The Kautham Project

A Software Tool for Robot
Motion Planning and Simulation

Service and Industrial Robotics
Master's Degree in Industrial Engineering
Planning and Implementation of Robotic Systems
Master's degree in Automatic Control and Robotics

Contents

1. Introduction

1. Objective and requirements

The **objective** is to provide an environment for testing and developing motion planning algorithms.

Teaching perspective: Allow students to easily gain insight into the different planning algorithms

- Availability of a wide range of planners,
- Possibility to flexibly use and parameterize planners
- Capacity to visualize 2D and 3D configuration spaces, as well, as projections of higher-dimensional configuration spaces.

Research perspective: Ease the development of motion planning strategies for hand-arm robotic systems

- Simple configurability of the coupling between degrees of freedom
- Dynamic simulation
- Benchmarking capabilities

- General design requirements:
  - Multiplatform and open source
  - Different interfaces (GUI, Console, ROS)
- Simulation requirements:
  - Geometric modelling
  - Visualization (workspace, configuration space)
  - Kinematics and dynamics
- Planning requirements:
  - Samplers (random, deterministic)
  - Collision-detection
  - Problem description using text files
  - Math utilities (Transform, PCA, …)
  - Benchmarking
  - Integration of OMPL planners

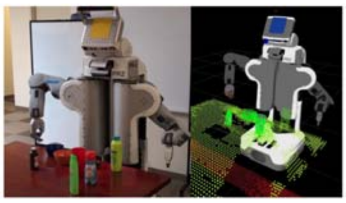
240IAU33 – Service and Industrial Robotics / 240AR032 - Planning and Implementation of Robotic Systems Jan Rosell

1. Introduction

2. Alternatives

MoveIt! is a software framework for motion planning in ROS.

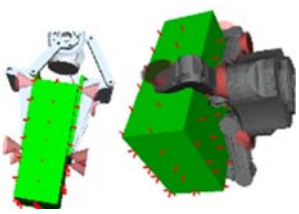
Uses **OMPL** as its core Motion Planning library.



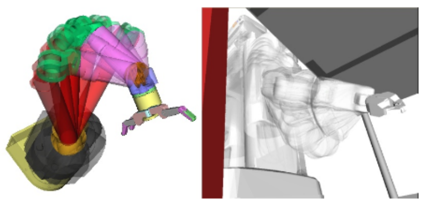
moveit.ros.org

Open Robotics Automation Virtual Environment (**OpenRAVE**) provides an environment for testing and developing motion planning algorithms

Offers some sampling-based planning algorithms for robot arms and also integrates **OMPL**. It has grasping and inverse kinematics modules.



Manages compiled inverse kinematics files for robots using ikfast.



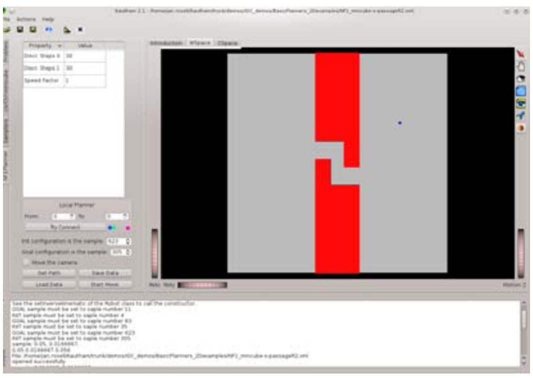
openrave.org

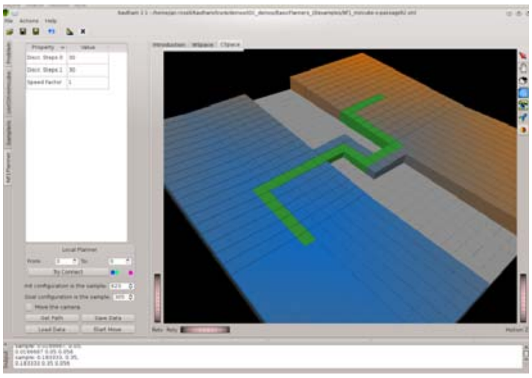
Master's Degree in Industrial Engineering / Master's degree in Automatic Control and Robotics

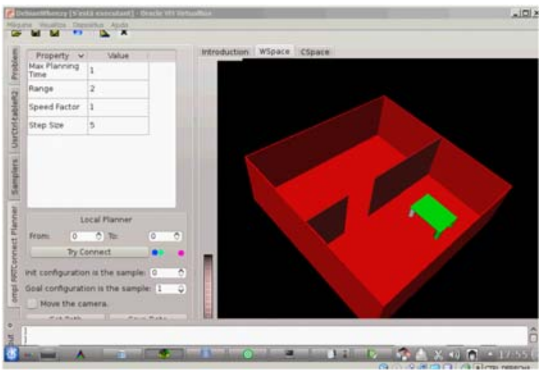
Contents

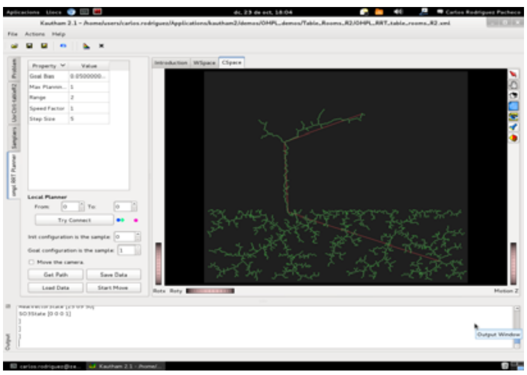
2. Results

1. Simple examples



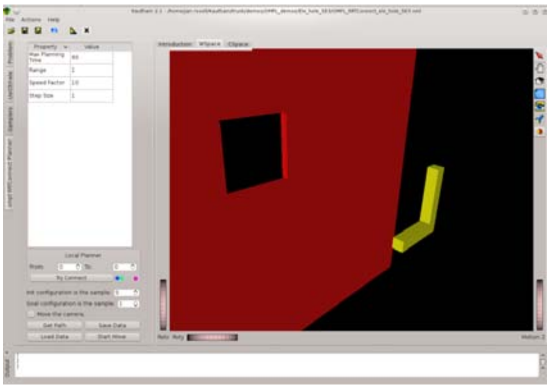
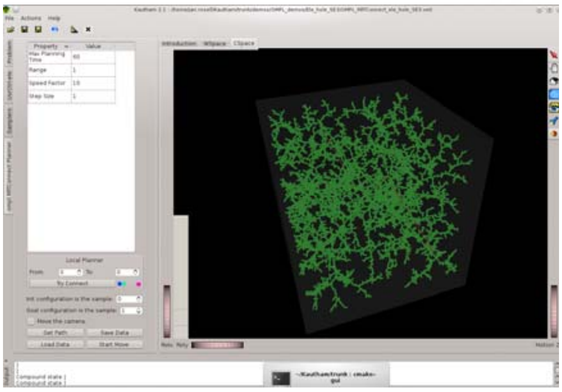
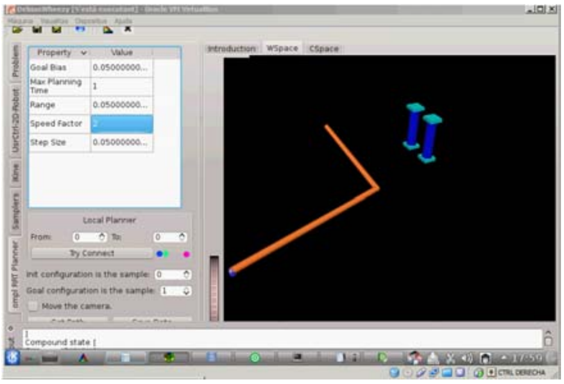
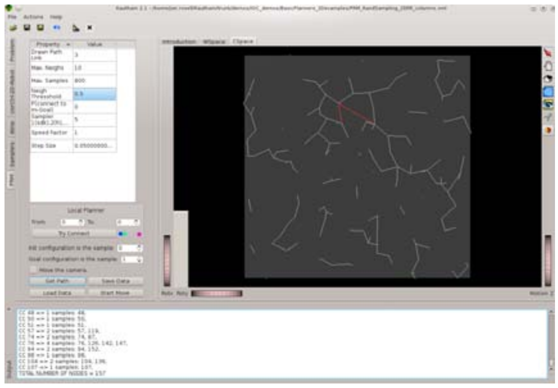






2. Results

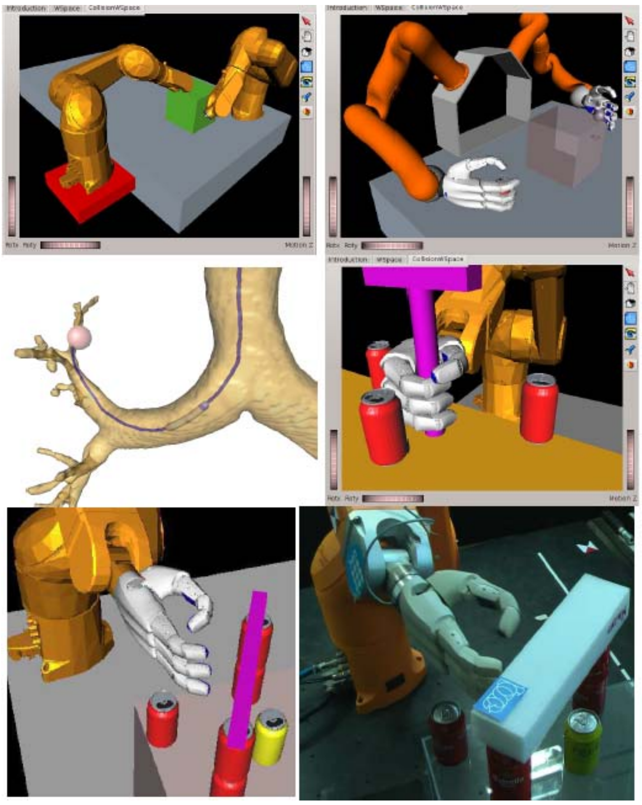
1. Simple examples

240IAU33 – Service and Industrial Robotics / 240AR032 - Planning and Implementation of Robotic Systems Jan Rosell

2. Results

2. Complex examples



- Cooperation with multiple robots
- Human-like motions for dual-arm anthropomorphic systems
- Path planning for bronchoscopes
- Human-like motions for mechanical hands
- Path planning of hand-arm systems in cluttered environments

Master's Degree in Industrial Engineering / Master's degree in Automatic Control and Robotics

2. Results

2. Complex examples

Contents

3. Basic features

1. Problem description

XML file with four basic main sections

Robot(s)

Obstacle(s)

Controls

Planner

```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>
```


3. Basic features

2. Modelling robots and obstacles

Obstacles: defined as robots (fixed obstacles will have no controls defined)

Robots: tree-like kinematic structure with an optional mobile base ($SE3 \times \mathbb{R}^n$) defined using

a) Universal Robotic Description Language (URDF)

```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>
```

b) Denavit-Hartenberg (DH) parameters coded in XML

```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>
```

Include: the path to a file describing the geometry of the rendering models and (optionally) the collision models and an inverse kinematic tag.

Supported geometry formats: The native supported format is VRML. If the ASSIMP package is available, many others like DAE or STL are also supported.

3. Basic features

3. Planners

Type of planners:

- Potential field-based planners: NF1, HF
- Sampling-based planners (OMPL): PRM, RRT, RRTconnect, RRT*, SBL, EST, KPIECE

Features:

a) The GUI is easily adaptable to new planners

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

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

b) Controls

- *Default*: one control per each joint defined as moveable, plus six for the mobile base (provided that the problem definition file includes the translational limits of motion thus indicating that the base is mobile)
- *Defined by an XML file*

3. Basic features

3. Planners

Industrial robot with a fixed base

```xml
<?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>
```

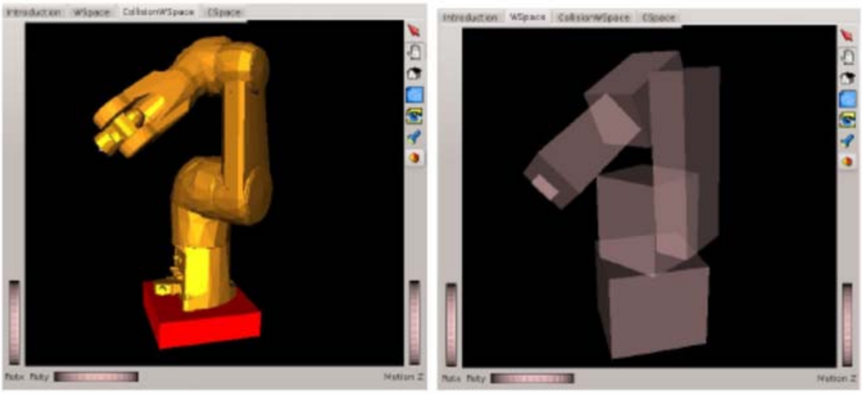
Free-flying robot with two translational d.o.f.

```xml
<?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>
```

3. Basic features

4. Visualization

1. Visualization of the workspace



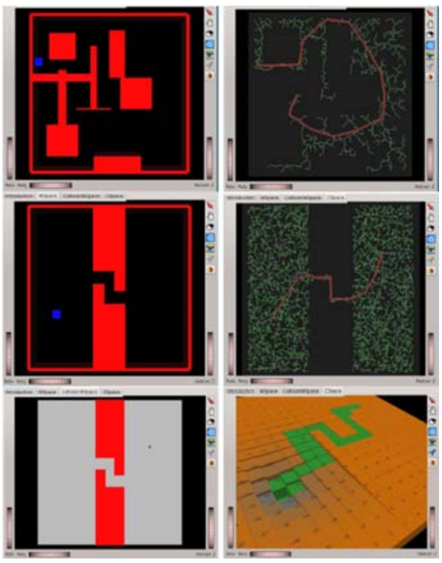
- The workspace can be visualized in two different tabs. The *WSpace* tab shows the rendering models and the *CollisionWSpace* tab shows the collision models (oriented bounding boxes can be computed if no collision models are provided).
- The translational trajectories of the end-effectors can also be shown in the *WSpace* tab.

240IAU33 – Service and Industrial Robotics / 240AR032 - Planning and Implementation of Robotic Systems Jan Rosell

3. Basic features

4. Visualization

1. Visualization of the workspace
2. Visualization of the configuration space (*CSpace* tab)
   - *For 2- or 3-dimensional problems*: the roadmaps, trees or potential landscapes
   - *For higher dimensional problems*: the projections of the configuration space onto the first two or three degrees of freedom of each robot .

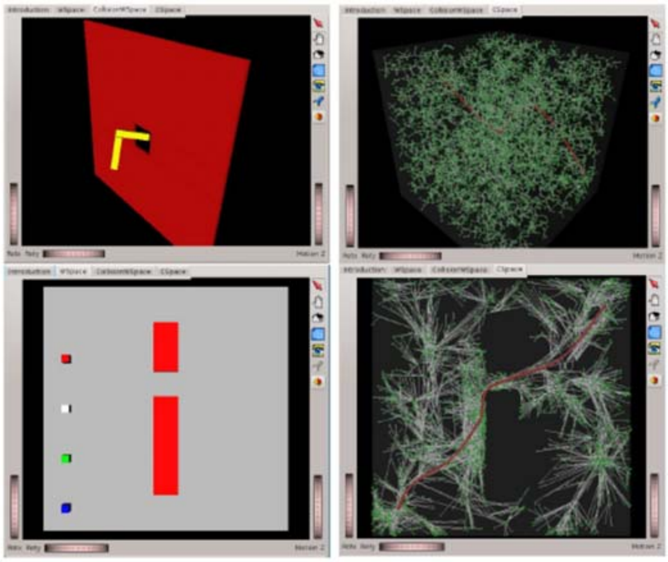


Master's Degree in Industrial Engineering / Master's degree in Automatic Control and Robotics

3. Basic features

4. Visualization

1. Visualization of the workspace
2. Visualization of the configuration space (*CSpace* tab)
   - *For 2- or 3-dimensional problems*: the roadmaps, trees or potential landscapes
   - *For higher dimensional problems*: the projections of the configuration space onto the first two or three degrees of freedom of each robot .



Contents

4. Advanced features

1. Coupling between degrees of freedom

Configuration space for *m* robots: $\mathcal{C} = \mathcal{C}_1 \times \cdots \times \mathcal{C}_i \times \cdots \times \mathcal{C}_m$

$\mathcal{C}_i = SE3 \times \mathbb{R}^{n_i}$

Dimension *d*: $d = \sum_{i=1}^{m} (6+n_i)$

Configuration with normalized components in [0,1]: $\hat{q} = (\hat{q}_1, \ldots, \hat{q}_d)^T$ with $\hat{q}_i = \frac{q_i - q_i^{\min}}{q_i^{\max} - q_i^{\min}}$

It can be determined using a vector of *p* of controls, and a *dxp* mapping matrix, K.

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

Controls and offsets take values in the range [0, 1]

The mapping matrix defines how the degrees of freedom are actuated:

- An identity mapping matrix indicates that all the d.o.f. independently actuated.
- A mapping matrix with some zero rows indicates that some d.o.f. are not actuated.
- A mapping matrix with columns with several non-zero elements indicates coupling between d.o.f.

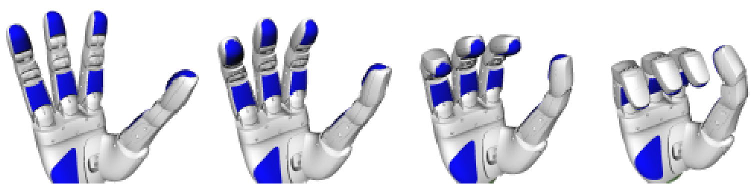
4. Advanced features

1. Coupling between degrees of freedom

XML file defining one of the columns of matrix K that determines the coupling between the d.o.f. of a mechanical hand.

```
<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>
```

Associated coupled motion



4. Advanced features

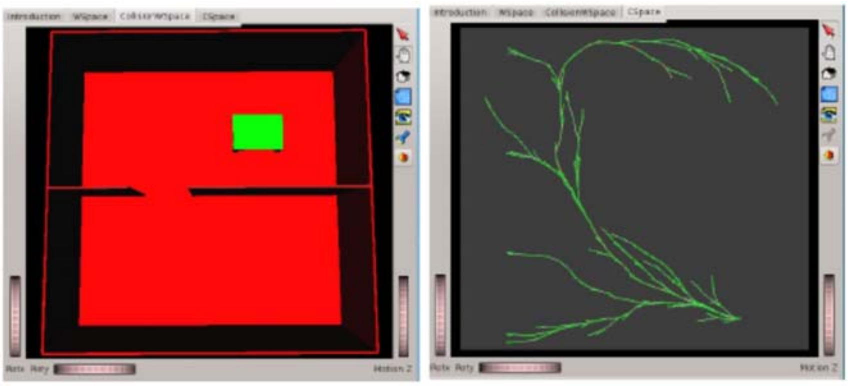
2. Constrained motion planning

OMPL has two sets of planners: *geometric* and *based on controls*

The ones based on controls allow to plan under motion constraints.

The Kautham Project defines a class for each planner and type of robot that includes:

- The kinematic constrained model
- The bounds of the valid controls

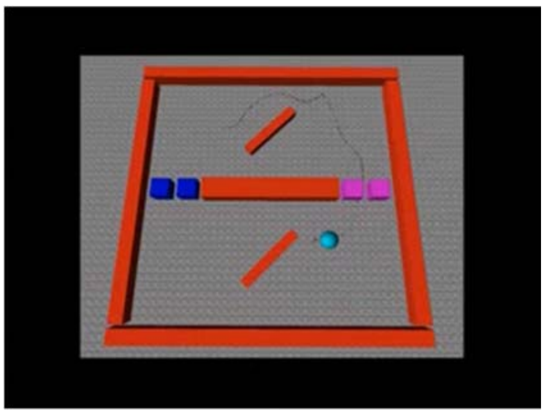


4. Advanced features

3. Dynamic simulation

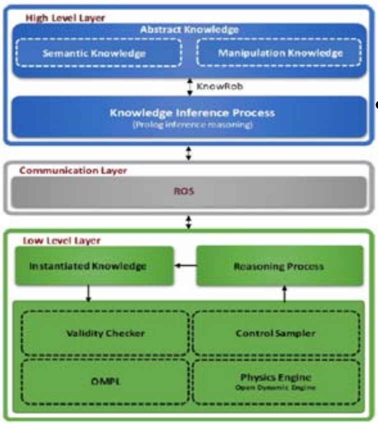
- ODE is used in The Kautham Project for dynamic simulation.
- OMPL library has the OpenDE *state space*, designed to represent the Cspace of the dynamic environment modeled with ODE.
- Dynamic information of robots/obstacles must be introduced using the URDF descriptions. Then, the dynamic environment 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 bodies.

```
<?xml version="1.0"?>
<robot name="SphereBlueDE">
    <link name="base">
    <inertial>
         <origin xyz="0 0 0" rpy="0 0 0"/>
         <mass value="0.1"/>
         <inertia ixx="0.36"  ixy="0"  ixz="0" iyy="0.36" iyz="0" izz="0.36" />
    </inertial>
    <visual>
        <origin xyz="0 0 0" rpy="0 0 0" />
            <geometry>
                <sphere radius="0.009"/>
            </geometry>
            <material>
                <color rgba="0 0 1 1" />
            </material>
    </visual>
    </link>
</robot>
```



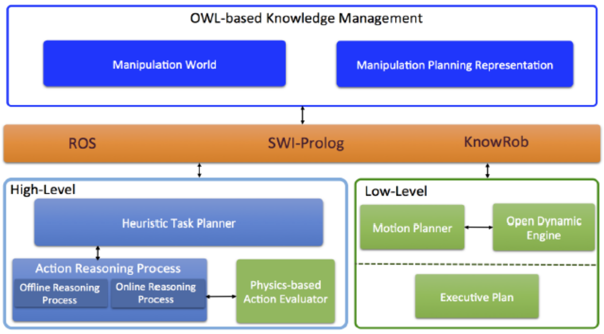
4. Advanced features

4. Incorporating knowledge and task planning



- *Enhancing planners with high-level knowledge*:

  Using the ROS middleware, Kautham is encapsulated as a motion planning ROS client and a Prolog-based knowledge-based manager as a ROS server.



- *Integration of task and motion planning*:

  Using the ROS middleware, Kautham is encapsulated as a motion planning ROS service and a Prolog-based task planner as a ROS client.

240IAU33 – Service and Industrial Robotics / 240AR032 - Planning and Implementation of Robotic Systems Jan Rosell

4. Advanced features

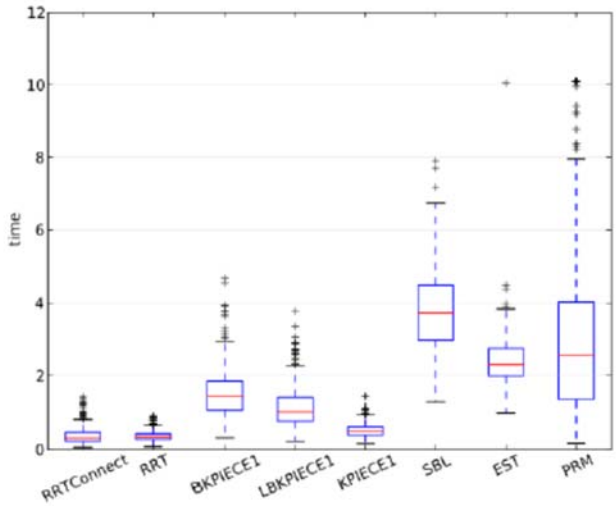
5. Benchmarking

- A console application, called *kautham-console*, is provided to run in batch mode and execute different benchmarkings, making use of the OMPL benchmarking utility.

- Solves a motion planning problem repeatedly with different planners, different samplers, or even differently configured versions of the same planning algorithm.

```
<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>
```

```
<Benchmark Name="experiment">
    <Problem File="OMPL_PRM_bigcube-maze2D_R2_r005.xml" PlannerAlias="PRM_r005" />
    <Problem File="OMPL_PRM_bigcube-maze2D_R2_r015.xml" PlannerAlias="PRM_r015" />
    <Problem File="OMPL_PRM_bigcube-maze2D_R2_r050.xml" PlannerAlias="PRM_r050" />
    <Problem File="OMPL_PRM_bigcube-maze2D_R2_h005.xml" PlannerAlias="PRM_h005" />
    <Problem File="OMPL_PRM_bigcube-maze2D_R2_h015.xml" PlannerAlias="PRM_h015" />
    <Problem File="OMPL_PRM_bigcube-maze2D_R2_h050.xml" PlannerAlias="PRM_h050" />
    <Parameter Name="maxTime" Value="1.5" />
    <Parameter Name="maxMem" Value="8000" />
    <Parameter Name="runCount" Value="10" />
    <Parameter Name="displayProgress" Value="true" />
    <Parameter Name="filename" Value="resultPRM.log" />
</Benchmark>
```



UPC ETSEIB Master’s Degree in Industrial Engineering / Master's degree in Automatic Control and Robotics

Contents

5. The code

1. General Structure

demos
- **IOC_grid_demos**
- **OMPL_ctrl_demos**
- **OMPL_geo_demos**
- **OMPL_ode_demos**
- **models**
    - **robots**
    - **obstacles**

apps
src
- external
- planner
- problem
- sampling
- util

The demos folder contains the demos related to the different families of planners, and the model folder with the robots, the obstacles.

IOC grid demos: Folder with basic 2D examples using potential-field planners.

OMPL demos: Folders with examples using sampling-based planners (based on controls, geometric, and physics-based). Each folder is divided in several subfolders, one for each problem, containing the problem files defined to solve the problem with different planners. Optionally there are folders describing the controls used.

Models: Contains folders describing robots and obstacles. The structure of the robots is defined with *.dh* or *.urdf* files, and the geometry with *.iv* or *.wrl* files (if the ASSIMP package is available then *.dae*, *.stl* or other file types can also be used) .

5. The code

1. General Structure

demos
apps
  console
  graphical
  ros
src
  external
  planner
  problem
  sampling
  util

The apps folder contains the three Kautham applications: Console, GUI and ROS node.

console folder:
Class benchmark: Used to define OMPL benchmarks.
kauthamplanner_console application file: The main function that runs Kautham without graphical output. It is used to launch benchmarks or run some problems.

graphical folder:
Widget classes: Classes defining the widgets of the GUI
GUI application file: The main function to run the GUI.

ros folder:
Kautham ROS node: Provide access, using ROS services, to all the utilities offered by Kautham through the kauthamshell class.
Messages and services in msg and srv subfolders.

240IAU33 – Service and Industrial Robotics / 240AR032 - Planning and Implementation of Robotic Systems Jan Rosell

5. The code

1. General Structure

demos
apps
src
  kauthamshell
  external
    gdiam
    libann
    pqp
    pugixm
  planner
  problem
  sampling
  util

Class kauthamshell: Class that encapsulates all the Kautham classes. The console and the ROS applications use it to get access to all the Kautham classes.

Master's Degree in Industrial Engineering / Master's degree in Automatic Control and Robotics

5. The code

1. General Structure

demos
apps
src
kauthamshell
external
drawstuff
gdiam
pqp
pugixm
planner
problem
sampling
util

The external folder contains third party libraries:

Drawstuff: Graphical viewer for physics-based planners (where ODE is used as state propagator).

Gdiam: Library used to compute bounding volumes for collision checking.

PQP: Collision detection library.

Pugixml: Library used to parse input XML files.

240IAU33 – Service and Industrial Robotics / 240AR032 - Planning and Implementation of Robotic Systems Jan Rosell

5. The code

1. General Structure

demos
apps
src
kauthamshell
external
planner
ioc
omplc
omplg
omplOpenDE
problem
sampling
util

The planner folder contains the source files of the planners: it includes the abstract class planner and the planners libraries *ioc*, *omplc* and *omplg*.

ioc: Includes potential field planners computed on grids (NF1 and HF).

Includes the following classes:
- gridPlanner, HFPlanner, NF1Planner.

Master's Degree in Industrial Engineering / Master's degree in Automatic Control and Robotics

5. The code

1. General Structure

demos
apps
src
  kauthamshell
  external
  planner
    ioc
    omplc
    omplg
    omplOpenDE
  problem
  sampling
  util

The planner folder contains the source files of the planners: it includes the abstract class planner and the planners libraries *ioc*, *omplc* and *omplg*.

omplg: Contains the planners based on the geometric OMPL planners. Includes the following classes:

- Abstract planner class (*omplgplanner*): Sets the state space and the solve function.
- Derived wrapper classes for standard planners: *omplESTplanner*, *omplKPIECEplanner*, *omplPRMplanner*, *omplRRTplanner*, *omplRRTConnectplanner*, *omplLazyRRTplanner omplRRTStarplanner*, *omplpRRTplanner*, *omplSBLplanner*.
- omplValidityChecker
- Auxiliar classes to define optimization objectives, synergies and vector fields.

Some planners like PRM, TRRT, RRTStar planners has been slightly modified to be able to tune some parameters that are fixed in the OMPL implementations.

5. The code

1. General Structure

demos
apps
src
  kauthamshell
  external
  planner
    ioc
    omplc
    omplg
    omplOpenDE
  problem
  sampling
  util

The planner folder contains the source files of the planners: it includes the abstract class planner and the planners libraries *ioc*, *omplc* and *omplg*.

omplc: Contains the planners based on the control OMPL planners. Includes the following classes:

- General classes: Eulerintegrator, KinematicRobotModel and omplcPlannerStatePropagator.
- Abstract planner class (*omplcplanner*): Sets the solve function and the state space. It has two derived classes, omplcRRTplanner and omplcSSTplanner, that set the parameters for the OMPL RRT and SST planners, resepctively.
- Classes that particularize the RRT and SST planners to use the kinematic constraints of a car and of a f16 plane (coded in classes KinematicF16Model and KinematicCarModel that derive from KinematicRobotModel class).

5. The code

1. General Structure

demos
apps
src
kauthamshell
external
planner
ioc
omplc
omplg
omplOpenDE
problem
sampling
util

The planner folder contains the source files of the planners: it includes the abstract class planner and the planners libraries *ioc*, *omplc* and *omplg*.

omplOpenDE: Contains the physics-based planners using the control OMPL planners in the OpenDE state space. Includes the following main classes:

- *KauthamOpenDEEnvironment*: Class that creates the ODE world. Its derived classes define the contact modelling, the validity checker, the projections, the distance function, the control space and how controls are applied.
- *KauthamOpenDEGoalRegion*: Class to define the distance to goal function.
- Planner classes: *KauthamOpenDEPlanner* base class for physics-based planners and derived classes particularizing the robot and the environment.
- *InstantiatedKnowledge:* Class that stores information on how to manipulate objects.

240IAU33 – Service and Industrial Robotics / 240AR032 - Planning and Implementation of Robotic Systems Jan Rosell

5. The code

1. General Structure

demos
apps
src
kauthamshell
external
planner
problem
sampling
util

The classes to set the scene and the problem to be solved.

- Classes element, ivelement, ivpqpelement, odeelement
- Class link
- Classes robot, urdf
- Class obstacle
- Classes workspace, ivworkspace
- Class problem

Master's Degree in Industrial Engineering / Master's degree in Automatic Control and Robotics

5. The code

1. General Structure

demos
apps
src
kauthamshell
external
planner
problem
sampling
util

The classes to define configurations and the samples

- Classes conf, rnconf, se2conf, se3conf, robconf.
- Classes sample, sdksample.
- Class sampleset.
- Classes sampler, randomsampler, haltonsampler, sdksampler gaussiansampler, gaussianlikesampler.

5. The code

1. General Structure

demos
apps
src
kauthamshell
external
planner
problem
sampling
util

Auxiliary classes.

Libmt: library for the management of transformations.

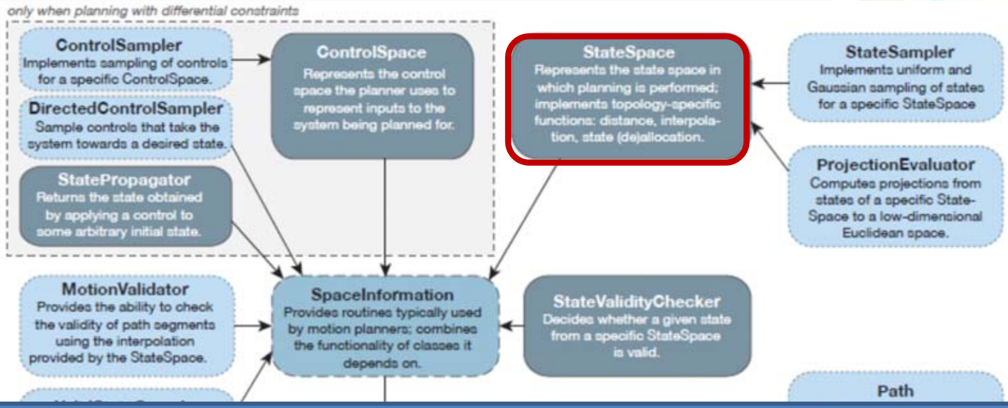
Kthutil: Basic structures and enumerates.

Libkin: inverse kinematics for some robots.

5. The code

2. OMPL: integration issues

State space




```
//! Constructor
omplPlanner::omplPlanner
```

```
//create the SE3 state space
spaceSE3[i] = ((ob::StateSpacePtr) new ob::SE3StateSpace());
```

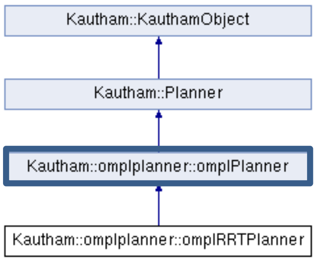
```
//load to the compound state space of robot i
compoundspaceRob.push_back(spaceSE3[i]);
weightsRob.push_back(1);
```

```
//create the Rn state space
spaceRn[i] = ((ob::StateSpacePtr) new weigthedRealVectorStateSpace(nj));
```

```
//load to the compound state space of robot i
compoundspaceRob.push_back(spaceRn[i]);
weightsRob.push_back(1);
```

```
//the compound state space for robot i is (SE3xRn), and either SE3 or Rn may be missing
spaceRob[i] = ((ob::StateSpacePtr) new ob::CompoundStateSpace(compoundspaceRob,weightsRob));
```

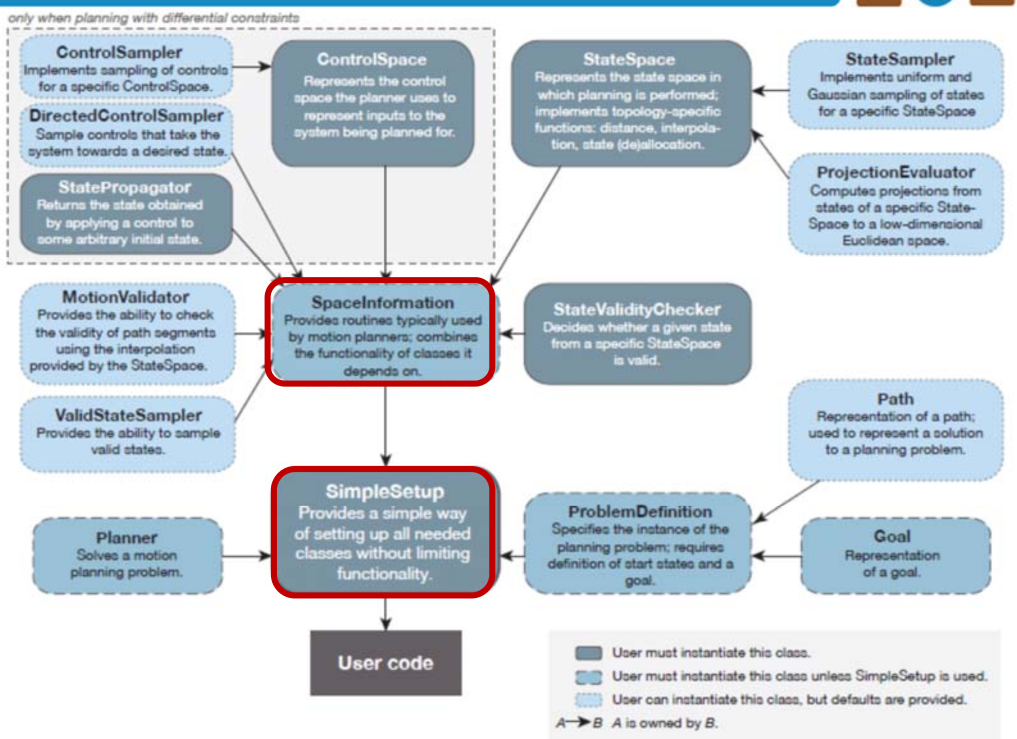
```
//the state space for the set of robots. All the robots have the same weight.
space = ((ob::StateSpacePtr) new ob::CompoundStateSpace(spaceRob,weights));
```



5. The code

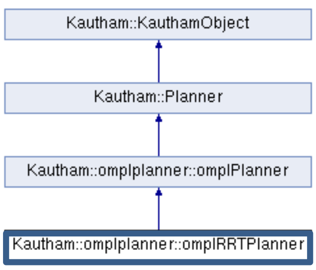
2. OMPL: integration issues

The simple setup




```
//! Constructor
omplRRTPlanner::omplRRTPlanner
```

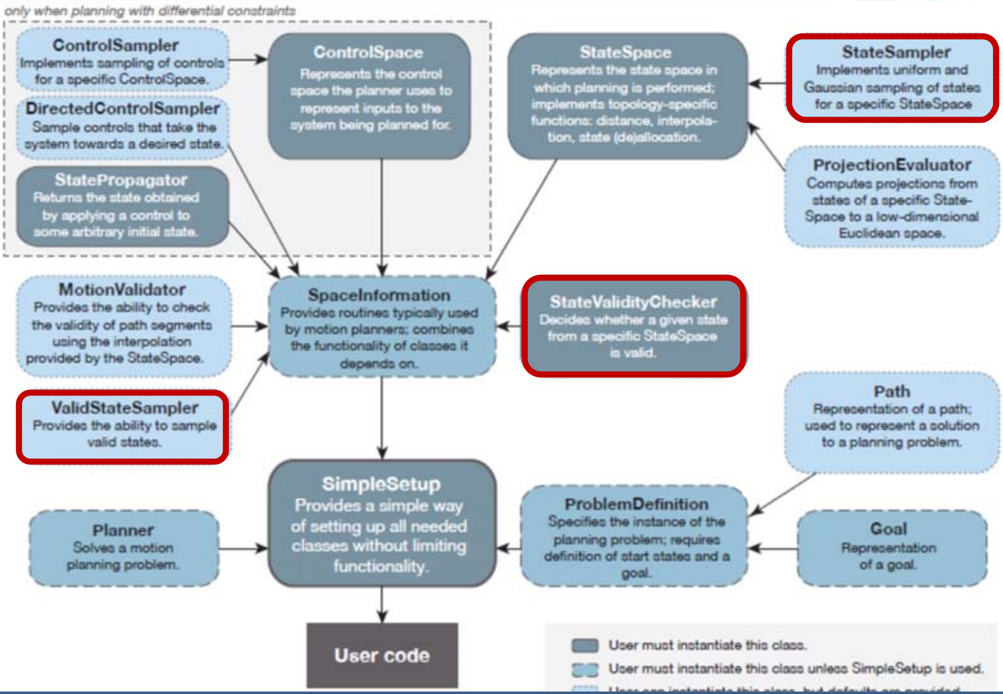
```
//create simple setup
ss = ((og::SimpleSetupPtr) new og::SimpleSetup(space));
ob::SpaceInformationPtr si=ss->getSpaceInformation();
```

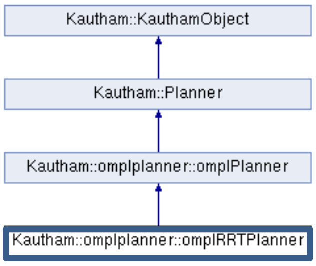


5. The code

2. OMPL: integration issues

The validity checker





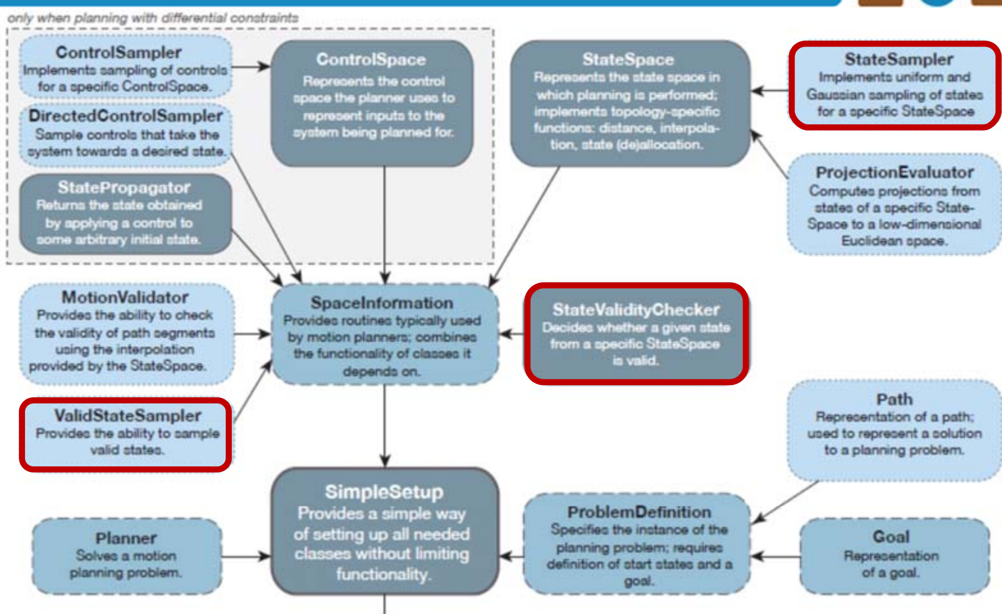

```
//! Constructor
omplRRTPlanner::omplRRTPlanner
```

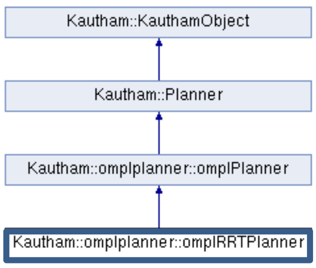
```
//set validity checker
ss->setStateValidityChecker(boost::bind(&omplplanner::isStateValid, si.get(), _1, (Planner*)this));
//alloc valid state sampler
si->setValidStateSamplerAllocator(boost::bind(&omplplanner::allocValidStateSampler, _1, (Planner*)this));
//alloc state sampler
space->setStateSamplerAllocator(boost::bind(&omplplanner::allocStateSampler, _1, (Planner*)this));
```


5. The code

2. OMPL: integration issues

The validity checker



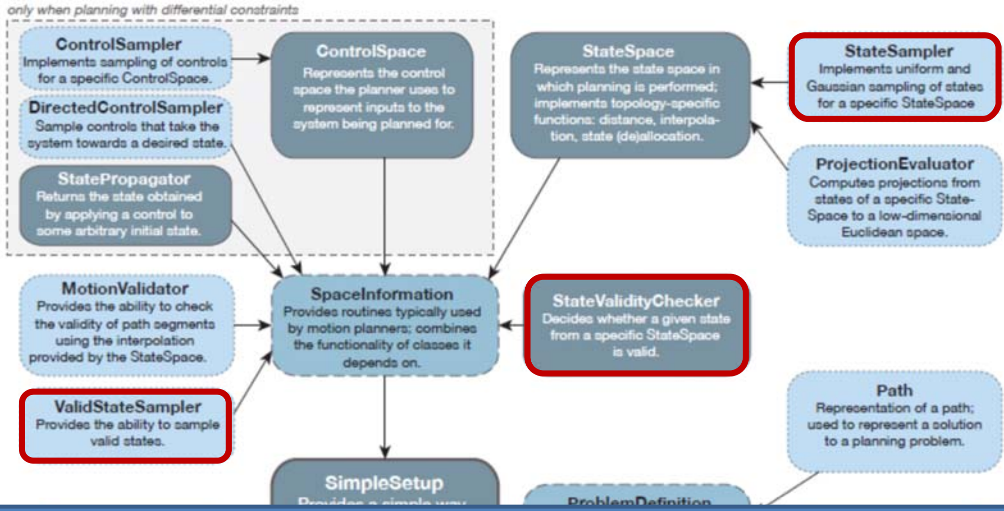


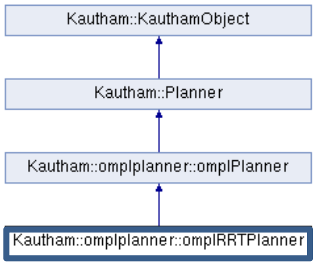

```
////////////////////////////////////////////////////////////////////////////////////////
//! This function is used to allocate a state sampler
ob::StateSamplerPtr allocStateSampler(const ob::StateSpace *mysspace, Planner *p)
{
    return ob::StateSamplerPtr(new KauthamStateSampler(mysspace, p));
}

////////////////////////////////////////////////////////////////////////////////////////
//! This function is used to allocate a valid state sampler
ob::ValidStateSamplerPtr allocValidStateSampler(const ob::SpaceInformation *si, Planner *p)
{
    return ob::ValidStateSamplerPtr(new KauthamValidStateSampler(si, p));
}
```

5. The code

2. OMPL: integration issues

The validity checker





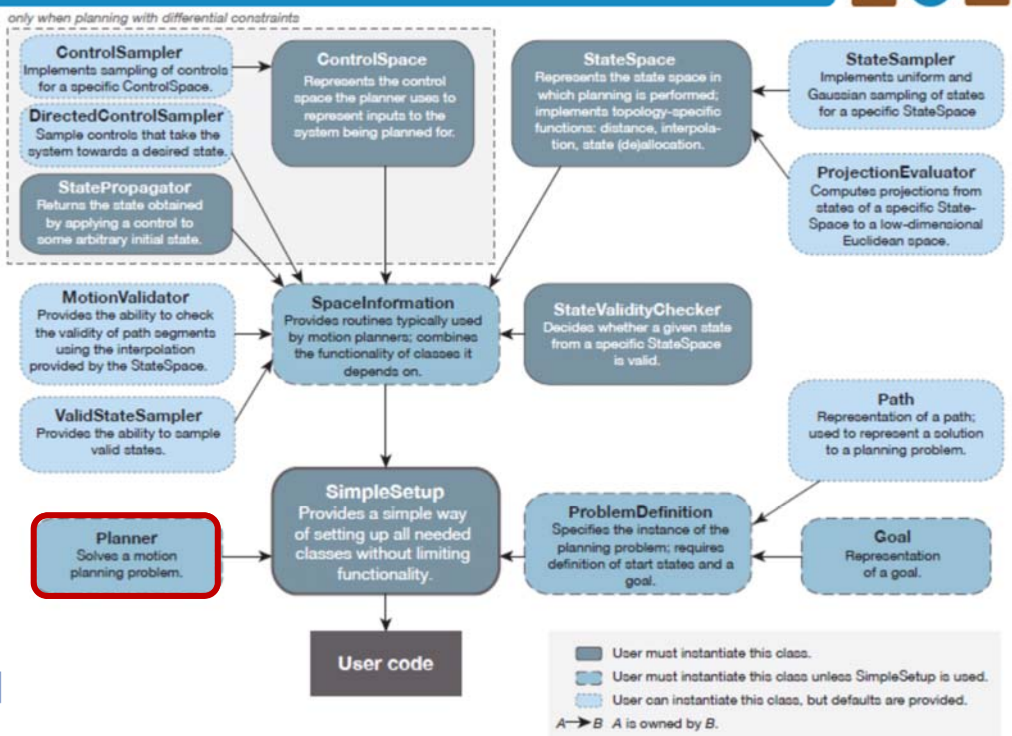

```
//! This function converts a state to a smp and tests if it is in collision or not
bool isStateValid(const ob::SpaceInformation *si, const ob::State *state, Planner *p)
{
    //verify bounds
    if(si->satisfiesBounds(state)==false)
        return false;
    //create sample
    int d = p->wkSpace()->getDimension();
    Sample *smp = new Sample(d);
    //copy the conf of the init smp. Needed to capture the home positions.
    smp->setMappedConf(p->initSamp()->getMappedConf());
    //load the RobConf of smp form the values of the ompl::state
    ((omplPlanner*)p)->omplState2smp(state,smp);
    //collision-check
    if( p->wkSpace()->collisionCheck(smp) )
        return false;
    return true;
}
```

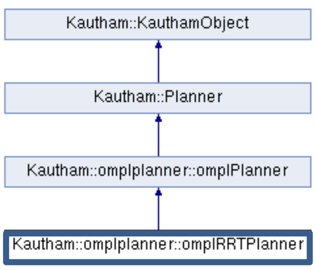

240IAU33 – Service and Industrial Robotics / 240AR032 - Planning and Implementation of Robotic Systems Jan Rosell

5. The code

2. OMPL: integration issues

The planner






```
//! Constructor
omplRRTPlanner::omplRRTPlanner
```

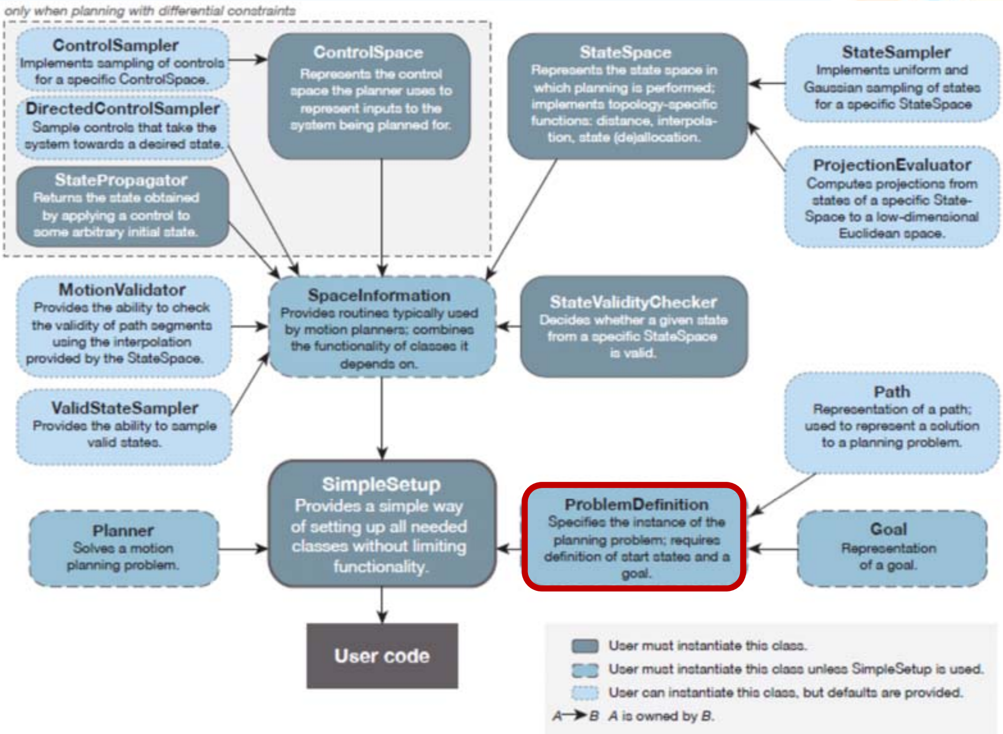
```
//create planner
ob::PlannerPtr planner(new og::RRT(si));
```

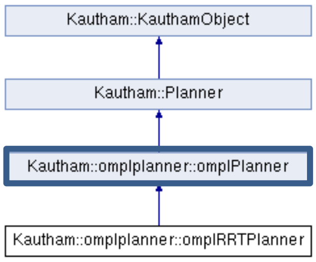
```
//set the planner
ss->setPlanner(planner);
```

5. The code

2. OMPL: integration issues

The problem definition






```
//! function to find a solution path
bool omplPlanner::trySolve()

    // set the start and goal states
    ss->setStartAndGoalStates(startompl, goalompl);

    ss->setup();
    ob::PlannerStatus solved = ss->solve(_planningTime);
```


Contents

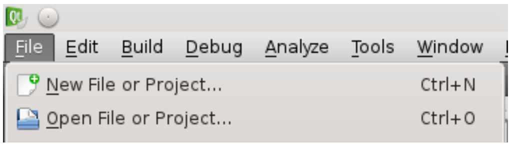
6. Getting started

1. If not installed, download and follow the instructions to install the Debian or Ubuntu packages available at sir.upc.edu/projects/kautham

2. Open the GUI application
   > **kautham-gui**

3. To browse the code, download the sources from github.com/iocroblab/kautham and use qtcreator to open the project defined by the CMakeLists.txt file

   > qtcreator&



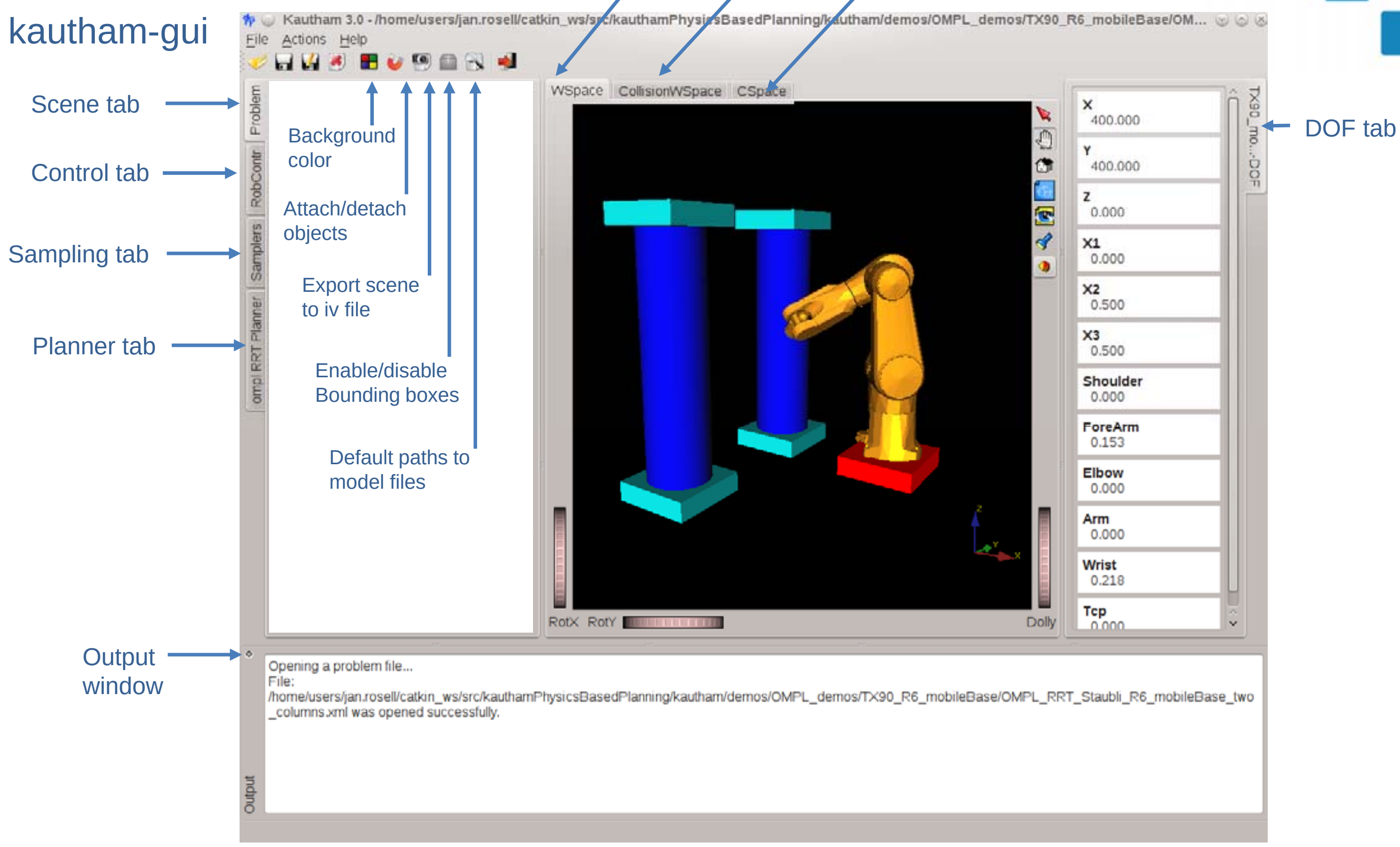
~/src/CMakeList.txt

240IAU33 – Service and Industrial Robotics / 240AR032 - Planning and Implementation of Robotic Systems Jan Rosell

6. Getting started

Workspace tab: rendering models
Workspace tab: collision models
Configuration space tab:

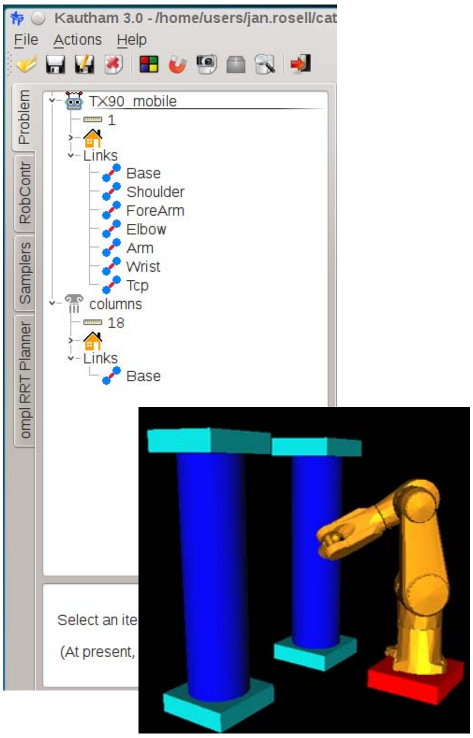
kautham-gui



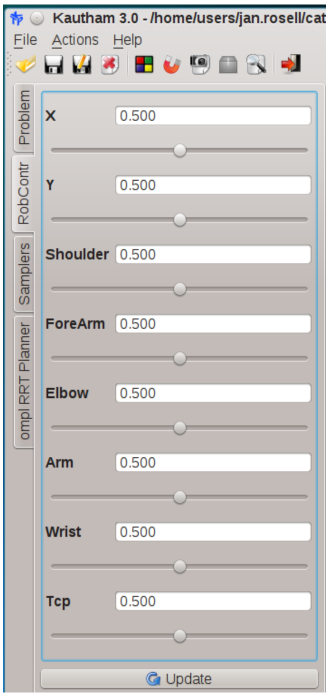
Master's Degree in Industrial Engineering / Master's degree in Automatic Control and Robotics

6. Getting started

Problem tab:
Describes the robots and obstacles in the scene



Control tab:
Allows to move the robot

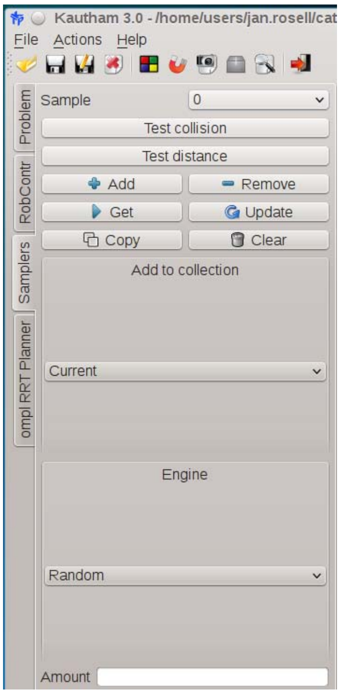


In this example each DOF is independently actuated by a single control

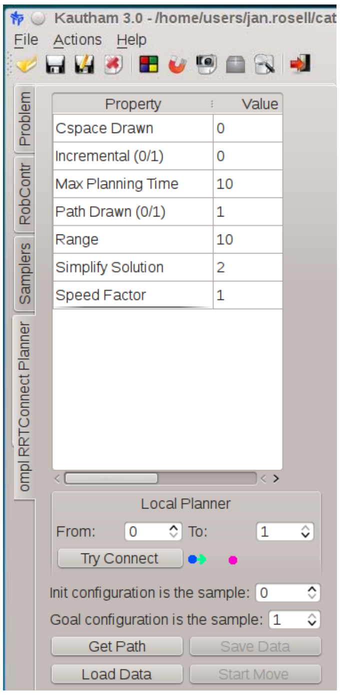
Controls are defined in the range [0,1]

6. Getting started

Sampling tab:
Used to visualize the sample set of a given planner, or to generate a sample set and to test for collision or distance.



Planner tab:
Used to set the planner parameters, launch a query and visualize the solution if available.
Also the rectilinear path in Cspace between two samples can be tested.



General parameters:

- *Cspace Drawn*: to select the robot, if there is more than one. Default is 0.
- *Incremental (0/1):* To reuse previous planner data (1) or not (0).
- *Max Planning Time*: Maximum allowed planning time (not considering the simplification of the solution).
- *Path Drawn (0/1):* If set to 1 the path followed by the TCP is drawn.
- *Simplify Solution:* If set to 1 or 2 the solution path is simplified, or left as is if set to 0.
- *Speed Factor*: Increases the speed of the motion along the solution path.

Planner-dependent parameters:

e.g. *Range* for an RRTConnect planner

6. Getting started

Launching the benchmarking utility using the kautham-console application:

a) A benchmarking file indicates which are the problems to be run (each one with a planner and some given values of its parameters) and compared:

```
<Benchmark Name="experiment">
    <Problem File="OMPL_PRM_bigcube-maze2D_R2_r005.xml" PlannerAlias="PRM_r005" />
    <Problem File="OMPL_PRM_bigcube-maze2D_R2_r015.xml" PlannerAlias="PRM_r015" />
    <Problem File="OMPL_PRM_bigcube-maze2D_R2_r050.xml" PlannerAlias="PRM_r050" />
    <Problem File="OMPL_PRM_bigcube-maze2D_R2_h005.xml" PlannerAlias="PRM_h005" />
    <Problem File="OMPL_PRM_bigcube-maze2D_R2_h015.xml" PlannerAlias="PRM_h015" />
    <Problem File="OMPL_PRM_bigcube-maze2D_R2_h050.xml" PlannerAlias="PRM_h050" />
    <Parameter Name="maxTime" Value="1.5" />
    <Parameter Name="maxMem" Value="8000" />
    <Parameter Name="runCount" Value="10" />
    <Parameter Name="displayProgress" Value="true" />
    <Parameter Name="filename" Value="resultPRM.log" />
</Benchmark>
```

A planner alias is only required if the same type of planners are benchmarked then the benchmarking file should include an alias name for each.

b) The benchmarking is launched using the kautham-console application:

```
> kautham-console -b absolute_path/benchmarking.xml
```

where the absolute path is e.g. `/home/linux/demos/OMPL_demos/Cube-sPassage_R2`

A file *result.log* is created with the data of the executed runs (the name of the log file is set in the xml file).

c) The results can be visualized by creating a database file and the pdf with the plots as follows:

```
> ompl_benchmark_statistics result.log -d result.db
> ompl_benchmark_statistics -d result.db -p result.pdf
```

Alternatively, the results of the benchmarking can be visualized by uploading the database file to plannerarena.org.

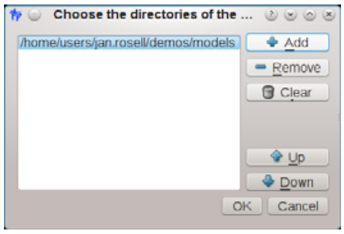
6. Getting started

Exercises:

First, copy the demos folder to a user directory (i.e. the home directory) so that you have writing permits:

```
> cp -r /usr/share/kautham/demos .
```

Then, set the new path to the models subfolder as a default path to models files using the corresponding icon .



Exercise 1: Solving some queries

Open de problem: demos/OMPL_geo_demos/Cube_Maze_R2/OMPL_RRTconnect_bigcube-maze2D_R2.xml

a) Check the d.o.f. of the robot by moving the sliders in the controls tab (*RobContr*).

b) Solve the query by pressing the *Get Path* button in the *Planner* tab.

c) Visualize the CSpace (press the “eye” button for a complete view) in the *CSpace* tab.

d) Animate the solution found by pressing the *Start Move* button. Visualize it in the *WSpace* tab.

Repeat for:

demos/OMPL_geo_demos/Ele_hole_SE3/OMPL_RRTConnect_ele_hole_SE3.xml

demos/OMPL_geo_demos/TX90_R6_mobileBase/OMPL_RRTConnect_Staubli_R6_mobileBase_two_columns.xml

6. Getting started

Exercise 2: Changing common parameters of the planners

Open de problem demos/OMPL_geo_demos/Cube_Maze_R2/OMPL_RRTconnect_bigcube-maze2D_R2.xml and vary the following general planning parameters:

a) *Max Planning Time*: Change it and think about probabilistic completeness of sampling-based planners.

b) *Simplify Solution*: Set to 0, 1 and 2 and check the type of smoothing done.

c) *Speed Factor*: Change it to accelerate the animation of the solution.

d) *Incremental*: Set it to 1 and the *Max Planning Time* to a very low value and repeatedly press the *Get Path* button to visualize in the CSpace tab how the trees grow.

6. Getting started

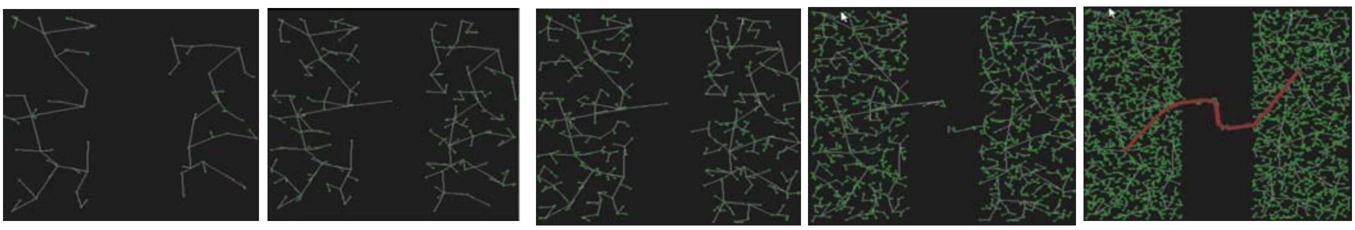
Exercise 3: Visualize the PRM construction

Open de problem demos/OMPL_geo_demos/Cube-sPassage_R2/OMPL_PRM_bigcube_s-passage_R2.xm and visualize the construction of the roadmap by executing the following steps:

a) Set the planner parameters to: MaxPlanningTime (**0.02**), Incremental (**0**), MinExpandTime (0), MaxNearestNeighbors (1000), DistanceThreshold (0.2), Simplify Solution (0) and press *Get Path*.

b) Change Incremental to **1** and press *Get Path* repeatedly to see how the roadmap is built, until a solution is found.

c) Repeat the previous steps several times to take into account the randomness.



6. Getting started

Exercise 4: Visualize the RRT and RRTconnect construction

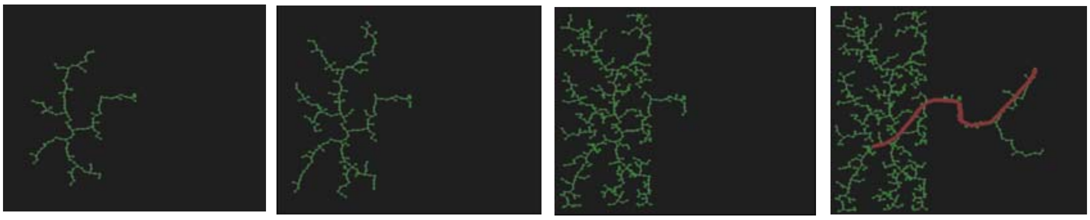
Open de problem demos/OMPL_geo_demos/Cube-sPassage_R2/OMPL_RRT_bigcube_s-passage_R2.xml

and visualize the construction of the tree by executing the following steps:

a) Set the planner parameters to: MaxPlanningTime (**0.02**), Incremental (**0**), Range (0.02), Goal Bias (0.05), Simplify Solution (0), and press *Get Path*.

b) Change Incremental to **1** and press Get Path repeatedly to see how the tree is built until a solution is found.

c) Repeat the previous steps several times to take into account the randomness.



Repeat for demos/OMPL_geo_demos/Cube-sPassage_R2/OMPL_RRTconnect_bigcube_s-passage_R2.xml