J Intell Robot Syst
https://doi.org/10.1007/s10846-017-0698-z

κ-PMP: Enhancing Physics-based Motion Planners with Knowledge-Based Reasoning

Muhayyuddin[1] · **Aliakbar Akbari**[1] · **Jan Rosell**[1]

Received: 13 March 2017 / Accepted: 8 August 2017

Abstract Physics-based motion planning is a challenging task, since it requires the computation of the robot motions while allowing possible interactions with (some of) the obstacles in the environment. Kinodynamic motion planners equipped with a dynamic engine acting as state propagator are usually used for that purpose. The difficulties arise in the setting of the adequate forces for the interactions and because these interactions may change the pose of the manipulatable obstacles, thus either facilitating or preventing the finding of a solution path. The use of knowledge can alleviate the stated difficulties. This paper proposes the use of an enhanced state propagator composed of a dynamic engine and a low-level geometric reasoning process that is used to determine how to interact with the objects, i.e. from where and with which forces. The proposal, called *κ-PMP* can be used with any kinodynamic planner, thus giving rise to e.g. *κ-RRT*. The approach also includes a preprocessing step that infers from a semantic abstract knowledge described in terms of an ontology the manipulation knowledge required by the reasoning process. The proposed approach has been validated with several examples involving an holonomic mobile robot, a robot with differential constraints and a serial manipulator, and benchmarked using several state-of-the art kinodynamic planners. The results showed a significant difference in the power consumption with respect to simple physics-based planning, an improvement in the success rate and in the quality of the solution paths.

Keywords Physics-based motion planning · Kinodynamic motion planning · Knowledge-based reasoning

1 Introduction

Motion planning is one of the important issues in robotics, either as a stand-alone problem or in conjunction with other tasks such as grasping or manipulation. In the past decades, the focus of many motion planning approaches has often been to compute collision-free paths in the configuration space $\mathcal{C}$ (the set of all possible configurations of the robot) while satisfying some geometric constraints. Approaches like grid-based methods or potential fields were first proposed for such planning [17]. Although being practical for simple scenarios, these algorithms were computationally intensive and difficult to implement in higher dimensional configuration spaces. Moreover, some problems arouse when executing the computed geometric path in the real robot due to the possible existence of kinematic and dynamic constraints. Therefore, new motion planning algorithms appeared to cope with these problems [6, 18].

Sampling-based motion planning algorithms (such as RRT [19]) were proposed to plan in higher dimensional configuration spaces, since these algorithms do not require

This work was partially supported by the Spanish Government through the projects DPI2013-40882-P, DPI2014-57757-R and DPI2016-80077-R. Muhayyuddin is supported by the Generalitat de Catalunya through the grant FI-DGR 2014. Aliakbar Akbari is supported by the Spanish Government through the grant FPI 2015.

✉ Muhayyuddin
muhayyuddin.gillani@upc.edu

Aliakbar Akbari
aliakbar.akbari@upc.edu

Jan Rosell
jan.rosell@upc.edu

[1] Institute of Industrial and Control Engineering (IOC), Universitat Politècnica de Catalunya (UPC) – Barcelona Tech, Barcelona, Spain

Published online: 22 September 2017

the explicit representation of the obstacles in $\mathcal{C}$ and showed to comply very well to problems with kinodynamic constraints. In this line, the approaches such as Kinodynamic Motion Planning by Interior-Exterior Cell Exploration (KPIECE [29]) were recently proposed to plan efficiently for systems with complex dynamics in high dimensional configuration spaces. Moreover, these approaches can also take into account the physics-based constraints such as friction and gravity along with the kinodynamic constraints.

The possibility of taking into account physics-based constraints has also made it possible to consider not only the search of collision-free paths but also the search of paths where the interaction between the robot and the environment is possible. This new class of planning algorithms is known as physics-based motion planning, and can be considered as an extension to kinodynamic motion planning. Physics-based motion planners evaluate the interactions between bodies based on the principle of basic physics and their results influence the planning process. Therefore, planning becomes a challenging task due to various factors such as the high dimensionality of the state space (where the environment may change as a result of the bodies interactions), the large planning search space and the possibly highly constrained solution set. Moreover, the accurate modeling of the objects interactions with dynamical effects such as friction and momentum is required. To alleviate these challenging issues, a few approaches have been proposed that develop strategies to reduce the search space, e.g. [21, 33, 34].

Since physics-based motion planning deals with the kinodynamic and physics-based constraints along with the dynamic interactions between rigid bodies, it is desirable to compute an efficient solution in terms of dynamics measures, such as power consumption, instead of just considering the planning time or path length, as usually done in other motion planning problems (this is particularly true for task and motion planning problems where dynamic cost computed by the motion planner significantly influence the planning decision at task level [1, 2]). To the best of our knowledge, there is not any approach within the framework of physics-based planning that searches for power-efficient motion plans.

With the physics-based motion planning in mind, one of the problems that can be stated is the following. Find a path for a robot (either a mobile or a manipulator) to move from an initial state to a goal state while interacting, if necessary, with the obstacles of the environment, obtaining a power-efficient solution that satisfies the constraints (regarding which obstacles can be collided, from where and with which interacting force). This paper aims to find a solution to this problem statement by proposing a power-efficient approach for physics-based motion planning based on the use of manipulation knowledge. Compared with plain physics-based motion planners, the proposal results in an improvement in the success rate and in the quality of the solution paths. A preliminary version of this approach was presented in [21], where the focus was on the introduction of a knowledge-based reasoning process for an efficient planning (in terms of time). This paper extends the approach for a power-efficient solution by proposing an improved framework and a new planning algorithm. Moreover, a detailed comparison of the approach with other approaches is presented in a variety of different scenarios.

Contributions The main contribution of this work is the proposal of a physics-based motion planning method equipped with a control sampling strategy that allows the search of a power-efficient motion plan, which may include free robot motions along with interactions with manipulatable objects that may be obstructing the path. The main components of the proposal are: *(1)* the partition of $\mathcal{C}$ into different regions as a function of whether the robot can either move freely or interact with manipulatable objects; *(2)* an instantaneous reasoning process that performs low-level geometric reasoning in order to analyze the configuration space regions and update the sampling range; *(3)* a detailed representation of knowledge, which is categorized into semantic knowledge and manipulation knowledge, to help improving the knowledge-based inference process. The proposed framework consists of a high-level and a low-level layer, that are connected through a ROS-based communication layer. The high-level layer contains the knowledge about the robot and the environment, that is used by the low-level motion planner for reasoning about the sampled controls and to update the manipulation constraints. This hierarchical structure results in smart motion plans for the robot to efficiently interact with the objects in the environment. The performance of the proposed planning approach is tested with three different scenarios: an holonomic mobile robot, a car-like mobile robot and a planar manipulator. The results are compared (in terms of power consumption, planning time and success rate) with other physics-based planning approaches.

The paper is structured as follows. First, Section 2 details some relevant related work and Section 3 formulates the problem and sketches the solution. Afterwards, Section 4 explains the high and low level representation of knowledge, Section 5 details the high level knowledge inference process and the low level reasoning process and Section 6 explains the framework and the proposed planning algorithm. Finally, Section 7 describes the obtained results and Section 8 concludes the study.

2 Related Work

The simplest form of motion planning is a geometric problem devoted to compute a collision-free path from a start to

a goal state in the configuration space while satisfying some geometric constraints like joint limits and collision avoidance. Sampling-based motion planners such as RRTs and PRMs [15] are able to solve problems in high dimensional configuration spaces, by connecting collision-free samples forming a graph or a tree-like structure that capture the connectivity of the free configuration space, or of the part of the free space relevant to the query to be solved. In some cases the kinematic and dynamic constraints of the robot must be taken into account while planning due to the difficulty that may arise in the following of a geometric path. This need gave rise to kinodynamic motion planners.

2.1 Kinodynamic Motion Planning

Physical systems may be subject to kinematic constraints that may be holonomic or nonholonomic. The former are constraints on system configurations, $q_0, q_1, ..., q_n$, and can be expressed as $f(q_0, q_1, ..., q_n; t) = 0$, i.e. they only depend on the coordinates and time, whereas the latter cannot be expressed in this form; they are constraints on velocities. Moreover, the system may be subject to dynamic constraints due to the dynamics laws and bounds on velocities, accelerations and applied forces. In robotics, the term kinodynamic planning was introduced by [10] to refer to motion planning problems where both kinematic and dynamic constraints were considered, i.e. a motion planning approach devoted to the search of a solution path that complies to the kinematic and dynamic laws and the bounds over the applied forces, the velocities and the accelerations. Basically, these kinodynamic algorithms search a solution in a higher dimensional state space $\mathcal{X}$ that records the dynamics of the system. For any particular configuration $q \in \mathcal{C}$, the state of the system is represented as $x = (q, \dot{q}) \in \mathcal{X}$, and the state propagation is performed by a transition function defined as:

$$\dot{x} = f(x, u) \tag{1}$$

with $u \in U$, the set of valid control inputs. The solution to a kinodynamic problem is found by determining the set of appropriate control inputs that applied using Eq. 1 brings the robot from the initial to the goal state while satisfying all the constraints.

Sampling-based motion planners (particularly those using tree-like structures) have the ability to efficiently plan in the presence of kinodynamic constraints [5, 30]. These planners can be divided into three main categories:

1. Planners that sample the states, such as RRTs and Expansive-Spaces Tree planners (EST) [13, 14]. The RRT grows a tree rooted at the start state by iteratively selecting a random sample x_{rand} and expanding the tree from the node that is nearest to x_{rand} by applying a randomly sampled control. The EST builds a tree-like roadmap by selecting a node with a probability inversely proportional to the density of the node neighborhood and extending it by applying a randomly sampled control.
2. Planners that sample path segments instead of states and that do not require the use of a metrics, such as the Path Directed Subdivision Tree (PDST) [16] and KPIECE [30]. The PDST planner subdivides the state space into regions (cells), each one containing a path segment. The tree is extended by iteratively sampling a cell, randomly selecting a state from that cell, and applying a randomly sampled control to generate the new path-segment.
   The KPIECE planner also samples the path segments (called motions) by using a grid-based decomposition of a projection space (defined either with random projections or user-defined) where the tree of motions is projected and where a sampling procedure is defined to select the important regions to explore.
3. Hybrid planners such as Synergistic Combination of Layers of Planning (SyCLoP) [24] and the Linear Temporal Logic (LTL) motion planner [4, 23]. The SyCLoP planner splits the planning problem into a discrete (high-level) layer and a continuous (low-level) layer of planning. The former is based on the decomposition of the workspace, whereas the latter consists of a sampling-based motion planner like EST or RRT that is guided by the discrete layer. LTL is an extension of the SyCLoP planer in which the discrete layer encodes a complex motion planning task using an abstract graph computed from a decomposition of the workspace and an automaton that represents a linear temporal logic formula describing the task.

In all the above stated planners the control sampling range is usually set at the start and remains the same in the entire planning process; on each state the controls are randomly sampled from the given control range that results in the robot motion. Beside sampling-based algorithms there are some other recently proposed approaches for kinodynamic motion planning, such as the Covariant Hamiltonian Optimization for Motion Planning (CHOMP [35]). These approaches mainly focus on the optimization (such as smoothness) but can be used as stand alone motion planners for computing collision-free trajectories.

2.2 Physics-Based Motion Planning

All the above stated kinodynamic motion planning strategies are focused on computing a collision-free trajectory from start to the goal state, i.e. interactions with the objects in the environment are forbidden (and the contacts of a

mobile robot with the floor not considered). This constraint leads to the neglecting of the physics-based dynamic features such as friction between the objects and the ground, pressure distribution under the objects surfaces, gravitational effect over the objects in the environment and the interaction dynamics (such as direction of interaction force and momentum). If interactions are allowed, however, all these features should be considered, and this is what physics-based motion planning does, and the key difference with respect to the kinodynamic and other motion planning approaches. Therefore, physics-based motion planning has recently emerged as a step further toward physical realism. It simultaneously considers the kinodynamic constraints and physics-based constraints (such as friction and gravity), and also incorporates the purposeful manipulation of objects by evaluating the dynamic interactions between rigid bodies simulated using the basic physics (rigid-body dynamics).

Physics-based planning approaches implicitly use a sampling-based kinodynamic planner, that is responsible for sampling the states and constructing the solution path, but using for the propagation step a rigid-body dynamic simulator (dynamic engine), such as ODE [26] or Bullet [9]. The dynamic engine models the dynamical world with all the physical properties and has the ability to simulate the properties of the physical interactions, such as force-based inter-body collision and momentum. The physics-based planner evaluates the results of the action after propagation, if it satisfies all the constraints then the action is selected, and discarded otherwise.

The complexity of the physics-based motion planning is very high due to the high-dimensional state space, large search space and highly constrained solution set. A few physics-based motion planning approaches have been proposed that addressed the above mentioned issues, such as the Behavioral Kinodynamic Balanced Growth Trees (BK-BGT) and the Behavioral Kinodynamic Rapidly-Exploring Random Trees (BK-RRT) proposed in [33] that use a strategy to reduce the search space based on a nondeterministic tactic modeled using a finite state machine, along with skills used to control the sampling. The propagation step is performed using PhysX [22]. A hybrid approach is proposed in [21] that equips the physics-based motion planner with knowledge (in the form of ontologies) about the robot's manipulation world. It uses a knowledge-based reasoning process to reduce the robot search space and guide the motion planner by defining the way objects can be manipulated. It uses RRT and KPIECE as kinodynamic motion planner and ODE as state propagator. This approach has also been used in task planning approaches [1, 2] for the physics-based reasoning process to determine the feasibility of the plan by evaluating the dynamic cost of each subaction in the task plan.

Some other approaches address problem related to physics-based motion planning such as physics-based grasping and rearrangement planning [7, 8]. These approaches evaluate the dynamic interaction by executing the straight line trajectories under the quasi static assumption. Moreover some approaches (such as [12, 28]) studied the rearrangement planing in conjunction with the physics-based motion planning, but none of them addressed the issue of robust control selection for the power efficient solution.

The determination of the appropriate set of controls and durations such that, if sequentially applied, the robot moves from an initial to a goal state following a power-efficient trajectory is a challenge. With this aim, the current proposal presents a control sampler that uses a knowledge-based reasoning process to determine the appropriate values of controls that may result in this behavior. The next section will formally describe the problem statement and outline the proposed solution.

3 Problem Formulation and Solution Overview

3.1 Problem Statement

Let a physics-based motion planning problem be defined as the tuple $(\mathcal{X}, \mathcal{U}, f, \mathcal{K}, \mathcal{F}, x_{init}, \mathcal{X}_{goal})$, where:

- $\mathcal{X}$ represents the state space; it is a differential manifold.
- $\mathcal{U}$ represents the control space; it contains the set of all possible control inputs that can be applied to the robot.
- $f : \mathcal{X}^i \times \mathcal{U} \longrightarrow \mathcal{X}^{i+1}$ is the propagation function.
- $\mathcal{K}$ is the abstract knowledge containing all the available knowledge about the world such as object classification, manipulation regions, and physical properties. $\kappa \subset \mathcal{K}$ is the instantiated knowledge that represents the knowledge that is valid for a particular instance of time.
- $\mathcal{F} : \kappa \times \mathcal{X} \longrightarrow \{0, 1\}$ is the physics-based state validity checker. It evaluates the state generated by applying f, and returns 1 if it satisfies all the constraints imposed by κ, or returns 0 otherwise.
- $x_{init} \in \mathcal{X}$ is the initial state.
- $\mathcal{X}_{goal} \subset \mathcal{X}$ is the goal region.

Consider a motion planning problem where no collision free trajectory from start to the goal exist (e.g. either the goal or the way toward it might be occupied with manipulatable objects). The objective is to determine the set of efficient control inputs $\{u_1, \ldots, u_n\} \in \mathcal{U}$ and the set of associated time durations $\{t_1, \ldots, t_n\}$ such that, if sequentially applied to the system using f and starting from x_{init}, a goal state $x_n \in \mathcal{X}_{goal}$ is reached, being the resultant trajectory power-efficient and satisfying all the constraints (i.e. a trajectory that avoids collisions with fixed obstacles but that may collide with manipulatable objects to push them away from the solution path). The proposed approach (at each step) will determine the robust control using the low level reasoning

about the object dynamics, in such a way that the resultant solution will be power efficient (no optimization of any kind such as path length will be considered).

3.2 Problem Modeling

We consider a dynamical workspace with several objects (each one composed of one or more rigid bodies), that are categorized into *fixed objects* and *manipulatable objects*. Fixed objects, such as walls, remain static during the entire planning process and no collision is allowed with them. Manipulatable objects, on the contrary, can be pushed and hence their pose is not fixed. Manipulatable objects are further categorized into *constraint-oriented manipulatable objects* (*co-mObjects*) and *free-manipulatable objects* (*free-mObjects*), depending on whether they are subject to some kinematic constraints or not (e.g. a car-like object can only be pushed forwards or backwards in a single direction). All manipulatable objects have associated regions, called manipulation regions (*mRegions*), from where the robot can exert forces in order to move them, i.e. the robot can interact with the object through these regions and it is not allowed to contact the object from any other part. For instance, as shown in Fig. 1, a car-like robot has one *mRegion* located at its front and one at its rear, and a vertical box has the *mRegions* around it but below its center of mass.

The state s of each object is represented by its position r, orientation o, linear velocity v, angular velocity w, and the associated manipulation constraints η:

$$s = \{r, o, v, w, \eta\} \tag{2}$$

The state of the workspace (composed of n objects) at a given instant of time t is represented as:

$$\mathbf{q}_t = \{s_1, s_2 \ldots, s_n, t\} \tag{3}$$

Regarding the robot, its configuration space $\mathcal{C}$, i.e. the set of all possible configurations of the robot, can be divided into the set of geometrically accessible regions, called $\mathcal{C}_{\text{free}}$, and the set of the forbidden ones (those corresponding to collision with obstacles), called $\mathcal{C}_{\text{obs}}$. The condition $\mathcal{C}_{\text{free}} \cup \mathcal{C}_{\text{obs}} = \mathcal{C}$ holds. In this work, $\mathcal{C}_{\text{free}}$ is further divided into $\mathcal{C}_{\text{move}}$ and $\mathcal{C}_{\text{interaction}}$ representing, respectively,

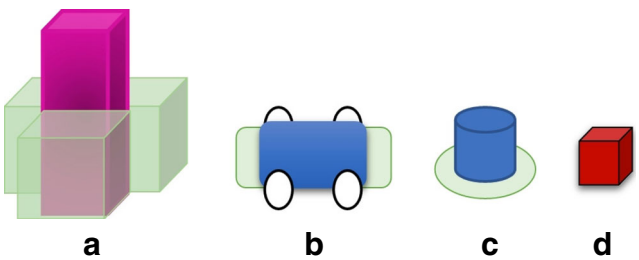

Fig. 1 Different bodies with their *mRegions* (the red cube represents the fixed object and hence have no associated *mRegion*)

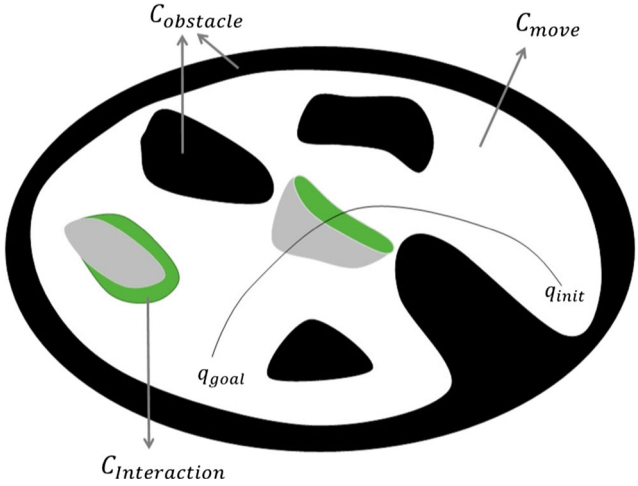

Fig. 2 Configuration space: $\mathcal{C}_{\text{obs}}$ shown in black (collisions with fixed obstacles) and gray (collision with manipulatable obstacles), $\mathcal{C}_{\text{interaction}}$ in green and $\mathcal{C}_{\text{move}}$ in white

the regions where the robot can move freely and those where the robot can enter in contact with the manipulatable objects, i.e. the set of configurations corresponding to the robot[1] being placed in an *mRegion* (see Fig. 2). The condition $\mathcal{C}_{\text{move}} \cup \mathcal{C}_{\text{interaction}} = \mathcal{C}_{\text{free}}$ holds.

3.3 Solution Overview

In order to find a power-efficient solution, we have developed an approach inspired by our daily life experience where, in order to perform the task robustly, we dynamically update the forces according to the interaction with the environment, e.g. we do not exert the same force while pushing a plate, a table, or when moving freely. We propose the use of any kinodynamic planner, such as KPIECE or RRT, and to equip it with a state transition model that computes the next state based on a dynamic engine and a reasoning process that uses instantiated knowledge. This knowledge defines from where objects should be manipulated and with which range of forces. The knowledge representation used is detailed in Section 4 and the knowledge inference and reasoning process in Section 5.

The knowledge about the task and the workspace is modeled in two levels:

– The *abstract knowledge* $\mathcal{K}$: It is a high-level representation of knowledge composed of the *semantic knowledge* $\mathcal{K}_{\text{S}}$ and the *manipulation knowledge* $\mathcal{K}_{\text{M}}$. The semantic knowledge describes, using ontologies, information of the task such as the kinematic and dynamic properties of the robot, of the objects, and the manipulation constraints. From $\mathcal{K}_{\text{S}}$ the manipulation knowledge $\mathcal{K}_{\text{M}}$

[1] In the case of manipulators the pose of the tool will be considered.

is inferred. It contains all the necessary information that is required for the motion planner, such as the type of objects and how they can be manipulated. Abstract knowledge remains the same during the whole planning process.

– The *instantiated knowledge* κ: It is a low-level representation, updated at each instance of time by the reasoning process, based on the manipulation knowledge and on the feedback received from the motion planner (e.g. if at a particular instance of time one of the *mRegion* of an *mObject* is occupied by some other object, then if there is an *mRegion* in the opposite side, it will be deactivated by the reasoning process and κ will be updated accordingly). The instantiated knowledge is used in the state transition model as explained next.

The dynamic flow of knowledge is shown in Fig. 3, where it is illustrated that the manipulation knowledge is extracted from semantic knowledge (ontologies) and used by the reasoning process along with the feedback from the motion planner to update the instantiated knowledge. This instantiated knowledge, as shown in Fig. 4, is the key module of the state transition process, that takes the state of the world $\mathbf{q}_t$ as input and generates the next state $\mathbf{q}_{t+1}$ by applying an appropriate control based on this knowledge:

$$\kappa_t = \xi(\mathcal{K}_M, \mathbf{q}_t) \tag{4}$$

$$\mathbf{q}_{t+1} = f(\mathbf{q}_t, \mathbf{u}_t(\kappa_t)) \tag{5}$$

That is, the state transition process consists of three modules: the physics engine, the instantiated knowledge, and the low-level reasoning. The reasoning process is responsible for updating the instantiated knowledge κ with the manipulation constraints (determining which are the active manipulation regions) and with the region of the configuration space where the robot is located (either $\mathcal{C}_{\text{move}}$ or $\mathcal{C}_{\text{interaction}}$). With this information the instantiated knowledge determines the set of controls and selects one to be applied (the set will be determined in such a way that if the robot is in a region where interaction with an object is possible then the range of controls must allow the manipulation of the object). Finally, the physics engine (i.e. physics-based state propagator) generates $\mathbf{q}_{t+1}$ by applying the randomly sampled control $\mathbf{u}_t$ to $\mathbf{q}_t$. The validity of the resulting state will be checked using the physics-based state validity checker $\mathcal{F}$ that takes into account the instantiated knowledge (i.e. a collision state is only valid if the robot collides with manipulatable object from one of its manipulation regions).

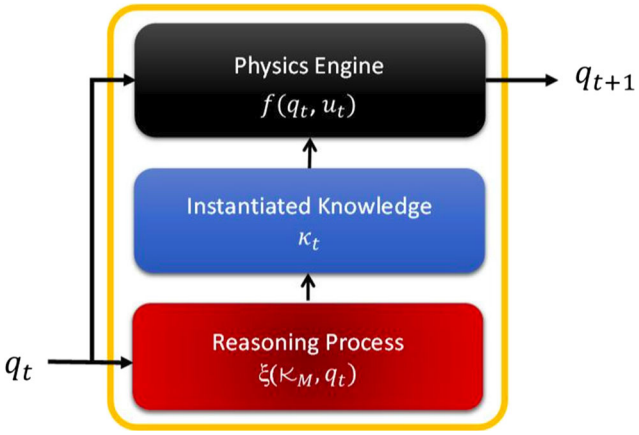

Fig. 4 State transition model for the κ-*PMP*

The preliminary work in [21] demonstrated the importance of using knowledge (described by ontologies) to guide physics-based motion planning. The present proposal greatly enhances the state transition model, which is the core of the proposal, by introducing the low-level geometric reasoning process that allows the use of an adaptive control range, thus obtaining power-efficient solutions. On the other hand, it reduces the computational cost by introducing the offline inference process that prevents the online query to the ontological knowledge.

4 Knowledge Representation

This section presents the proposal done to represent and manage knowledge for motion planning purposes, from the abstract knowledge $\mathcal{K}_S$ using ontologies to the instantiated knowledge κ used at the motion planning level, through the manipulation knowledge $\mathcal{K}_M$ inferred from $\mathcal{K}_S$ and used to maintain κ.

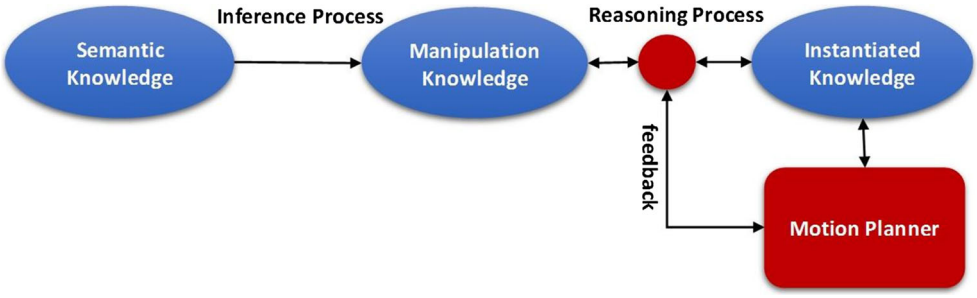

Fig. 3 Flow of knowledge from the high-level abstract knowledge to the low-level instantiated knowledge

4.1 Representation of Knowledge Using Ontologies

Ontologies can be employed to model and organize knowledge within different domains. In particular, they have recently been used to organize knowledge for motion and manipulation planning (e.g. [2, 10]) in order to enhance inference capabilities. Ontologies can be easily edited using the *Protégé* editor [27] and can be encoded and stored using the Web Ontology Language (OWL) [3]. In this way, the knowledge can be shared by different devices, being accessed through the world wide web. Using ontologies, knowledge is mainly classified in classes (which entails a collection of objects), individuals (in which instances of classes are stored), relations (expressing the correspondence among objects as well as individuals), and properties (specifying data values for objects).

4.2 Abstract Knowledge

The abstract knowledge is the high-level representation of knowledge which remains fixed throughout the planning process. It is divided into the *Semantic Knowledge* containing information about the workspace and the robot, coded as an OWL taxonomy, as depicted in Fig. 5, and the *Manipulation Knowledge* which involves knowledge related to how the robot can interact with the workspace, and which is inferred from the semantic knowledge.

4.2.1 Semantic Knowledge ($\mathcal{K}_S$)

Semantic knowledge categorizes information within the following classes:

- Class *"RobotProperties"* describes the properties of the robot in two subclasses. Geometric constraints of the robot such as joint limits are stored in the class *KinematicProperties*; differential properties of the robot such as bounds on forces, torques, velocities, and accelerations (global properties that condition the maximum capacity of the robot) are stored in the class *DynamicProperties*.
- Class *"ObjectClassification"* is used to describe the objects in the workspace such as, fixed (*fixedObject*), free manipulatable (*free-mObject*) and constraint-oriented manipulatable (*co-mObject*).
- Class *"ManipulationConstraint"* describes orientation constraints on the motion of bodies and objects.

The hierarchy of knowledge among the classes can be represented using Description Logic (DL). For instance the hierarchy for a constraint-oriented manipulatable object is described below:

$$
\begin{aligned}
&\mathit{Thing} := \mathit{object}\\
&\exists \mathit{hasSuperclass}(\mathit{Thing}, \mathit{SemanticKnowledge})\\
&\wedge \exists \mathit{hasClass}(\mathit{SemanticKnowledge}, \mathit{ObjClassification})\\
&\wedge \exists \mathit{hasSubclass}(\mathit{ObjectClassification}, \mathit{co\text{-}mObject})
\end{aligned}
$$

where $\wedge$ and $\exists$ represents *conjunction* and *exist*, respectively.

The semantic properties (that are stored on OWL) are divided into object properties and data properties. The former are used to describe the relationships between the individuals, and the latter are used to assign the values to the physical attributes. As an example, some of the semantic properties of the car-like object are depicted in Fig. 6 and explained below in terms of DL.

$$
\begin{aligned}
&\mathit{Object} := \mathit{Car}\\
&\wedge \exists \mathit{hasWheel}(\mathit{Car}, \mathit{Wheel})\\
&\wedge \exists \mathit{hasBody}(\mathit{Car}, \mathit{Body})\\
&\wedge \exists \mathit{hasWheel}(\mathit{Wheel}, \mathit{alongYaxis})\\
&\wedge \exists \mathit{canMove}(\mathit{Car}, \mathit{alongXaxis})
\end{aligned}
$$

The above stated DL description of an object property (*hasConst-yAxis fourwheeldrive*) explain that car is composed of wheels and body, the motion of the car-like object is constraint by the wheels that are along the y axis of the

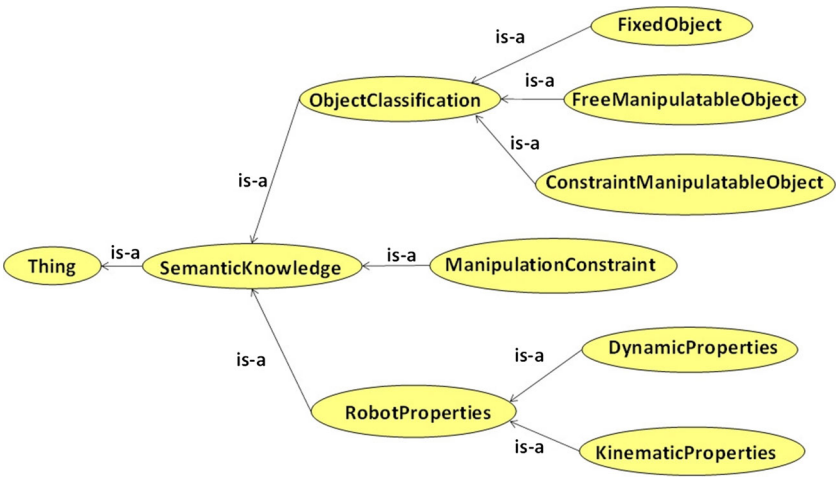

Fig. 5 OWL-based semantic knowledge taxonomy. https://sir.upc.edu/projects/ontologies

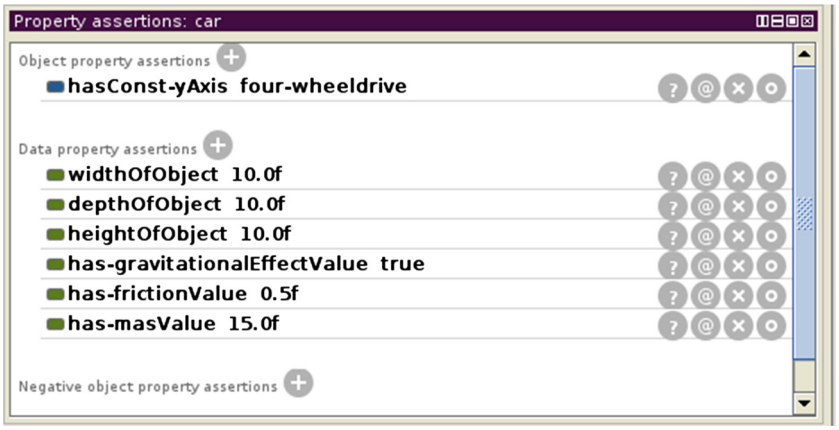

Fig. 6 Screen shot of the Protégé editor showing the semantic properties of a car-like object

chassis so that the car can only move along the associated x-axis.

The data properties in terms of description logic are represented as follows:

$$
\begin{aligned}
&Object := Car \\
&\exists hasWidth(Car, Value) \\
&\wedge \exists hasDepth(Car, Value) \\
&\wedge \exists hasHeight(Car, Value) \\
&\wedge \exists hasGravity(Car, Value) \\
&\wedge \exists hasFriction(Car, Value) \\
&\wedge \exists hasMass(Car, Value)
\end{aligned}
$$

It describe the dimension of the car, response to the gravitational (if considered as dynamic object, the value will be true and false otherwise) and the values of friction (between wheels and road) and mass of the car respectively.

4.2.2 Manipulation Knowledge ($\mathcal{K}_M$)

Manipulation knowledge is inferred from $\mathcal{K}_S$ using a Prolog inference process that will be detailed later in Section 5.1. It contains the classification of objects (*fixedObject*, *freemObject*, or *co-mObject*), a complete set of *mRegions* for manipulatable objects, physical attributes of objects, and kinematic and dynamic properties of the robot (such as joint limits and bounds on forces and velocities). Manipulation knowledge remains fixed throughout the motion planning process. $\mathcal{K}_S$ contains the maximum possible knowledge about the world and $\mathcal{K}_M$ that related to a particular motion planning problem (e.g. it must be updated if the features of the robot change).

4.3 Instantiated Knowledge (κ)

Instantiated knowledge is the low level representation of knowledge (at the motion planning level). It is dynamic and valid for a particular instant of time (containing all possible constraints that are valid for that instant of time) and is updated for the next time step. The instantiated knowledge contains two main components: *(1)* the knowledge about the valid and invalid manipulation regions, as well as dynamical properties (such as masses, friction coefficients) of the objects; *(2)* the control sampling range (such as bounds on control forces and torques) to be used by the motion planner at the current instant of time (by using the state propagator, i.e. the dynamic engine).

The instantiated knowledge is updated by the low-level reasoning process based on the high-level manipulation knowledge and the feedback from the motion planner. The reasoning process is detailed in Section 5.2).

5 Knowledge Inference and Reasoning Process

The flow of knowledge was graphically illustrated in Fig. 3. It includes the knowledge inference process and the reasoning process. The knowledge inference process is the pre-processing step responsible of inferring the manipulation knowledge $\mathcal{K}_M$ from the ontological semantic knowledge $\mathcal{K}_S$. On the other hand, the reasoning process is the responsible of updating the instantiated knowledge at each instant of time while planning, based on $\mathcal{K}_M$ and the current state fed back by the motion planner.

5.1 Knowledge Inference Process

The inference process is performed using Knowrob [32], a knowledge-based processing tool for robotics applications that allows operating over OWL data bases, extended with the following predicates (coded in Prolog) tailored to extract the necessary information for the physics-based motion planner:

– *object_classification(?Obj, ?ObjType)*: Given an object *Obj* returns the type of the associated object *ObjType* by evaluating its category. Those manipulatable objects that are too heavy for the robot to be manipulated are changed to fixed.

- *manipulatable_region(?Obj, ?ManipRgns)*: Given a manipulatable object *Obj* computes the set of associated manipulatable regions *ManipRgns* taking into account the manipulation constraints, if any.
- *object_properties(?Obj, ?PhysicalProps, ?Dimension)*: Given an object *Obj* returns its physical properties *PhysicalProps*, including mass and friction values, as well as gravitational effect and the dimension *Dimension* of the object.
- *robot_properties(?DynamicsProps, ?KinematicProps)*: Returns the dynamics properties of the robot (forces and velocities limits) in *DynamicsProps*, and the kinematic properties (joint limits) in *KinematicProps*.

At any given state fed back by the motion planner (corresponding to the node to be expanded), the reasoning process uses geometric reasoning to update the instantiated knowledge with the current information on the manipulation regions and on the control sampling range. Those manipulation regions that become useless are deactivated; the others are set active. A manipulation region becomes useless if it is occupied by an obstacle (i.e. the robot cannot access it), or if the motion of the object is not possible when the robot interacts with it from the manipulation region, e.g. if the front manipulation region of the car-like object is blocked with a *mObject* it is deactivated, as well as the rear *mRegion* because the robot can not exert forces from there until the front *mRegion* becomes free.

5.2 Reasoning Process

Kinodynamic planners sample both the direction and module of the control to be applied to extend the tree data structure. The proposed reasoning process computes the module range from where to sample depending on whether the robot is located in $\mathcal{C}_{\text{move}}$ or $\mathcal{C}_{\text{interaction}}$, and in this latter case depending on whether the robot is in collision or not. The normal control range to move the robot freely in $\mathcal{C}_{\text{move}}$ will be diminished when being in $\mathcal{C}_{\text{interaction}}$, and if contact occurs then it will be increased based on the weight of the object to be pushed. Let F be the module range from where to sample, and let F take the value $F = [f_{\min}, f_{\max}]$ when the robot is in $\mathcal{C}_{\text{move}}$. Then, when the robot is in $\mathcal{C}_{\text{interaction}}$:

- $F = \alpha[f_{\min}, f_{\max}]$ with $\alpha < 1$ if no contact occurs,
- $F = [f_{\min} + f_{\text{obj}}, f_{\max} + f_{\text{obj}}]$ with $f_{\text{obj}} = \mu_{\text{obj}}\, m_{\text{obj}}\, g$, if contact occurs,

where μ_{obj} is the friction coefficient between the object and the floor, m_{obj} is the mass of the object, and g the gravitational force. In case of a manipulator (kinematic chain), the range of forces is converted to a range of joint torques using the transposed Jacobian.

To illustrate this, Fig. 7 qualitatively depicts the difference between the use of the proposed control sampling range and a fixed range, as done in standard physics-based motion planning approaches [21, 29]. In our proposal, the higher value of control forces will only be used when the robot is in contact with the object, and according to its weight. If a fixed lower control range is set then it may not be able to push the objects (to clear the regions) and fails to compute the path, on the other hand, if a higher control range is set then it consumes unnecessary power and may result in a huge displacement of the object. Moreover, prior to contact, the proposed controls slow down to have a smooth transition from no contact to contact.

6 The *κ-PMP* approach

6.1 Proposed Framework

The proposed framework for power-efficient physics-based motion planning is depicted in Fig. 8. It is a hybrid planning framework that consists of three main layers: the high-level

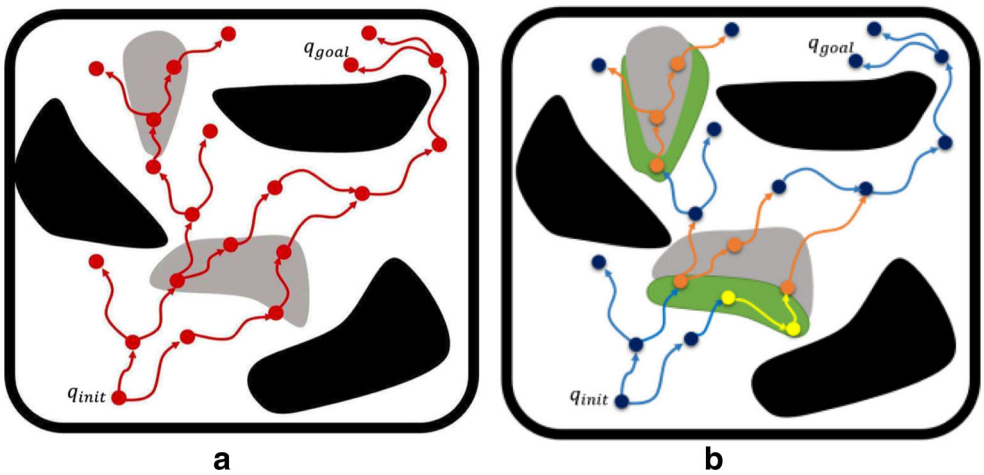

Fig. 7 Sample and propagation using: **a** the simple physics-based motion planner; **b** κ-PMP. The control sampling range is selected based on the dynamics properties of the target object and the manipulation region. Blue samples represent the lower values of control range, whereas orange samples represent the higher values. Yellow samples represents the decrease in forces during the transition between free and contact motions

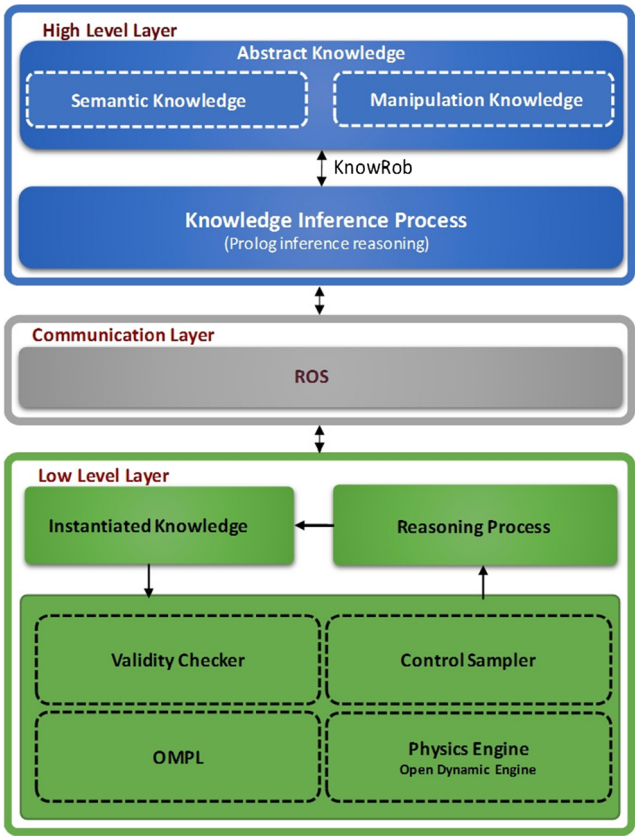

Fig. 8 Knowledge-oriented physics-based motion planning (κ-*PMP*) for power-efficient motion plan

layer devoted to the knowledge management and inference process, the communication layer, and the low-level layer devoted to the physics-based motion planning.

The high-level layer that contains the high-level representation of knowledge is divided into the semantic knowledge coded in the form of ontologies, and the manipulation knowledge inferred from the semantic knowledge using the knowledge inference process module.

The low-level layer consists of the instantiated knowledge κ (that contains the temporary manipulation constraints and the bounds on the control forces), the reasoning process (responsible of updating the instantiated knowledge from the current state and the manipulation knowledge), and the motion planner (whose main modules are the physics-based state validity checker $\mathcal{F}$, the control sampling module, a standard kinodynamic planner like KPIECE or RRT from the Open Motion Planning Library [31], and the ODE-based physics engine). This layer is developed within *The Kautham Project* [25], an open source framework for motion planning that includes geometric, differential and physics-based motion planners (including those that require ontological knowledge).

The communication layer is based on ROS and communicates the high-level abstract knowledge and the low-level motion planning layer: the abstract knowledge is encapsulated as a ROS service and the motion planning layer accesses it as a ROS client.

6.2 Algorithm

The planning process is sketched in Algorithm 1. It takes as input the initial state $\mathbf{q}_{init}$, the goal region $\mathbf{Q}_{goal}$, and the maximum allowed planning time T_{max}. If a solution is found, it returns the path from $\mathbf{q}_{init}$ to $\mathbf{q}_{goal} \in \mathbf{Q}_{goal}$, or NULL otherwise. To sample the states and construct the planner data structure, the approach provides the flexibility to use any sampling-based kinodynamic motion planner (such as RRT, KPIECE, SyCLoP) offered by OMPL, together with the Open Dynamic Engine as state propagator.

Lines [1–3] of the algorithm are the preprocessing steps for the motion planner; lines [4–15] contain the planning process. The main functions used are the following:

- *WorkspaceInit:* Initializes the state of the robot and of the objects in the environment.
- *SemanticKnowledgeGenerator:* Creates the semantic knowledge $\mathcal{K}_S$ from the ontologies.
- *ManipulationKnowledgeInference:* Infers $\mathcal{K}_M$ from $\mathcal{K}_S$.
- *ReasoningProcess:* Updates the instantiated knowledge by updating the manipulation constraints and the range of the control, as detailed in Algorithm 2.
- *SelectNodeToExpand:* Selects the node to expand following the node selection process of the kinodynamic planner used.

Algorithm 1 κ-*PMP*

Input: Initial state $\boldsymbol{q}_{\text{init}}$, Goal region $\boldsymbol{Q}_{\text{goal}} \in \mathcal{C}$, Threshold T_{max}
Output: A path from $\boldsymbol{q}_{\text{init}}$ to $\boldsymbol{q} \in \boldsymbol{Q}_{\text{goal}}$

```
1:  WorkspaceInit()
2:  K_S ←SemanticKnowledgeGenerator()
3:  K_M ←ManipulationKnowledgeInference(K_S)
4:  while t < T_max do
5:      κ ← ReasoningProcess(K_M, q)
6:      SelectNodeToExpand()
7:      {u, n} ← SampleControlsAndSteps(κ)
8:      for i = 0 to i < n do
9:          q_new ← Propagate(q, u)
10:         if ! StateValidityChecker(q_new, κ) then
11:             Break
12:         else
13:             UpdateConnections()
14:         end if
15:     end for
16:     if q_new ∈ Q_goal then
17:         return Path(q_new)
18:     end if
19: end while
20: return NULL
```

Algorithm 2 ReasoningProcess($\mathcal{K}_\mathrm{M}$, $\mathbf{q}$)

1: $\Gamma \leftarrow$ UpdateManipulationConstraints($\mathcal{K}_M$, $\mathbf{q}$)
2: $\mathcal{L} \leftarrow$ ComputeRobotLocation($\mathbf{q}$)
3: $\{f_{\min}, f_{\max}\} \leftarrow$ComputeControlRange($\mathcal{L}$)
4: $\kappa_{\text{new}} \leftarrow$UpdateInstantiatedKnowledge(Γ, $\{f_{\min}, f_{\max}\}$)
5: **return** κ_{new}

- *SampleControlsAndSteps:* Samples the controls and the number of steps describing the number of times the selected control will be applied repeatedly. To sample the controls any control sampling strategy, such as steering control sampling, can be used.
- *Propagate:* Applies the sampled control $\mathbf{u}$ on the state $\mathbf{q}$ for Δt time using the Open Dynamic Engine as state propagator.
- *StateValidityChecker*: Is the physics-based state validity checker ($\mathcal{F}$) that validates the newly generated state by taking into account the instantiated knowledge.
- *UpdateConnections:* Adds the accepted state to the planner data structure and updates the connections accordingly.
- *Path:* Returns a path from $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ if the last generated state lies in the goal region, and NULL otherwise.

Algorithm 2 implements the reasoning process, it contains the following steps:

- *UpdateManipulationRegions:* Activates and deactivates the manipulation regions using the geometric reasoning based on $\mathcal{K}_M$ and the current state.
- *ComputeRobotLocation:* Determines the region $\mathcal{L}$ where the robot lies.
- *ComputeControlRange:* Determines the control range as a function of $\mathcal{L}$.
- *UpdateInstantiatedKnowledge:* Updates the data structures of the instantiated knowledge with the updated manipulation regions and the control sampling range.

The key point of the algorithm is that, during planning, the instantiated knowledge is updated at each instant of time by the reasoning process (Line 5), conditioning the sampling of the controls (Line 7) and the state validity checking procedure (Line 9).

6.3 Simulation Setups

The proposed approach is validated using three different robot models, depicted in Fig. 9. The scenario presented in

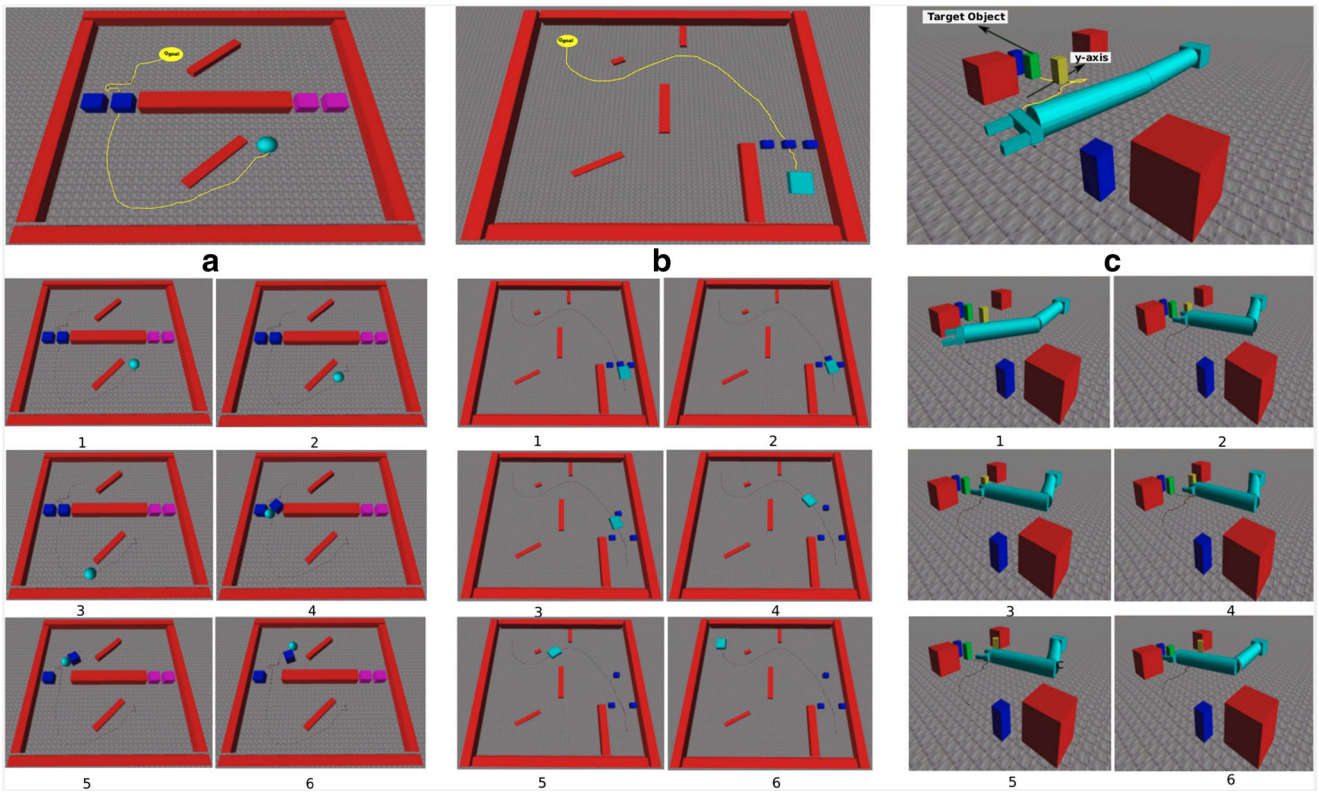

Fig. 9 Planning scenes: **a** an holonomic mobile robot; **b** a car-like mobile robot; **c** a planar kinamatic chain. Video: https://sir.upc.edu/projects/kautham/videos/k-PMP1.mp4

Fig. 9a is for an holonomic mobile robot. It consists of a robot (sphere), *free-mObjects* (blue and purple cubes, being the purple ones heavier than blue cubes) and *fixed-objects* (red walls). The problem is to go from $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$, being the possible solutions blocked by the blue or the purple cubes. Since the motion of this robot is controlled by applying the control force, the range of the control force will be selected by the reasoning process according to the object to be pushed (blue or purple), as explained in Section 5.2. The higher values of controls will only be applied when the robot is in interaction with the object, unlike the conventional planners that fix the control range at the start, our approach dynamically varies the control range. The power $\mathcal{P}$ consumed by the robot while moving along the path is computed as

$$\mathcal{P} = \sum_{i}^{n} \frac{\mathbf{f}_i \cdot \mathbf{d}_i}{\Delta t_i}, \tag{6}$$

with $\mathbf{f}$ and $\mathbf{d}$ being, respectively, the applied force and displacement vectors, and Δt the time duration.

Figure 9b describes the scene for the car-like robot. The position and orientation of the car is controlled by adding

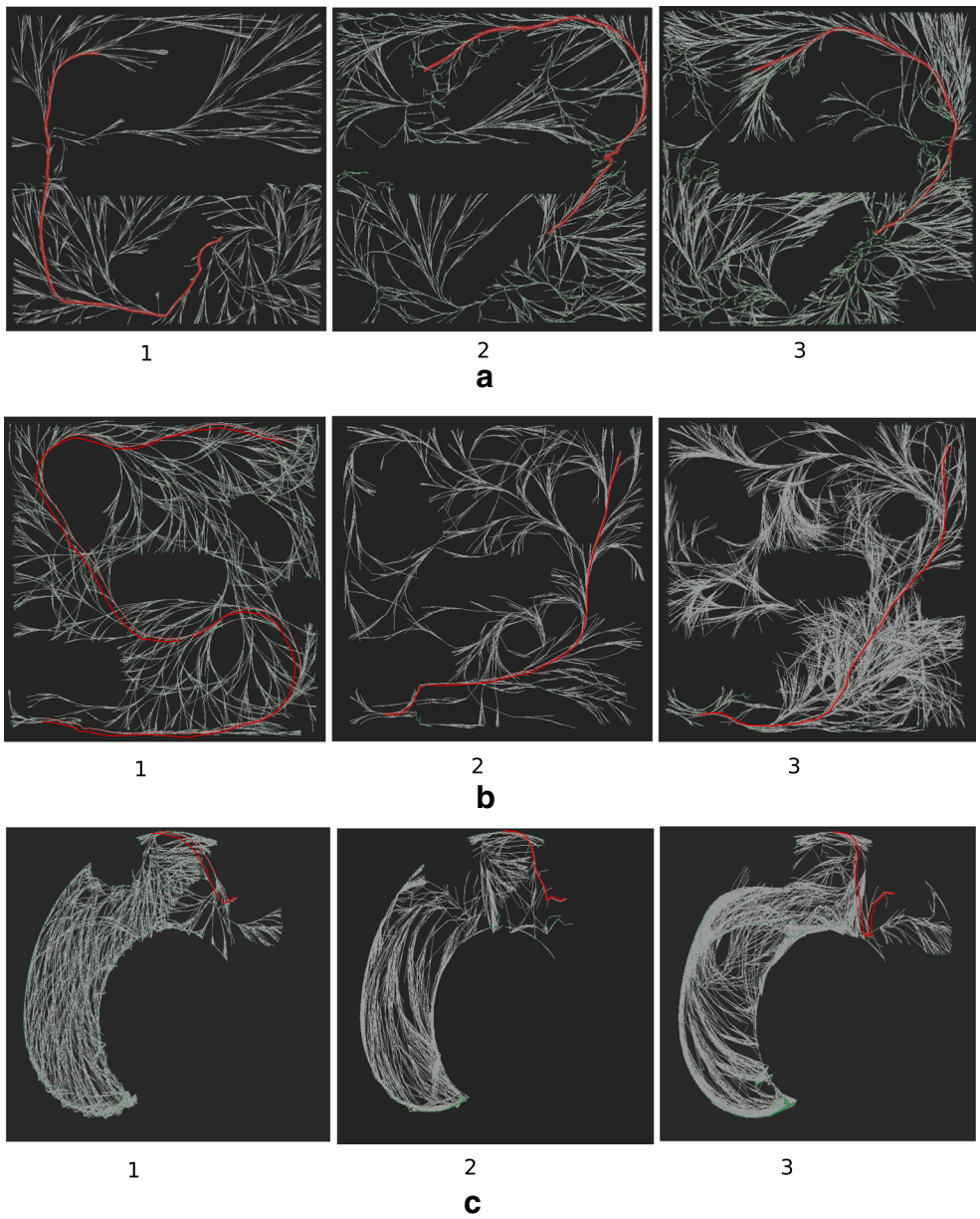

Fig. 10 State space projection onto the workspace for the scenes with the holonomic mobile robot (top), the car-like robot (middle) and the planar manipulator (bottom) using: (1) κ-*KPIECE*, (2) κ-*RRT* and (3) κ-*SyCLoP* planners

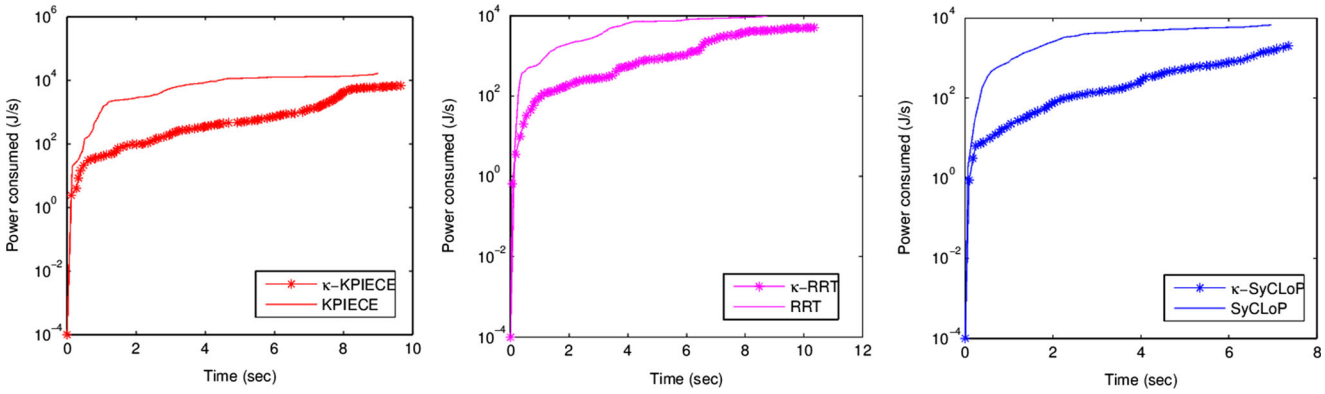

Fig. 11 Logarithmic plot of the power consumed while moving along the path for the holonomic mobile robot

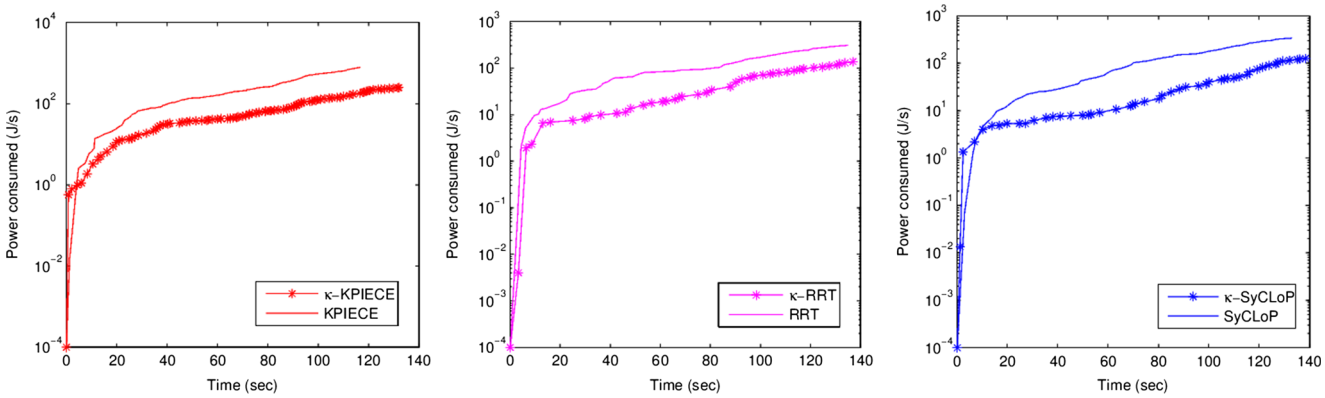

Fig. 12 Logarithmic plot of the power consumed while moving along the path for the car-like robot

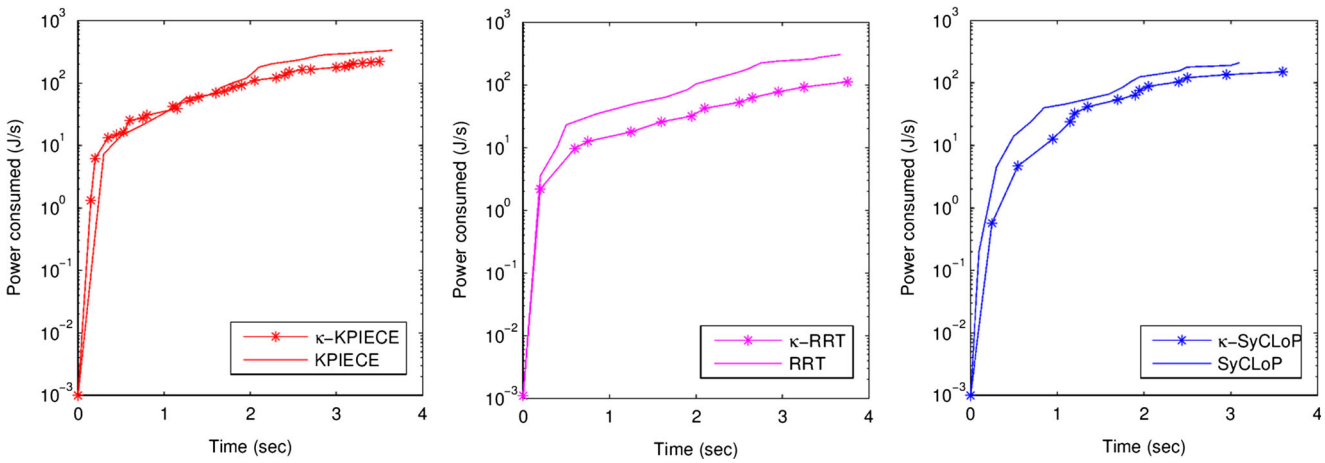

Fig. 13 Logarithmic plot of the power consumed while moving along the path for the planar manipulator

the torque to the wheels and to the steering. The path to the goal is blocked with the *free-mObjects* (blue cubes) and in order to reach the goal the car has to clear the way by pushing the objects away. The high amount of torque is only required while pushing the objects. The reasoning process updates the torque bounds by determining the forces that are required to push the object, and transforms it into the torque exerted by each wheel. The power consumed while moving along the path is computed as

$$\mathcal{P} = \sum_{i}^{n} \tau_i \cdot \omega_i, \tag{7}$$

where τ is the torque exerted by the wheels and ω is the corresponding angular velocity.

The scenario for the planar manipulator is depicted in Fig. 9c, it consists of *free-mObject* (yellow and blue boxes), the target object (green box) and the *fixedObjects* (red cubes). The manipulator is shown in its initial state and the goal is to grasp the target object, being the way to that object blocked by the yellow box. Since one region of the yellow box is occupied with the target object, the reasoning process will change its type to *co-mObject* and it can only be manipulate along y-axis (Once all the manipulation regions of the yellow box will be free, its type will again update to *free-mObject*). In-order to reach the goal the manipulator has to push it away. The control forces are transformed into joint torques using the transposed Jacobian and the power is computed as

$$\mathcal{P} = \sum_{i}^{n} \tau_i \cdot \omega_i, \tag{8}$$

where now τ is the torque exerted by the joints and ω is the corresponding angular joint velocity. Fig. 10 depicts the configuration space for the κ-*KPIECE*, κ-*RRT* and κ-*SyCLoP* planners for the above stated scenarios.

7 Results and Discussion

We compared κ-*PMP* with simple physics-based planning, using RRT, KPIECE and SyCLoP as kinodynamic motion planners, because a recent benchmarking study [11] of the physics-based motion planning showed these planners as being the most suitable for physics-based motion planning: KPIECE computed the efficient solutions in terms of time whereas SyCLoP and RRT computed the efficient solution in terms of power. No comparison has been done with other planners that seek different goals, such as [12, 28] that cope with the rearrangement of the objects in the workspace or such as [20] that are focus on optimization issues.

The parameter used in the comparison were:

- *Power consumption:* The total power consumed by the robot while moving along the solution path.
- *Planning time:* The total time consumed by the planner to compute the solution path.
- *Success rate:* The number of successful runs in the maximum limit of planning time.

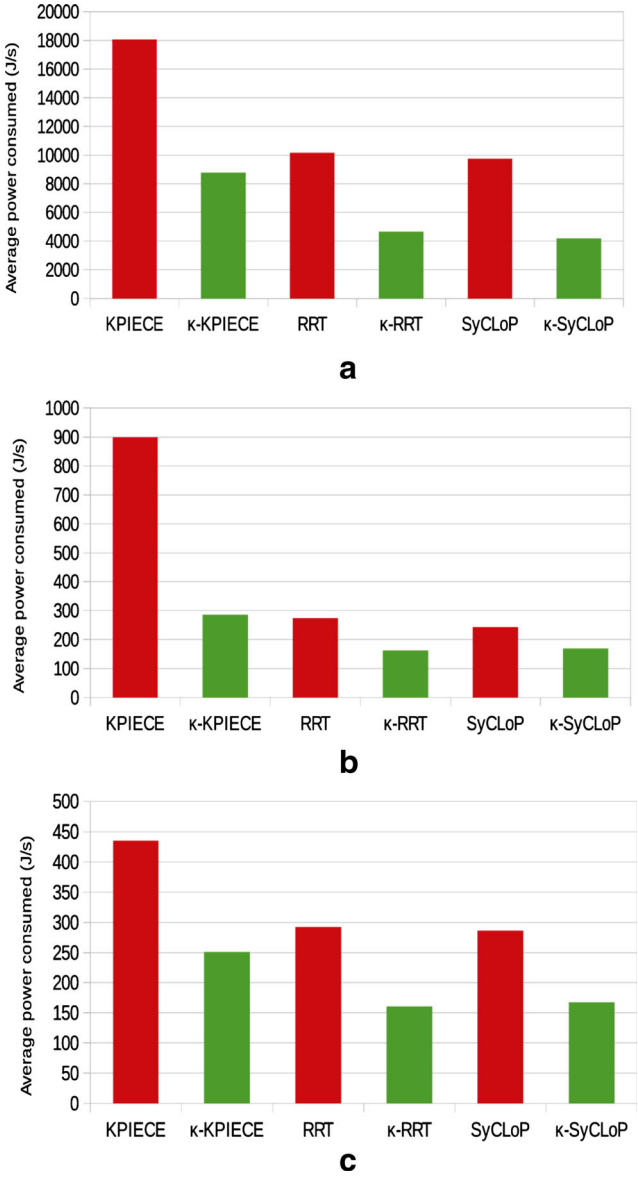

Fig. 14 Histogram of the average power consumed by the robots: **a** holonomic mobile robot, **b** car-like robot, **c** planar manipulator

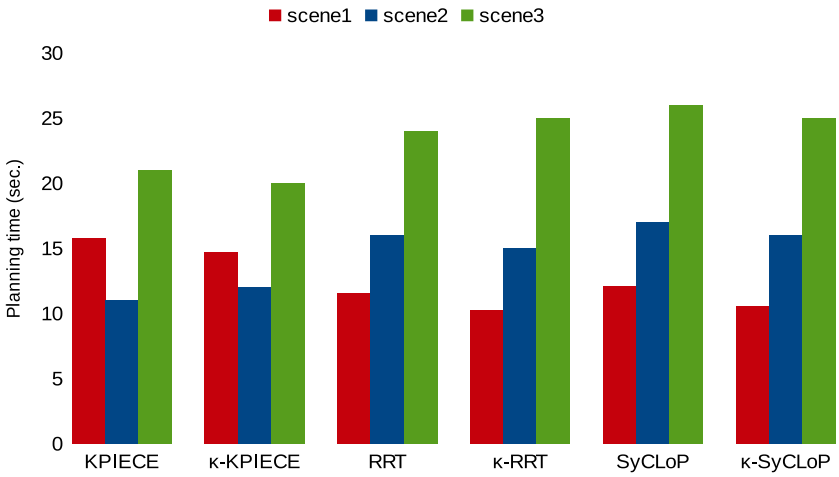

Fig. 15 Histogram of the average planning time. Scene1, scene2 and scene3 correspond to the scenes illustrated in Fig. 9

7.1 Quantitative Analysis

κ-PMP computes the power efficient solution as compared to the simple physics-based motion planning approach. We compared the best results that we obtained using both approaches with different kinodynamic motion planners. Figures 11, 12, and 13 show the logarithmic plot of power consumption vs time corresponding to the holonomic mobile robot, the car-like robot and the planar manipulator, respectively. In all cases *κ-PMP* is the most efficient plan in terms of power. The average power consumed by each planner in several runs is shown in the form of histogram (Fig. 14) corresponding to the three different robot models. There is a significant difference in terms of power consumption when using *κ-PMP* and simple physics-based planning approaches.

The average planing time of several runs for three scenes is presented in Fig. 15. *κ-PMP* computes the solution almost in the same time as the traditional approach does. To compute the success rate, we set the maximum planning time to 150 s and each query is executed 10 times. Figure 16 shows the histogram of the success rate where it can be appreciated that *κ-PMP* has a similar success rate as the traditional one. In some cases (particularly for the manipulator) our approach has the higher success rate than the traditional one.

7.2 Qualitative Analysis

The analysis of the results shown in the previous section illustrates that *κ-PMP* preserves the behavior of the kinodynamic planner used (such as RRT or KPIECE) and significantly reduce the power consumed. This is due to the fact that *κ-PMP* dynamically varies the control sampling range according to the physical properties of the target object. Moreover, *κ-PMP* uses knowledge to determine the way to manipulate objects in order to reach the goal in a more realistic way. For instance, Fig. 17 shows a sequence of snapshots of an execution using a traditional physics-based planning approach. Since it lacks the knowledge about the way of manipulating the objects, the yellow

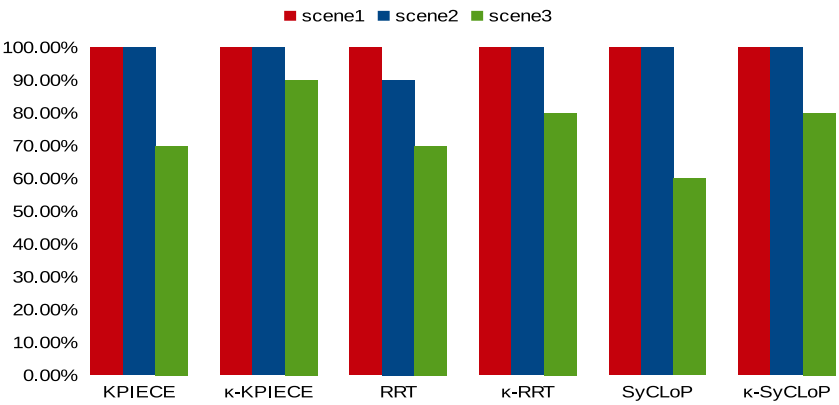

Fig. 16 Histogram of the success rate. Scene1, scene2 and scene3 correspond to the scenes illustrated in Fig. 9

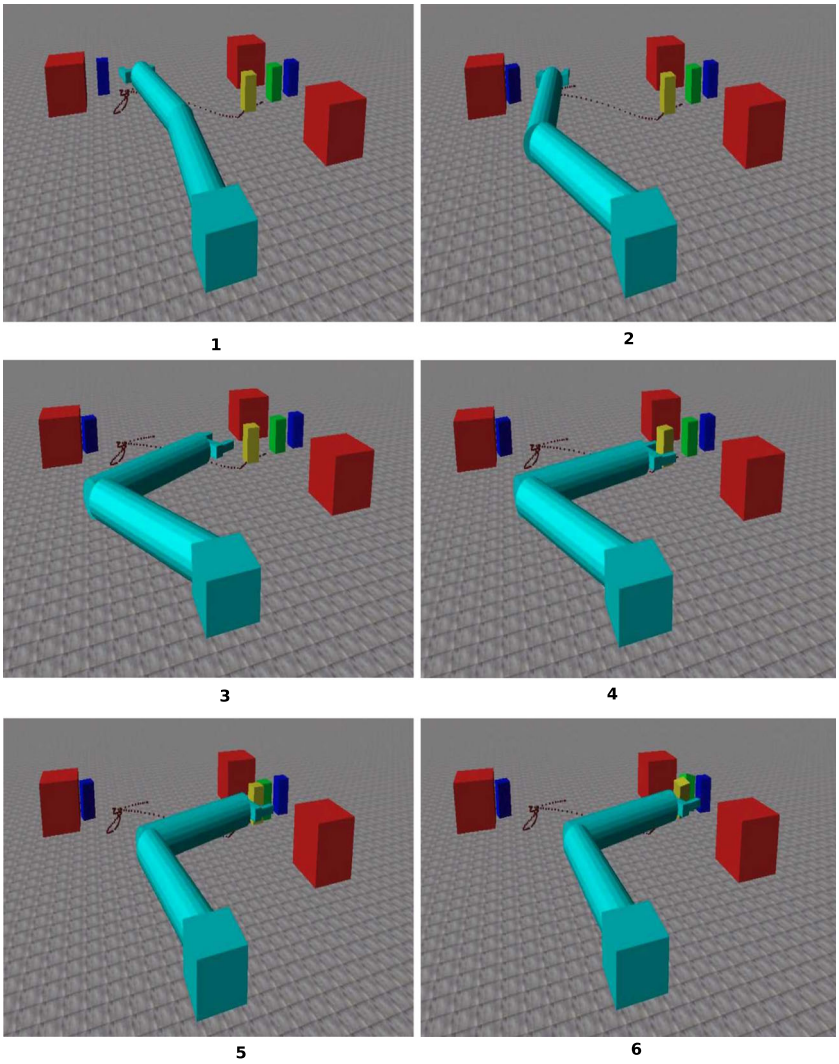

Fig. 17 Execution of the computed motion plan by the simple physics-based motion planner. It can be observed that the task fails because the manipulatable yellow object ends at the gripper, thus preventing the manipulator to grasp the target green object

box ends stuck in the gripper, thus preventing the manipulator to grasp the green box, reducing the success rate of the planner.

κ-PMP manipulates the objects in a natural way without putting any extra constraints (such as quasi static push). Figure 18 shows the results of the dynamic interaction between the robot and the object using both approaches. In most of the cases simple physics-based planning approach threw the object away (Fig. 18a) because it is unaware of how much force is required to push. In contrast, Fig. 18b shows the smoother interaction resulting from the use of *κ-PMP*. It applies the forces based on the physical properties (such as mass and friction) and pushes the object in a smooth way. Furthermore the solution path computed by the κ-PMP is naturally biased toward the manipulation regions, which helps to effectively manipulate the objects.

7.3 Practical Application of the Proposed Approach

From the analysis done, it has been shown that the proposed approach computes more robust and power efficient motion trajectories, as compared to the kinodynamic or simple physics-based planning approaches (mentioned in the related work). Moreover, the current proposal handles dynamic interaction in a more natural way. With a practical perspective, therefore, this approach may be significantly important in two directions. On the one hand, it can be a part of an integrated task and motion planner. The availability of power-efficient motion trajectories may have a relevant contribution in the final performance of the integrated task and motion planner, since the costs of the actions in a plan are critical and greatly influence the decisions of the task planner. On the other hand, as a stand alone planner, it results in a smart and powerful tool to manipulate objects in the

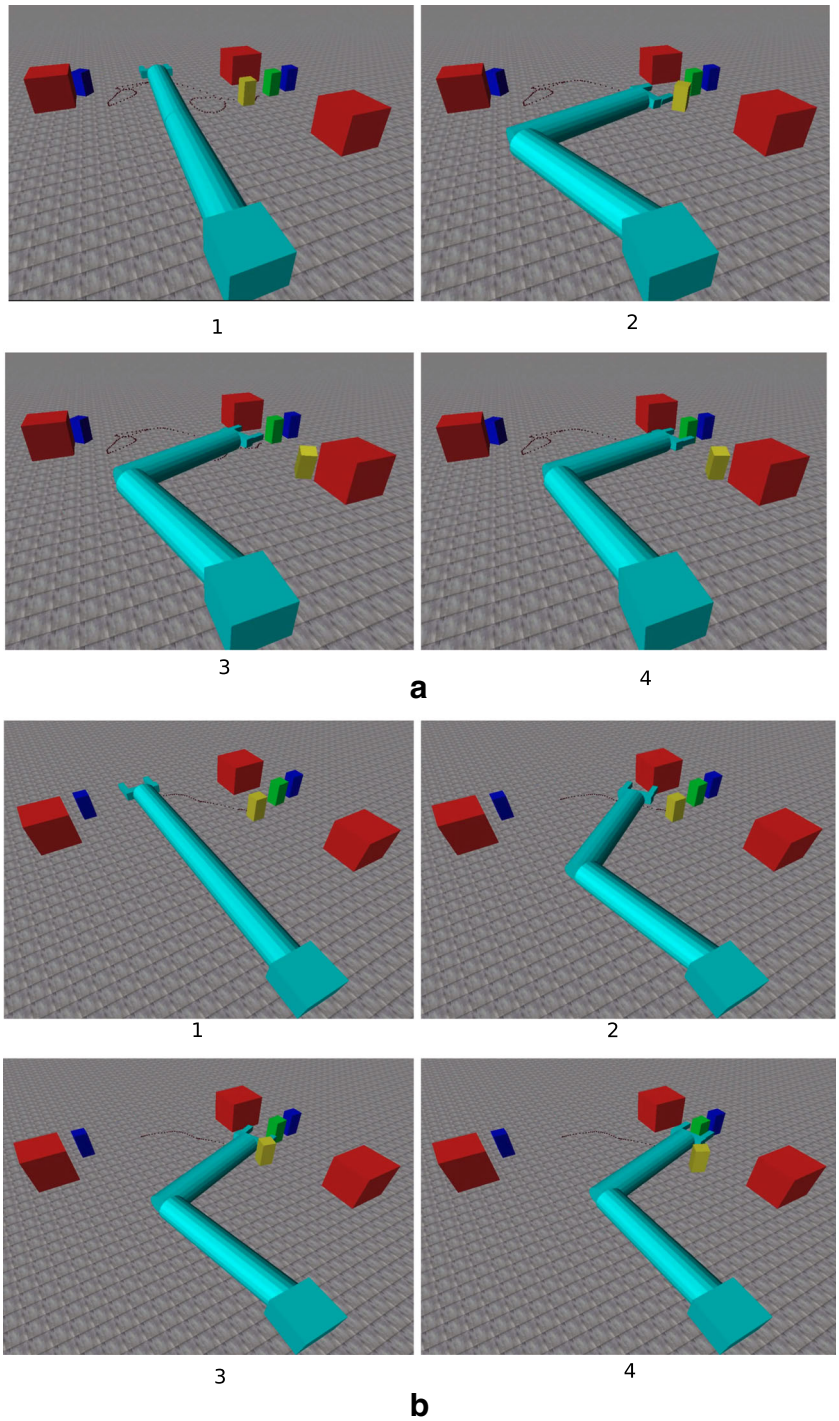

Fig. 18 Snapshots of the dynamic interactions between the robot and the yellow object corresponding to the simple physics-based motion planning (**a**) and to the κ-*PMP* planner (**b**), respectively. Videos: https://sir.upc.edu/projects/kautham/videos/k-PMP2.mp4

clutter, without the need of reasoning at task level, giving practical solutions to quite difficult problems.

8 Conclusion

This paper has proposed a framework, called κ-*PMP*, to use knowledge to enhance physics-based motion planners based on any kinodynamic algorithm, like RRT or KPIECE, and on any dynamic engine, like ODE or Bullet. Manipulation knowledge, inferred from an abstract knowledge ontology coded using the Web Ontology Language (OWL), is used to incorporate a reasoning process within the state transition model. This allows to dynamically update: a) the control sampling range based on the region of the configuration space where the robot is located and on the physical properties of the objects to be interacted; b) the regions from where an object can be manipulated. A comparison of

physics-based motion planners based on RRT, KPIECE and SyCLoP with and without using the *κ-PMP* framework has been carried out in three different scenarios involving a holonomic mobile robot, a car-like robot and a planar manipulator. The proposal resulted in an improvement of the success rate and of the performance in terms of power consumed and quality of the solutions.

References

1. Akbari, A., Gillani, M., Rosell, J.: Reasoning-based evaluation of manipulation actions for efficient task planning. In: Robot 2015: Second Iberian Robotics Conference, pp. 69–80. Springer (2016)
2. Akbari, A., Muhayyuddin, Rosell, J.: Task and motion planning using physics-based reasoning. In: Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–7 (2015)
3. Antoniou, G., van Harmelen, F.: Web Ontology Language: OWL. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies in Information Systems, pp. 67–92. Springer (2004)
4. Bhatia, A., Maly, M.R., Kavraki, L.E., Vardi, M.Y.: Motion planning with complex goals. IEEE Robot. Autom. Mag. **18**(3), 55–64 (2011)
5. Carpin, S.: Randomized motion planning: a tutorial. Int. J. Robot. Autom. **21**(3), 184–196 (2006)
6. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G.A., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Pittsburgh (2005). https://mitpress.mit.edu/books/principles-robot-motion. ISBN 0-262 -03327-5
7. Dogar, M., Srinivasa, S.: A framework for push-grasping in clutter. In: Robotics: Science and Systems VII (2011)
8. Dogar, M.R., Hsiao, K., Ciocarlie, M., Srinivasa, S.: Physics-based grasp planning through clutter. In: Robotics: Science and Systems (2012)
9. Erwin, C.: Bullet physics library. http://bulletphysics.org (2013)
10. Feyzabadi, S., Carpin, S.: Knowledge and data representation for motion planning in dynamic environments. In: Robot Intelligence Technology and Applications 2, pp. 233–240. Springer (2014)
11. Gillani, M., Akbari, A., Rosell, J.: Physics-based motion planning: evaluation criteria and benchmarking. In: Robot 2015: Second Iberian Robotics Conference, pp. 43–55. Springer (2016)
12. Haustein, J.A., King, J., Srinivasa, S.S., Asfour, T.: Kinodynamic randomized rearrangement planning via dynamic transitions between statically stable states. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3075–3082 (2015)
13. Hsu, D., Kindel, R., Latombe, J.C., Rock, S.: Randomized kinodynamic motion planning with moving obstacles. Int. J. Robot. Res. **21**(3), 233–255 (2002)
14. Hsu, D., Latombe, J.C., Motwani, R.: Path planning in expansive configuration spaces. In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, pp. 2719–2726. IEEE (1997)
15. Kavraki, L.E., Švestka, P., Latombe, J.C., Overmars, M.H.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **12**(4), 566–580 (1996)
16. Ladd, A., Kavraki, L.: Fast tree-based exploration of state space for robots with dynamics. In: Algorithmic Foundations of Robotics VI, pp. 297–312. Springer (2005)
17. Latombe, J.-C.: Robot Motion Planning. Kluwer Academic Publishers, Norwell (1991). https://www.springer.com/us/book/9780792392064. ISBN 079239206X
18. LaValle, S.M.: Planning Algorithms. Cambridge University Press, New York. https://www.cambridge.org/core/books/planning-algorithms/FC9CC7E67E851E40E3E45D6FE328B768. ISBN 05218 62051 (2006)
19. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. **20**(5), 378–400 (2001)
20. Li, Y., Littlefield, Z., Bekris, K.E.: Asymptotically optimal sampling-based kinodynamic planning. Int. J. Robot. Res. **35**(5), 528–564 (2016)
21. Muhayyuddin, Akbari, A., Rosell, J.: Ontological physics-based motion planning for manipulation. In: Proceedings of the IEEE International Conference on Emerging Technologies Factory Automation (ETFA), pp. 1–7 (2015)
22. NVIDIA: Physx. https://developer.nvidia.com/physx-sdk
23. Plaku, E.: Planning in discrete and continuous spaces: from ltl tasks to robot motions. In: Advances in Autonomous Robotics, pp. 331–342. Springer (2012)
24. Plaku, E., Kavraki, L., Vardi, M.: Motion planning with dynamics by a synergistic combination of layers of planning. IEEE Trans. Robot. **26**(3), 469–482 (2010)
25. Rosell, J., Pérez, A., Aliakbar, A., Muhayyuddin, Palomo, L., García, N.: The Kautham project: a teaching and research tool for robot motion planning. In: Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation (2014)
26. Russell, S.: Open dynamic engine. http://www.ode.org/ (2007)
27. Stanford2007: Protégé. http://protege.stanford.edu/ (2007)
28. Stilman, M., Kuffner, J.J.: Navigation among movable obstacles: real-time reasoning in complex environments. Int. J. Humanoid Rob. **2**(04), 479–503 (2005)
29. Sucan, I., Kavraki, L.E.: A sampling-based tree planner for systems with complex dynamics. IEEE Trans. Robot. **28**(1), 116–131 (2012)
30. Şucan, I.A., Kavraki, L.E.: Kinodynamic motion planning by interior-exterior cell exploration. In: Algorithmic Foundation of Robotics VIII, pp. 449–464. Springer (2010)
31. Şucan, I.A., Moll, M., Kavraki, L.E.: The open motion planning library. IEEE Robot. Autom. Mag. **19**(4), 72–82 (2012)
32. Tenorth, M., Beetz, M.: Knowrob knowledge processing for autonomous personal robots. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4261–4266 (2009)
33. Zickler, S., Veloso, M.: Efficient physics-based planning: sampling search via non-deterministic tactics and skills. In: Proceedings of the 8Th International Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 27–33 (2009)
34. Zickler, S., Veloso, M.M.: Variable level-of-detail motion planning in environments with poorly predictable bodies. In: Proceedings of the European Conference on Artificial Intelligence Montpellier, pp. 189–194 (2010)
35. Zucker, M., Ratliff, N., Dragan, A.D., Pivtoraiko, M., Klingensmith, M., Dellin, C.M., Bagnell, J.A., Srinivasa, S.S.: Chomp: covariant hamiltonian optimization for motion planning. Int. J. Robot. Res. **32**(9–10), 1164–1193 (2013)

Muhayyuddin is a Ph.D. student in Automatic Control, Robotics and Computer Vision, at the Institute of Industrial and Control Engineering, Universitat Politècnica de Catalunya (UPC), Barcelona Spain. His current research area includes physics-based motion planning for grasping and manipulation, dynamic simulations and mobile manipulation. He Received the BS degree in Computational Physics from University of the Punjab, Lahore, Pakistan in 2008 and the MS degree in Computer Science from GC University Lahore, Pakistan in 2011.

Aliakbar Akbari received the B.Sc. in Mechanical Engineering at the Islamic Azad University Parand Branch and the M.Sc. degree in Mechanical Engineering at the Eastern Mediterranean University (EMU). He is a PhD. student in Automatic Control, Robotics and Computer Vision at the Institute of Industrial and Control Engineering, Universitat Politècnica de Catalunya (UPC). His current research areas include combination of knowledge-based task and physics-based motion planning and mobile manipulation.

Jan Rosell received the BS degree in Telecommunication Engineering and the Ph.D. degree in Advanced Automation and Robotics from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, in 1989 and 1998, respectively. He joined the Institute of Industrial and Control Engineering (IOC) in 1992 where he has developed research activities in robotics. He has been involved in teaching activities in Automatic Control and Robotics as Assistant Professor since 1996 and as Associate Professor since 2001. His current technical areas include task and motion planning, mobile manipulation, and robot co-workers.