Ontological Physics-based Motion Planning for Manipulation

Muhayyuddin, Aliakbar Akbari and Jan Rosell
Institute of Industrial and Control Engineering (IOC)
Universitat Politcnica de Catalunya (UPC) – Barcelona Tech
Barcelona, Spain, jan.rosell@upc.edu

Abstract—**Robotic manipulation involves actions where contacts occur between the robot and the objects. In this scope, the availability of physics-based engines allows motion planners to comprise dynamics between rigid bodies, which is necessary for planning this type of actions. However, physics-based motion planning is computationally intensive due to the high dimensionality of the state space and the need to work with a low integration step to find accurate solutions. On the other hand, manipulation actions change the environment and conditions further actions and motions. To cope with this issue, the representation of manipulation actions using ontologies enables a semantic-based inference processe that alleviates the computational cost of motion planning. This paper proposes a manipulation planning framework where physics-based motion planning is enhanced with ontological knowledge representation and reasoning. The proposal has been implemented and is illustrated and validated with a simple example. Its use in grasping tasks in cluttered environments is currently under development.**

Index Terms—**Physics-based motion planning, Dynamic simulations, Knowledge-based reasoning, Manipulation.**

I. INTRODUCTION

Motion planning problems deal with computing the collision-free trajectory from a start to a goal state in the configuration space, that is the set of all possible configurations of the robot [1]. During the last decade, the field of motion planning has evolved from the basic geometric problems (such as the piano mover's problem) to the kinodynamic motion planning problems for complex robotic systems [2]. The kinodynamic motion planning algorithms consider both the kinematic (geometric) and the dynamic (differential) constraints while planning. The most significant class of motion planning methods are those based on the sampling of the configuration space, such as the *Rapidly-exploring Random Trees* (RRTs [3]) and *Kinodynamic Planning by Interior-Exterior Cell Exploration* (KPIECE [4]). Both of these methods are sampling-based kinodynamic motion planners for systems with differential constraints, and KPIECE is particularly suited for systems with complex dynamics.

Typically, all these kinodynamic algorithms focus on computing the collision-free trajectories in the configuration space.

A new class of motion planning strategies, known as physics-based, has recently emerged that focus not only on computing collision-free trajectories, but also on considering the purposeful manipulation of objects. Physics-based planning is an extension to the kinodynamic motion planing that brings up the interaction between rigid bodies in terms of considering their dynamics [5]. Unlike what it is done in collision-free motion planning, physics-based motion planning involves the complex interaction between rigid bodies and manipulation actions to achieve the desired goal [6], i.e. based on Newtonian physics the behavior of the bodies resulting from physical interactions is simulated and taken into account in the planning decisions. These algorithms implicitly use the sampling based algorithms for sampling the states. To evaluate the interaction between rigid bodies, the state propagator uses dynamic simulators like the Open Dynamic Engine (ODE [7]).

A physics-based motion planner has been proposed in [8] that uses non-deterministic tactics and skills modeled using a finite state machine (i.e. the rigid inter-body dynamical simulations are performed using a physics-based state and transition model). Their proposal is called Behavioral Kinodynamic Rapidly-exploring Random Trees and the Behavioral Kinodynamic Balance Growth Trees. A different approach proposed a physics-based temporal projection to provide, for manipulation tasks, a reasoning capability on motion planning parameters [9] (the study applied Bullet physics engine to reason about stability, visibility, and reachability). Focused in grasping, a physics-based grasp planner has been proposed that computes the motions of the objects using a physics-based pushing analysis [10]. It simultaneously contacts and moves the obstacle in a controlled way for clearing the required path (to make it computationally efficient the interactions of the robot with the objects are precomputed). In a similar line, a physics-based trajectory optimization approach is proposed in [11] that samples the straight line trajectories and evaluates the cost based on various parameters such as collision costs, clutter behavior costs, and trajectory matching costs.

None of these referred works, however, takes advantage of using reasoning on knowledge within the physics-based motion planning procedures, although knowledge-based representation techniques have usually been used as one of the significant keys to model manipulation problems, i.e. the integration of knowledge-based reasoning process with physics
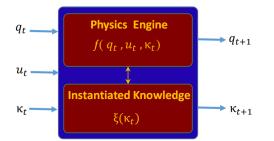
Fig. 1: State transition process for ontological physics-based planning.

engines still is an open research challenge for manipulation problems, and this paper tries to contribute in this line. Regarding knowledge representation and reasoning, the use of ontologies has emerged as a good way to describe manipulation actions and to facilitate the reasoning process in manipulation planning to provide inference capabilities from the abstract knowledge [12], [13]. With this in mind, the main purpose of this paper is to develop a framework where physics-based motion planning is enhanced with ontological knowledge and reasoning (about manipulation actions as well as about the geometry of the objects), i.e. the inferred knowledge is to be used to tune the physics-based motion planning. The proposed ontological physics-based motion planner is called the *smart motion planner*.

After this introduction the paper is structured as follows. Section II describes the problem statement and the solution overview, Sections III and IV present, respectively, the knowledge representation and the reasoning, Section V presents the framework, the planning algorithm, and the simulation results, and finally Sections VI concludes the work.

II. PROBLEM STATEMENT AND SOLUTION OVERVIEW

A. Problem statement

Consider a motion planning problem where a robot must move in an environment with obstacles that can be fixed or manipulatable (i.e. obstacles that can be pushed away). Let Q be the state space of the robot, $q_{init} \in Q$ the initial state and $Q_{goal} \in Q$ the goal region. The robot must move from q_{init} to $q_{goal} \in Q_{goal}$ avoiding collisions with fixed obstacles and, if necessary, pushing away manipulatable obstacles to clear the way to find a solution path (for instance a manipulatable obstacle may be placed at the goal region thus preventing a collision-free path to exist).

B. Modelling

We consider a workspace composed of rigid bodies, that are divided into two main categories, *fixed bodies* and *manipulatable bodies*. The former will remain static during the whole planning process, even if collisions occurs with other manipulatable objects. On the contrary, the latter can be manipulated during the planning. The manipulatable bodies are further divided into *free-manipulatable bodies* and *constraint-oriented manipulatable bodies*. Following the rigid body dynamics, the

free-manipulatable bodies can move in any direction when collisions occur. On the contrary, constraint-oriented manipulatable bodies, have the allowable motion directions restrained, i.e. they can only be moved if forces are applied in certain directions.

The manipulation constraints are modeled by defining some parts of the object from where the object can be pushed, and an associated region for each one (called manipulatable regions) where the robot must be located to exert the pushing forces. For instance, the manipulation constraints for a car-like body are defined by the forward and the backward directions as the only allowable motions directions (i.e. the body will not move if it receives a contact force on its lateral side). These constraints are modelled by two parts (the rear and the front) and their manipulatable regions.

At any time t the state s of each body is defined as:

$$s = \{p, o, v, w, \eta\}, \tag{1}$$

where p represents the position of body, o the orientation, v the linear velocity, w the angular velocity, and η describes the manipulation constraints.

Also, in order to model the problem, the knowledge about the task and the workspace is organized in two levels: a *high level* knowledge representation called abstract knowledge K, and a *low level* knowledge representation called *instantiated knowledge* κ_t. The former codes the semantic-based knowledge about the world, such as the type of bodies (i.e. fixed and manipulatable) and other features like mass or the manipulation constraints in case of constraint-oriented manipulatable bodies. This knowledge remains unchanged throughout the whole planning process. The latter, on the other hand, is a dynamic knowledge, i.e. the instantiated knowledge infers from the abstract knowledge and is continually updated by making use of the reasoning process (detailed in Section IV). For instance, if at a given instance of time the car-like body has its front manipulatable region overlapping another manipulatable object, then this region cannot be accessed by the robot and the body cannot be pushed backward. In this case the front manipulatable region is tagged as not valid in the instantiated knowledge.

Then, the state of the world at any instant of time can be described as the tuple $(\kappa_t, \mathbf{q}_t)$, with κ_t being the instantiated knowledge about the world at time t and $\mathbf{q}_t$ the corresponding states of the bodies:

$$\mathbf{q}_t = \{s_0, s_1, ..., s_n, t\}, \tag{2}$$

This state evolves with the state transition process depicted in Fig. 1, that is composed of two modules, the physics engine module and the instantiated knowledge module. Given the current state $(\kappa_t, \mathbf{q}_t)$ and the controls $\mathbf{u}_t$ to be applied, the physics engine module implements the dynamic state propagator that computes $\mathbf{q}_{t+1}$ and the instantiated knowledge module is then used to update the current instantiated knowledge:

$$\mathbf{q}_{t+1} = f(\mathbf{q}_t, \mathbf{u}_t, \kappa_t) \tag{3}$$
$$\kappa_{t+1} = \xi(\kappa_t) \tag{4}$$

This state transition process will be at the core of the physics-based planners to be used, and will only return the state whenever it is valid, i.e. the application of a control $\mathbf{u}_t$ that produces a non-allowable collision will not be accepted (collisions are not allowed with fixed obstacles nor with constraint-oriented manipulatable obstacles if the robot is not located at one of the manipulatable regions).

C. Solution overview

The problem will be solved with a physics-based motion planner enhanced with a knowledge-based reasoning process that on the one hand modifies how manipulatable objects behaves, and on the other modifies some parameters of the planner. The key points regarding κ_t and the reasoning process are:

- All fixed bodies are set as non-collisionable (the robot is not allowed to collide with them).
- All free manipulatable bodies are set as collisionable.
- Any constraint-oriented manipulatable body is set as collisionable if the robot is located at one of its manipulatable regions, otherwise it is set as non-collisionable.
- If the manipulatable region of a constraint-oriented manipulatable body is occupied, then this region is set as not valid.
- If a constraint-oriented manipulatable bodies result with no valid manipulatable regions, then the body is set as fixed.
- If the state transition process results with a not-allowed collision, then the new generated state is discarded by the planner.
- If the goal region is occupied by a constraint-oriented manipulatable body and the planner used has a sampling bias, then this sampling-bias is randomly set to one of the manipulatable regions of the body occupying the goal region, with the aim of pushing it away.

III. ONTOLOGY-BASED MANIPULATION REPRESENTATION

Ontology models have recently been used to represent robotic manipulation specifications, like manipulation actions or object descriptions and their properties. These models give robots the required knowledge to reason and to take better decisions while planning.

A. Ontology overview

Ontology models organize knowledge within specific domains. They enable a flexible access by defining things as well as their relations. Ontologies are composed of five components:

- *Classes*, also referred to as a concepts, describe types or collections of objects that share common properties.
- *Individuals*, also called instances, represent specific elements of classes, e.g., people are individuals of a person class.
- *Properties* express how classes and individuals are related to one another, e.g., sibling relationship among people.
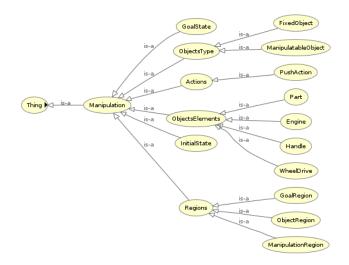

Fig. 2: Ontology-based manipulation classes.

- *Attributes* specify features, unique properties, and particular characteristics of objects, such as the age of a person.
- *Axioms* define constraints on the values of classes and individuals.

Ontologies can be encoded using the Web Ontology Language (OWL) [14]. OWL is intended to collect and organize ontology-based knowledge on a world-wide accessible database in an XML-based file format in order that multiple systems can use and share such knowledge.

Ontologies can be written using the *Protégé* editor [15]. *Protégé* is an open source platform providing an ontology editor to develop knowledge-based applications. It, moreover, can be used to portray the visualization of ontology in terms of showing relations of classes or individuals as a graph.

B. Manipulation knowledge representation

For representing a manipulation problem, it is required to specify actions, objects with their properties, initial and goal states of a robot, and valid regions. In the present study, the *Protégé* editor is employed to represent this knowledge in the OWL. Six classes, derived from a general *"Manipulation"* class have been defined (Fig. 2):

- Class *"InitialState"* contains the initial location of the robot.
- Class *"GoalState"* contains the goal location of the robot.
- Class *"Actions"* contains different manipulation actions, PickAction, PlaceAction, and PushAction, although only the latter one will be used in the present paper.
- Class *"Regions"* defines different types of regions: ManipulationRegion, ObjectRegion, and GoalRegion. The ManipulationRegions are the regions around the constrained-oriented manipulatable bodies from where the interaction between the robot and the bodies are allowed (these regions are currently defined as boxes), i.e. the robot can only interact with the bodies through the ManipulationRegions. ObjectRegion is the bounding

box of an object. The GoalRegion is a circular region defined around the goal state.

- Class *"ObjectsType"* collects the types of objects in the world: ManipulatableObject or FixedObject. If necessary, the manipulatable status of an object can be temporary changed to fixed by using the assertion process.
- Class *"ObjectsElements"* describes elements and features of objects.

IV. KNOWLEDGE-BASED REASONING PROCESS

Robots can be made more autonomous to carry out their complex tasks if some kind of reasoning process upon knowledge is provided, instead of having a purely symbolic knowledge representation. Reasoning on manipulation actions or on robot geometries provides the possibility of understanding the tasks in an analogous way a human does, e.g. of being able to answer questions like *"Which specific actions should be selected for a given type of object?"*, or *"Do I need to move some manipulatable objects to reach the goal?"*.

A. Types of reasoning process

Two types of reasoning are proposed: reasoning about the manipulation actions and about the geometry of the objects. The aim of the reasoning about the manipulation actions is to guide the motion planner to find an appropriate action in accordance with the object type. As introduced before, objects are classified into *fixed* and *manipulatable*, and those manipulatable are divided into *constraint-oriented manipulatable* and *free manipulatable*. The type of object can be predefined, although it can be identified by a reasoning process based on the parts and features of the object and on the constraints they define, since these constraints may restrict the movement of the object. For instance, if when defining a car-like object the body is attached with a wheel-drive, the allowable directions of motion are automatically constrained along the plane of the wheel and the pushing manipulation actions can then only be exerted from some parts of the object (the front and the rear). Also, some parameters like the size and the weight of objects may constrain which actions can be selected according to the capabilities of a robot.

The reasoning about the geometry of objects tackles two matters. First, it determines whether the goal region is free or occupied by other objects (using the bounding boxes of the objects and of the goal region). Second, it determines possible manipulation regions on the object (from where to apply pushing forces). To address this, a reasoning process is done by inferring over the properties of the objects.

B. Prolog-based knowledge inference process

The reasoning predicates have been defined in Prolog and been integrated within Knowrob, a knowledge processing framework for robotic systems [16]. Knowrob is a powerful reasoning tool that works over ontology models: its implementation is based on SWI Prolog and the Semantic Web library, enabling the use of Prolog predicates to fetch the knowledge
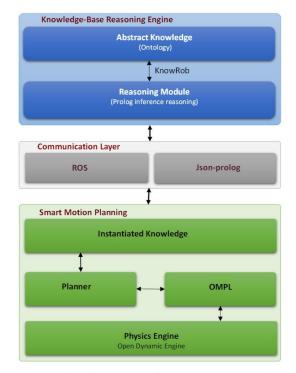

Fig. 3: Integration framework for ontological physics-based motion planning

stored inside the OWL. Therefore, Knowrob provides a flexible access to such knowledge to facilitate the reasoning process.

The following Prolog predicates have been implemented to infer manipulation knowledge from the OWL and to reason according to the states of the bodies.

- *object_classification(?Obj, ?ObjType, ?ManipType)*: Given an object *Obj* returns its type (fixed or manipulatable) in *ObjType*. In case of a manipulatable objects, its type (free or constrained) is inferred from its properties and returned in *ManipType*.
- *action_type(?Obj, ?ObjElement, ?Action)*: Given an object *Obj* returns in *ObjElement* a list of elements (parts) from where the object can be manipulated using the list of actions returned in *Action*, e.g. for the car-like object it will return the rear and the front as elements and push as the manipulation action for both.
- *determine_goal_region(?Obj)*: Returns in *Obj* the object that is occupying the goal region if any, or null otherwise.
- *valid_region(?Obj, ?ObjElement, ?ValidRegion)*: Given an object *Obj* and its part *ObjElement* returns in *ValidRegion* whether the associated manipulatable region is valid or not.

V. IMPLEMENTATION

A. Framework

The proposed framework, depicted in Fig. 3, is composed of three main modules: the *knowledge-based reasoning engine*, the *communication layer*, and the *smart motion planner*.

The *knowledge-based reasoning engine* has two main components, the *Abstract Knowledge* which represents the whole knowledge using an ontology model[1], and the *Reasoning Module* which infers ontological knowledge using Prolog predicates integrated within Knowrob.

The *smart motion planning* module consists of the *Instantiated Knowledge* module and the *Planner* module. The *Instantiated Knowledge* module contains the dynamic knowledge that describes the temporary manipulation constraints (inferred form the abstract knowledge based on the reasoning process), that are updated at each instance of time and that describes the way the bodies can be manipulated at a given particular instance of time. The *Planner* module is composed of the *The Kautham Project* [17], an open-source tool for motion planing, developed in C++, that uses the *Open Motion Planning Library (OMPL)* [18] as the core set of planning algorithms. OMPL allows planning under geometric constraints as well as under differential constraints, including those that required dynamic simulations (OMPL uses the *Open Dynamic Engine* for the dynamic simulation).

The role of the *communication* layer is to transfer information between the other two layers. It includes ROS [19] and Json-prolog (which is provided by Knowrob). The Json-prolog library enables the access to Prolog predicates through the ROS communication protocols, thus allowing to encapsulate the knowledge-based reasoning process as a ROS service node that can be called from a ROS client node containing the motion planner. In this way, it is possible to integrate the Prolog-based reasoning process within the motion planner.

B. Planning algorithm

Algorithm 1 Ontological Physics-Based Motion Planning.

Input: Initial state q_{init}, Goal region $Q_{\text{goal}} \in \mathcal{C}$, Threshold T_{max}
Output: A path from q_{init} to $q \in Q_{\text{goal}}$
1: ReadTheWorld()
2: K ← FormulateOntology()
3: **while** $t < T_{max}$ **do**
4:     κ ← KnowledgeReasoner(K, κ)
5:     SelectNodeToExpand()
6:     u ← SampleControls()
7:     q_{new} ← Propagate(u, κ)
8:     **if** Valid(q_{new}) **then**
9:         UpdateConnections()
10:        **if** $q_{\text{new}} \in Q_{\text{goal}}$ **then**
11:            **return** Path(q_{new})
12:        **end if**
13:    **end if**
14: **end while**
15: **return** NULL

The ontological physics-based planning procedure is sketched in Algorithm 1. It takes as input the initial configuration q_{init}, the goal region Q_{goal} and a time limit T_{max}, and returns a path from q_{init} to $q_{goal} \in Q_{goal}$, if found, or NULL otherwise. The planning is performed by any of the OMPL control planners, like RRT, KPIECE, or PDST [20], tuned to

[1]OWL files are accessible at: https://sir.upc.edu/projects/ontologies/.

use ODE as state propagator. The instantiated knowledge is updated at each iteration and conditions the behavior of the bodies in the ODE world.

The steps of the algorithm are the following:

- *ReadTheWorld*: Sets the initial state of the robot and the environment by reading the p, o, v, w, and η of each body.
- *FormulateOntology*: Extracts the abstract knowledge K about the world, such as the type of the bodies, the manipulation constraints and the geometrical positions of bodies and determines whether the GoalRegion is occupied or not. Further it computes the Manipulation-Region for each constraint-oriented manipulatable object. All these attributes are stored in the form of abstract knowledge.
- *KnowledgeReasoner*: Evaluates the state of the world and updates the instantiated knowledge κ. For instance, if the GoalRegion is occupied by an object, then the sampling of the planner will be biased towards one of the ManipulationRegion of the object occupying it (instead of the standard bias towards the goal). When the robot enters in the ManipulationRegion, the standard bias is reset and the instantiated knowledge is updated with the manipulation constraints of that object.
- *SelectNodeToExpand*: Selects the node to expand the tree data structure of the planner.
- *SampleControls*: Samples the controls within the given range.
- *Propagate*: Applies the controls to the bodies (controls can be applied by applying a force, a torque or by setting a linear and angular velocity), while applying the controls, the state propagator will take into account the manipulation constraints provided by the instantiated knowledge, and returns the new generated state.
- *Valid*: Returns true if no forbidden collision has occurred during the generation of the new state, and false otherwise.
- *UpdateConnections*: Updates the tree data structure by adding the edge between the previous and the newly generated state.
- *Path(q)*: Returns a path from q_{init} to $q \in Q_{goal}$ by backtracking along the planner data strcuture.

C. Simulation Results

A simple scenario has been set in order to put the focus on how the planner makes use of the knowledge to manage a complex situation involving a manipulatable obstacle. It consists of: a) the walls of a room which are defined as fixed bodies; b) a spherical shaped robot with two degrees of freedom; and c) a car-like obstacle which is defined as a constraint-oriented manipulatable body that can only be moved in the forward and backward directions. The goal region Q_{goal} is occupied by the car-like obstacle, so no collision-free path exists, i.e. the robot needs to push the car-like obstacle away in order to reach Q_{goal}.
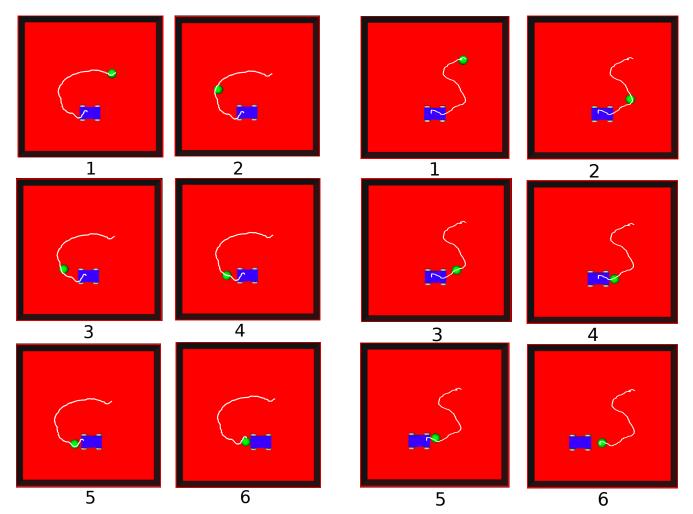
Fig. 4: Simulation results of the ontological physics-based motion planner using KPIECE.

Fig. 5: Simulation results of the ontological physics-based motion planner using RRT.

The knowledge-based reasoning engine reads the initial state of the world and extracts the abstract knowledge related to the world such as the type of bodies, the manipulation constraints, and the state of the goal. Based on a reasoning process, the instantiated knowledge is inferred from the abstract knowledge and fills the data structures in the motion planning layer (such as the current manipulation constraints, the valid manipulation regions, the goal state - occupied or free -, etc.), that are periodically updated. In this example, the knowledge-based reasoning engine determines that the goal is occupied and extracts the manipulation constraints associated to the car-like obstacle (they have been defined as two ManipulationRegions corresponding to its front and rear parts, since contact is allowed with the front and the rear sides and is forbidden with the two lateral sides).

The planning of paths has been performed using two different physics-based kinodynamic motion planners, KPIECE and RRT. At each instance of time, the instantiated knowledge evaluates the state of the world and the manipulation constraints, changing the standard sampling bias towards the goal by a bias towards one of the ManipulationRegions of the car-like obstacle that occupies the GoalRegion. Once the robot reaches the ManipulationRegion, the standard bias towards the goal is restored. Fig. 4 and 5 show, respectively, sequences of snapshots of the solutions found using KPIECE and RRT. In both cases, it can be appreciated how the robot moves towards one of the ManipulationRegions and then hits the car-like obstacle at its front/rear side and pushes it to free the GoalRegion and reach the goal.

Without using the aid provided by the instantiated knowledge, these planners were not able to find good solutions, as illustrated in Fig. 6. Since in this case the planner did not know how to manipulate the car-like object, many tree edges tried unfruitfully to grow towards the goal by hitting the car at its side (in the solution shown the robot hits the car at the wheel and bounces back). Therefore, the computational cost was greater (it took approximately twice the time used by the ontological physics-based planners, either with KPIECE or RRT). The configuration spaces for all the above stated executions are shown in Fig. 7.
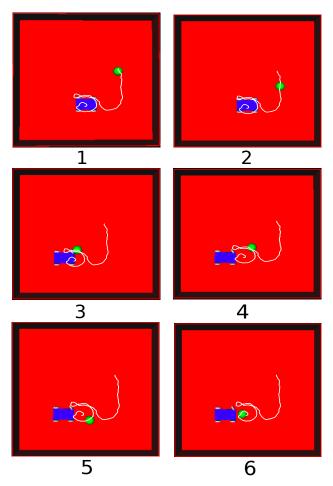
Fig. 6: Simulation results using KPIECE without using instantiated knowledge.
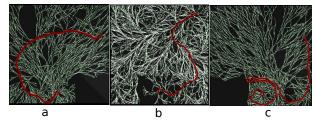

Fig. 7: Configuration spaces with the tree structures of the planners used and the corresponding solution paths (in red): a) KPIECE using instantiated knowledge; b) RRT using instantiated knowledge; c) KPIECE without using instantiated knowledge.

VI. CONCLUSIONS AND FUTURE WORK

The present study has explored the integration of knowledge-based reasoning with physics-based motion planning. A framework has been presented to combine both phases, aiming to enhance the planning process for manipulation problems. Two types of reasoning (about manipulation actions and about the geometry of objects) is considered. The result

of the inference process is stored inside the instantiated knowledge and provided to the motion planning. The proposed algorithm has been illustrated with a manipulation problem where the goal region is occupied by a manipulatable obstacle that should be removed by the robot using pushing actions. The results described show that the proposed algorithm has a better performance than the simple physics-based planning.

Future work will be directed towards the integration of physics-based reasoning with a task planner, and the consideration of other manipulation actions besides pushing, like pick and place actions.

REFERENCES

[1] T. Lozano-Pérez, "Spatial Planning: A Configuration Space Approach." *IEEE Trans. on Computers*, vol. 32, no. 2, pp. 108–120, 1983.
[2] I. A. Şucan and L. E. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *IEEE Trans. on Robotics*, vol. 28, no. 1, pp. 116–131, 2012.
[3] S. M. Lavalle and J. J. Kuffner, "Rapidly-Exploring Random Trees: Progress and Prospects," B. R. Donald, K. M. Lynch, and D. Rus, Eds. A K Peters, 2001, pp. 293–308.
[4] I. A. Şucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Algorithmic Foundation of Robotics VIII*. Springer, 2010, pp. 449–464.
[5] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM*, vol. 40, no. 5, pp. 1048–1066, 1993.
[6] S. Zickler and M. M. Veloso, "Variable level-of-detail motion planning in environments with poorly predictable bodies." in *Proc. of the European Conf. on Artificial Intelligence Montpellier*, 2010, pp. 189–194.
[7] S. Russell, "Open Dynamic Engine," http://www.ode.org/, 2007.
[8] S. Zickler and M. Veloso, "Efficient physics-based planning: sampling search via non-deterministic tactics and skills," in *Proc. of The 8th Int. Conf. on Autonomous Agents and Multiagent Systems-Volume 1*, 2009, pp. 27–33.
[9] L. Mosenlechner and M. Beetz, "Fast temporal projection using accurate physics-based geometric reasoning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2013, pp. 1821–1827.
[10] M. R. Dogar, K. Hsiao, M. Ciocarlie, and S. Srinivasa, "Physics-based grasp planning through clutter," in *Robotics: Science and Systems*, 2012.
[11] N. Kitaev, I. Mordatch, S. Patil, and P. Abbeel, "Physics-based trajectory optimization for grasping in cluttered environments," in *Proc, of the IEEE Int. Conf. on Robotics and Automation*, 2015, pp. 3102 – 3109.
[12] S. Feyzabadi and S. Carpin, "Knowledge and data representation for motion planning in dynamic environments," in *Robot Intelligence Technology and Applications 2*. Springer, 2014, pp. 233–240.
[13] A. Akbari, Muhayyudin, and J. Rosell, "Task and motion planning using physics-based reasoning," in *Proc. of the IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2015.
[14] G. Antoniou and F. van Harmelen, "Web Ontology Language: OWL," in *Handbook on Ontologies in Information Systems*, S. Staab and R. Studer, Eds. Springer-Verlag, 2003, pp. 67–92.
[15] Stanford2007, "Protégé," http://protege.stanford.edu/, 2007.
[16] M. Tenorth and M. Beetz, "Knowrob knowledge processing for autonomous personal robots," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009, pp. 4261–4266.
[17] J. Rosell, A. Pérez, A. Aliakbar, Muhayyuddin, L. Palomo, and N. García, "The kautham project: A teaching and research tool for robot motion planning," in *Proc. of the IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2014.
[18] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, no. 4, pp. 72–82, December.
[19] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2, 2009, p. 5.
[20] A. Ladd and L. Kavraki, "Fast tree-based exploration of state space for robots with dynamics," in *Algorithmic Foundations of Robotics VI*. Springer, 2005, pp. 297–312.