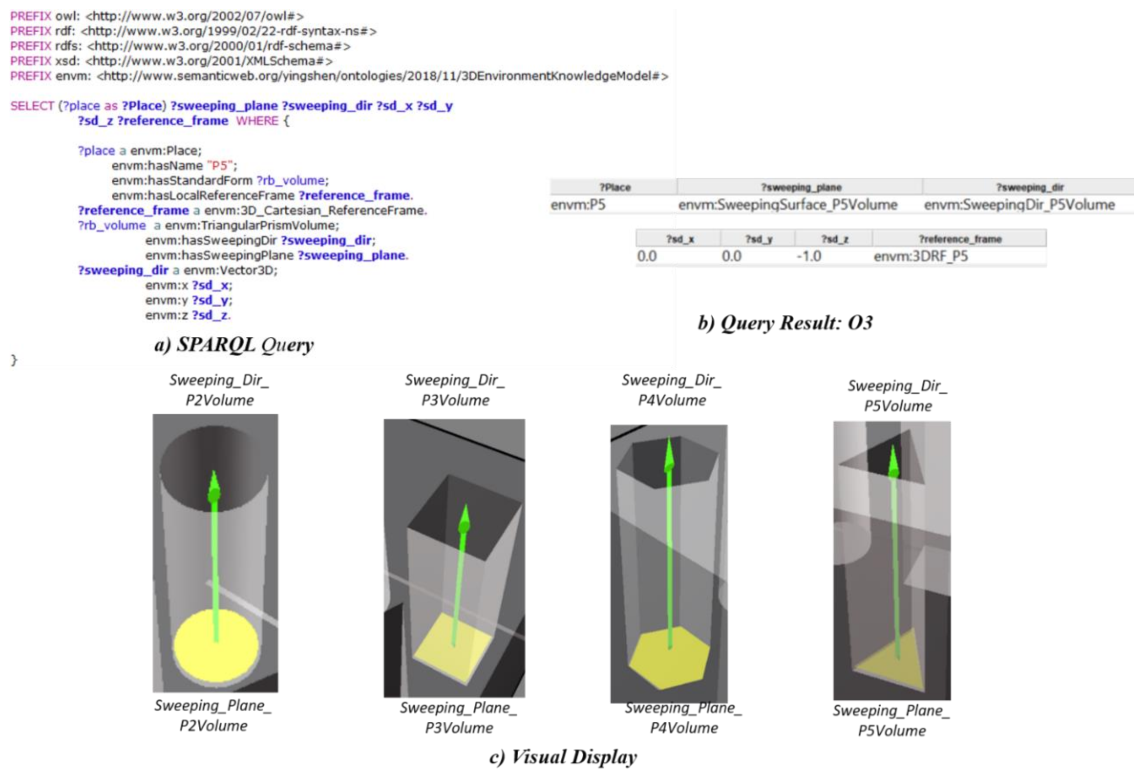
?reference_frame
envm:3DRF_O3

b) Query Result: O3

Sweeping_Dir_ PentagonalPrismVolume1
Sweeping_Dir_ BlockVolume1
Sweeping_Dir_ TriangularPrismVolume1
Sweeping_Dir_ CylindricalVolume1
Sweeping_Plane_ PentagonalPrismVolume1
Sweeping_Plane_ BlockVolume1
Sweeping_Plane_ TriangularPrismVolume1
Sweeping_Plane_ CylindricalVolume1

c) Visual Display

Figure 60 SPARQL Query Result - O3



```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX envm: <http://www.semanticweb.org/yingshen/ontologies/2018/11/3DEnvironmentKnowledgeModel#>

SELECT (?place as ?Place) ?sweeping_plane ?sweeping_dir ?sd_x ?sd_y
          ?sd_z ?reference_frame  WHERE {

          ?place a envm:Place;
                envm:hasName "P5";
                envm:hasStandardForm ?rb_volume;
                envm:hasLocalReferenceFrame ?reference_frame.
          ?reference_frame a envm:3D_Cartesian_ReferenceFrame.
          ?rb_volume  a envm:TriangularPrismVolume;
                    envm:hasSweepingDir ?sweeping_dir;
                    envm:hasSweepingPlane ?sweeping_plane.
          ?sweeping_dir a envm:Vector3D;
                    envm:x ?sd_x;
                    envm:y ?sd_y;
                    envm:z ?sd_z.
}
```

a) SPARQL Query

?Place	?sweeping_plane	?sweeping_dir
envm:P5	envm:SweepingSurface_P5Volume	envm:SweepingDir_P5Volume

?sd_x	?sd_y	?sd_z	?reference_frame
0.0	0.0	-1.0	envm:3DRF_P5

b) Query Result: O3

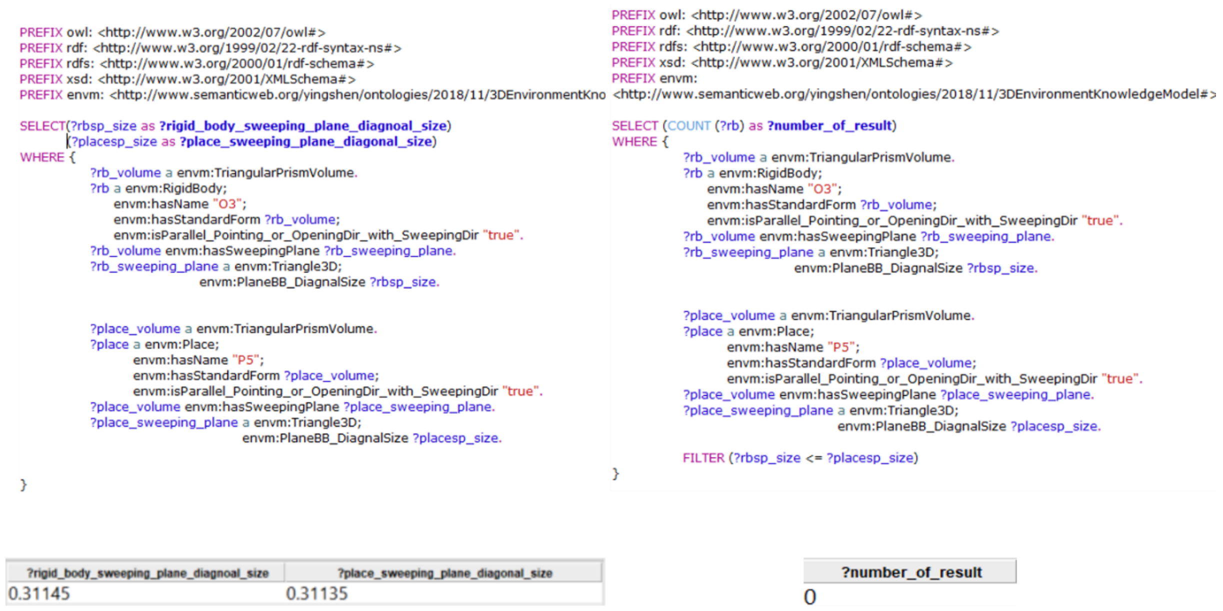
c) Visual Display

Figure 61 SPARQL Query Result - P5

In Table 28, we design two derivations: the first one specifies that the diagonal size of P5's sweeping plane is 0.3115 and the second one 0.31135. A competency question is defined to check whether O3 can be inserted into P5. Because both O3 and P5 have standard forms *TriangularPrismVolume*, the result relies on the diagonal size of O3's and P5's sweeping planes. Because the diagonal size of O3's sweeping plane is smaller than the one of P5, O3 can be inserted into the P5 in derivation 1. Otherwise, it is impossible to perform this primitive action (see derivation 2).

Table 28 Competency Question – The possibility of inserting the triangular prism pen into P5

Competency Questions	**Derivation**	**Result**
Whether O3 can be inserted into P5?	Derivation 1	1 (meaning: Yes) O3_SweepingPlane_DiagnalSize: 0.31145 P5_SweepingPlane_DiagnalSize: 0.3115
	Derivation 2	0 (meaning: No) O3_SweepingPlane_DiagnalSize: 0.31145 P5_SweepingPlane_DiagnalSize: 0.31135



```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX envm: <http://www.semanticweb.org/yingshen/ontologies/2018/11/3DEnvironmentKno

SELECT(?rbsp_size as ?rigid_body_sweeping_plane_diagnoal_size)
      (?placesp_size as ?place_sweeping_plane_diagonal_size)
WHERE {
        ?rb_volume a envm:TriangularPrismVolume.
        ?rb a envm:RigidBody;
            envm:hasName "O3";
            envm:hasStandardForm ?rb_volume;
            envm:isParallel_Pointing_or_OpeningDir_with_SweepingDir "true".
        ?rb_volume envm:hasSweepingPlane ?rb_sweeping_plane.
        ?rb_sweeping_plane a envm:Triangle3D;
                        envm:PlaneBB_DiagnalSize ?rbsp_size.


        ?place_volume a envm:TriangularPrismVolume.
        ?place a envm:Place;
                envm:hasName "P5";
                envm:hasStandardForm ?place_volume;
                envm:isParallel_Pointing_or_OpeningDir_with_SweepingDir "true".
        ?place_volume envm:hasSweepingPlane ?place_sweeping_plane.
        ?place_sweeping_plane a envm:Triangle3D;
                            envm:PlaneBB_DiagnalSize ?placesp_size.


}
```

```
?rigid_body_sweeping_plane_diagnoal_size    ?place_sweeping_plane_diagonal_size
0.31145                                     0.31135
```

a) Sweeping plane's radius or diagonal size

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX envm:
<http://www.semanticweb.org/yingshen/ontologies/2018/11/3DEnvironmentKnowledgeModel#>

SELECT (COUNT (?rb) as ?number_of_result)
WHERE {
        ?rb_volume a envm:TriangularPrismVolume.
        ?rb a envm:RigidBody;
            envm:hasName "O3";
            envm:hasStandardForm ?rb_volume;
            envm:isParallel_Pointing_or_OpeningDir_with_SweepingDir "true".
        ?rb_volume envm:hasSweepingPlane ?rb_sweeping_plane.
        ?rb_sweeping_plane a envm:Triangle3D;
                        envm:PlaneBB_DiagnalSize ?rbsp_size.


        ?place_volume a envm:TriangularPrismVolume.
        ?place a envm:Place;
                envm:hasName "P5";
                envm:hasStandardForm ?place_volume;
                envm:isParallel_Pointing_or_OpeningDir_with_SweepingDir "true".
        ?place_volume envm:hasSweepingPlane ?place_sweeping_plane.
        ?place_sweeping_plane a envm:Triangle3D;
                            envm:PlaneBB_DiagnalSize ?placesp_size.

        FILTER (?rbsp_size <= ?placesp_size)
}
```

```
?number_of_result
0
```

b) Result

Figure 62 SPARQL Query - – The possibility of inserting O3 into P5 (derivation 2)

In Figure 62-a, we demonstrate an example of SPARQL query to search for the radius of the sweeping plane of O3 (*?rigid_body_sweeping_plane_diagonal_size*) and the diagonal size of the sweeping plane of P5 (*?place_sweeping_plane_diagonal_size*). In Figure 62-b, we demonstrate an example of SPARQL query to determine whether O3 can be inserted into P5. No result can be found (*?number_of_result = 0*) so that O3 cannot be inserted into P5 in the derivation 2.

5.6.2 Validation of the generation of the path planning query for a primitive action

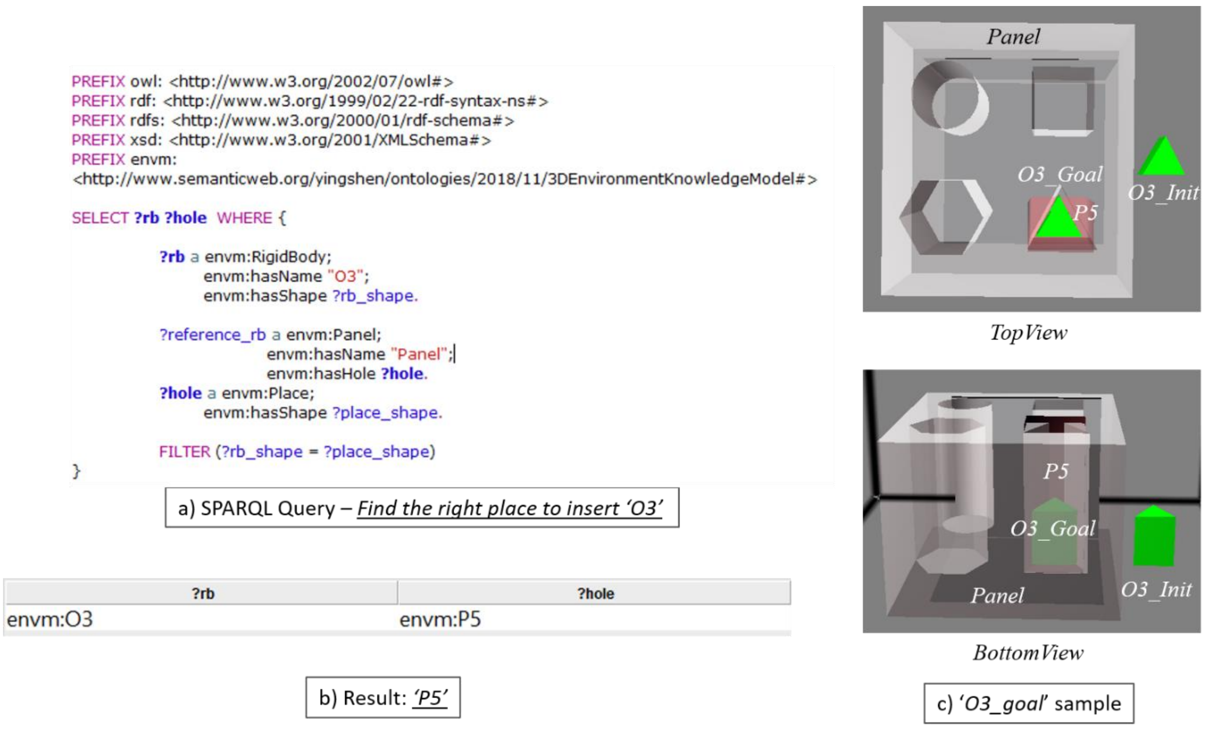
The performed action of this use case is to insert the rigid bodies (O2 to O5) into the right holes (i.e., P2 to P5). Here, we take the O3 (a rigid body with triangular prism shape) for example. The performed action of this use case (Table 29) is to insert O3 into P5 (a hole with triangular prism shape), along with a spatial constraint that O3 should be shape aligned with P5.

Table 29 The primitive action specification

Primitive action (PA) to perform	Insert
The manipulated object	O3
The reference geometry	Panel
PA constraints	O3 <u>*ShapeAligned*</u> P5

The corresponding path planning query is built from the specification of this primitive action.

Firstly, the goal configuration of the path planning query is obtained by random sampling in the goal place. The sampling process must obey the geometric constraints associated with the goal place. In Figure 63-a, we show an example of SPARQL query of finding the right place to insert O3. Figure 63-b shows the result we obtained, i.e., P5 is the goal place. Figure 63-c shows the goal configuration of O3 (i.e., *O3_Goal*) is obtained using a random sampling within P5.



```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX envm:
<http://www.semanticweb.org/yingshen/ontologies/2018/11/3DEnvironmentKnowledgeModel#>

SELECT ?rb ?hole  WHERE {

        ?rb a envm:RigidBody;
              envm:hasName "O3";
              envm:hasShape ?rb_shape.

        ?reference_rb a envm:Panel;
                      envm:hasName "Panel";
                      envm:hasHole ?hole.
        ?hole a envm:Place;
              envm:hasShape ?place_shape.

        FILTER (?rb_shape = ?place_shape)
}
```

a) SPARQL Query – <u>*Find the right place to insert 'O3'*</u>

?rb	?hole
envm:O3	envm:P5

b) Result: <u>*'P5'*</u>

c) '*O3_goal*' sample

Figure 63 The search of goal to reach: 'O3_Goal'

Secondly, the spatial constraint is mapped into the corresponding geometric constraint, through an inference process using SWRL Rule 1.

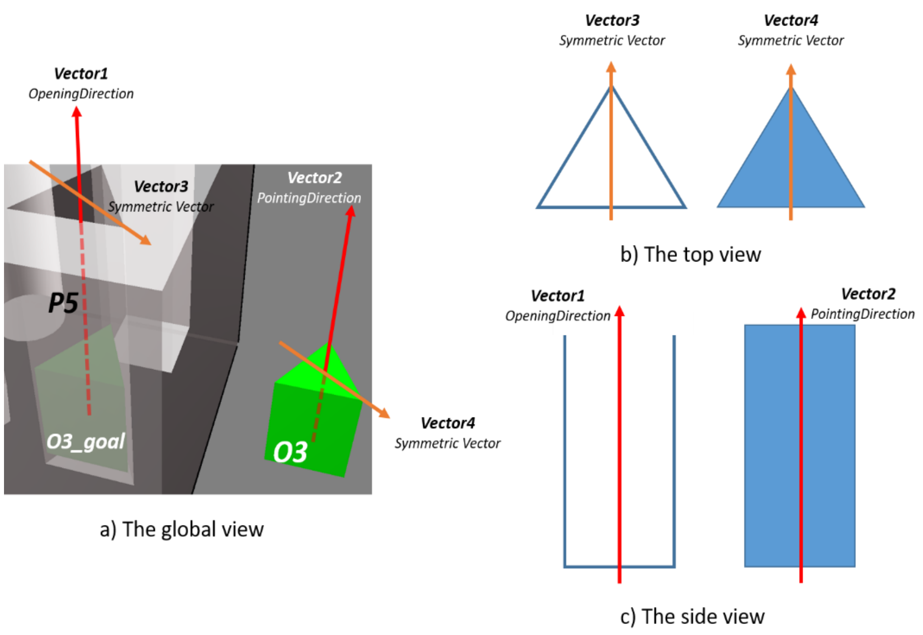


Figure 64 The shape alignment for O3 and P5

SWRL Rule 1: O3 ShapeAligned P5

A SWRL Rule for spatial-geometric constraint mapping

```
IF {
 a instance_of Place and a has3DShape RSQTriangular_Prism
 b instance_of RigidBody and b has3DShape RSQTriangular_Prism
 vector_1, vector_2, vector_3, vector_4 instance_of Vector and
 a hasPointingDirection vector_1 and a hasSymmetricVector vector_3 and
 b hasOpeningDirection  vector_2 and b hasSymmetricVector vector_4 and
 sconstraint instance_of SpatialConstraint and
    sconstraint  hasRelation "ShapeAligned" and
    sconstraint  hasRefGeometry "b" and
    sconstraint  hasTargetGeometry "a" and
gconstraint1, gconstraint2 instance_of GeometricConstraint and
   gconstraint1 hasRelatedSC "sconstraint"
   gconstraint2 hasRelatedSC "sconstraint"
}
THEN {
 gconstraint1 hasRefGeoElement "vector_2" and
 gconstraint1 hasTargetGeoElement "vector_1" and
 gconstraint1 hasRelation "Against"
 gconstraint2 hasRefGeoElement "vector_3" and
 gconstraint2 hasTargetGeoElement "vector_4" and
 gconstraint2 hasRelation "SameDirection"
}
```

Table 30 The corresponding path planning query

Path planning(PP) query	
The manipulated object	O3
Goal configuration	O3_Goal (Figure 64-a)
PP constraints	Vector1 *Against* Vector2 Vector3 *SameDirection* Vector4 (see Figure 64-b,c)

Finally, the description of the generated path planning query is shown in Table 30. This use case aims to take advantages of "Vector1 *Against* Vector2" and "Vector3 *SameDirection* Vector4" to control the trajectory definition when O3 is inserted into P5. Appendix 2 describes how to use this geometric constraint to control the possible orientation of O3.

5.6.3 Validation of non-interactive path planning for a primitive action

The objective of the shaped game scenario is to insert 4 different objects (O2-O5) into O1 (type: *Panel*). In another word, these 4 different objects should be inserted into the corresponding holes on the panel having the same shape, i.e., O2 into P2 (*RSQ_Cylinder*), O3 into P5 (*RSQ_TriangularPrism*), O4 into P3 (*RSQ_Cuboid*), O5 into P4 (*RSQ_PentagonalPrism*). Section 5.4.2 has taken O3 and P5 as an example. Two geometric constraints are generated. They are associated with a path planning query for the primitive action "*Insert (O3, P5)*". A corresponding trajectory is then generated to demonstrate the feasibility of motions of performing such primitive action. This section compares different path planning strategies (G, GT, GTS, GO, GTO), and it demonstrates how effective the task level information can contribute to path planning of a primitive action in Figure 65 (number of random samples), Figure 66 (total computational time). The computed trajectories for O3 are shown in Figure 67.

- In all cases (i.e., O2 into P2, O3 into P5, O4 into P3, O5 into P4), the G, GT and GTS strategies cannot find collision-free trajectories (reaching the maximum number of allowed random samples) and take a huge amount of computational time (more than 5 hours). The reason for this phenomenon is that RRT with uniform random sampling encounters the difficulty of finding collision-free samples in narrow passages.
- Comparing to the G, GT and GTS strategies, the GO strategy is able to find collision-free trajectory with much less number of random samples needed. The reason of this phenomenon is that the geometric constraints (associated to the path planning query of a primitive action) restrict the allowed pose of random samples, so that the search space of random samples is then reduced. Because the margin between the 3d volumes of O5

and P4 is larger than other cases (O2 and P2, O3 and P5, O4 and P3), the path planning of inserting O5 into P4 uses less computational time than others.

- However, the GO strategy still take more than 50 minutes of the computational time to find collision-free trajectories (i.e., O2, O3, O4). In the meanwhile, the GTO strategy takes less than 19 seconds to find collision-free trajectories. Such results are because the GTO strategy is able to find the key collision-free geometric configurations at the borders between P1 and P2 to P5. These geometric configurations at the borders provide guidance leading the manipulated object into the holes.

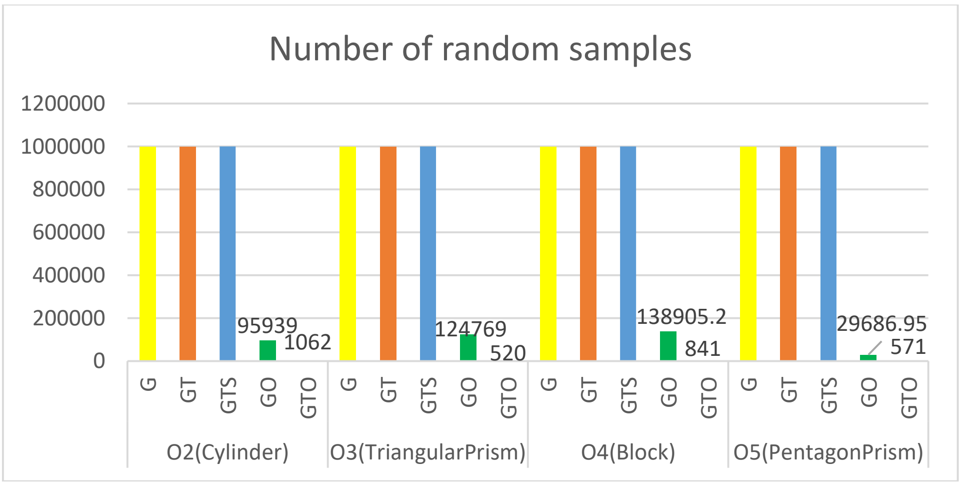


Figure 65 Path planning results (G, GT, GTS, GO, GTO): Number of random sample used

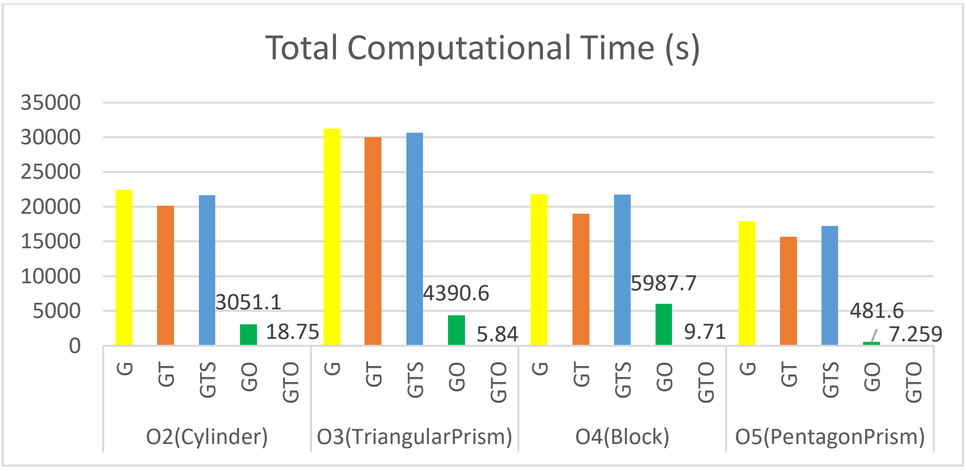


Figure 66 Path planning results (G, GT, GTS, GO, GTO): Total Computational Time

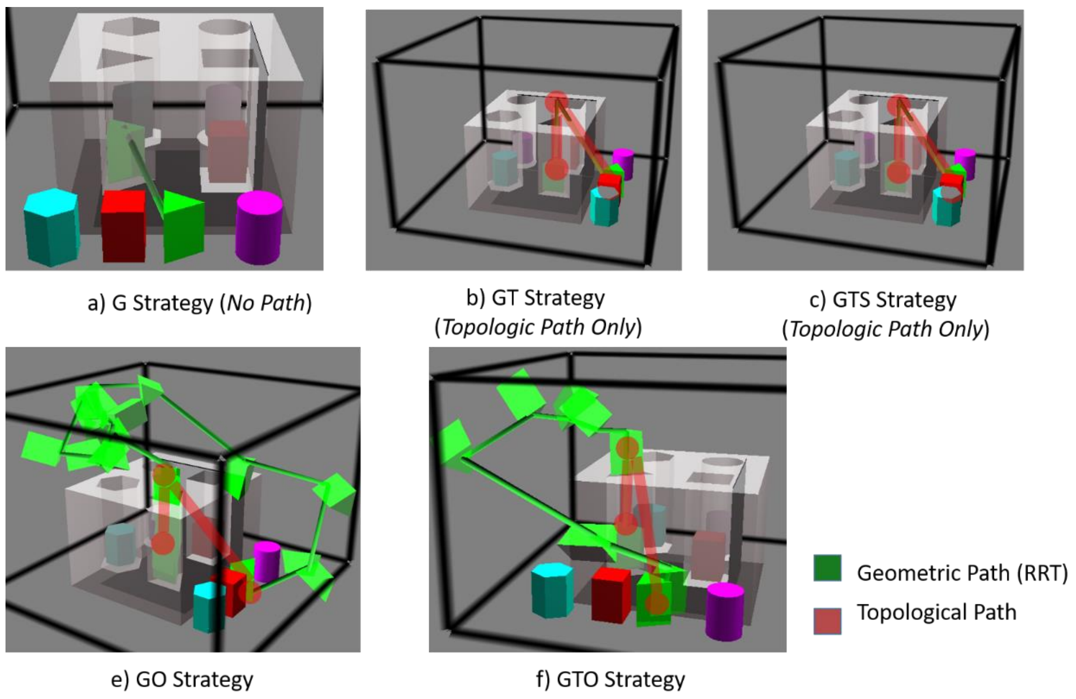

a) G Strategy (*No Path*)
b) GT Strategy (*Topologic Path Only*)
c) GTS Strategy (*Topologic Path Only*)
e) GO Strategy
f) GTO Strategy
🟩 Geometric Path (RRT)
🟥 Topological Path

Figure 67 The computed path using 'G, GT, GTO, GO, GTS' Strategies

The influence of the border size using the GTO strategy

We have noticed that the key geometric configurations found at the borders between the P1 and (P2 to P5) have a great inference on the computational time of finding collision-free trajectories. The location of these key geometric configurations at the border (a bit inside or outside of holes, i.e., P2 to P5) also influences on the computational time.

In the multi-level environment model, a border (B) is overlapped by two places (B_{place1} and B_{place2}). In the case of B_{place1} (free) and B_{place2} (narrow), we propose to extend the border (B) with a certain size towards B_{place1} that is not narrow. The extension direction is determined by a vector pointing from the center of B_{place2}'s oriented bounding box to the center of B's oriented bounding box. The extension size is a coefficient regarding the diagonal length of the bounding box of the manipulated object (noted as BEC, border extension coefficient).

In the shape embedding game scenario, the borders between P1 and (P2 to P5) are extended towards P1, because P1 is free and P2 to P5 are narrow. Taking O3 and P5 for example, Figure 68 shows the extension of the border (B_{o3p5}) with the coefficient 1/5, 1/10, 1/20.

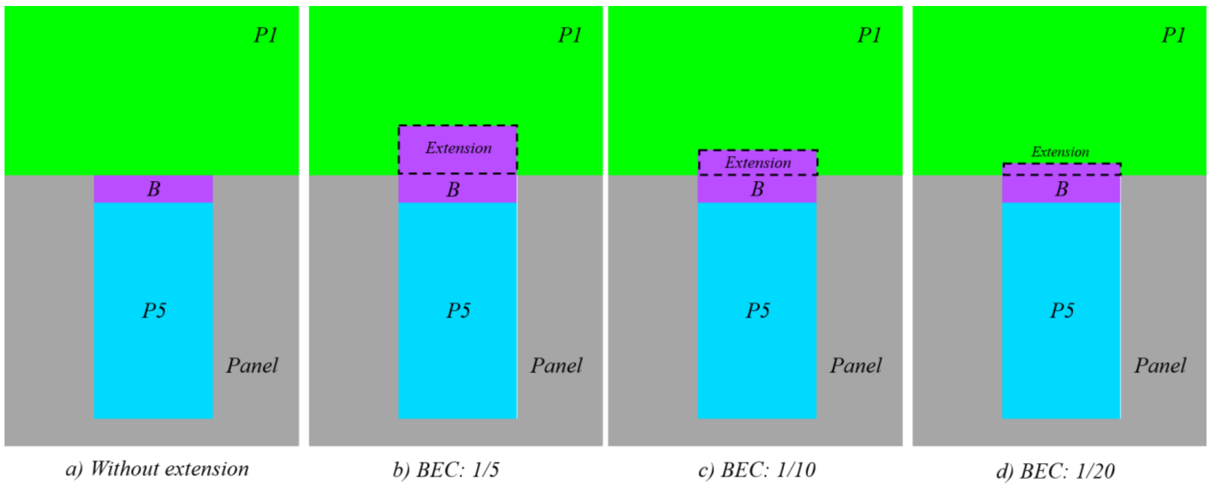


Figure 68 The extension of the border with different coefficient

In all cases (i.e., O2 into P2, O3 into P5, O4 into O3, O5 into O4), the results in Figure 69 show that the collision-free trajectories are easier to be found using the GTO strategy with the larger extension of the borders' size. It is because that the key geometric configuration found at a border (e.g., B_{o3p5}) with the small extension size is more inside of a hole than the one found at a border (e.g., B_{o3p5}) with the large extension size. If the key geometric configuration is more inside of the hole, it becomes more difficult to access it from the outside place (i.e., P1).

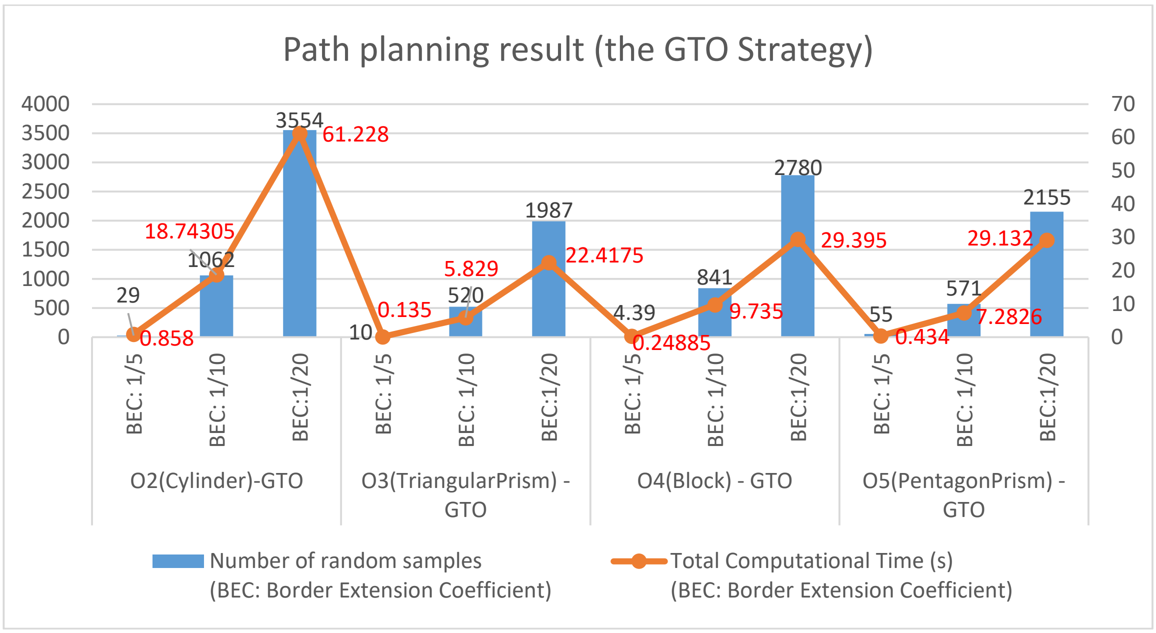


Figure 69 Path planning results (the GTO Strategy with different BEC)

Therefore, the location of the key geometric configuration at a border (also in a narrow place) is affected by the extension size of a border. More concretely speaking, the location of the key geometric configuration (a bit inside or outside of a narrow place) can affect how difficult it is to access this geometric configuration from the outer place (free).

5.6.4 Synthesis on the experimental results of the shape embedding game scenario

The objective of the shape embedding game scenario is to insert the objects to be manipulated (O2 to O5) into the holes with the right shape (P2 to P5). Comparing to scenario 1, scenario 2 is more complex because the manipulated object has to be shape aligned with the targeted hole when performing an "insert" action.

Firstly, the ontology of 3D environment is instantiated with the environment data of the shape embedding game scenario. The O3 and P5 have been taken as an example to visualize a part of the instantiated ontology model. Different competency questions developed on O3 and P5 validate and verify the correctness of the ontology of 3D environment.

Secondly, a path planning query for the primitive action in a task plan, e.g., insert (O3, P5), is firstly generated. Two geometric constraints expressing the shape alignment, e.g., between O3 and P5, is associated with the path planning query.

Thirdly, different path planning strategies are compared and their results are analyzed.

- The G, GT, GTS strategies cannot find collision-free trajectories within a given threshold (i.e., the maximum number of random samples). Because the scenario 2 is more geometrically constrained than Scenario 1, these strategies suffer more difficulties in narrow passages using RRT with uniform random sampling.
- The GO and the GTO strategies show a significant improvement comparing to the G, GT, GTS strategies. The random sampling process of RRT is controlled/restricted by the task-related geometric constraints, which guides the trajectory definition to the goal configuration.
- The GTO strategy improves further the GO strategy. The key geometric configuration found at a border (e.g., B_{o3p5}) guides the search of a collision-free trajectory into a narrow passage (e.g., P5).
- Different locations of the key geometric configuration at a border (e.g., B_{o3p5}) affect the efficiency of finding a collision-free path. The result implies that the key geometric configuration that is more outside of a narrow passage (e.g., P5) can improve the path planning performance.

From the experimental results, we can see that the task-related information (e.g., the task-related geometric constraints) has a significant effect to improve the path planning performance (i.e., less random samples and lower computational time) in a highly geometrically constrained environment. It opens the possibility to better control on the path planning algorithm to check the motion feasibility of the primitive actions of a task plan. It may possibly further lead to better performance of the joint usage of task and path planning to simulate complex tasks.

5.7 SUMMARY

The experiments performed in this chapter allows to validate different propositions in this dissertation. Two different use cases with the increasingly geometrically constrained environment are presented.

First of all, the experiments validate and verify the ontology of 3D environment where the simulated tasks take place. Competency questions are designed and are defined to check whether the proposed ontology can reply with the right answers. This ontology of 3D environment could give fast access to any geometric details of a given rigid body or a given place. It could also enhance the reasoning at the task level, e.g., to check whether a manipulated object can be inserted into a target place.

Second, we exploit the proposed ontology to automatically generate a path planning query associated with a target primitive action of a task plan. Through a reasoning process involving the primitive actions instantiated in the ontology, we are able to infer the start and the goal configurations, as well as task-related geometric constraints.

Finally, the experiments compare the results of different path planning strategies. The results obtained demonstrate that using task-related information allows better control on the RRT path planning algorithm involved to check the motion feasibility for the primitive actions of a task plan, leading to lower computational time and more relevant trajectories for primitive actions.

Chapter 6 General conclusions and perspectives

6.1 SYNTHESIS OF CONTRIBUTIONS

This thesis work proposed an ontology-based approach of using task-related information to generate a path planning query for a primitive action of a task plan. In order to realize this objective, we proposed:

An ontology of the 3D environment where a simulated manipulation task takes place

Inspired from the multi-level environment model proposed in [Cailhol et al. 2019], the originality of the proposed ontology lies in the fact that it conceptualize the heterogeneous environment information (i.e., semantics, topology, and geometry) for both the rigid bodies and the free space models.

- At the geometric level, rigid bodies are described either using the volume representation (BREP) or the surface representation (CSG) of CAD models; the free space model is described using cell decomposition, the unbalanced octree is chosen as the preference.
- At the topologic level, places identify different locations of an environment and borders are the overlapped areas between places. The connectivity between places and borders are described in a topological graph.
- At the semantic level, the taxonomy of rigid bodies, places and borders are given according to the application; their properties are also formalized.

The ontology of 3D environment constitutes the basis of our second contribution of generating path planning query for a primitive action of a task plan.

An ontology-based approach to generate path planning query for a primitive action of a task plan

This contribution explores the interest of using action-specific knowledge to generate a path planning query for a given primitive action of a task plan. The start and the goal locations, as well as the task-related geometric constraints, are generated.

Firstly, an ontology of action-specific knowledge related to a primitive action is proposed. Such kind of knowledge consists of:

- The specification of a primitive action, and its associated so-called "spatial constraints": The spatial constraint restricts the relative position and orientation (i.e., spatial relation)

between two rigid bodies or between a rigid body and a place, in terms of human's semantic spatial understanding of an environment.
- The description of a path planning query, and its associated geometric constraints: The geometric constraint restricts the relative position and orientation (i.e., spatial relation) between two geometric elements, in terms of the geometric path computation.

Secondly, an ontology-based method of generating the path planning query from a given primitive action is discussed. A set of SWRL rules are pre-defined. Through a reasoning process involving a primitive action instantiated in the ontology of action-specific knowledge,

- The start and the goal locations can be determined by exploiting the ontology of 3D environment, e.g., to find the right place to reach.
- The spatial constraints are mapped into the geometric constraints by exploiting the ontology of 3D environment to find the involving geometric elements.

6.2 DISCUSSION

The interest of having an ontology of 3D environment rather than traditional environment models is to link different levels of environment information using formal semantics tightly. Such knowledge formalization allows the path planning system answering to semantically meaningful queries, such as "Does *ObjectA* fits *HoleB*?". "What is the central axis of *ObjectA*?". Moreover, to a certain extent, the knowledge reasoning capability using ontology (in terms of DL logics) allows a path planning system to make decisions on its own. For example, answering to a given "*Insert*" primitive action to be performed, the path planning system can decide the most appropriate "*Hole*" to reach by exploring the environment ontology.

Last but not least, we consider each level of environment information as an individual module of the ontology of 3D environment. Some modules are application-independent (i.e., Geometric description module) and others must be adapted to tasks proceed by applications (i.e., Topologic and Semantic description modules). Such a modular architecture allows to easily adapt the proposed ontology in different applications by updating or changing the affected modules. Using the power of semantic web tools, the ontology of 3D environment can be shared, and its modules can be easily reused by other domain ontologies.

The second part of our research work discussed an ontology-based method to generate the path planning query for a primitive action of a task plan. In two increasingly complex scenarios of manipulating objects,

- we have demonstrated the simplicity of defining spatial constraints to a primitive action rather than non-intuitive geometric constraints;
- we have also shown the use of a reasoning process involving SWRL rules to generate geometric constraints from spatial constraints of a primitive action;
- taking the generated path planning query with geometric constraints as input, we have demonstrated the improvement of path planning results by reducing the computational time by 10 times more.

Comparing to hard-coded geometric constraints, such an ontology-based method introduced a more flexible way of defining geometric constraints through an inference process involving SWRL rules. To some extent, by sampling updating SWRL rules and the ontology of 3D environment, this method can be adapted into different applications of manipulation tasks. Moreover, a library of mathematical implementations of geometric constraints is needed so that the generated geometric constraints can be used in the path planning of a primitive action. This library should be as complete as possible so that a path planning system is able to deal with geometric constraints encountered in different applications easily.

Although significant path planning results have shown the benefits of using the task-related information at the path planning level, this work still encounters its limitations.

The ontology of 3D environment is still a preliminary work. In the geometric description module, the geometries of rigid bodies (based on CAD models) has not taken into account all the criterions of the STEP standard. The geometry of the free space model, currently, considered only the octree decomposition of the simulation environment. In the topologic description module, "border as node, place as arc" is not necessarily the schema in all cases. In the semantic description module, the taxonomies of rigid bodies, places and border are very locally defined and the modeling of the relative locations between rigid bodies or between a rigid body and a place in 3D space should also be considered. Last but not least, in general, different levels of abstraction (top-level, domain- and application-specific) should be considered in building an ontology. By aligning with the top-level ontology (e.g., Basic Formal Ontology, Suggested Upper Merged Ontology), the proposed ontology becomes interoperable with other domain ontologies by making the correspondence between similar concepts having different names.

The definition of spatial constraints should make a difference between those related to a single primitive action and those obeyed by all primitive actions of a task plan. Therefore, it is necessary to define spatial constraints in a task description and to assign them individually in the specification of each primitive action of a task plan.

The generation of geometric constraints currently relies on the spatial constraints assigned to a primitive action and the involving SWRL rules. Sometimes, the task-related geometric constraints can be directly obtained from the beliefs of the path planning system, e.g., when putting a cup full of water on the table, the cup should be put strictly upwards.

Task planning has not been taken into consideration. This thesis work assumes that a task plan is already generated. It concerns the path planning for a primitive action of the task plan and studies the benefits of using the task-related information.

6.3 PERSPECTIVES

Regarding the synthesis and the discussion on this thesis work, we have identified several perspectives as future improvements.

6.3.1 Joint use of task planning and path planning

Rather than the hypothesis in our current research that a task plan is already generated, one of our next steps is to develop the task planning level to use task planning and path planning jointly in the following aspects:

Using the task-related information along with the joint task and path planning process: In our research, using the task-related information in only one primitive action is discussed. It would be interesting to apply the task-related information in all primitive actions of a task plan: 1) spatial constraints are automatically distributed to each primitive action of a task plan, 2) and geometric constraints can be automatically inferred according to the task context. In order to achieve this objective, the potential efforts for the next phase of work should consider:

- ***The definition of spatial constraints*** is leveraged in the task description. Two different kinds are concerned: the ones related to a primitive action and the ones obeyed by all possible actions. In the joint task and path planning process, they are distributed individually to each primitive action in the task plan.
- ***The definition of geometric constraints*** associated to the path planning query of a primitive action is not only generated from the spatial constraints of a primitive action but also inferred from the knowledge of the path planning system (e.g., commonsense knowledge).

Developing an iterative task and path planning process: The joint task and path planning process should consider the loopback between the task planning level and the path planning level:

- Using the task level information at the path planning level, e.g., to control the trajectory definition;
- Using the path planning result at the task planning level, e.g., to guide the search of an optimal task plan.

Current research work remains in using the task-level information at the path planning level (i.e., top-down, or task level to motion level). Therefore, it is interesting to study the benefits of using path planning results in the task planning process. Furthermore, considering both levels of information, it opens the possibility to develop an iterative task and path planning process.

6.3.2 Ontologies integration between the task planning level and the path planning level

Current research work has demonstrated the benefits of using ontologies (conceptualizing the task-related information) at the path planning level. In the joint use of task and path planning, we indeed have also to study the benefits of using ontologies at the task planning level. In order to achieve this objective in the next phase of our research work, we have to carefully consider the integration of ontologies used at the task planning level and of ontologies used at the path planning level. For example, how should primitive actions of a task plan be mapped to a topological path (made of topological steps) computed at the path planning level? Should a primitive action map to a single topological step, or should it map to a complete topological path?

6.3.3 An ontology of CAD models

When we built the ontology of 3D environment where a simulated manipulation task takes place, we realized that the existing ontologies of modeling environment information in robotic applications rarely consider the formalization of the geometric models of objects using formal semantics. An important reason is that the commonly used polygonal models based on Delaunay triangulation contain a large amount of raw data that is semantically meaningless.

On the contrary, the geometry of CAD models is semantically meaningful. For example, rather than thousands of meaningless triangular faces, the “Cylindrical” surface can be defined as a surface having an origin point, a central axis and a radius. Similar semantically meaningful geometric information can be found everywhere in CAD models. However, the existing format (e.g., STEP) of CAD models does not take advantages of the semantically meaning data to allow knowledge querying and reasoning. Therefore, conceptualizing CAD models using ontology is worth of research.

REFERENCES

[Abbott et al. 2007] Abbott, Jake J., Panadda Marayong and Allison M. Okamura 2007. "Haptic virtual fixtures for robot-assisted manipulation". Robotics research, *Springer*: 49-64.

[Akbari et al. 2019] Akbari, Aliakbar, Muhayyuddin and Jan Rosell 2019. "Knowledge-oriented task and motion planning for multiple mobile robots." *Journal of Experimental Theoretical Artificial Intelligence* **31**(1): 137-162.

[Akbari et al. 2015] Akbari, Aliakbar and Jan Rosell 2015. "Ontological physics-based motion planning for manipulation". *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, IEEE.

[Alt et al. 1988] Alt, Helmut and Emo Welzl 1988. "Visibility graphs and obstacle-avoiding shortest paths." *Zeitschrift für Operations-Research* **32**(3-4): 145-164.

[Arp et al. 2015] Arp, Robert, Barry Smith and Andrew D. Spear 2015. "Building ontologies with basic formal ontology", *Mit Press*. Book.

[Aztiria et al. 2008] Aztiria, Asier, Juan Carlos Augusto and Alberto Izaguirre 2008. "Spatial and temporal aspects for pattern representation and discovery in intelligent environments". *Workshop on spatial and temporal reasoning at 18th European conference on artificial intelligence (ECAI 2008)(to be published)*.

[Baader 2003] Baader, Franz 2003. "The description logic handbook: Theory, implementation and applications", *Cambridge university press*. Book.

[Barbau et al. 2012] Barbau, Raphael, Sylvere Krima, Sudarsan Rachuri, Anantha Narayanan, Xenia Fiorentini, Sebti Foufou and Ram D. Sriram 2012. "OntoSTEP: Enriching product model data using ontologies." *Computer-Aided Design* **44**(6): 575-590.

[Bastianelli et al. 2013] Bastianelli, Emanuele, Domenico Bloisi, Roberto Capobianco, Guglielmo Gemignani, Luca Iocchi and Daniele Nardi 2013. "Knowledge representation for robots through human-robot interaction". *International Conference On Logic Programmming*.

[Belouaer et al. 2011] Belouaer, Lamia, Maroua Bouzid and Abdel-Illah Mouaddib 2011. "Spatial knowledge in planning language". *International Conference on Knowledge Engineering and Ontology Development*.

[Benslimane et al. 2006] Benslimane, Djamal, Ahmed Arara, Gilles Falquet, Zakaria Maamar, Philippe Thiran and Faiez Gargouri 2006. "Contextual ontologies". *International Conference on Advances in Information Systems*, Springer.

[Bidot et al. 2017] Bidot, Julien, Lars Karlsson, Fabien Lagriffoul and Alessandro Saffiotti 2017. "Geometric backtracking for combined task and path planning in robotic systems." *Artificial Intelligence* **247**: 229-265.

[Blodow et al. 2011] Blodow, Nico, Lucian Cosmin Goron, Zoltan-Csaba Marton, Dejan Pangercic, Thomas Rühr, Moritz Tenorth and Michael Beetz 2011. "Autonomous semantic mapping for robots performing everyday manipulation tasks in kitchen environments". *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, IEEE.

[Böhm et al. 1984] Böhm, Wolfgang, Gerald Farin and Jürgen Kahmann 1984. "A survey of curve and surface methods in CAGD." *Computer Aided Geometric Design* **1**(1): 1-60.

[Bonet et al. 1999] Bonet, Blai and Hector Geffner 1999. "Planning as heuristic search: New results". *European Conference on Planning*.

[Borrmann et al. 2009] Borrmann, A, J Hyvärinen and E Rank 2009. "Spatial constraints in collaborative design processes". *Proceedings of the International Conference on Intelligent Computing in Engineering (ICE'09). Berlin, Germany, Berlin, Germany*.

[Botsch et al. 2007] Botsch, Mario, Mark Pauly, Leif Kobbelt, Pierre Alliez, Bruno Lévy, Stephan Bischoff and Christian Röossl 2007. "Geometric modeling based on polygonal meshes."

[Bouquet et al. 2003] Bouquet, Paolo, Fausto Giunchiglia, Frank Van Harmelen, Luciano Serafini and Heiner Stuckenschmidt 2003. "C-owl: Contextualizing ontologies". *International Semantic Web Conference*, Springer.

[Brooks et al. 1982] Brooks, Rodney A. and Tomas Lozano-Perez. "A Subdivision Algorithm in Configuration Space for Findpath with Rotation". *DTIC Document*. 1982

[Cailhol 2015] Cailhol, Simon 2015. "Planification interactive de trajectoire en Realite Virtuelle sur la base de donnees geometriques, topologiques et semantiques."

[Cailhol et al. 2015] Cailhol, Simon, Philippe Fillatreau, Jean-Yves Fourquet and Yingshen Zhao 2015. "A hierarchic approach for path planning in virtual reality." *International Journal on Interactive Design and Manufacturing (IJIDeM)* **9**(4): 291-302.

[Cailhol et al. 2019] Cailhol, Simon, Philippe Fillatreau, Yingshen Zhao and Jean-Yves Fourquet 2019. "Multi-layer path planning control for the simulation of manipulation tasks: Involving semantics and topology." *Robotics Computer-Integrated Manufacturing* **57**: 17-28.

[Caldiran et al. 2009] Caldiran, Ozan, Kadir Haspalamutgil, Abdullah Ok, Can Palaz, Esra Erdem and Volkan Patoglu 2009. "Bridging the gap between high-level reasoning and low-level control". Logic Programming and Nonmonotonic Reasoning, *Springer***:** 342-354.

[Cambon et al. 2009] Cambon, Stephane, Rachid Alami and Fabien Gravot 2009. "A hybrid approach to intricate motion, manipulation and task planning." *The International Journal of Robotics Research* **28**(1): 104-126.

[Cantamessa et al. 2012] Cantamessa, Marco, Francesca Montagna and Paolo Neirotti 2012. "An empirical analysis of the PLM implementation effects in the aerospace industry." *Computers in Industry* **63**(3): 243-251.

[Challam et al. 2007] Challam, Vishnu, Susan Gauch and Aravind Chandramouli 2007. "Contextual search using ontology-based user profiles". *Large Scale Semantic Access to Content (Text, Image, Video, and Sound)*, LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

[Chitkara 1998] Chitkara, KK 1998. "Construction project management", *Tata McGraw-Hill Education*. Book.

[Choset et al. 2005] Choset, Howie M, Seth Hutchinson, Kevin M Lynch, George Kantor, Wolfram Burgard, Lydia E Kavraki and Sebastian Thrun 2005. "Principles of robot motion: theory, algorithms, and implementation", *MIT press*. Book.

[Clark et al. 1995] Clark, Alan L and Scott M Staley 1995. "STEP AP203 data exchange study". *Proceedings of the third ACM symposium on Solid modeling and applications*, ACM.

[Consortium 2004] Consortium, World Wide Web 2004. "SWRL." from https://www.w3.org/Submission/SWRL/.

[Cox 1986] Cox, Brad J 1986. "Object-oriented programming: an evolutionary approach", *CUMINCAD*. Book.

[Dandois et al. 2013] Dandois, Jonathan P and Erle Ellis 2013. "High spatial resolution three-dimensional mapping of vegetation spectral dynamics using computer vision." *Remote Sensing of Environment* **136**: 259-276.

[Dang-Vu et al. 2016] Dang-Vu, Bao-Anh, Oliver Porges and Maximo A. Roa 2016. "Interpreting manipulation actions: From language to execution". *Robot 2015: Second Iberian Robotics Conference*.

[de Silva et al. 2013] de Silva, Lavindra, Amit Kumar Pandey, Mamoun Gharbi and Rachid Alami 2013. "Towards combining HTN planning and geometric task planning". *RSS Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*.

[Dearden et al. 2013] Dearden, Richard and Chris Burbridge 2013. "An Approach for Efficient Planning of Robotic Manipulation Tasks". *The International Conference on Automated Planning and Scheduling (ICAPS)*.

[Dechter et al. 2003] Dechter, Rina and David Cohen 2003. "Constraint processing", *Morgan Kaufmann*. Book.

[Dedeoglu et al. 1999] Dedeoglu, Goksel, Maja J Mataric and Gaurav S Sukhatme 1999. "Incremental online topological map building with a mobile robot". *Mobile Robots XIV*, International Society for Optics and Photonics.

[Demyen et al. 2006] Demyen, Douglas and Michael Buro 2006. "Efficient triangulation-based pathfinding". *Aaai*.

[Diab et al. 2017] Diab, Mohammed, Aliakbar Akbari and Jan Rosell 2017. "An ontology framework for physics-based manipulation planning". *Iberian Robotics conference*, Springer.

[Diab et al. 2019] Diab, Mohammed, Aliakbar Akbari, Muhayy Ud Din and Jan Rosell 2019. "PMK—A Knowledge Processing Framework for Autonomous Robotics Perception and Manipulation." *Sensors* **19**(5): 1166.

[Dornhege et al. 2009] Dornhege, Christian, Marc Gissler, Matthias Teschner and Bernhard Nebel 2009. "Integrating symbolic and geometric planning for mobile manipulation". *2009 IEEE International Workshop on Safety, Security \& Rescue Robotics (SSRR 2009)*.

[Erdem et al. 2011] Erdem, Esra, Kadir Haspalamutgil, Can Palaz, Volkan Patoglu and Tansel Uras 2011. "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation". *2011 IEEE International Conference on Robotics and Automation (ICRA)*.

[Eyerich et al. 2010] Eyerich, Patrick, Thomas Keller and Bernhard Nebel 2010. "Combining action and motion planning via semantic attachments". *Proc. of Workshop on Combining Action and Motion Planning at ICAPS*.

[FCL 2014] FCL 2014. "FCL." Flexible Collision Library. from http://gamma.cs.unc.edu/FCL/fcl_docs/webpage/generated/index.html.

[Firat 2003] Firat, Aykut. "Information integration using contextual knowledge and ontology merging". *Ph.D. Thesis*. 2003

[Forsberg et al. 1991] Forsberg, Kevin and Harold Mooz 1991. "The relationship of system engineering to the project cycle". *INCOSE International Symposium*, Wiley Online Library.

[Fox et al. 2003] Fox, Maria and Derek Long 2003. "PDDL2. 1: An extension to PDDL for expressing temporal planning domains." *Journal of Artificial Intelligence Research* **20**: 61-124.

[Galindo et al. 2008] Galindo, Cipriano, Juan-Antonio Fernández-Madrigal, lez Gonzá, Javier and Alessandro Saffiotti 2008. "Robot task planning using semantic maps." *Robotics and Autonomous Systems* **56**(11): 955-966.

[Garrett et al. 2014] Garrett, Caelan Reed, Lozano-Pérez Tomás and Leslie Pack Kaelbling 2014. "Heuristic search for task and motion planning". *Proceedings of the Workshop on Planning and Robotics (PlanRob)*.

[Giunchiglia et al. 2004]Giunchiglia, Enrico, Joohyung Lee, Vladimir Lifschitz, Norman McCain and Hudson Turner 2004. "Nonmonotonic causal theories." *Artificial Intelligence* **153**(1-2): 49-104.

[Giunchiglia et al. 1999]Giunchiglia, Fausto and Paolo Traverso 1999. "Planning as model checking". *European Conference on Planning*.

[Gruber 1993] Gruber, Thomas R. 1993. "A translation approach to portable ontology specifications." *Knowledge acquisition* **5**(2): 199-220.

[Grüninger et al. 1995] Grüninger, Michael and Mark S Fox 1995. "The role of competency questions in enterprise engineering". Benchmarking—Theory and practice, *Springer*: 22-31.

[Guitton et al. 2009] Guitton, Julien and Jean-Loup Farges 2009. "Taking into account geometric constraints for task-oriented motion planning." *Proc. Bridging the gap Between Task And Motion Planning, BTAMP* **9**: 26-33.

[Gursel et al. 2009] Gursel, IPEK, SEVIL Sariyildiz, RUDI STOuFFS, ÖMER Akin, T Tidafi and T Dorta 2009. "Contextual ontology support as external knowledge representation for building information modeling." *Joining Languages, Cultures Visions: CAADFutures* **2009**: 487-500.

[Hernandez 1994] Hernandez, Daniel 1994. "Qualitative representation of spatial knowledge", *Springer Science & Business Media*. Book.

[Hirschberg 2000] Hirschberg, Morton J Crosstalk. "The V model". *Army Research Laboratory*. 2000

[Hirtle et al. 1985] Hirtle, Stephen C, John J Memory Jonides and cognition 1985. "Evidence of hierarchies in cognitive maps." **13**(3): 208-217.

[Hoffmann 1989] Hoffmann, Christoph M 1989. "Geometric and solid modeling", *Morgan Kaufmann Pub*. Book.

[Hoffmann et al. 2001] Hoffmann, Jörg and Bernhard Nebel 2001. "The FF planning system: Fast plan generation through heuristic search." *Journal of Artificial Intelligence Research* **14**: 253-302.

[Hu et al. 2004] Hu, Haibo and Dik-Lun Lee 2004. "Semantic location modeling for location navigation in mobile environment". *2004 IEEE International Conference on Mobile Data Management, Proceedings*, IEEE.

[Javvadi May 2011] Javvadi, Lakshminadh. "Introduction to Product Lifecycle Management". *MPHASIS (An HP Company)*. May 2011

[Kaelbling et al. 2011] Kaelbling, Leslie Pack and Lozano-Pérez Tomás 2011. "Hierarchical task and motion planning in the now". *2011 IEEE International Conference on Robotics and Automation (ICRA)*.

[Kaiser et al. 2018] Kaiser, Adrien, Jose Alonso Ybanez Zepeda and Tamy Boubekeur 2018. "A survey of simple geometric primitives detection methods for captured 3D data". *Computer Graphics Forum*, Wiley Online Library.

[Karniel et al. 2011] Karniel, Arie and Yoram Reich 2011. "Managing the dynamics of new product development processes: a new product lifecycle management paradigm", *Springer Science & Business Media*. Book.

[Karray et al. 2019] Karray, Mohamed Hedi, Farhad Ameri, Melinda Hodkiewicz and Thierry Louge 2019. "ROMAIN: Towards a BFO compliant reference ontology for industrial maintenance." *Applied Ontology*(Preprint): 1-24.

[Kavraki et al. 1996] Kavraki, Lydia E., Petr Svestka, Jean-Claude Latombe and Mark H. Overmars 1996. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." *Robotics and Automation, IEEE Transactions on* **12**(4): 566-580.

[Kemke et al. 2006] Kemke, Christel and Erin Walker 2006. "Planning with action abstraction and plan decomposition hierarchies". *IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2006. IAT'06.* .

[Khatib 1986] Khatib, Oussama 1986. "Real-time obstacle avoidance for manipulators and mobile robots." *The International Journal of Robotics Research* **5**(1): 90-98.

[Kortenkamp et al. 1994] Kortenkamp, David and Terry Weymouth 1994. "Topological mapping for mobile robots using a combination of sonar and vision sensing". *AAAI*.

[Kuipers et al. 2004] Kuipers, Benjamin, Joseph Modayil, Patrick Beeson, Matt MacMahon and Francesco Savelli 2004. "Local metrical and global topological maps in the hybrid spatial semantic hierarchy". *2004 IEEE International Conference on Robotics and Automation, Proceedings. ICRA'04.* .

[Ladeveze 2010] Ladeveze, Nicolas 2010. "Apport des methodes de planification automatique dans les simulations interactives d'industrialisation et de maintenance en realite virtuelle."

[Ladeveze et al. 2009] Ladeveze, Nicolas, Jean Yves Fourquet, Bernard Puel and Michel Taix 2009. "Haptic assembly and disassembly task assistance using interactive path planning". *Virtual Reality Conference, 2009. VR 2009. IEEE*.

[Lagriffoul et al. 2016] Lagriffoul, Fabien and Benjamin Andres 2016. "Combining task and motion planning: A culprit detection problem." *The International Journal of Robotics Research* **35**(8): 890-927.

[Lagriffoul et al. 2014] Lagriffoul, Fabien, Dimitar Dimitrov, Julien Bidot, Alessandro Saffiotti and Lars Karlsson 2014. "Efficiently combining task and motion planning using geometric constraints." *The International Journal of Robotics Research* **33**(14): 1726-1747.

[LaValle 1998] LaValle, Steven M. 1998. "Rapidly-Exploring Random Trees A New Tool for Path Planning."

[LaValle 2006] LaValle, Steven M. 2006. "Planning algorithms", *Cambridge university press*. Book.

[Lozano-Perez 1990] Lozano-Perez, Tomas 1990. "Spatial planning: A configuration space approach". Autonomous robot vehicles, *Springer*: 259-271.

[Lozano-Pérez et al. 2014] Lozano-Pérez, Tomás and Leslie Pack Kaelbling 2014. "A constraint-based method for solving sequential manipulation planning problems". *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE.

[Marton et al. 2008] Marton, Zoltan Csaba, Nico Blodow, Mihai Dolha, Moritz Tenorth, Radu Bogdan Rusu and Michael Beetz 2008. "Autonomous mapping of kitchen environments and applications". *Proc. of the 1st Int. Workshop on Cognition for Technical Systems, Munich, Germany*.

[Matta et al. 2011] Matta, Nada, Guillaume Ducellier, Yannick Charlot, Mohammed Ridha Beldjoudi, François Tribouillois and Edouard Hibon 2011. "Traceability of design project knowledge using PLM". *2011 International Conference on Collaboration Technologies and Systems (CTS)*, IEEE.

[Matta et al. 2013] Matta, Nada, Guillaume Ducellier and Chaker Djaiz 2013. "Traceability and structuring of cooperative Knowledge in design using PLM." *Knowledge Management Research Practice* **11**(1): 53-61.

[McCain et al. 1997] McCain, Norman and Hudson Turner 1997. "Causal theories of action and change". *AAAI/IAAI*.

[McGuinness et al. 2004] McGuinness, Deborah L and Frank Van Harmelen 2004. "OWL web ontology language overview." *W3C recommendation* **10**(10): 2004.

[McNamara 1986] McNamara, Timothy P 1986. "Mental representations of spatial relations." *Cognitive psychology* **18**(1): 87-121.

[Meagher 1982] Meagher, Donald 1982. "Geometric modeling using octree encoding." *Computer graphics image processing* **19**(2): 129-147.

[Minsky 1974] Minsky, Marvin. "A framework for representing knowledge". *Massachusetts Institute of Technology Cambridge, MA, USA*. 1974

[Mozos et al. 2007] Mozos, Oscar Martınez, Patric Jensfelt, Hendrik Zender, Geert-Jan M Kruijff and Wolfram Burgard 2007. "From labels to semantics: An integrated system for conceptual spatial representations of indoor environments for mobile robots". *ICRA Workshop: Semantic Information in Robotics*.

[Nau et al. 2003] Nau, Dana S., Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, J. William Murdock, Dan Wu and Fusun Yaman 2003. "SHOP2: An HTN planning system." *Journal of Artificial Intelligence Research* **20**: 379-404.

[Neubert et al. 2004] Neubert, Gilles, Yacine Ouzrout and Abdelaziz Bouras 2004. "Collaboration and integration through information technologies in supply chains." *International Journal of Technology Management* **28**.

[Nüchter et al. 2008] Nüchter, Andreas and Joachim Hertzberg 2008. "Towards semantic maps for mobile robots." *Robotics Autonomous Systems* **56**(11): 915-926.

[OWLAPI 2018] OWLAPI 2018. "OWLAPI." *Web Ontology Language - Application Program Interface*.

[Perzylo et al. 2015] Perzylo, Alexander, Nikhil Somani, Markus Rickert and Alois Knoll 2015. "An ontology for CAD data and geometric constraints as a link between product models and semantic robot task descriptions". *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE.

[Piegl et al. 1987] Piegl, Leslie and Wayne Tiller 1987. "Curve and surface constructions using rational B-splines." *Computer-Aided Design* **19**(9): 485-498.

[Plögert 1996] Plögert, Klaus 1996. "The tailoring process in the german v-model." *Journal of systems architecture* **42**(8): 601-609.

[Pratt 2001] Pratt, Michael 2001. "Introduction to ISO 10303—the STEP standard for product data exchange." *Journal of Computing Information Science in Engineering* **1**(1): 102-103.

[Pronobis 2011] Pronobis, Andrzej 2011. "Semantic mapping with mobile robots."

[Protégé 2017] Protégé 2017. "Protégé." from https://protegewiki.stanford.edu/wiki/Main_Page.

[Requicha 1980] Requicha, Aristides G. 1980. "Representations for rigid solids: Theory, methods, and systems." *ACM Computing Surveys (CSUR)* **12**(4): 437-464.

[Roth et al. 1993] Roth, Gerhard and Martin Levine 1993. "Extracting geometric primitives." *CVGIP: Image Understanding* **58**(1): 1-22.

[Russell et al. 2003] Russell, Stuart Jonathan, Peter Norvig, John F. Canny, Jitendra M. Malik and Douglas D. Edwards 2003. "Artificial intelligence: a modern approach", *Prentice hall Upper Saddle River*. Book.

[Rusu 2010] Rusu, Radu Bogdan 2010. "Semantic 3D object maps for everyday manipulation in human living environments." *Künstliche Intelligenz* **24**(4): 345-348.

[Samet 1984] Samet, Hanan 1984. "The quadtree and related hierarchical data structures." *ACM Computing Surveys* **16**(2): 187-260.

[Sánchez-Macián et al. 2007] Sánchez-Macián, Alfonso, Encarna Pastor, Jorge E de López Vergara and David López 2007. "Extending SWRL to enhance mathematical support". *International Conference on Web Reasoning and Rule Systems*, Springer.

[Sarcar et al. 2008] Sarcar, MMM, K Mallikarjuna Rao and K Lalit Narayan 2008. "Computer aided design and manufacturing", *PHI Learning Pvt. Ltd.* Book.

[Siméon et al. 2004] Siméon, Thierry, Juan Cortés, Anis Sahbani and Jean-Paul Laumond 2004. "A general manipulation task planner". Algorithmic Foundations of Robotics V, *Springer***:** 311-327.

[Srivastava et al. 2014] Srivastava, Sanjeev, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stephen Russell and Pieter Abbeel 2014. "Combined task and motion planning through an extensible planner-independent interface layer". *2014 IEEE International Conference on Robotics and Automation (ICRA)*.

[Standardization 2014] Standardization, International Organization for. "Industrial automation systems and integration -- Product data representation and exchange -- Part 1652: Application module: Basic geometry". *International Organization for Standardization*. 2014

[Stark 2015] Stark, John 2015. "Product lifecycle management". Product Lifecycle Management (Volume 1), *Springer***:** 1-29.

[Strobl 2017] Strobl, Christian 2017. "Dimensionally extended nine-intersection model (de-9im)." *Encyclopedia of GIS*: 470-476.

[Stuckenschmidt et al. 2004] Stuckenschmidt, Heiner, Frank Van Harmelen, Luciano Serafini, Paolo Bouquet and Fausto Giunchiglia. "Using C-OWL for the alignment and merging of medical ontologies". *University of Trento*. 2004

[Suh et al. 2007] Suh, Il Hong, Gi Hyun Lim, Wonil Hwang, Hyowon Suh, Jung-Hwa Choi and Young-Tack Park 2007. "Ontology-based multi-layered robot knowledge framework (OMRKF) for robot intelligence". *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007.*

[Sureephong et al. 2008] Sureephong, Pradorn, Nopasit Chakpitak, Yacine Ouzrout and Abdelaziz Bouras 2008. "An ontology-based knowledge management system for industry clusters". Global Design to Gain a Competitive Edge, *Springer*: 333-342.

[Sylvester 1851] Sylvester, James Joseph 1851. "Sketch of a memoir on elimination, transformation, and canonical forms." *Cambridge Dublin Mathematical Journal* **6**: 186-200.

[Systemes 2014] Systemes, Dassault 2014. "Catia." from http://www.3ds.com/fr/produits-et-services/catia/fonctionnalites/ingenierie-mecanique/.

[Systemes 2014] Systemes, Dassault 2014. "SolidWorks." from http://www.solidworks.fr/sw/products/3d-cad/cad-animation.htm.

[Systèmes 2006] Systèmes, Dassault 2006. "Virtools." from https://en.wikipedia.org/wiki/Virtools.

[Tching et al. 2010] Tching, Loïc, Georges Dumont and Jérôme Perret 2010. "Interactive simulation of CAD models assemblies using virtual constraint guidance." *International Journal on Interactive Design Manufacturing* **4**(2): 95-102.

[Tenorth et al. 2014] Tenorth, Moritz, Georg Bartels and Michael Beetz 2014. "Knowledge-based Specification of Robot Motions". *ECAI*.

[Tenorth et al. 2009] Tenorth, Moritz and Michael Beetz 2009. "KnowRob: knowledge processing for autonomous personal robots". *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*.

[Tenorth et al. 2010] Tenorth, Moritz, Lars Kunze, Dominik Jain and Michael Beetz 2010. "Knowrob-map-knowledge-linked semantic object maps". *2010 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*.

[Tsetsos et al. 2006] Tsetsos, Vassileios, Christos Anagnostopoulos, Panayotis Kikiras and Stathes Hadjiefthymiades 2006. "Semantically enriched navigation for indoor environments." *International Journal of Web Grid Services* **2**(4): 453-478.

[Tzafestas 2013] Tzafestas, Spyros G. 2013. "Introduction to mobile robot control", *Elsevier*. Book.

[Vásquez et al. 2017] Vásquez, Alex, Arnaud Dapogny, Kévin Bailly and Véronique Perdereau 2017. "Sequential recognition of in-hand object shape using a collection of neural forests". *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE.

[Vassev et al. 2012] Vassev, Emil and Mike Hinchey 2012. "Knowledge representation for cognitive robotic systems". *2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, IEEE.

[Voronoi 1908] Voronoi, Georges 1908. "Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les parallélloèdres primitifs." *Journal für die reine und angewandte Mathematik* **134**: 198-287.

[Wolfe et al. 2010] Wolfe, Jason, Bhaskara Marthi and Stuart J. Russell 2010. "Combined Task and Motion Planning for Mobile Manipulation". *The International Conference on Automated Planning and Scheduling (ICAPS)*.

[Woods 1975] Woods, William A. 1975. "What's in a link: Foundations for semantic networks." *Representation and understanding: Studies in cognitive science*: 35-82.

[Xu 2017] Xu, Da. "Contribution to the elaboration of a decision support system based on modular ontologies for ecological labelling". *Ph.D. Thesis*. 2017

[Zacharias et al. 2011] Zacharias, Franziska and Christoph Borst 2011. "Knowledge representations for high-level and low-level planning". *The International Conference on Automated Planning and Scheduling (ICAPS)*.

[Zender et al. 2008] Zender, Hendrik, O. Martínez Mozos, Patric Jensfelt, G.-J. M. Kruijff and Wolfram Burgard 2008. "Conceptual spatial representations for indoor mobile robots." *Robotics and Autonomous Systems* **56**(6): 493-502.

APPENDIX

In order to control motions of a manipulated object, this appendix respectively describes how geometric constraints have been used in the pen-penbox insertion scenario (section 5.4.1) and the shape embedding game (section 5.4.2). We should note that the geometric constraints used in these two scenarios can be also applied in other scenarios.

Appendix 1: Pen – Penbox Insertion Scenario

A. *The representation of geometric constraint*

As we can see from section 5.5.2, the geometric constraint inferred from a spatial constraint "Pen1 *PointTo* Penbox1" is :

Vector_1 *Against* Vector_2

The visual representation of this geometric constraint is given in Figure 70-a.

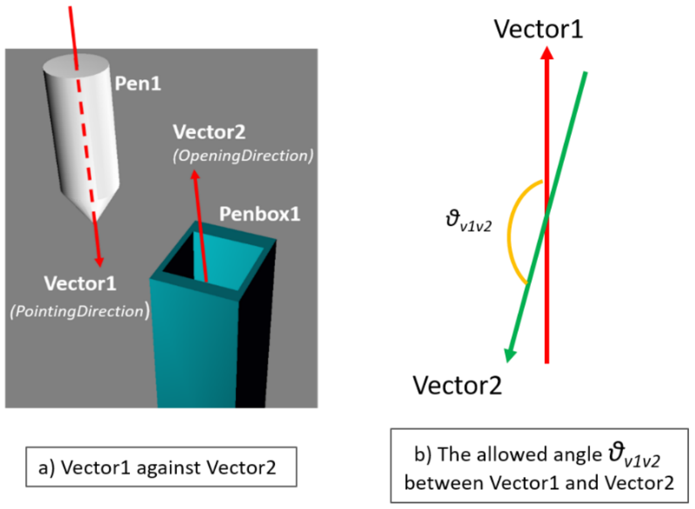


Figure 70 The visual representation of a geometric constraint 'Vector1 Against Vector2'

This geometric constraint defines that the two vectors (i.e., Vector1, Vector2) are pointed to the opposite direction (spatial relation: *Against*). In this thesis work, rather than exclusively opposite, the amplitude between Vector1 and Vector2 is allowed and this amplitude is limited by a threshold value ($\theta_{threshold}$). Therefore, θ_{v1v2} between Vector1 and Vector2 (Figure 70-b) should be inside an interval ($\theta_{threshold}$, 180°).

B. *Current status of the environment*

Before using the geometric constraint to control motions of Pen1, we will firstly introduce some current status of the environment in Figure 71.

- Local_RF1 and Local_RF2 are respectively the local reference frame of Penbox1 and Pen1
- o1 and o2 are the origins of the Local_RF1 and Local_RF2
- Vector1 (x1, y1, z1) and Vector2 (x2, y2 z2) is respectively defined in Local_RF1 and Local_RF2
- θ_{xv1}, $\theta_{yv1,}$ and θ_{zv1} is the angle between the x, y, z axes of Local_RF1 and Vector1
- $\theta_{x'v2}$, $\theta_{y'v2}$ and $\theta_{z'v2}$ is the angle between the x', y', z' axes of Local_RF2 and Vector2
- The global reference frame is the Global_RF located at the origin o_W and x_W, y_W, z_W axes.

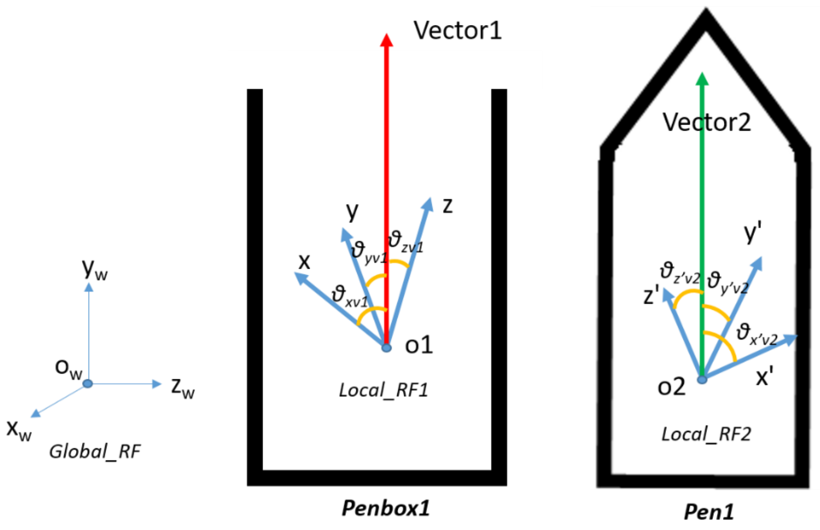


Figure 71 The graphical representation of the pattern 1

C. The usage of geometric constraint

Pre-condition:

- The Penbox1 is fixed to the ground. Thus, the rotation matrix from Local_RF1 to Global_RF ($R_{lrf1\_to\_grf}$) does not change
- Vector1 (x1, y1, z1) and Vector2 (x2, y2 z2) in local reference frame.
- θ_{xv1}, θ_{yv1}, θ_{zv1}, $\theta_{x'v2}$, $\theta_{y'v2}$, $\theta_{z'v2,}$ is fixed when Vector1 and Vector2 are given.

Objective: Compute the possible orientations of Pen1 that satisfying the geometric constraint, i.e., the rotation matrix from Local_RF2 to Global_RF ($R_{lrf2\_to\_grf}$), see Figure 72.

Computation:

1) Representing Vector1 in the global reference frame (Global_RF) as Vector1$_g$ (x1$_g$, y1$_g$, z1$_g$)

$$Vector1_g = R_{lrfl\_to\_grf} \times Vector1 \tag{1}$$

2) Getting an arbitrary vector 'Arbitr_vector1 (x_{ar1}, y_{ar1}, z_{ar1})', then computing an orthogonal vector 'Orth_vector1 (x_{or1}, y_{or1}, z_{or1})' regarding to Vector1$_g$ and Arbitr_vector1.

$$Arbit\_vector_1 = (x_{ar}, y_{ar}, z_{ar}) \tag{2}$$

$$\begin{gathered} Orth\_vector_1 = Vector1_g \times Arbit\_vector_1,\ \ where \\ x_{or1} = y1_g * z_{ar1} - z1_g * y_{ar1}, \\ y_{or1} = z1_g * x_{ar1} - x1_g * z_{ar1}, \\ z_{or1} = x1_g * y_{ar1} - y1_g * x_{ar1} \end{gathered} \tag{3}$$

3) Computing a rotation matrix RotationMatrix$_1$ regarding Orth_vector1 and the allowed angle θ_{v1v2}.

$$RotationMatrix_1 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, where \tag{4}$$

$$\begin{gathered} a_{11} = cos\theta_{v1v2} + (1 - cos\theta_{v1v2}) * {x_{or1}}^2 \\ a_{12} = (1 - cos\theta_{v1v2}) * x_{or1} * y_{or1} - sin\theta_{v1v2} * z_{or1} \\ a_{13} = (1 - cos\theta_{v1v2}) * x_{or1} * z_{or1} + sin\theta_{v1v2} * y_{or1} \\ a_{21} = (1 - cos\theta_{v1v2}) * x_{or1} * y_{or1} + sin\theta_{v1v2} * z_{or1} \\ a_{22} = cos\theta_{v1v2} + (1 - cos\theta_{v1v2}) * {y_{or1}}^2 \\ a_{23} = (1 - cos\theta_{v1v2}) * z_{or1} * y_{or1} - sin\theta_{v1v2} * x_{or1} \\ a_{31} = (1 - cos\theta_{v1v2}) * z_{or1} * x_{or1} - sin\theta_{v1v2} * y_{or1} \\ a_{32} = (1 - cos\theta_{v1v2}) * z_{or1} * y_{or1} + sin\theta_{v1v2} * x_{or1} \\ a_{33} = cos\theta_{v1v2} + (1 - cos\theta_{v1v2}) * {z_{or1}}^2 \end{gathered}$$

4) Rotating Vector1$_g$ regarding RotationMatrix$_1$ to obtain the Vector2$_g$ (x2$_g$, y2$_g$, z2$_g$) (i.e., Vector2 in global reference frame)

$$Vector2_g = RotationMatrix_1 \times Vector1_g \tag{5}$$

5) (Similar to the process 2 to 4) Finding an arbitrary vector Arbitr_vecor2 (x_{ar2}, y_{ar2}, z_{ar2}), computing an orthogonal vector 'Orth_vector2 (x_{or2}, y_{or2}, z_{or2})' regarding to Vector2$_g$ and Arbitr_vector2, computing a rotation matrix RotationMatrix2 regarding to Orth_vector2 and the angle $\theta_{z'v2}$, finally rotating Vector2g according to RotationMatrix2 to find the VectorZ'$_g$ ($x_{zg'}$, $y_{zg'}$, $z_{zg'}$) (i.e., z' in the global reference frame).

$$VectorZ'_g = RotationMatrix_2 \times Vector2_g \tag{6}$$

6) Solving an equation (7) to obtain VectorX'$_g$ ($x_{xg'}$, $y_{xg'}$, $z_{xg'}$) (i.e., x' presented in the global reference frame).

$$\left| VectorX'_g \times VectorZ'_g \right| = 0 \tag{7}$$

$$\left| VectorX'_g \times Vector2_g \right| = sin\theta z'v2$$

$$\left|VectorX'_g\right| = 1 \; and \; \left|Vector2_g\right| = 1$$

7) Using the right-hand rule to find the VectorY'$_g$ ($x_{yg'}$, $y_{yg'}$, $z_{yg'}$) (i.e., y' presented in the global reference frame).

$$\begin{gathered} VectorY'_g = VectorX'_g \times VectorZ'_g, \;\; where \\ x_{yg1} = y_{xg} * z_{zg} - z_{xg} * y_{zg}, \\ y_{yg1} = z_{xg} * x_{zg} - x_{xg} * z_{zg}, \\ z_{yg1} = \; x_{xg} * y_{zg} - y_{xg} * x_{zg} \end{gathered} \tag{8}$$

8) Obtaining the rotation matrix $R_{lrf2\_to\_grf}$ regarding VectorX'$_g$, VectorY'$_g$, VectorZ'$_g$

$$R_{lrf2\_to\_grf} = \begin{bmatrix} VectorX'_g \\ VectorY'_g \\ VectorZ'_g \end{bmatrix} = \begin{bmatrix} x_{xg}' & y_{xg}' & z_{xg}' \\ x_{yg}' & y_{yg}' & z_{yg}' \\ x_{zg}' & y_{zg}' & z_{zg}' \end{bmatrix} \tag{9}$$

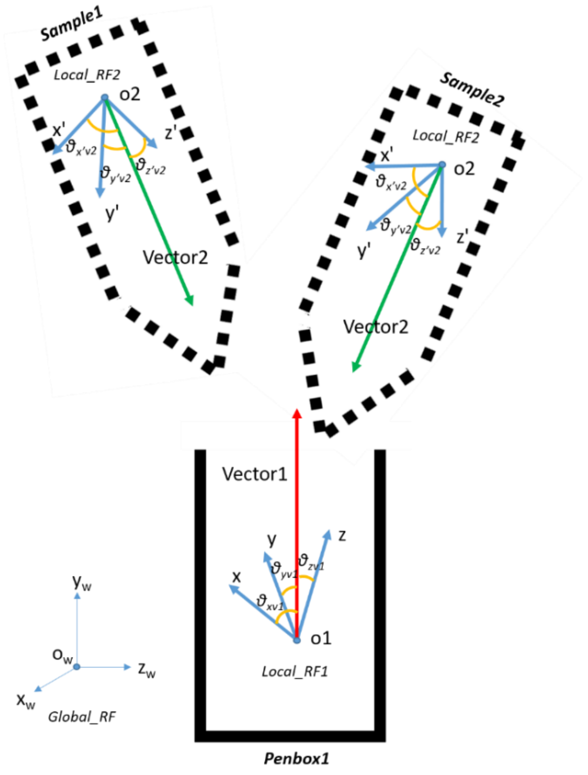


Figure 72 The possible orientations of Pen1 (Sample1, Sample2)

Appendix 2: The Shape Embedding Game

A. *The representation of geometric constraint*

As we can see from section 5.6.2, the geometric constraint inferred from a spatial constraint "O3 *ShapeAligned* P5" is :

Vector1 *Against* Vector2

Vector3 *SameDirection* Vector4

The visual representation of this geometric constraint is given in Figure 73-a,b,c.

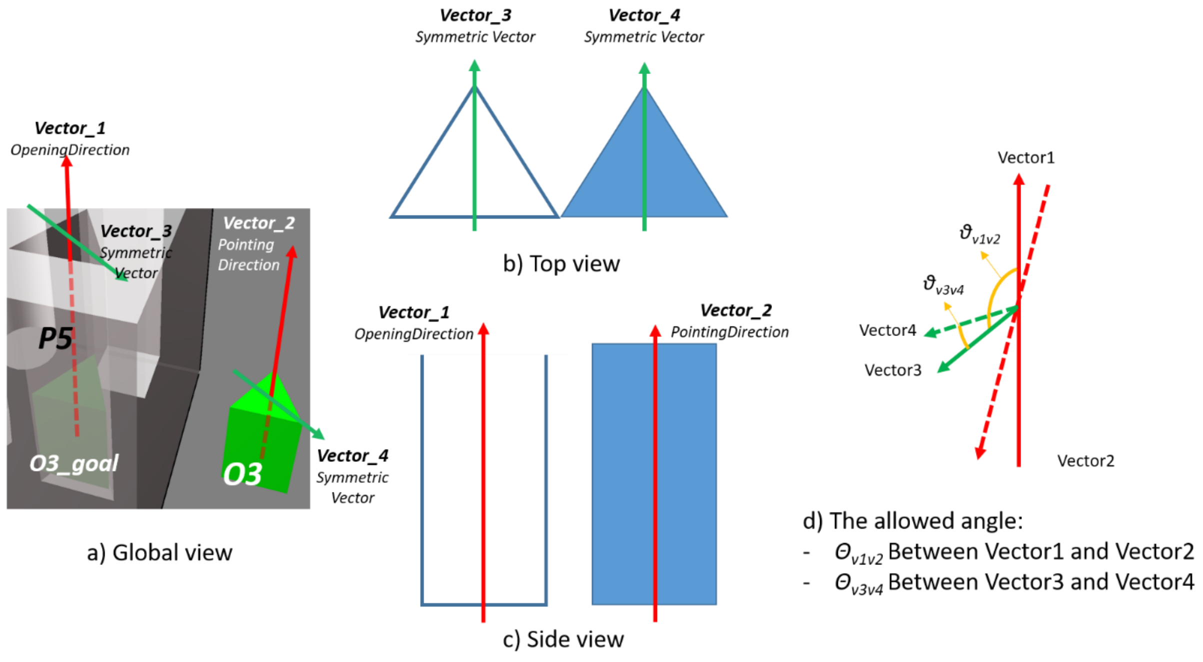


Figure 73 The shape alignment for O3 and P5

'Vector1 *Against* Vector2' defines that these two vectors are pointed to the opposite direction (spatial relation: *Against*). Its definition can refer to the one used in Appendix 1. 'Vector3 *SameDirection* Vector4' defines that these two vectors are pointed to the same direction (spatial relation: SameDirection). Similar to 'Vector1 *Against* Vector2' (see Appendix 1), in this thesis work, rather than exclusively opposite, the amplitude between Vector3 and Vector4 is allowed and this amplitude is limited by a threshold value ($\theta_{threshold}$). Therefore, θ_{v3v4} between Vector3 and Vector4 (Figure 73-d) should be inside an interval (0, $\theta_{threshold}$).

B. *Current status of the environment*

Before using the geometric constraint to control motions of O3, we will firstly introduce some current status of the environment in Figure 74.

- Local_RF1 and Local_RF2 are respectively the local reference frame of O3 and P5
- o1 and o2 are the origins of the Local_RF1 and Local_RF2

- Vector1 (x1, y1, z1) and Vector3 (x3, y3 z3) is defined in Local_RF1
- Vector2 (x2, y2, z2) and Vector4 (x4, y4 z4) is defined in Local_RF2
- θ_{xv1}, θ_{yv1}, and θ_{zv1} is the angle between the x, y, z axes of Local_RF1 and Vector1
- $\theta_{x'v2}$, $\theta_{y'v2}$ and $\theta_{z'v2}$ is the angle between the x', y', z' axes of Local_RF2 and Vector2
- $\theta_{x'v4}$, $\theta_{y'v4}$ and $\theta_{z'v4}$ is the angle between the x', y', z' axes of Local_RF2 and Vector4.
- The global reference frame is the Global_RF located at the origin o_W and x_W, y_W, z_W axes.

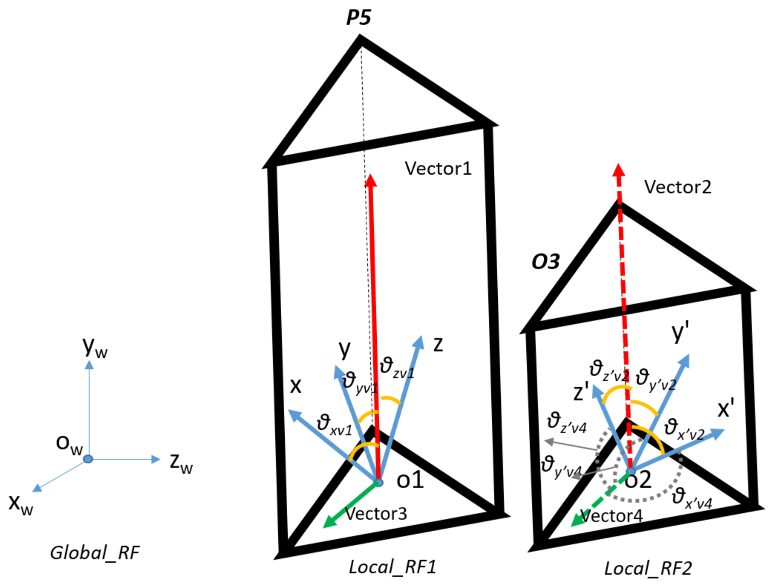


Figure 74 Geometric Constraint 2

C. *The usage of geometric constraint*

Pre-condition:

- P5 is a hole on the Panel. Because Panel is fixed to the ground, thus P5 is also fixed to the ground, i.e., the rotation matrix from Local_RF1 to Global_RF ($R_{lrf1\_to\_grf}$) does not change
- Vector1 (x1, y1, z1) and Vector3 (x3, y3 z3) is defined in Local_RF1
- Vector2 (x2, y2, z2) and Vector4 (x4, y4 z4) is defined in Local_RF2
- θ_{xv1}, θ_{yv1}, θ_{zv1}, $\theta_{x'v2}$, $\theta_{y'v2}$, $\theta_{z'v2}$, $\theta_{x'v4}$, $\theta_{y'v4}$, $\theta_{z'v4}$ are fixed when Vector1, Vector2, and Vector4 are given.

Objective: Compute the possible orientations of O3 that satisfying the geometric constraint, i.e., the rotation matrix from Local_RF2 to Global_RF ($R_{lrf2\_to\_grf}$), see Figure 75.

Computation:

1) Computing Vector2$_g$ ($x2_g$, $y2_g$, $z2_g$) (Vector2 in global reference frame) using formulas (1) to (5)

2) Computing Vector3$_g$ (x3$_g$, y3$_g$, z3$_g$) (Vector 3 in global reference frame) using R$_{lrf1\_to\_grf}$

$$Vector3_g = R_{lrfl\_to\_grf} \times Vector3 \tag{10}$$

3) Computing Vector4$_g$ (x4$_g$, y4$_g$, z4$_g$) using formulas (4) and (10)

$$Vector4_g = RotationMatrix_1 \times Vector3_g \tag{11}$$

4) Computing VectorZ'$_g$ (x$_{zg'}$, y$_{zg'}$, z$_{zg'}$) in the global reference frame by solving formula (12)

$$\left| Vector2_g \times VectorZ'_g \right| = sin\theta z'v2 \tag{12}$$

$$\left| Vector4_g \times VectorZ'{}_g \right| = sin\theta z'v4$$

$$\left|VectorZ'_g\right| = 1, \left|Vector2_g\right| = 1 \; and \; \left|Vector4_g\right| = 1$$

5) Computing VectorY'$_g$ (x$_{yg'}$, y$_{yg'}$, z$_{yg'}$) in the global reference frame by solving formula (13)

$$\left| Vector2_g \times VectorY'_g \right| = sin\theta y'v2 \tag{13}$$

$$\left| Vector4_g \times VectorY'{}_g \right| = sin\theta y'v4$$

$$\left|VectorY'_g\right| = 1, \left|Vector2_g\right| = 1 \; and \; \left|Vector4_g\right| = 1,$$

$$\left|VectorY'_g \times VectorZ'_g\right| = 0$$

6) Using the right-hand rule to find the VectorX'$_g$ (x$_{xg'}$, y$_{xg'}$, z$_{xg'}$) (i.e., y' presented in the global reference frame).

$$\begin{gathered} VectorX'_g = VectorY'_g \times VectorZ'_g, \;\; where \\ x_{xg1} = y_{yg} * z_{zg} - z_{yg} * y_{zg}, \\ y_{xg1} = z_{yg} * x_{zg} - x_{yg} * z_{zg}, \\ z_{xg1} = x_{yg} * y_{zg} - y_{yg} * x_{zg} \end{gathered} \tag{14}$$

7) Obtaining the rotation matrix R$_{lrf2\_to\_grf}$ regarding VectorX'$_g$, VectorY'$_g$, VectorZ'$_g$

$$R_{lrf2\_to\_grf} = \begin{bmatrix} VectorX'_g \\ VectorY'_g \\ VectorZ'_g \end{bmatrix} = \begin{bmatrix} x_{xg}{}' & y_{xg}{}' & z_{xg}{}' \\ x_{yg}{}' & y_{yg}{}' & z_{yg}{}' \\ x_{zg}{}' & y_{zg}{}' & z_{zg}{}' \end{bmatrix} \tag{15}$$

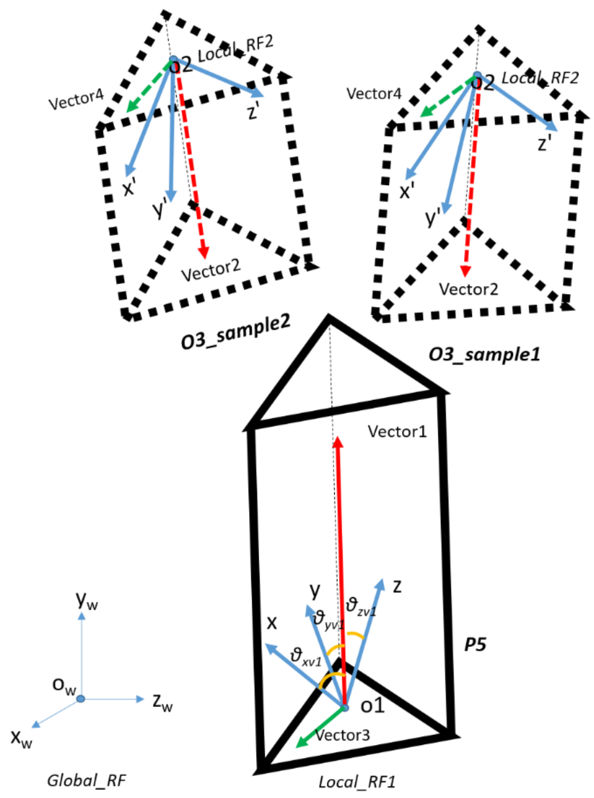


Figure 75 The possible orientations of O3 (O3_Sample1, O3_Sample2)