The Kautham Project: A teaching and research tool for robot motion planning

Jan Rosell, Alexander Pérez, Akbari Aliakbar, Muhayyuddin, Leopold Palomo and Néstor García
Institute of Industrial and Control Engineering (IOC)
Universitat Politècnica de Catalunya (UPC) – Barcelona Tech
Barcelona, Spain, jan.rosell@upc.edu

***Abstract*—This paper presents the software tool used at the Institute of Industrial and Control Engineering (IOC-UPC) for teaching and research in robot motion planning. The tool allows to cope with problems with one or more robots, being a generic robot defined as a kinematic tree with a mobile base, i.e. the tool can plan and simulate from simple two degrees of freedom free-flying robots to multi-robot scenarios with mobile manipulators equipped with anthropomorphic hands. The main core of planners is provided by the Open Motion Planning Library (OMPL). Different basic planners can be flexibly used and parameterized, allowing students to gain insight into the different planning algorithms. Among the advanced features the tool allows to easily define the coupling between degrees of freedom, the dynamic simulation and the integration with task planers. It is principally being used in the research of motion planning strategies for hand-arm robotic systems.**

I. Introduction

The recent software development in robotics, focused in valuable concepts of software engineering like reusability, genericity and interoperability, is greatly contributing to the improvement of the performance and capabilities of robotics systems. For instance, middleware frameworks, like ROS [1] and OROCOS [2], offer low-level control, hardware abstraction, and inter-process communications. Regarding simulation, several contributions are emerging as well. For example, we can find packages devoted to specific robotic tasks, like OpenGRASP [3] which is focused in grasping and dexterous manipulation, or others devoted to more general robotic tasks, like V-REP [4], which is focused to offer efficient real simulations including real-world physics, the simulation of sensor data, or of surface cutting or of paint or welding seams, or MoveIt! [5], which is focused in general mobile manipulation issues including motion planning, manipulation, 3D perception, kinematics, control and navigation. Regarding motion planning issues, MoveIt! uses the Open Motion Planning Library (OMPL) [6], which codes a wide set of the state-of-the-art sampling-based motion planning algorithms at an abstract level, i.e. without including issues related to robot modelling, collision-check or visualization. Regarding visualization MoveIt! can use the ROS visualizer, RViz, or Gazebo [7], a robot simulator that offers the ability to accurately and efficiently simulate populations of robots in complex environments using a robust physics engine and high-quality graphics.

At the Institute of Industrial and Control Engineering (IOC-UPC) we have been working in motion planning for many years, both in teaching and research activities. This work has lead to the development of The Kautham Project, whose aim is to provide an environment for testing and developing motion planning algorithms, since none of the existing software completely met all our needs. From the teaching perspective we required the availability of a wide range of planners, the possibility to flexibly use and parameterize planners, and the capacity to visualize 2D configuration spaces, with the aim of allowing students to easily gain insight into the different planning algorithms. From the research perspective, that we focus on motion planning strategies for hand-arm robotic systems, we required a tool to easily configure the coupling between degrees of freedom and the inclusion of dynamic simulation.

The OMPL library has been chosen as the planning core for Kautham because of its quality and the great amount of planners provided (that allow planning under geometric constraints as well as under differential constraints), including those requiring dynamic simulations. Not any other alternative could compete at this level. Although OMPL has a graphical application (OMPLapp), we have discarded it because it is currently restrained to motion planning problems involving static environments and free-flying rigid bodies in 2D and 3D. OMPL is also used in MoveIt!, but this option has not been considered because we desired an application not tied to any middleware like ROS. Besides, these two applications neither allow the visualization of the configuration space.

The paper is structured as follows. Section II briefly reviews the software tools used for the development, then the basic and advanced features are described in Sections III and IV, respectively. Finally, Section V describes its use in teaching and research at the IOC-UPC and Section VI concludes the work.

II. Tools

The Kautham Project has been developed in C++ with the open-source and cross-platform directives in mind [8]. This section lists the software tools used without discussing their choice (in some cases alternative tools could

This work was partially supported by the Spanish Government through the projects DPI2010-15446, DPI2011-22471 and DPI2013-40882-P. A. Pérez is also with the Escuela Colombiana de Ingeniería "Julio Garavito", Bogotá, Colombia.

```xml
<?xml version="1.0"?>
<Problem name="OMPL_RRTConnect_Staubli_R6_mobileBase_two_columns">
  <Robot robot="robots/TX90_mobile.dh" scale="1.0">
    <Limits name="X" min="-400.0" max="1200.0" />
    <Limits name="Y" min="-200.0" max="1000.0" />
    <Limits name="Z" min="0.0" max="1000.0" />
    <Home TH="180.0" WZ="1.0" WY="0.0" WX="0.0" Z="0.0" Y="400.0" X="-300.0"/>
  </Robot>
  <Obstacle obstacle="obstacles/columns.iv" scale="18.0">
    <Home TH="0.0" WZ="0.0" WY="0.0" WX="1.0" Z="0.0" Y="0.0" X="250.0" />
  </Obstacle>
  <Controls robot="controls/TX90_mobile.cntr" />
  <Planner>
    <Parameters>
      <Name>omplRRTConnect</Name>
      <Parameter name="Max Planning Time">10.0</Parameter>
      <Parameter name="Range">10</Parameter>
      <Parameter name="Speed Factor">1</Parameter>
      <Parameter name="Speed Factor">1</Parameter>
    </Parameters>
    <Queries>
      <Query>
        <Init dim="8">0.15 0.64 0.64 0.62 0.75 0.69 0.27 0.28</Init>
        <Goal dim="8">0.79 0.91 0.11 0.49 0.87 0.73 0.25 0.75</Goal>
      </Query>
    </Queries>
  </Planner>
</Problem>
```

Fig. 1: XML problem description file that includes information of: the robot, the obstacle, the controls and the planner.

have been selected as design options). They are the following: Qt (qt-project.org) is used for the user interface, Coin3D (www.coin3d.org) for the graphical rendering, PQP (gamma.cs.unc.edu/SSV) for the collision detection and ODE (www.ode.org) for the rigid body dynamic simulation. Input files describing the problem use the XML format (www.w3.org/XML) and the geometry of the robot and the scene is described in VRML format (web3d.org). The software uses the CMake build system (www.cmake.org) and is under the Git distributed revision control (git-scm.com). It can be downloaded and installed from sir.upc.edu/kautham.

III. Basic features

A. Problem description

Problems are defined using input XML files with four basic main sections (Fig. 1): robot(s), obstacle(s), controls and planner. The main information provided for each robot/obstacle is: a) the path to a robot/obstacle description file; b) the translational limits of motion if the robot/obstacle has a mobile base; c) the home location with respect to the world reference frame. The information for controls is optional and it consists of: a) the path of a file that describes which degrees of freedom (d.o.f.) of the robot(s) are going to be actuated and how (if not provided all d.o.f. are actuated and a control per d.o.f. is created); b) the path of a file with the same information for obstacle(s) (if not provided they are considered fixed and no controls are created). Finally, the main information provided for the planner is: a) the query to be solved (initial and goal configurations of the robot(s) and initial configuration of mobile obstacles, if any); b) the parameters of the planner.

B. Modelling robots and obstacles

Any robot will be considered, in a general way, as a tree-like kinematic structure with an optional mobile base, i.e. the configuration space will be $\mathcal{C} = \mathcal{C}_b \times \mathcal{C}_\theta$, with $\mathcal{C}_b$ being $SE3$ or null, and $\mathcal{C}_\theta = \mathbb{R}^n$, being n the number of joints of the kinematic structure ($n = 0$ for free-flying robots).

```xml
<?xml version="1.0"?>
<robot name="allegro_hand_right">
  <link name="Palm">
    <visual>
      <geometry>
        <mesh filename="AllegroHand/right/base_link_right.wrl" />
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0" />
      <material name="black">
        <color rgba="0.2 0.2 0.2 1" />
      </material>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="-0.009300 0 -0.0475" />
      <geometry>
        <box size="0.0408 0.1130 0.095" />
      </geometry>
    </collision>
  </link>
```

Fig. 2: Part of the URDF file that describes the Allegro Hand.

```xml
<?xml version="1.0"?>
<Robot name="KukaLWR" DHType="Modified" robType="Chain">
  <WeightSE3 rho_t="1.0" rho_r="1.0" />
  <Joints size="8">
    <Joint name="Base" ivFile="kukaLWR/base.wrl">
      <DHPars alpha="0.0" a="0.0" theta="0.0" d="0.0" />
      <Description rotational="false" movable="false" />
      <Limits Hi="0" Low="0" />
      <Weight weight="1.0" />
      <Parent name="" />
    </Joint>
    <Joint name="Link1" ivFile="kukaLWR/link1.wrl">
      <DHPars alpha="0.0" a="0.0" theta="0.0" d="310.0" />
      <Description rotational="true" movable="true" />
      <Limits Hi="170.0" Low="-170.0" />
      <Weight weight="1.0" />
      <Parent name="Base" />
    </Joint>
```

Fig. 3: Part of the XML file that describes a KUKA LWR robot using the modified DH parameters.

The kinematic structure will be defined using either the Universal Robotic Description Language (URDF, http://wiki.ros.org/urdf), or the Denavit-Hartenberg (DH) parameters (either standard or modified) coded in an XML file (Fig. 2 and 3). These files include the visualization models of the links as the path to a VRML file describing the geometry (in the case of URDF, the link geometry can also be defined using primitives like boxes, cylinders and spheres, or as the union of different geometries when multiple instances are provided). Also, collision models can be optionally provided in a similar way. When this is not the case, the user can choose between using the same visualization model for collision-checking (the default mode) or an oriented bounding box that is computed using the gdiam library (sarielhp.org/p/00/diameter/diam_prog.html). Boxes are stored as VRML files, not to repeat the computation if later required.

If needed, inverse kinematics should be coded for any new robot to be used (currently, the inverse kinematic for the Stäubli TX-90 and the Universal Robots UR5 robots and for the Schunk SAH mechanical hand are available). The inverse kinematics to be used is specified in the file that describes the kinematic structure of the robot.

Obstacles are described using the robot data structure. For the simple case of static obstacles, they are defined with none of the degrees of freedom being actuated. For mobile obstacles

Property	Value
Cspace Drawn	0
Incremental (0/1)	0
Speed Factor	1
Max Planning Time	10
Range	10
Simplify Solution	2

Fig. 4: Table of parameters for an RRTConnect planner (it includes the Range and other parameters general to all the planners).

```
addParameter("MinGrowTime", _MinGrowTime);
addParameter("MinExpandTime", _MinExpandTime);
addParameter("DistanceThreshold", _distanceThreshold);
addParameter("MaxNearestNeighbors", _MaxNearestNeighbors);
addParameter("BounceSteps", _BounceSteps);
```

Fig. 5: Code to add the parameters for a PRM planner contained in the creator of the PRM planner class.

they can range from simple free-flying bodies (i.e. robots with a single link and a mobile base) to complete robots. In this case a file with the controls that move the d.o.f. is provided, and their initial configuration is set in the query section of the problem description file. The configuration of mobile obstacles is controlled externally (a class called kauthamshell has been implemented to provide access from external applications and that can be connected, for example, to a ROS communication node).

C. Planners

Different planners are provided by The Kautham Project:

- Potential field-based planners: two planners are provided, one based on the navigation function NF1 [9] and the other one based on harmonic functions [10]. Although coded for n dimensions, these planners work correctly for $n \leq 3$ since they are based on a grid decomposition of the configuration space.
- Sampling-based planners: Many geometric planners from OMPL [6] are available (PRM [11], RRT [12], RRT-Connect [13], RRT* [14], SBL [15], EST [16], and KPIECE [17]) and any new OMPL planner can easily be incorporated. The Kautham Project defines a class for each planner that derives from a general abstract class that is responsible for generating the OMPL StateSpace from the description of the problem. This is done using the OMPL CompoundStateSpace class (the state space of a robot with configuration space $SE3 \times \mathbb{R}^n$ is the composition of an OMPL SE3StateSpace and an OMPL RealVectorStateSpace; and the complete state space of the multi-robot problem is the composition of the state space of each robot).

The following two features are of interest:

```
<?xml version="1.0"?>
<ControlSet>
  <Offset>
    <DOF name="bigcube/X" value="0.5" />
    <DOF name="bigcube/Y" value="0.5" />
  </Offset>
  <Control name="bigcube/x" eigValue="1.0">
    <DOF name="bigcube/X" value="1.0" />
  </Control>
  <Control name="bigcube/y" eigValue="1.0">
    <DOF name="bigcube/Y" value="1.0" />
  </Control>
</ControlSet>
```

Fig. 6: XML file describing the controls for a free-flying robot with two d.o.f. of translation.

```
<?xml version="1.0"?>
<ControlSet>
  <Offset>
    <DOF name="KukaLWR/Link1" value="0.5" />
    <DOF name="KukaLWR/Link2" value="0.5" />
    <DOF name="KukaLWR/Link3" value="0.5" />
    <DOF name="KukaLWR/Link4" value="0.5" />
    <DOF name="KukaLWR/Link5" value="0.5" />
    <DOF name="KukaLWR/Link6" value="0.5" />
    <DOF name="KukaLWR/Link7" value="0.5" />
  </Offset>
  <Control name="KukaLWR/Axis1" eigValue="1.0">
    <DOF name="KukaLWR/Link1" value="1.0" />
  </Control>
  <Control name="KukaLWR/Axis2" eigValue="1.0">
    <DOF name="KukaLWR/Link2" value="1.0" />
  </Control>
  <Control name="KukaLWR/Axis3" eigValue="1.0">
    <DOF name="KukaLWR/Link4" value="1.0" />
  </Control>
  <Control name="KukaLWR/Axis4" eigValue="1.0">
    <DOF name="KukaLWR/Link5" value="1.0" />
  </Control>
  <Control name="KukaLWR/Axis5" eigValue="1.0">
    <DOF name="KukaLWR/Link6" value="1.0" />
  </Control>
  <Control name="KukaLWR/Axis6" eigValue="1.0">
    <DOF name="KukaLWR/Link7" value="1.0" />
  </Control>
  <Control name="KukaLWR/Axis7" eigValue="1.0">
    <DOF name="KukaLWR/Link3" value="1.0" />
  </Control>
</ControlSet>
```

Fig. 7: XML file describing the controls for a KUKA robot with a fixed base.

a) The GUI is easily adaptable to new planners: Planner parameters are shown in the graphical user interface (GUI) as a table of names-values (Fig.4), and functions are provided to easily set or delete the parameters to be shown to the user (Fig.5), thus different planners can be interfaced without the need to change the GUI.

b) Controls: By default, one control is created per each joint defined as moveable, plus six for the mobile base (three for translations and three for orientations using the parameterization of quaternions detailed in [18]), provided that the problem definition file includes the translational limits of motion thus indicating that the base is mobile. This mode can be overridden by defining an XML file with the controls to be used, as shown

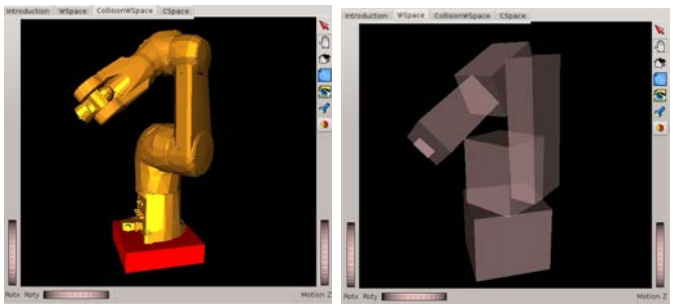

Fig. 8: Visualization and collision model tabs.

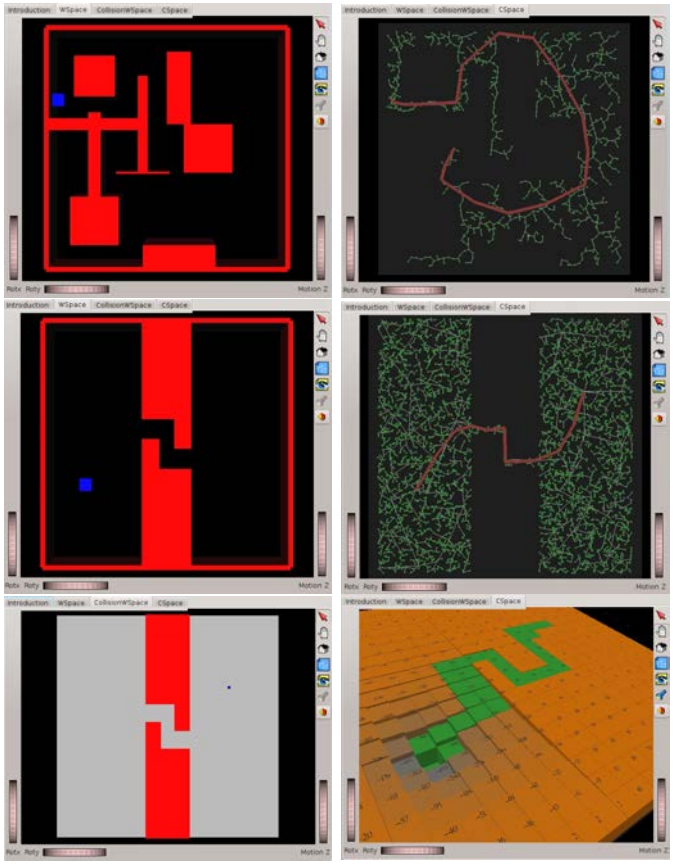

Fig. 9: Visualization of the configuration space of: a 2D maze problem solved with an RRT (top), a 2D narrow passage problem solved with a PRM (middle) and another one solved with harmonic functions (bottom).

in Fig. 6 for a free-flying robot with two translational d.o.f. and Fig. 7 for an industrial robot with a fixed base. Coupling between d.o.f. can also be defined as detailed in Section IV-A. All the controls are configured to take values in the range $[0, 1]$, and the sampling is done in this control space, which is an unitary hypercube. Alternatively, when there is a single control per degree of freedom, the OMPL samplers can also be used.

D. Visualization

The GUI shows the workspace, where the solution path is animated, in two tabs, one with the visualization models and

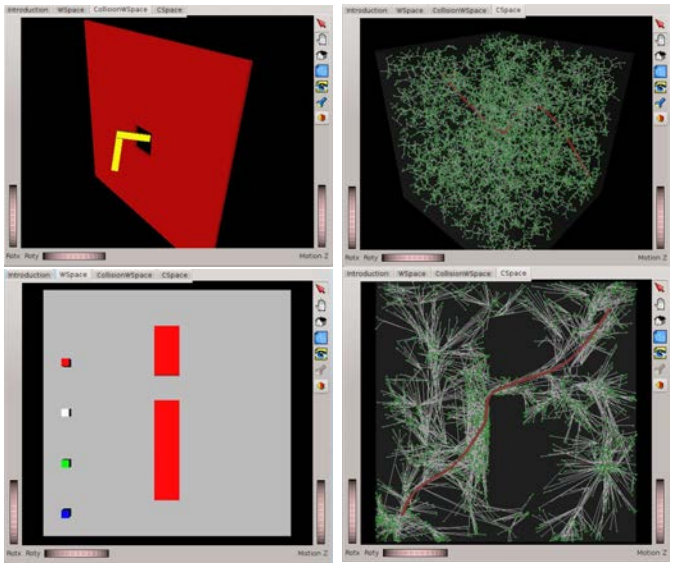

Fig. 10: Visualization of the configuration space of: a narrow passage in SE3 solved with a RRTConnect (the projection of the 6-dimensional configuration space onto the three transaltional d.o.f. is shown), and a multirobot problem solved with a RRT* (the projection of the 8-dimensional configuration space onto the two translational d.o.f. of the blue robot is shown).

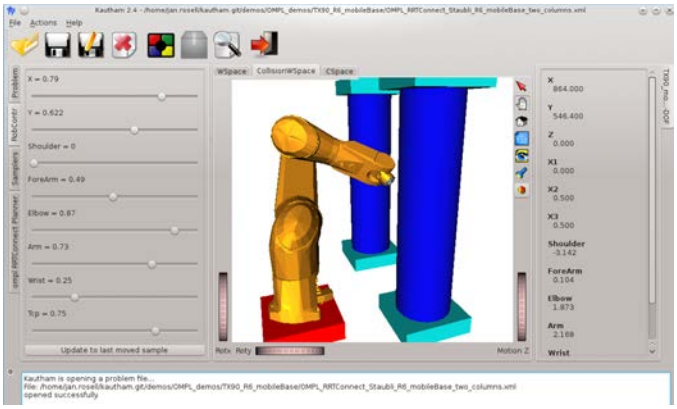

Fig. 11: Graphical User Interface.

the other with the ones used for collision checking (Fig. 8). For 2- or 3-dimensional configuration spaces, the roadmaps, trees or potential landscapes and the solution path are shown by the GUI in a separate tab (Fig. 9). For higher dimensional configuration spaces the software makes use of the OMPL projection class to show, in separate tabs, the projections of the configuration space onto the first two or three degrees of freedom of each robot (Fig. 10).

The GUI also allows to (Fig. 11): a) interactively change the control values to move the robots and the mobile obstacles (and to visualize the values of all the d.o.f.); b) obtain samples using different samplers; c) check if a sample is in collision or not and compute the distance to the obstacles; d) test if the

local planner can connect two samples; e) define the planner parameters and solve a query; f) set/unset the computation of bounding boxes and configure some visualization features.

IV. Advanced features

A. Coupling between degrees of freedom

The configuration space of a problem with m robots is the composition of the m configuration spaces $\mathcal{C}_i$:

$$\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_i \times \cdots \times \mathcal{C}_m, \tag{1}$$

where in the general case $\mathcal{C}_i = SE3 \times \mathbb{R}^{n_i}$. Configurations in $\mathcal{C}$ will be represented by vectors of dimension $d = \sum_{i=1}^{m}(6+n_i)$, with each component normalized in the range $[0, 1]$, i.e.:

$$\hat{\boldsymbol{q}} = (\hat{q}_1, \ldots, \hat{q}_d)^T \text{ with } \hat{q}_i = \frac{q_i - q_i^{\min}}{q_i^{\max} - q_i^{\min}} \tag{2}$$

The d degrees of freedom of the robotic system can be actuated either separately or in a coupled way. To model this, the following expression is used to determine a configuration $\hat{\boldsymbol{q}}$ using a vector of p controls, $(c_1, \ldots, c_p)^T$, a $d \times p$ mapping matrix, K, and a vector of offsets, $(o_1, \ldots, o_d)^T$:

$$\begin{bmatrix} \hat{q}_1 \\ \vdots \\ \hat{q}_d \end{bmatrix} = K \begin{bmatrix} c_1 - 0.5 \\ \vdots \\ c_p - 0.5 \end{bmatrix} + \begin{bmatrix} o_1 \\ \vdots \\ o_d \end{bmatrix} \tag{3}$$

Controls and offsets take values in the range $[0, 1]$, and the values $\hat{q}_i$ are forced to lie also in this range, i.e. whenever the ith row of the right-hand side of Eq. (3) is greater than 1 then $\hat{q}_i = 1$ and whenever it is below 0 then $\hat{q}_i = 0$.

The mapping matrix defines how the degrees of freedom are actuated. An identity mapping matrix indicates that all the degrees of freedom are independently actuated. A mapping matrix with some zero rows indicates that some degrees of freedom are not actuated, i.e. fixed. A mapping matrix with columns with several non-zero elements indicates that some degrees of freedom (that may be from the same robot or from different robots) are coupled. For instance, the joints of a mechanical hand can be coupled to mimic the synergies that there exist in the human hand motions [19]. This is illustrated in Fig. 12 that shows a portion of an XML file describing a column of the mapping matrix (the column is built by multiplying the eigenValue shown in the first line by the unitary vector whose components are the values of each d.o.f.). It can be seen that the control called handPMD_C1 simultaneously moves all the degrees of freedom of the fingers of the mechanical hand SAH, as shown in the snapshots of Fig 13.

Because, as stated in Section III-C, sampling is done in the control space, the dimension of the problem can be reduced when coupling between degrees of freedom is defined. For instance, in the case of the mechanical hand that mimics the human hand, the consideration of up to five controls is enough to cover a high percentage of motions. Then, the planning of the motions of a 19 d.o.f. hand-arm robotic system composed of a 6 d.o.f. industrial robot and a 13 d.o.f. mechanical hand as that considered in [19], can be performed in a 11-dimensional control space.

```
<Control name="SAH/handPMD_C1" eigValue="1.6550">
  <DOF name="SAH/Thumb" value="-0.0683" />
  <DOF name="SAH/Thumb-Base" value="-0.0683" />
  <DOF name="SAH/Thumb-Prox" value="0.2222" />
  <DOF name="SAH/Thumb-Med" value="0.3194" />
  <DOF name="SAH/Index-Base" value="0.2409" />
  <DOF name="SAH/Index-Prox" value="-0.4246" />
  <DOF name="SAH/Index-Med" value="0.2125" />
  <DOF name="SAH/Mid-Base" value="0.0" />
  <DOF name="SAH/Mid-Prox" value="-0.4884" />
  <DOF name="SAH/Mid-Med" value="0.2399" />
  <DOF name="SAH/Ring-Base" value="-0.1741" />
  <DOF name="SAH/Ring-Prox" value="-0.4451" />
  <DOF name="SAH/Ring-Med" value="0.1897" />
</Control>
```

Fig. 12: Description of a control that moves all the 13 d.o.f. of the SAH mechanical hand in a coupled way.

Fig. 13: Snapshots of the motion of the mechanical hand SAH when control handPMD_C1 is actuated.

B. Constrained motion planning

Besides the geometric planners, OMPL also includes a set of planners to plan under motion constraints. The Kautham Project defines a class for each planner and type of robot that includes the kinematic constrained model and the bounds of the valid controls. All this classes derive from a parent class that contain the state space of the robotic system, instantiated as before from the description of the problem, and procedures to compute the changes of the system state when controls are applied. This procedures are encapsulated in a class derived from the OMPL StatePropagator class and contain an instance of a template class that performs an Euler integration using a generic kinematic model of the robot that is later particularized. Any of the planers included in the OMPL control namespace can be used within The Kautham Project, although only the RRT has been currently incorporated. As an example Fig. 14 shows a path planning problem of a non-holonomic robot moving through a narrow passage solved with this planner.

C. Dynamic simulation

Dynamic simulation plays a significant role in robotics and is essential in order to achieve a more realistic behavior when there is physical interaction, like when performing grasping operations. Various physics engines are available for the simulation of rigid body dynamics such as Bullet (bullet-physics.org), PhysX (physxinfo.com) or ODE (www.ode.org).

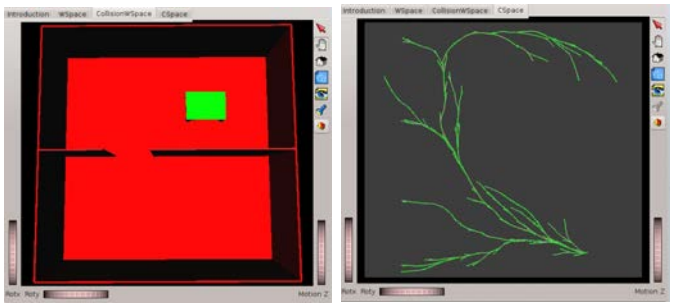

Fig. 14: Workspace and configuration space of a narrow passage problem with a non-holonomic mobile robot.

```
<Benchmark Name="experiment">
        <Problem File="OMPL_EST_2DRR_columns.xml" />
        <Problem File="OMPL_KPIECE_2DRR_columns.xml" />
        <Parameter Name="maxTime" Value="10.0" />
        <Parameter Name="maxMem" Value="100.0" />
        <Parameter Name="runCount" Value="2" />
        <Parameter Name="displayProgress" Value="true" />
        <Parameter Name="filename" Value="result.log" />
</Benchmark>
```

Fig. 15: XML file describing a benchmarking.

Even though many physics engines are used for animations and gaming purposes [20], ODE is at the core of many dynamic simulators for robotics, like Gazebo (gazebosim.org) or Webots (www.cyberbotics.com), for being efficient and stable [21]. Furthermore, the OMPL library has a state space, called OpenDE state space, designed to represent the configuration space of the dynamic environment modeled with ODE. Due to these reasons, ODE is used in The Kautham Project for dynamic simulation.

In order to consider dynamic simulation within Kautham, the dynamic information of robots/obstacles must be introduced using the URDF descriptions and a planner from the OMPL control-based suite of planners must be defined in the problem input file. Then, the dynamic environment is created by defining an ODE body for each robot/obstacle link and then generating the OMPL OpenDE state space corresponding to the set of resulting bodies (each body state has 12 dimensions, 3 for position, 3 for orientation, 3 for linear velocity and 3 for angular velocity). Planning parameters like world step size are defined in the planer section of the XML input file. The controls computed by the OMPL control-based planners are applied as forces/torques to the bodies. The goal is defined as a goal region and a distance to this goal is defined and used by the planners. When a path is obtained, it can be visualized by the GUI.

D. Integration with task planning

Task planning is the problem of finding the discrete and finite sequence of actions a robot needs to perform to achieve a goal, while motion planning copes with the computation of the motions to fulfill such actions. Task planning and motion planning need to be hierarchically coupled and simultaneously solved in order to make robots able to solve complex tasks in dynamic and semi-structured human environments.

To integrate task and motion planning, the ROS middleware is used. At the planning level, The Kautham Project has been encapsulated as a ROS service. At the task planning level, the Cognitive Robot Abstract Machine (CRAM, [22]), behaves as a ROS client. CRAM is provided as two ROS packages: the cram_language package and the cram_reasoning package. The cram_language defines the CRAM planning language (CPL) that contains extensions to the Common Lisp (CL) language especially designed for writing transparent robot control programs (CL eases the artificial intelligence planning and the interactive interface with the operator). The cram_reasoning package contains a full-featured Prolog interpreter written in CL that provides lightweight reasoning mechanisms to infer control decisions.

Therefore, with this framework, The Kautham Project planning environment can be flexibly configured (planner type, planner parameters, query to be solved,...) from a task level perspective and used when required, and the answers provided can be used as inputs for further task reasoning.

E. Benchmarking

A console application, called *Kautham Console*, is provided to run in batch mode and execute different benchmarkings, making use of the OMPL benchmarking utility. It facilitates solving a motion planning problem repeatedly with different planners, different samplers, or even differently configured versions of the same planning algorithm. It also allows the user to run the same benchmarking but with different number of repetitions or subject to different time and memory limits.

Kautham Console is configured by using an input file. This file uses XML format and contains the benchmarkings to run (Fig. 15). The information provided for each benchmarking consists on a list of paths to the problem descriptions files and the parameters to configure the benchmarking.

In every specified problem description file, a planner will be defined. It is important to assure that robots, controls, obstacles and the query to be solved are the same for all the benchmarking's files. Otherwise, the setup of the first problem will be used for the benchmarking.

Some of the benchmarking parameters that can be set up are the maximum amount of time and memory a planner is allowed to spend in planning, the number of times to run each planner, whether progress is to be displayed or not, a name for the experiment and a file path where results will be saved. Nevertheless, code provides default values for all the parameters.

Once the code has been executed, a log file for every benchmarking is generated. These files contain information about the settings of the planners, the parameters of the problem tested on, etc. To visualize this information, the user can make use of OMPL scripts to parse the log files and generate a series of plots showing the results for each planner (Fig. 16). A dump file, that can be loaded in a MySQL database, can also be generated to process the data in alternative ways.

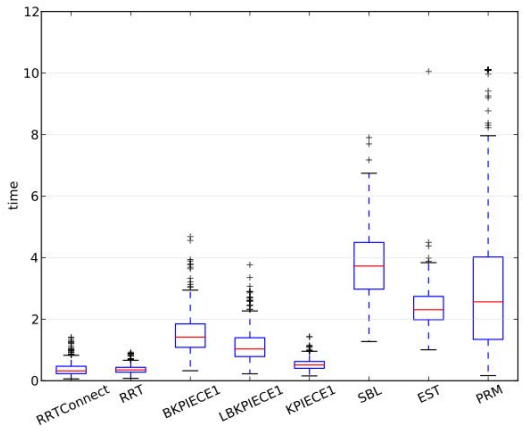

Fig. 16: Example of the benchmarking output: plot comparing the execution times (in seconds).

V. Using Kautham

A. Teaching

The Kautham Project is being used with success for the practicals of the course "Planning and Implementation of Robotic Systems" of the Master in Automatic Control and Robotics taught at the Universitat Politècnica de Catalunya. The course is mainly devoted to describing motion planning approaches, both the classic ones (based on potential fields, roadmaps or cell decompositions) and those based on sampling. The practicals have been designed to illustrate the basic potential field planners and the basic RRT and PRM sampling-based planners, evaluate their parameters and understand the resolution/probabilistic completeness of these planners:

- *Potential field methods*: Students must apply the NF1 navigation function and the Harmonic Function planner to different 2D scenarios and, in the later case, observe the potential landscape for different iteration values and different boundary conditions.
- *RRT*: Students must analyze the effect of the goal bias and of the advance step, applying the RRT to different 2D scenarios (a maze, a cluttered workspace and a narrow passage).
- *PRM*: Students must analyze the effect of the distance threshold, applying the PRM to the same 2D scenarios, and evaluate the utility of the expansion phase for narrow passages (a variant of the PRM provided by the OMPL library has been coded that permits to vary the amount of time devoted to each of the expansion and growing phases).

Optionally, final work projects are proposed that consists in reviewing some of the more recent and advanced planners offered by OMPL, like KPIECE or RRT*, comparing different planning algorithms using the benchmarking option, or analyzing the basic planers for problems with a high number of degrees of freedom. With these practicals students get a deep insight of the basic planners and get aware of the effect of their parameters.

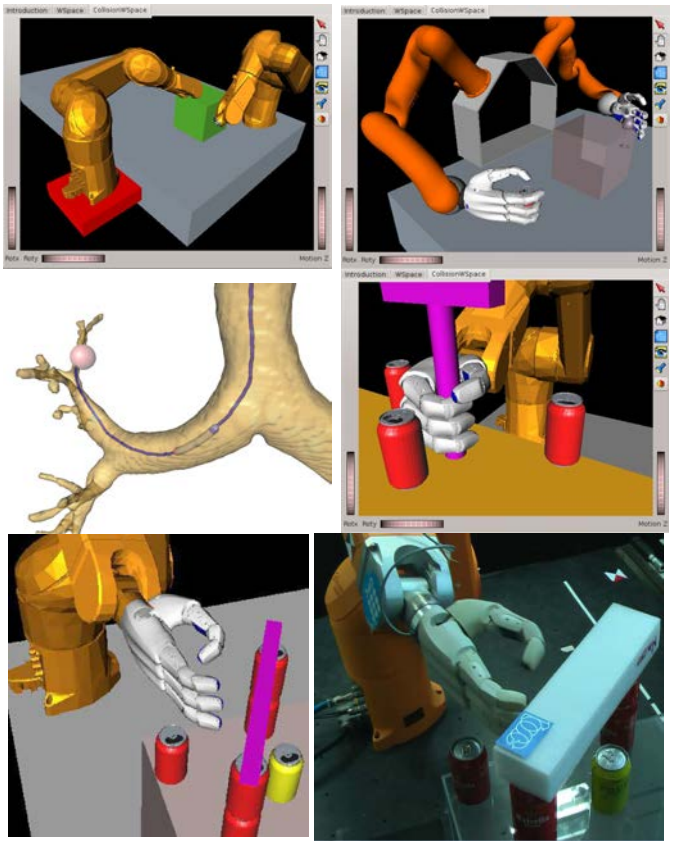

Fig. 17: Samples of complex planning problems managed with The Kautham Project: cooperation in an industrial robotic cell with a fixed and a mobile manipulator, coordination of a two-arm torso, motion of a bronchoscope modelled as a robot with kinematics constrains and grasping in cluttered environments.

Starting to work with Kautham is quite easy because many problem examples are provided ready to be run, as well as many robots and obstacles that can be used to define new scenarios.

B. Research

The Kautham Project has been used for the development of many planners related to the motion of hand-arm robotic systems, like those reported in [19][23][24][25], that reproduce in the mechanical hand the coupling between the motion of the fingers of the human hand. In a different scope, it has been used within a virtual bronchoscopy system for the motion planning of a bronchoscope (with kinematic constraints) from the trachea to a peripheral lung lesion [26].

Fig. 17 illustrates several problems solved with The Kautham Project, that shows that it has been used to design planners to solve difficult motion planning problems. Also, in some of the problems set, the solutions found have been run in real environments with successful performance.

The current research is focused in taking into account the coupling between degrees of freedom of the two arms of a robotic torso, the consideration of dynamic simulation to

obtain robust grasps, and the integration with task planning to allow the performance of tasks in human environments.

VI. Conclusions

This paper has presented a software package for the teaching and the research of robot motion planning. The package allows to cope with problems with one or more robots, which may range from simple free-flying robots to mobile manipulators equipped with mechanical hands. Based on the OMPL suite of planners, The Kautham Project offers some advanced features like the easy parametrization of planners, an easy way to define coupled degrees of freedom and tailor the planning accordingly, the dynamic simulation, the integration with task planners or the benchmarking. The software is an open source project and is available at sir.upc.edu/kautham.

References

[1] M. Quigley, B. Gekey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," *Workshop on Open Source Robotics. IEEE Int. Conf. on Robotics and Automation*, May 2009.

[2] H. Bruyninckx, P. Soetens, and B. Koninckx, "The real-time motion control core of the Orocos project," in *IEEE Int. Conf. on Robotics and Automation*, 2003, pp. 2766–2771.

[3] B. León, S. Ulbrich, R. Diankov, G. Puche, M. Przybylski, A. Morales, T. Asfour, S. Moisio, J. Bohg, J. Kuffner, and R. Dillmann, "OpenGRASP: A toolkit for robot grasping simulation," in *Simulation, Modeling, and Programming for Autonomous Robots*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6472, pp. 109–120.

[4] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira, "Virtual robot experimentation platform V-REP: A versatile 3D robot simulator," in *Simulation, Modeling, and Programming for Autonomous Robots*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6472, pp. 51–62.

[5] I. A. Suçan and S. Chitta, "MoveIt!" http://moveit.ros.org.

[6] I. A. Suçan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012.

[7] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2004, pp. 2149–2154.

[8] A. Pérez and J. Rosell, "A Roadmap to Robot Motion Planning Software Development," *Computer Applications in Engineering Education*, vol. 18, no. 4, pp. 651–660, 2010.

[9] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," in *Advanced Robotics, 1991. 'Robots in Unstructured Environments', Fifth International Conference on*, June 1991, pp. 1012–1017 vol.2.

[10] C. I. Connolly and R. A. Grupen, "The use of harmonic functions in robotics," *Journal of Robotic Systems*, vol. 10, no. 7, pp. 931–946, 1993.

[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. K. Overmars, "Probabilistic roadmaps for path planning in high - dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, August 1996.

[12] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," TR 98-11, Computer Science Dept., Iowa State University, October 1998.

[13] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2000, pp. 995–1001.

[14] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[15] G. Sanchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Int. Symp. Robotics Research*, 2001, pp. 403–417.

[16] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 3, Apr 1997, pp. 2719–2726 vol.3.

[17] I. Suçan and L. Kavraki, "A sampling-based tree planner for systems with complex dynamics," *Robotics, IEEE Transactions on*, vol. 28, no. 1, pp. 116–131, Feb 2012.

[18] J. J. Kuffner, "Effective sampling and distance metrics for 3D rigid body path planning," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 4, 2004, pp. 3993–3998.

[19] J. Rosell, R. Suárez, C. Rosales, J. A. García, and A. Pérez, "Motion planning for high dof anthropomorphic hands," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2009, pp. 4025–4030.

[20] T. Y. Yeh, G. Reinman, S. J. Patel, and P. Faloutsos, "Fool me twice: Exploring and exploiting error tolerance in physics-based animation," *ACM Trans. Graph.*, vol. 29, no. 1, pp. 5:1–5:11, Dec. 2009.

[21] E. Drumwright, J. Hsu, N. Koenig, and D. Shell, "Extending open dynamics engine for robotics simulation," in *Simulation, Modeling, and Programming for Autonomous Robots*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6472, pp. 38–50.

[22] M. Beetz, D. Jain, L. Mösenlechner, and M. Tenorth, "Towards Performing Everyday Manipulation Activities," *Robotics and Autonomous Systems*, vol. 58, no. 9, pp. 1085–1095, 2010.

[23] R. Suárez, J. Rosell, A. Pérez, , and C. Rosales, "Efficient search of obstacle-free paths for anthropomorphic hands," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and System*, 2009, pp. 1773–1778.

[24] J. Rosell, R. Suárez, C. Rosales, and A. Pérez, "Autonomous motion planning of a hand-arm robotic system based on captured human-like hand postures," *Autonomous Robots*, vol. 31, no. 1, pp. 87–102, 2011.

[25] J. Rosell, R. Suárez, and A. Pérez, "Path planning for grasping operations using an adaptive PCA-based sampling method," *Autonomous Robots*, vol. 35, no. 1, pp. 27–36, 2013.

[26] J. Rosell, A. Pérez, P. Cabras, and A. Rosell, "Motion planning for the virtual bronchoscopy," in *IEEE Int. Conf. on Robotics and Automation*, 2012, pp. 2932–2937.