View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Open Archive Toulouse Archive Ouverte

Open Archive Toulouse Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of some Toulouse researchers and makes it freely available over the web where possible.

This is an author's version published in: http://oatao.univ-toulouse.fr/22789

Official URL: https://doi.org/10.1109/AICCSA.2018.8612805

To cite this version:

Zhao, Yingshen and Fillatreau, Philippe and Karray, Mohamed Hedi and Archimède, Bernard An ontology-based approach towards coupling task and path planning for the simulation of manipulation tasks. (2018) In: 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA), 28 October 2018 - 1 November 2018 (Aqaba, Jordan).

Any correspondence concerning this service should be sent to the repository administrator:

tech-oatao@listes-diff.inp-toulouse.fr

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

YINGSHEN ZHAO
INP-ENIT, Université Fédérale de Toulouse Midi-Pyrénées
Tarbes, France
yingshen.zhao@enit.fr

Philippe FILLATREAU
INP-ENIT, Université Fédérale de Toulouse Midi-Pyrénées
Tarbes, France
philippe.fillatreau@enit.fr

Mohamed Hedi KARRAY
INP-ENIT, Université Fédérale de Toulouse Midi-Pyrénées
Tarbes, France
mkarray@enit.fr

Bernard ARCHIMEDE
INP-ENIT, Université Fédérale de Toulouse Midi-Pyrénées
Tarbes, France
bernard.archimede@enit.fr

***Abstract*—Simulating complex industrial manipulation tasks (e.g., assembly, disassembly and maintenance tasks) under strong geometric constraints in a virtual environment, requires the joint usage of task and path planning, not only to compute a sequence of primitive actions (i.e., a task plan) at task planning level to identify the order to manipulate different objects (e.g., assembly order), but also to generate and validate motions for each of these primitive actions in a virtual environment by computing valid collision-free paths for these actions at path planning level. Although task and path planning have been respectively welly discussed by artificial intelligence and robotic domain, the link between them still remains an open issue, in particular because path planning for a primitive action often uses purely geometric data. This purely geometric path planning suffers from the classical failures (i.e., high-possibility of failure, high processing time and low path relevance) of automated path planning techniques when dealing with complex geometric models. Thus, it can possibly lead to high computational time of the joint task and path planning process and can probably produce a poor implementation of a task plan. Instead of geometric data, involving higher abstraction level information related to a task to be performed in the path planning of a primitive action could lead to a better relevance of simulations. In this work, we propose an ontology-based approach to generate a specific path planning query for a primitive action, using a well-structured task-oriented knowledge model. This specific path planning query aims at obtaining an increased control on the path planning process of the targeted primitive action.**

Keywords—Coupling task and path planning, ontology, task-oriented knowledge, knowledge reasoning

I. INTRODUCTION

Nowadays, while industrial products are more and more complex and integrated, their development time and cost must be reduced, whereas health, environmental and quality standards are increasingly demanding. Facing these challenges, industry expresses its increasingly strong need to use integrated numerical tools all along the stages of the product lifecycle management (PLM). In particular, virtual prototypes should provide a way to validate, from design stage on, tasks (e.g., assembly, disassembly or maintenance) related to the lifecycle of a product through simulations.

In this work, we focus on simulating and validating manipulation tasks. Researchers from the robotics community studied the simulation of manipulation tasks following two approaches. The first approach [1] follows the task planning endpoint. It solves a complex manipulation task problem by generating a sequence of primitive actions, for example, to reach a goal state where a complex product is assembled. The second approach [2] follows the path planning endpoint. It focuses on generating and validating motions notably for manipulated objects possibly under strong geometric constraints. Among these two approaches, path planning cannot determine the sequence to manipulate different objects (e.g., assembly order of a product), whereas task planning can resolve the object manipulation sequence problem by computing a sequence of primitive actions (i.e., a task plan) for the manipulations. Moreover, task planning uses higher abstraction level information (e.g., symbolic terms such as On (book1, table1)) instead of geometric data, and only path planning could allow to geometrically check motion feasibility (i.e., to compute a valid collision-free path) for the primitive actions in a task plan. Therefore, in order to solve a complex manipulation task and to generate valid geometric paths for manipulated objects in a virtual environment, it requires considering them jointly.

In such joint approaches [3, 4, 5, 6, 7, 8], a task planner searches a feasible sequence of primitive actions to manipulate different objects (i.e., a task plan), and a geometric path planner is called to check the motion feasibility of these actions. Although task and path planning have been respectively welly discussed by artificial intelligence and robotic domain, the link between them still remains an open issue, in particular because path planning for a primitive action uses purely geometric data. In a highly geometrically constrained environment, this geometric path planning suffers from the classical failures of automated path planning techniques. Thus, it can possibly lead to high computational

time of the joint task and path planning process and can probably produce a poor implementation of a task plan.

Instead of purely geometric data, involving higher abstraction level information related to a task could produce a better relevance of simulations. The work here investigates the benefits of using task oriented knowledge in the path planning process of a primitive action. The first step aims at generating a specific path planning query for a primitive action, using a well-structured task-oriented knowledge model. We propose an ontology-based strategy to generate a specific path planning query for a primitive action, based on the development of 1) an ontology-based knowledge model, including a well-structured environment representation related to a task, the specification of primitive action and the description of path planning query; 2) an inference mechanism to generate a path planning query description from a primitive action specification, using SWRL (Semantic Web Rule Language) rules.

The rest of the paper is organized as follows. Section II presents the literature in task planning and path planning. Section III gives a use-case used throughout our discussion. Section IV illustrates task-oriented knowledge in which we are interested. Section V presents in detail the modeling of task-related information. Using this information, section VI generally discusses how to generate a specific path planning query for a primitive action. Section VII describes in detail the generation process. The conclusion is presented in section VIII.

II. Related works

A. Task planning and Path planning

1) Task planning, as a thoroughly sub-field of artificial intelligence, has been studied largely since 1960. The classical task planning process [1] aims at generating a sequence of successive primitive actions [9], transiting from an initial state of the world to a pre-defined goal state, using forward-searching [10] or backward-searching [3] algorithms.

A world state [11] is described by symbolic terms (e.g., On (book1, table1)) using symbols and prepositions. These symbols and prepositions respectively represent environment instances (e.g., book1, table1) and relations (e.g., On, In, Left).

A primitive action [11] allows state transition where some terms are removed from a world state, and some are added to it. A primitive action consists of two parts: precondition and effect. Pre-condition describes the terms that current world state must satisfy when an action is performed. Effect describes the terms that will be added or removed from current world state after an action is performed.

Task planning solves the object manipulation sequence problem, and it uses higher abstraction level information (symbolic terms, such as On (book1, table1)) instead of geometric data (e.g., coordinates or angles). Using this abstract representation does not allow to check the motion feasibility of primitive actions of a task plan and to compute valid collision-free paths for the manipulated objects in a virtual environment.

2) Path Planning: Since 1980, automated path planning techniques have been deeply and widely discussed by the robotics community [2, 12]. They aim at producing a collision-free path (a sequence of free configurations) moving a manipulated object from an initial configuration (position and orientation) to a goal configuration. [13] proposed a four-quadrant categorization of path planning techniques [14, 15, 16, 17], from deterministic to probabilistic and from global to local. Because most of the path planning works often uses purely geometric models and little higher abstraction level information has been taken into account, in a highly geometrically constrained environment, it could probably fail, could result in high-processing time and the generated paths could be of low-relevance. More importantly, path planning is unable to determine the necessary sequence of actions to manipulate different objects.

Therefore, in order to determine the feasible sequence of primitive actions for the manipulations and compute a valid collision-free path for each action, solving a complex manipulation task in a virtual environment expresses a strong need to consider jointly task and path planning.

B. The joint usage of task planning and path planning

To use jointly task and path planning [3, 4], a task planner searches for a feasible task plan that consists of a sequence of primitive actions, e.g., to manipulate objects, and a geometric path planner is called to check the motion feasibility (i.e., path planning) of these actions.

In terms of how task and path planning relate to each other, two approaches have been identified. Firstly, some researchers [5, 6] discussed them in a discrete and iterative way: *task planning first, then path planning*. A task plan is constructed firstly, and a geometric path planner is then called to check the motion feasibility of all primitive actions in this plan. If no path is found in a given time threshold, the geometric path planner informs the task planner to find an alternative task plan. Secondly, some others considered them in a continuous way [3, 7, 8]: *task planning querying path planning*. The motion feasibility of a primitive action is verified by a geometric path planner during task planning process, when this action is applied by a task planner.

A key issue in the joint usage is to effectively link task and path planning to improve the performance of the planning process. [5] used *external predicates/functions* at task planning level to call a geometric path planner, to check the motion feasibility of a primitive action applied by task planner. [4] used a list of *geometric predicates* (reachability / visibility of objects), computed by geometric path planner or other geometric reasoners, in the task planning process to determine the primitive actions that can be applied by task planner. Yet, the link them is still an open issue due to the fact that,

1) path planning for a primitive action uses purely geometric data, and it suffers from the classical failures of automated path planning techniques, and it can further lead to high computational time of the joint task and path planning process and can produce a poor implementation (low path relevance) of a task plan.

2) to search for an optimal task plan for manipulation, more kinds of feedback from path planning results, e.g., complexity or relevance of computed paths, should be taken into account.

We concentrate on the first issue of the joint task and path planning. Rather than purely geometric data, higher abstraction level information should be considered. [13] proposed a multi-level environment model and path planning architecture to use semantic (complexity to cross a place, object shape, obstacle mobility), topological (places and borders, a topological graph to describe the connectivity of places) and geometric (3D object meshes and octree decomposition to describe 3D free space) information in a two-step path planning process (coarse planning computes a least cost topological path, and fine planning finds a collision free geometric path for each step on this topological path). It improved the relevance of computed paths and the performance of the planning system. Nonetheless, it is still a path planning approach, and the semantic level is still poor with only text information.

Our work further develops this multi-level path planning architecture [13] to investigate the usage of richer semantic information and to develop towards a joint task and path planning approach to solve a task problem. Different kinds of high abstraction level information exist in a task, such as the spatial representation of an environment (e.g., the location of an object, the relative location between two objects), and potential constraints related to a task to be performed (e.g., spatial relations to be obeyed between two objects). Yet, the employment of this task-related information in the path planning process of a primitive action is rarely issued.

In this paper, we investigate the benefits of using the task-oriented knowledge for the path planning process of a primitive action. In particular, we focus on the knowledge related to a primitive action to be performed (see Section III) and generating a specific path planning query for a targeted primitive action. This specific path planning query aims at obtaining a better control on the path planning of this action, and furthermore, a better relevance of simulations.

III. A PEN-PENBOX INSERTION USE CASE

In this paper, 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 fig. 1-(a) to fig. 1-(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.

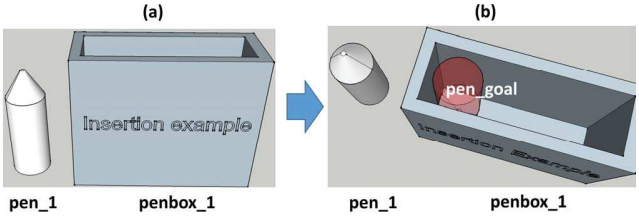

Figure 1 A pen-penbox insertion use case

IV. TARGETED TASK-ORIENTED KNOWLEDGE

Action-specific knowledge

In order to correctly execute a primitive action for manipulation (such as "Put a book on a table", "Insert a pen into a penbox"), the planning system should understand and has to process action-specific knowledge [18]. This action-specific knowledge is used not only to identify current location and goal place of a manipulated object but also to provide specific constraints during the manipulation. In this paper, the targeted task-oriented knowledge concerns the constraints about how a manipulated object should be placed in the 3D (3-Dimensional) space in relation to a reference object during the manipulation. For example, when inserting a pen into a penbox, the pen should point to the penbox.

Spatial Constraints

The relation describing the relative location (i.e., position and orientation) between two objects in the 3D space is specified as a spatial relation [19]. Typical spatial relations include, among others: Left, Right, Inside, Outside, PointTo and so on. Then, we define a spatial constraint as a constraint that describes a spatial relation between a manipulated object and a reference object. Widely speaking, a spatial constraint can specify a spatial relation not only between two objects but also between an object and an area (a closed bounded volume of 3D space). The spatial constraint is described by symbolic terms, e.g., PointTo (pen, penbox). It is easily understandable to humans, but it is insufficient to be used for path planning, which mainly deals with geometric models of an environment.

Geometric Constraints

A geometric constraint specifies a spatial relation between two geometric elements [20], e.g., the origin of a pen is on the left of the origin of a penbox. The geometric elements refer to the ones used in the geometric modeling, such as point, line, face, bounding box, vector. Moreover, geometric constraints have been discussed and used long ago in path planning [21].

Linking Spatial and Geometric Constraints

Spatial constraints often come from the action-specific knowledge related to a primitive action to be performed. In order to employ those spatial constraints in the path planning process of a primitive action, it is required to map spatial constraints into geometric constraints. In particular, we are interested in generating the geometric constraints from the spatial constraints, depending on the task-oriented environment knowledge (the type and the shape of an object or an area involved in an action) and the action itself. These generated geometric constraints are to be used in the control of the path planning process of a primitive action.

V. AN ONTOLOGY-BASED ENVIRONMENT KNOWLEDGE REPRESENTATION

In order to describe objects or areas involved in a spatial constraint and geometric elements used in a geometric constraint, a spatial knowledge representation of an environment not only should describe high abstraction level information related to a task (e.g., world knowledge about the

taxonomy of objects and areas), but also has to represent geometric models of an environment. Furthermore, spatial relations should also be considered in the modeling of a spatial constraint and a geometric constraint. Therefore, the environment knowledge representation here is going to conceptualize in the knowledge base:

- The knowledge of the multi-level environment model [13], consisting of information on the semantic, topologic and geometric level (see Section II-B), since our work further develops the multi-level path planning approach [13] and the multi-level environment information allows to welly describe the necessary elements (objects, areas, geometric elements) in the spatial/geometric constraint description.
- The qualitative spatial knowledge about an environment, including 1) world knowledge about the environment in a particular task, and 2) spatial relation knowledge describing relative location between two objects, between an object and an area, and between two geometric elements.

A. The conceptualization of a multi-level environment model and world knowledge

A multi-level environment model knowledge

The conceptualization of the multi-level environment model (Fig. 2) is decomposed into three levels, following the one proposed by [13]: semantics, topology, and geometry. It further develops the semantic level by introducing world knowledge about an environment in a particular task.

1) Knowledge on the geometric level: It concerns the concepts and the relations used in the geometric modeling of a 3D environment. Two abstract concepts are discussed.

- ***Geometry*** describes the geometric models of a 3D environment. *ObjectGeometry* and *SpaceGeometry* are two sub-concepts concerning respectively geometric models of *RigidBody* and *FreeSpace*. A *RigidBody* is an *ObjectGeometry*, and it is solid with no deformation allowed. *FreeSpace* is a *SpaceGeometry*, and it represents continuous geometric 3D space with no obstacles overlapped.
- ***Geometric element*** describes the elements belonging to the geometric models of a *RigidBody* or *FreeSpace*. A *BoundingBox* represents the smallest enclosing box of a *RigidBody* or *FreeSpace*. *GeometricPrimitive* is the common set of primitives used in 3D modeling such as *Point*, *Line*, *Face*. *Vector* is a more mathematical concept, and it is often used to describe other concepts such as the position of a *Point* and the pointing direction of a *Geometry*. Orientation reference frame (*OrientationRF*) assists describing orientation relation.

2) Knowledge on the topological level: It aims at conceptualizing places and borders, constructed at topological level in [13]. *Place* and *Border* are sub-concepts of *Area*, representing a closed bounded volume of geometric 3D free space. A *Border* connects two *Places*.

3) Knowledge on the semantic level: The environment information on this level is strongly related to a task to be performed. It is described by the world knowledge about an environment of a task, and it aims to obtain a detailed classification of the concepts on the two other levels (e.g., *RigidBody* at the geometric level and *Place* defined at the topological level).

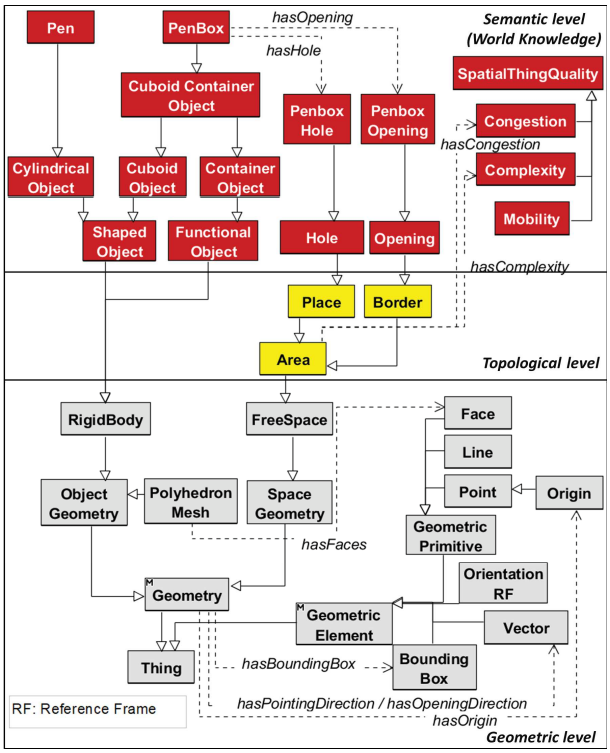

Figure 2 A taxonomy of multi-level environment knowledge (Concepts)

World Knowledge

World knowledge concerns the detailed taxonomy of *RigidBody*, *Place* and *Border*, their properties, and relations, under a particular task context.

- ***ShapedObject* and *FunctionalObject***: A *RigidBody* is further classified into *ShapedObject* and *Functional Object* respectively, depending on the shape and the functionality of an object. A particular *RigidBody* can be both a *ShapedObject* and a *FunctionalObject*. For example, a *Pen* is a *CylindricalObject*, and a *PenBox* is a *CuboidContainerObject*.
- ***Hole* and *Opening***: *Place* and *Border* are further classified according to some characteristics of the 3D free space. A *Hole* is a *Place* within an object. An *Opening* is a *Border* that connects a *Hole* with the rest of geometric free space. And they are further respectively classified into a *PenBoxHole* and a *PenboxOpening* that belong to a *PenBox*.
- ***SpatialThingQuality***: It concerns all quality information belonging to *RigidBody*, *Place*, and *Border*. *Congestion* and *Complexity* describe how difficult to cross a *Place* or a *Border*. *Mobility* identifies whether a *RigidBody* can be moved or not.

B. *The Conceptualization of Spatial Relation Knowledge*

Spatial relation knowledge should also be modeled in order to describe the relative location between two geometries (object or area) in a spatial constraint or between two geometric elements in a geometric constraint. Inspired by [22, 23], the taxonomy specializes into three categories (Fig. 3).

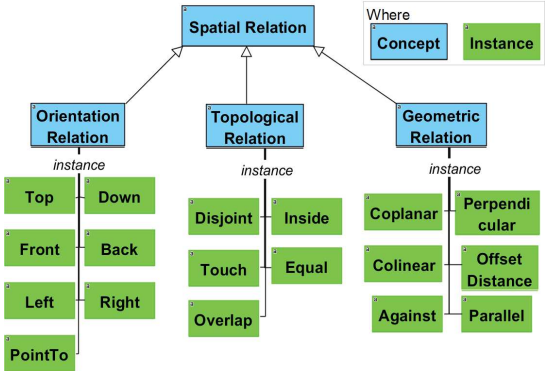

Figure 3 Spatial Relation Knowledge

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

Orientation Relation describes how two geometries (object or area) or two geometric elements are placed relative to one another. *Top*, *Down*, *Front*, *Back*, *Left*, *Right*, and *PointTo* are all instances of this concept. This relation is described according to a frame of reference.

Geometric Relation describes how two geometric elements (e.g., line, plane) relate to each other. *Coplanar*, *Perpendicular*, *Offset*, *Collinear*, *Against*, *Parallel* are all instances of this concept.

VI. Generation of a Path Planning Query for a Primitive Action in a Task Plan

In this section, we want to consider jointly task and path planning and focus on generating a specific path planning query for a primitive action, using action-specific knowledge. Especially, we are interested in generating geometric constraints from spatial constraints, where geometric constraints are specified in a path planning query and spatial constraints are defined in a specification of a primitive action. Finally, geometric constraints are used to guide the search in the path planning, which we have not yet discussed in the paper. Subsection VI-A discusses the knowledge modeling of a primitive action and a path planning query. Subsection VI-B shows an overview of the generation process.

A. *The modeling of 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. Fig. 4 shows the partial modeling of a primitive action specification (PAS) and a path planning query description (PPQD).

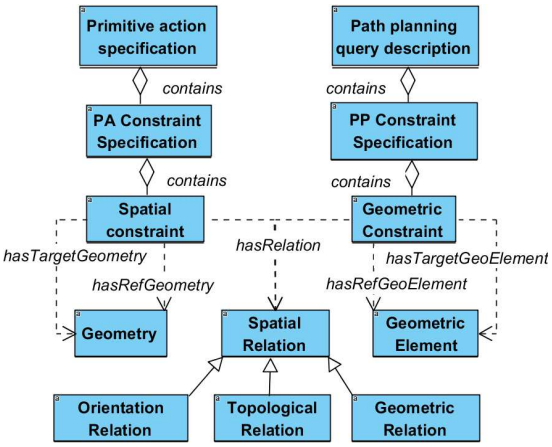

Figure 4 The knowledge modeling of PAS and PPQD (Concepts)

Primitive action (PA) specification (PAS): A primitive action specification 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 rigid body that a system manipulates.
- Reference Geometry (RG): a rigid body or a place to which the action references.
- PA Constraint Specification: The spatial constraints for the primitive action.

A spatial constraint here specifies a spatial relation between two geometries (object or area), currently, that is between two rigid bodies or between a rigid body and a place. It consists of three parts: 1). A reference geometry, 2). A target geometry, 3). A related spatial relation

Path planning (PP) query description (PPQD): A path planning query description consists of three parts:

- Manipulated object: the rigid body that a system manipulates.
- Goal Configuration: The configuration to reach
- PP Constraint Specification: The geometric constraints in a path planning query

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.

B. *Generation of a path planning query description through an inference process*

As the first step of the path planning query generation process, 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 targeted primitive action. A list of spatial constraints is

manually assigned by human operator and is associated to the action specification (Fig. 5).

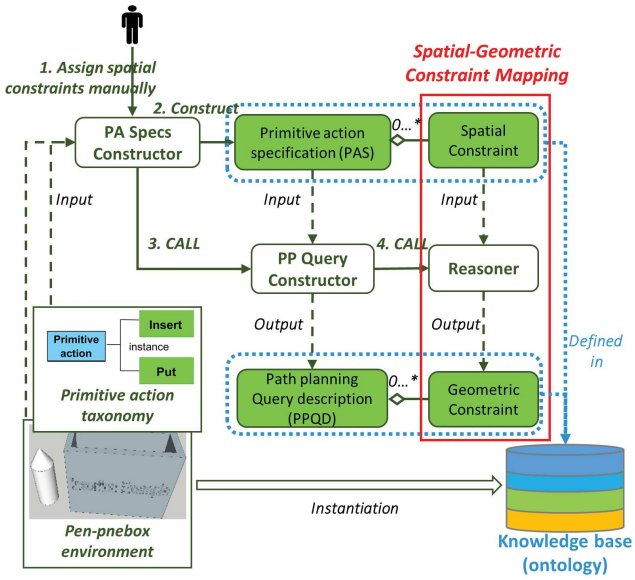

Figure 5 The process to construct a PAS and to generate a related PPQD

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 obtained in the goal region, while satisfying the geometric constraints that are associated to the PPQD.

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 Fig. 5. 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.

VII. A Pen-Penbox Insertion Example

With the help of a pen-penbox insertion case, this section starts to describe in detail the generation of a path planning query for a primitive action, especially the generation of geometric constraints for path planning from spatial constraints defined in a primitive action.

The subsections discuss 1) the instantiation of a pen-penbox environment in the knowledge base, 2) the specification of an insertion action with the associated spatial constraints, 3) through an inference process, the generation of a path planning query description with geometric constraints.

A. The instantiation of Pen-Penbox case in knowledge base

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).

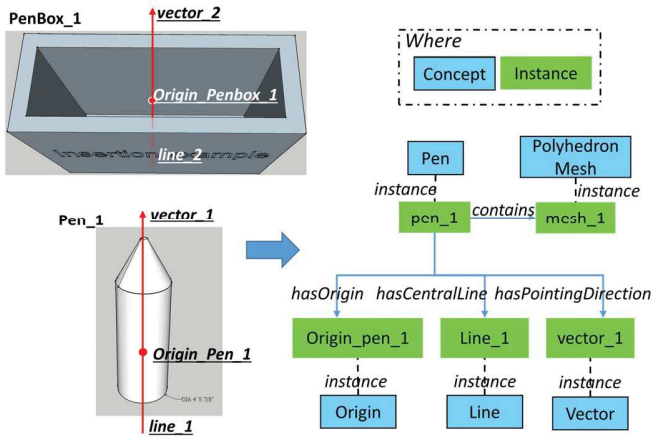

Figure 6 An instantiation of a pen in the environment model knowledge

Fig. 6 shows an instantiation of a pen in the environment model knowledge. "pen_1" is an instance of the concept *Pen*, it has a *PolyhedronMesh* "mesh_1", an *Origin* "Origin_pen_1", a central line (*Line*) "Line_1" and a pointing direction (*Vector*) "vector_1". This information is either manually annotated or automatically generated. Although this annotation would be long and tedious, it can be saved as a config file and be reused in the next time.

B. The instantiation of spatial relation knowledge

In the pen-penbox insertion scenario, the penbox is regarded as a reference geometry. To correctly define the orientation relation between a pen and a penbox, we specify an orientation reference frame, located at the penbox's origin. Then, the orientation relations are described by different vectors, based on this reference frame, see from Fig. 7.

- +-X corresponds to the *Front* and *Back* of penbox
- +-Y corresponds to the *Right* and *Left* of penbox
- +-Z corresponds to the *Top* and *Bottom* of penbox.

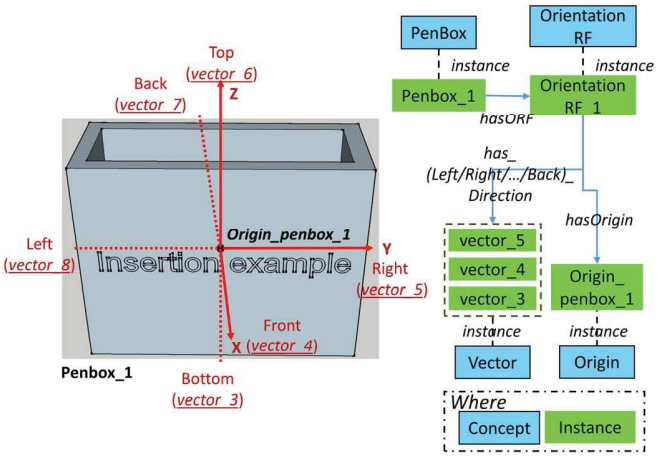

Figure 7 The orientation relations referred to penbox's origin

C. *Primitive action specification for pen-penbox insertion*

A primitive action specification is constructed from an insertion action between "pen_1" and "penbox_1" (Fig. 8). The primitive action identity ("Insert"), the manipulated object ("pen_1") and the reference geometry ("penbox_1") are directly extracted from this insertion action.

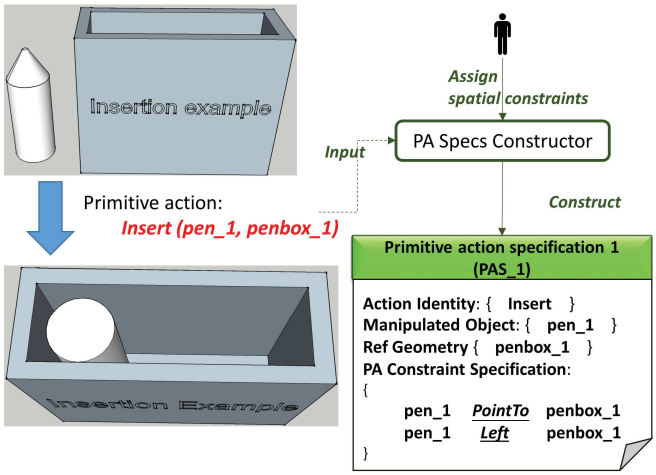

Figure 8 The construction of a primitive action specification

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 "pen_1" and "penbox_1", and then they are associated to the specification of this insertion action. Two spatial constraints are used here.

pen_1 *PointTo* penbox_1

pen_1 *Left* penbox_1

D. *The generation of a path planning query description using spatial-geometric constraint mapping*

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 Fig. 9. The manipulated object ("pen_1") in "PPQD_1" is directly extracted from "PAS_1". The goal configuration ("Goal_pen_1") is obtained by a 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 VII-C, through an inference process using the pre-defined inference rules (Fig. 9).

Pen_1 *PointTo* Penbox_1: For simplicity, we use a spatial constraint "pen_1 *PointTo* penbox_1" as an example to show how a geometric constraint, "vector_1 *Against* vector_2", 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 holds, then the terms in consequent must holds.

A SWRL Rule for spatial-geometric constraint mapping

```
1. IF {
2.  a instance_of CylindericalObject and
3.  b instance_of ContainerObject and
4.  vector_1, vector_2 instance_of Vector and
5.  a hasPointingDirection vector_1 and
6.  b hasOpeningDirection  vector_2 and
7.  sconstraint instance_of SpatialConstraint and
8.     sconstraint  hasRelation "PointTo" and
9.     sconstraint  hasRefGeometry "b" and
10.    sconstraint  hasTargetGeometry "a" and
11. gconstraint instance_of GeometricConstraint and
12.    gconstraint hasRelatedSC "sconstraint"
13. }
14. THEN {
15.  gconstraint hasRefGeoElement "vector_2" and
16.  gconstraint hasTargetGeoElement "vector_1" and
17.  gconstraint hasRelation "Against"
18. }
```

1) **Antecedent** (line 1-13): 1) Environment specification (line 2-6): "object a" is an instance of *CylindericalObject* (line 2), "object b" is an instance of *ContainerObject* (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.

Fig. 10 visually shows why the "pen_1 *PointTo* penbox_1" results in "vector_1 *Against* vector_2". In our pen-penbox insertion example, the corresponding geometric constraints (see in Fig. 9), generated from the spatial constraints specified in VII-C, are

vector_1 *Against* vector_2

Origin_pen_1 *Left* Origin_penbox_1

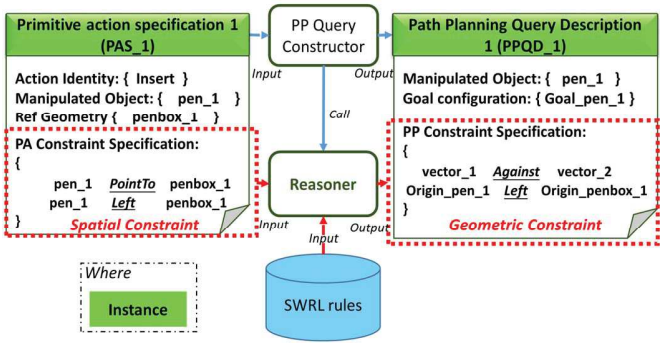

Figure 9 The generation of a path planning query description

Finally, when all geometric constraints are inferred and are generated from the spatial constraints, they are specified in a path planning query description. Using this specific path planning query description as an input, a path planner computes a relevant trajectory/path for this pen-penbox insertion action while respecting the geometric constraints.

VIII. Conclusion and Future Work

The work here presents an attempt to generate a specific path planning query for a primitive action by using a well-structure task-oriented knowledge model. In particular, this work concentrates on using spatial constraints (defined in a primitive action specification) to generate geometric constraints (specified in a path planning query) through an inference process using the pre-defined rules. Then, the specific path planning query aims at obtaining a better control (e.g., lower-processing time, higher path relevance) on the path planning process during the execution of a primitive action. Currently, the work is still preliminary. Firstly, the ontology-based knowledge model has limited number of concepts and rules, and it has to be further developed, validated and evaluated (e.g., to evaluate the metrics of the ontology model, to verify whether the system's responding time is acceptable using the inference engine). Secondly, we believe that more experiments are necessary to validate this work. The objective is to show that a better control (e.g., lower processing time, higher path relevance) on the path planning process of a primitive action can be obtained, by using a proper set of task-related semantic information instead of purely geometric data. Thirdly, the work should be fully evaluated in a real-world scenario, such as an industrial product assembly task. Fourthly, the current work generates and uses geometric constraints in a geometric path planning process, and it requires a further development towards a complete usage of the multi-level path planning architecture [13] (coarse/topologic and fine/geometric planning). Last but not the least, the generation of a task plan is not issued in this work, and it would be necessary to develop towards a complete task and path planning approach.

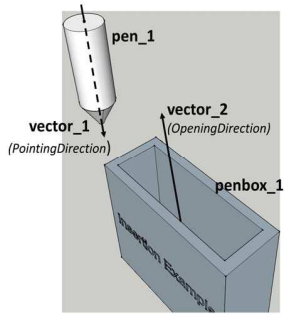

Figure 10 A graphical representation of the SWRL rule

References

[1] B. Bonet and H. Geffner, "Planning with incomplete information as heuristic search in belief space," in Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems. AAAI Press, 2000, pp. 52–61.

[2] J.-C. Latombe, Robot motion planning. Springer Science & Business Media, 2012, vol. 124.

[3] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in 2011 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2011, pp. 1470–1477.

[4] L. de Silva, A. K. Pandey, M. Gharbi, and R. Alami, "Towards combining htn planning and geometric task planning," arXiv preprint arXiv:1307.1482, 2013.

[5] E. Erdem, K. Haspalamutgil, C. Palaz, V. Patoglu, and T. Uras, "Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation," in 2011 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2011, pp. 4575–4581.

[6] R. Dearden and C. Burbridge, "An approach for efficient planning of robotic manipulation tasks." in ICAPS, 2013.

[7] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014, pp. 639–646.

[8] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," The International Journal of Robotics Research, vol. 28, no. 1, pp. 104–126, 2009.

[9] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, Artificial intelligence: a modern approach. Prentice hall Upper Saddle River, 2003, vol. 2, no. 9.

[10] F. Giunchiglia and P. Traverso, "Planning as model checking," in European Conference on Planning. Springer, 1999, pp. 1–20.

[11] S. G. Tzafestas, Introduction to mobile robot control. Elsevier, 2013.

[12] W. Burgard et al., "Principles of robot motion," 2005.

[13] S. Cailhol, P. Fillatreau, J.-Y. Fourquet, and Y. Zhao, "A hierarchic approach for path planning in virtual reality," International Journal on Interactive Design and Manufacturing (IJIDeM), vol. 9, no. 4, pp. 291–302, 2015.

[14] R. A. Brooks, "Solving the find-path problem by good representation of free space," IEEE Transactions on Systems, Man, and Cybernetics, no. 2, pp. 190–197, 1983.

[15] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," The international journal of robotics research, vol. 5, no. 1, pp. 90–98, 1986.

[16] S. M. LaValle, "Rapidly-exploring random trees a new tool for path planning," Citeseer, 1998.

[17] M. H. Overmars et al., A random approach to motion planning. Citeseer, 1992.

[18] B.-A. Dang-Vu, O. Porges, and M. A. Roa, "Interpreting primitive manipulation actions: From language to execution," in Robot 2015: Second Iberian Robotics Conference. Springer, 2016, pp. 175–187.

[19] H. Hüttenrauch, K. S. Eklundh, A. Green, E. Topp et al., "Investigating spatial relationships in human-robot interaction," in 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2006, pp. 5052–5059.

[20] B. Brüderlin and D. Roller, Geometric constraint solving and applications. Springer Science & Business Media, 2012.

[21] J. J. Abbott, P. Marayong, and A. M. Okamura, "Haptic virtual fixtures for robot-assisted manipulation," in Robotics research. Springer, 2007, pp. 49–64.

[22] L. Belouaer, M. Bouzid, and A.-I. Mouaddib, "Spatial knowledge in planning language," in International Conference on Knowledge Engineering and Ontology Development, 2011.

[23] D. Hernandez, Qualitative representation of spatial knowledge. Springer Science & Business Media, 1994, vol. 804.