The Kautham Project

A Software Tool for Robot
Motion Planning and Simulation

Planning and Implementation of Robotic Systems

Master's degree in Automatic Control and Robotics

Contents

1. Introduction

1. **Objective**
2. **Requirements**
3. **Alternatives**

2. Results
3. Tools
4. Basic features
5. Advanced features
6. The code
7. Getting started

1. Objectives and requirements

1. Objective

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 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

1. Objectives and requirements

2. Requirements

- General:
  - Multiplatform and open source
  - Different interfaces (GUI, Console, ROS)

- Planning:
  - Samplers
  - Collision-detection
  - Problem description
  - Math utilities (Transform, PCA, …)
  - Benchmarking

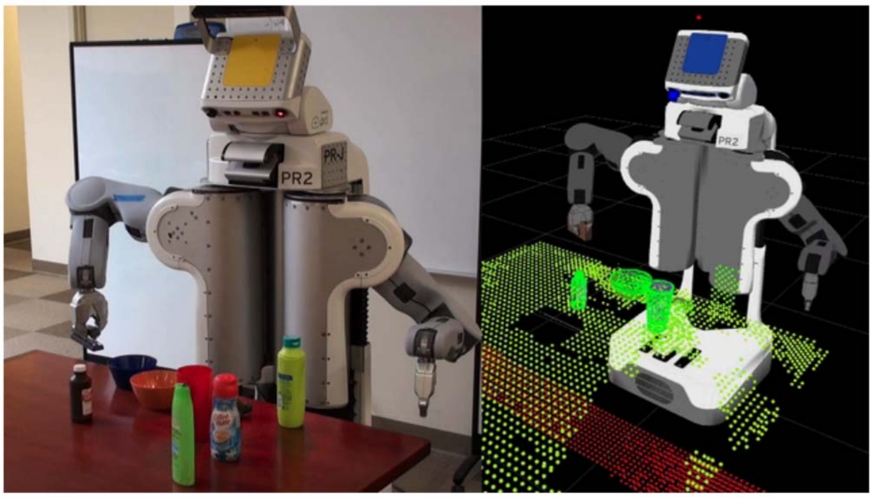
- Simulation:
  - Geometric modelling
  - Visualization
  - Kinematics and dynamics

1. Objectives and requirements

3. Alternatives

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

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



moveit.ros.org

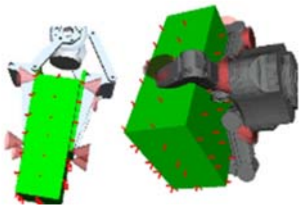
1. Objectives and requirements

3. Alternatives

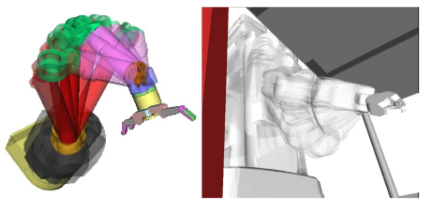
Open Robotics Automation Virtual Environment (**OpenRAVE**) provides an environment for testing, developing motion planning algorithms

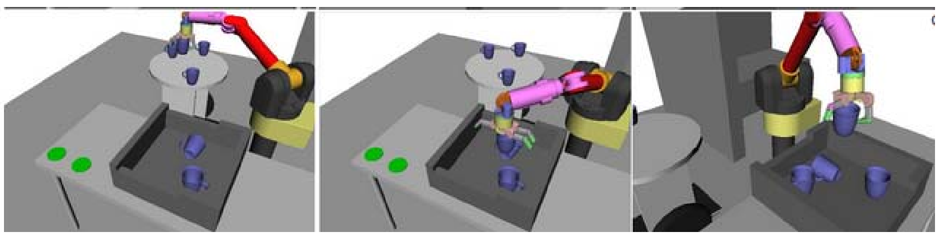
Offers some sampling-based planning algorithms for robot arms.

Has graspping and inverse kinematics modules.



Manages compiled inverse kinematics files for robots using ikfast.



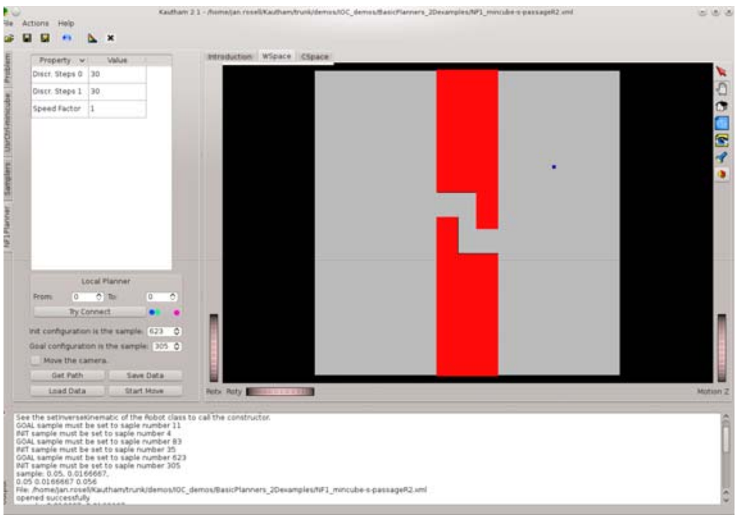


Contents

1. Introduction
2. **Results**
   1. **Simple examples**
   2. **Complex examples**
3. Tools
4. Basic features
5. Advanced features
6. The code
7. Getting started

2. Results

1. Simple examples


Kautham 2.1 - /home/jan.rosell/Kautham/trunk/demos/IOC_demos/BasicPlanners_2Dexamples/NF1_mincube-s-passageR2.xml

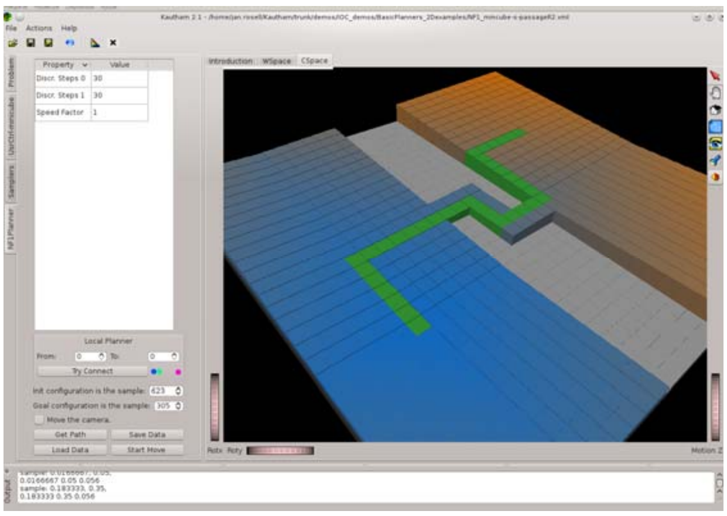
Property	Value
Discr. Steps 0	30
Discr. Steps 1	30
Speed Factor	1

Local Planner — From: 0 To: 0

Init configuration is the sample: 623

Goal configuration is the sample: 305

```
See the setInverseKinematic of the Robot class to call the constructor.
GOAL sample must be set to saple number 11
INIT sample must be set to saple number 4
GOAL sample must be set to saple number 83
INIT sample must be set to saple number 35
GOAL sample must be set to saple number 623
INIT sample must be set to saple number 305
sample: 0.05, 0.0166667,
0.05 0.0166667 0.056
File: /home/jan.rosell/Kautham/trunk/demos/IOC_demos/BasicPlanners_2Dexamples/NF1_mincube-s-passageR2.xml
opened successfully
```


Kautham 2.1 - /home/jan.rosell/Kautham/trunk/demos/IOC_demos/BasicPlanners_2Dexamples/NF1_mincube-s-passageR2.xml

Property	Value
Discr. Steps 0	30
Discr. Steps 1	30
Speed Factor	1

Local Planner — From: 0 To: 0

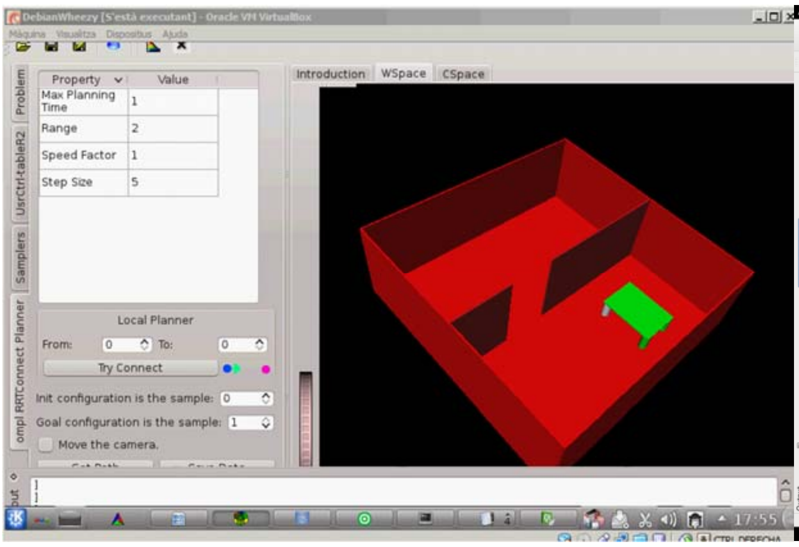
Init configuration is the sample: 623

Goal configuration is the sample: 305

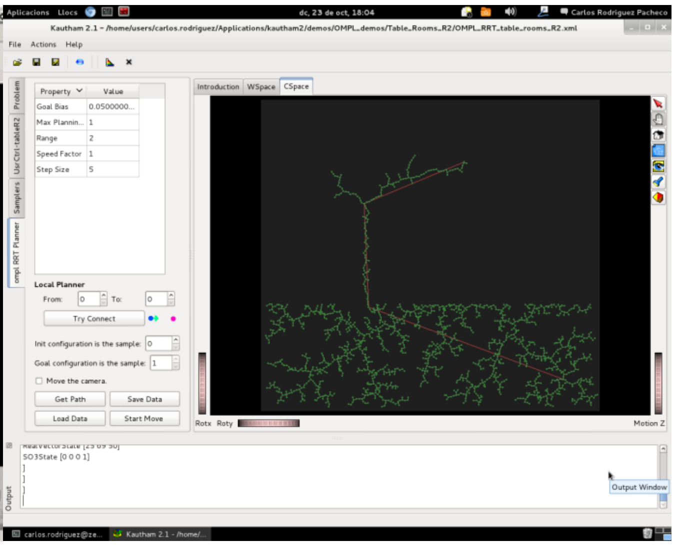
```
sample: 0.0166667, 0.05,
0.0166667 0.05 0.056
sample: 0.183333, 0.35,
0.183333 0.35 0.056
```

2. Results

1. Simple examples


DebianWheezy [S'està executant] - Oracle VM VirtualBox

Property	Value
Max Planning Time	1
Range	2
Speed Factor	1
Step Size	5


Kautham 2.1 – /home/users/carlos.rodriguez/Applications/kautham2/demos/OMPL_demos/Table_Rooms_R2/OMPL_RRT_table_rooms_R2.xml

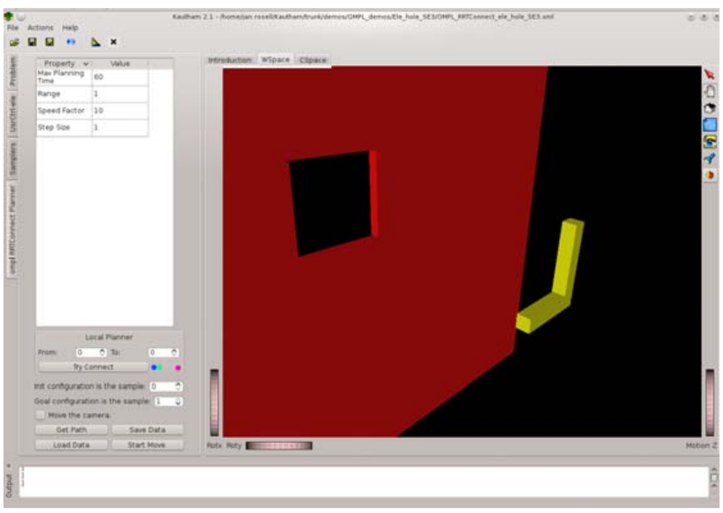
Property	Value
Goal Bias	0.0500000...
Max Plannin...	1
Range	2
Speed Factor	1
Step Size	5

Init configuration is the sample: 0
Goal configuration is the sample: 1

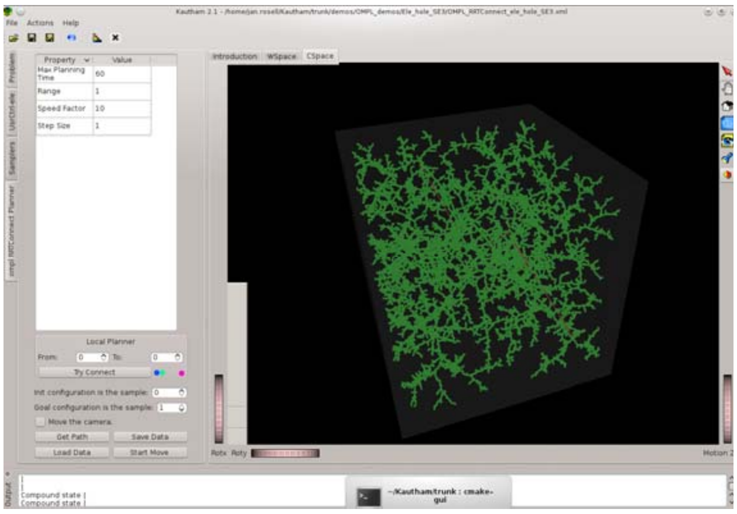
```
SO3State [0 0 0 1]
]
]
]
```

2. Results

1. Simple examples


Kautham 2.1 - /home/jan.rosell/Kautham/trunk/demos/OMPL_demos/Ele_hole_SE3/OMPL_RRTConnect_ele_hole_SE3.xml

Property	Value
Max Planning Time	60
Range	1
Speed Factor	10
Step Size	1

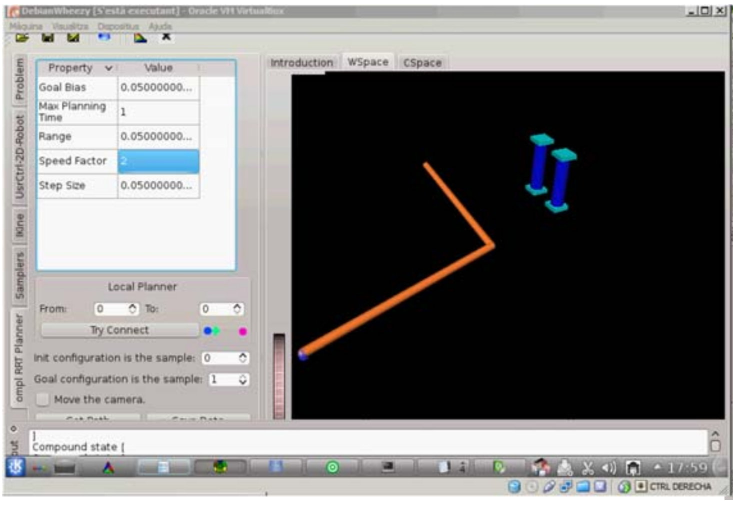

Kautham 2.1 - /home/jan.rosell/Kautham/trunk/demos/OMPL_demos/Ele_hole_SE3/OMPL_RRTConnect_ele_hole_SE3.xml

Property	Value
Max Planning Time	60
Range	1
Speed Factor	10
Step Size	1

```
Compound state [
Compound state [
```

2. Results

1. Simple examples



Property	Value
Goal Bias	0.05000000...
Max Planning Time	1
Range	0.05000000...
Speed Factor	2
Step Size	0.05000000...

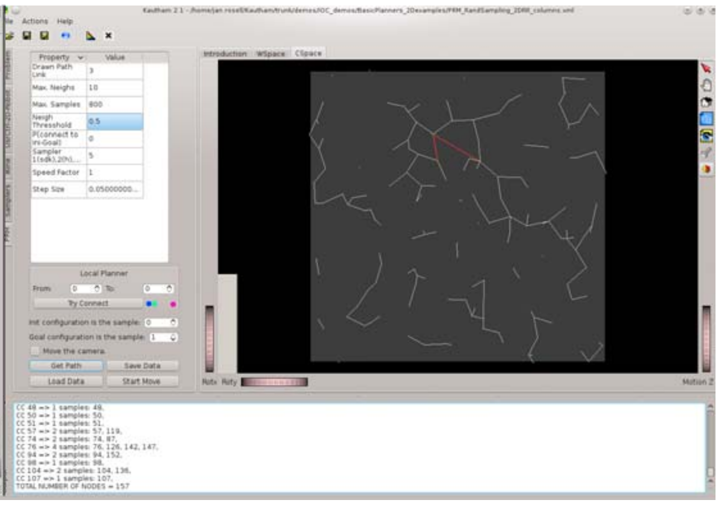
Local Planner

From: 0 To: 0

Init configuration is the sample: 0

Goal configuration is the sample: 1

```
Compound state [
```


Kautham 2.1 - /home/jan.rosell/Kautham/trunk/demos/IOC_demos/BasicPlanners_2Dexamples/PRM_RandSampling_2DRR_columns.xml
Actions
Help
Introduction
WSpace
CSpace
Try Connect
Move the camera.
Get Path
Save Data
Load Data
Start Move
RotX
RotY
Motion Z

Property	Value
Drawn Path Link	3
Max. Neighs	10
Max. Samples	800
Neigh Threshold	0.5
P(connect to Ini-Goal)	0
Sampler 1(sdk),2(h),...	5
Speed Factor	1
Step Size	0.05000000...

Local Planner

From: 0 To: 0

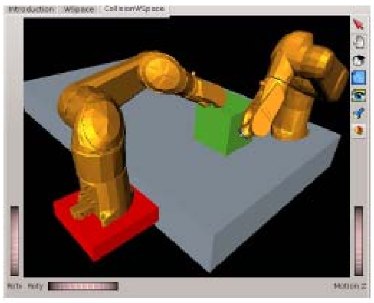
Init configuration is the sample: 0

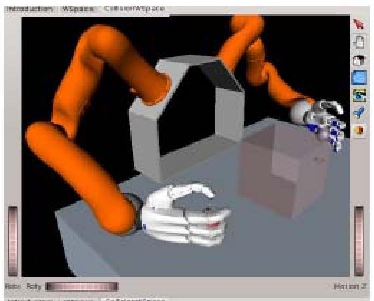
Goal configuration is the sample: 1

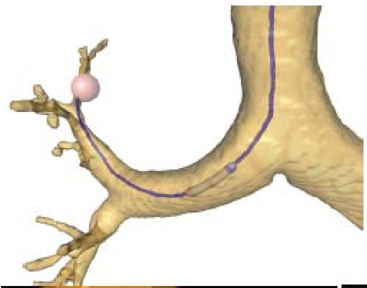
```
CC 48 => 1 samples: 48,
CC 50 => 1 samples: 50,
CC 51 => 1 samples: 51,
CC 57 => 2 samples: 57, 119,
CC 74 => 2 samples: 74, 87,
CC 76 => 4 samples: 76, 126, 142, 147,
CC 94 => 2 samples: 94, 152,
CC 98 => 1 samples: 98,
CC 104 => 2 samples: 104, 136,
CC 107 => 1 samples: 107,
TOTAL NUMBER OF NODES = 157
```

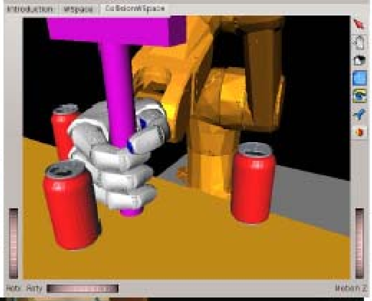
2. Results

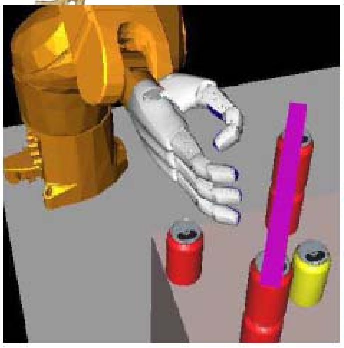
2. Complex examples

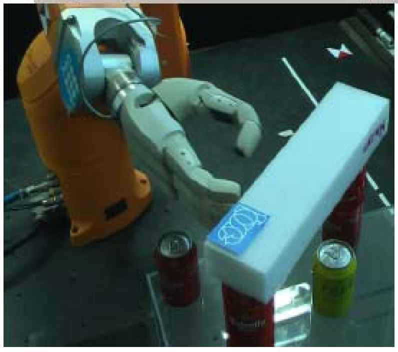






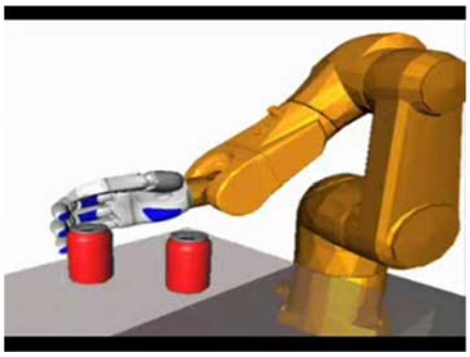
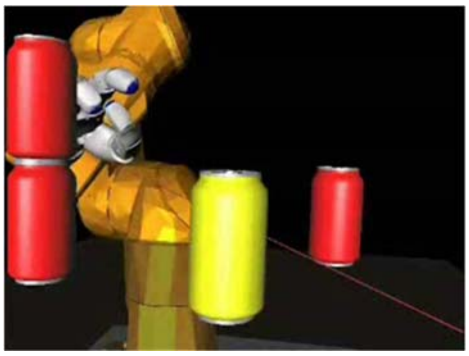
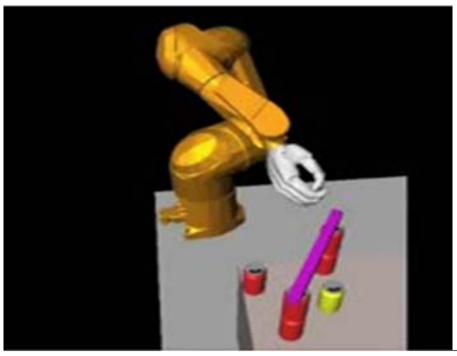


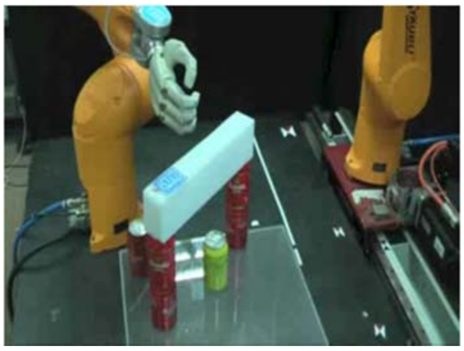




2. Results

2. Complex examples



Contents

1. Introduction
2. Results
3. Tools
   1. **Programming Language - C++**
   2. **Input Data - XML**
   3. **Graphics Rendering - COIN3D**
   4. **Collision Detection – PQP**
   5. **Dynamic Simulation - ODE**
   6. **Maths utilities: Mt / Armadillo**
   7. **Motion Planning: OMPL**
4. Basic features
5. Advanced features
6. The code
7. Getting started

3. Tools

1. Programming Language (C++)

C++:

- Object Oriented Programming (class hierarchy)
- Generic Programming (templates)

Doxygen: www.doxygen.org

The de facto standard tool for generating documentation from annotated C++ sources

```
/*! A test class */

class Test
{
  public:
    /** An enum type.
     *  The documentation block cannot be put after the enum!
     */
    enum EnumType
    {
      int EVal1,     /**< enum value 1 */
      int EVal2      /**< enum value 2 */
    };
    void member();   //!< a member function.

  protected:
    int value;       /*!< an integer value */
};
```

3. Tools

1. Programming Language (C++)

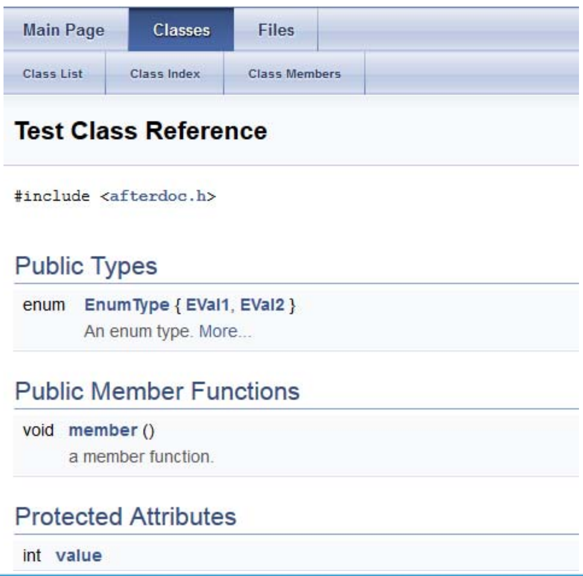
C++:

- Object Oriented Programming (class hierarchy)
- Generic Programming (templates)

Doxygen: www.doxygen.org

The de facto standard tool for generating documentation from annotated C++ sources

```
/*! A test class */

class Test
{
  public:
    /** An enum type.
     *  The documentation block cannot be put after the enum!
     */
    enum EnumType
    {
      int EVal1,     /**< enum value 1 */
      int EVal2      /**< enum value 2 */
    };
    void member();   //!< a member function.

  protected:
    int value;       /*!< an integer value */
};
```



Test Class Reference

`#include <afterdoc.h>`

Public Types

enum **EnumType** { **EVal1**, **EVal2** }
An enum type. More...

Public Member Functions

void **member** ()
a member function.

Protected Attributes

int **value**

3. Tools

1. Programming Language (C++)

CMake: http://www.cmake.org/

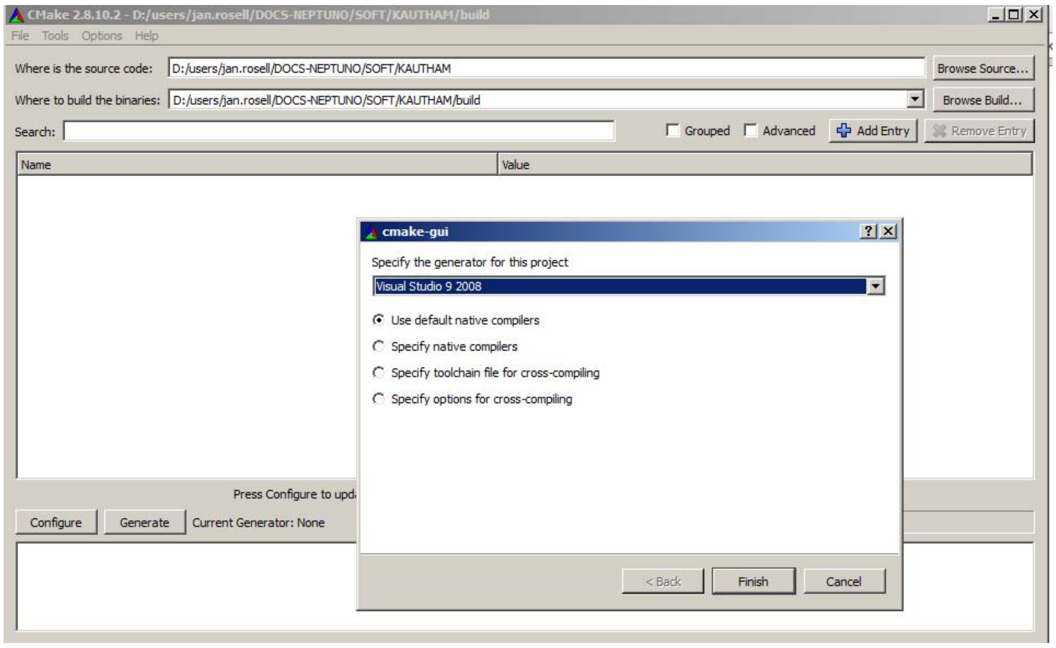
- Cross-platform, open-source build system.
- Used to control the software compilation process.
- Uses simple platform and compiler independent configuration files (CMakefile.txt).

```
# Make sure the compiler can find include files from our Hello library.
include_directories (${HELLO_SOURCE_DIR}/Hello)

# Make sure the linker can find the Hello library once it is built.
link_directories (${HELLO_BINARY_DIR}/Hello)

# Add executable called "helloDemo" that is built from the source files
# "demo.cxx" and "demo_b.cxx". The extensions are automatically found.
add_executable (helloDemo demo.cxx demo_b.cxx)

# Link the executable to the Hello library.
target_link_libraries (helloDemo Hello)
```

3. Tools

1. Programming Language (C++)

CMake generates native makefiles and workspaces that can be used in the compiler environment of your choice.



Where is the source code: D:/users/jan.rosell/DOCS-NEPTUNO/SOFT/KAUTHAM

Where to build the binaries: D:/users/jan.rosell/DOCS-NEPTUNO/SOFT/KAUTHAM/build

Search:

Press Configure to upd

Current Generator: None

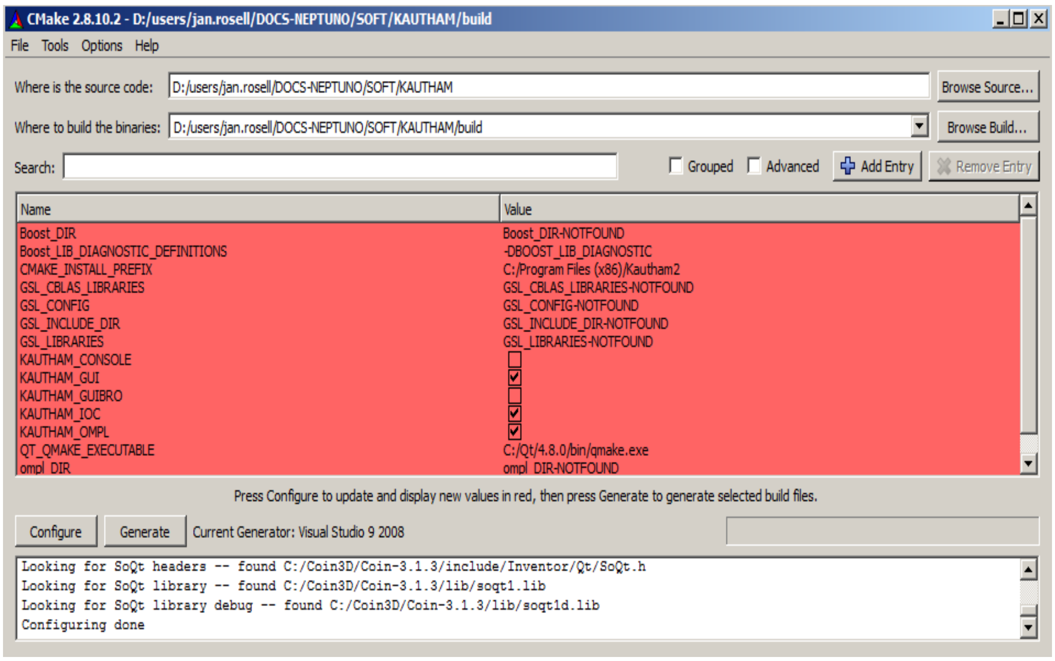
Specify the generator for this project: Visual Studio 9 2008

(●) Use default native compilers
( ) Specify native compilers
( ) Specify toolchain file for cross-compiling
( ) Specify options for cross-compiling

3. Tools

1. Programming Language (C++)

CMake generates native makefiles and workspaces that can be used in the compiler environment of your choice.



CMake 2.8.10.2 - D:/users/jan.rosell/DOCS-NEPTUNO/SOFT/KAUTHAM/build

Where is the source code: D:/users/jan.rosell/DOCS-NEPTUNO/SOFT/KAUTHAM

Where to build the binaries: D:/users/jan.rosell/DOCS-NEPTUNO/SOFT/KAUTHAM/build

Name	Value
Boost_DIR	Boost_DIR-NOTFOUND
Boost_LIB_DIAGNOSTIC_DEFINITIONS	-DBOOST_LIB_DIAGNOSTIC
CMAKE_INSTALL_PREFIX	C:/Program Files (x86)/Kautham2
GSL_CBLAS_LIBRARIES	GSL_CBLAS_LIBRARIES-NOTFOUND
GSL_CONFIG	GSL_CONFIG-NOTFOUND
GSL_INCLUDE_DIR	GSL_INCLUDE_DIR-NOTFOUND
GSL_LIBRARIES	GSL_LIBRARIES-NOTFOUND
KAUTHAM_CONSOLE	☐
KAUTHAM_GUI	☑
KAUTHAM_GUIBRO	☐
KAUTHAM_IOC	☑
KAUTHAM_OMPL	☑
QT_QMAKE_EXECUTABLE	C:/Qt/4.8.0/bin/qmake.exe
ompl_DIR	ompl_DIR-NOTFOUND

Press Configure to update and display new values in red, then press Generate to generate selected build files.

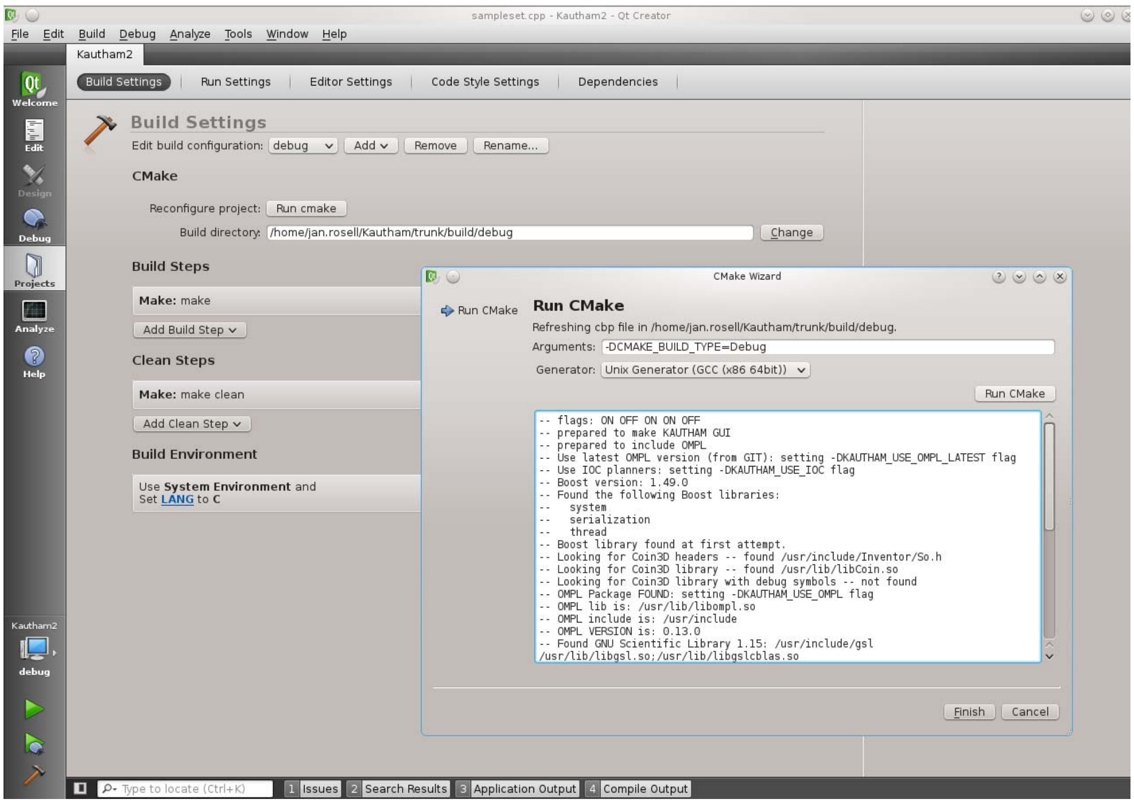
Current Generator: Visual Studio 9 2008

```
Looking for SoQt headers -- found C:/Coin3D/Coin-3.1.3/include/Inventor/Qt/SoQt.h
Looking for SoQt library -- found C:/Coin3D/Coin-3.1.3/lib/soqt1.lib
Looking for SoQt library debug -- found C:/Coin3D/Coin-3.1.3/lib/soqt1d.lib
Configuring done
```

3. Tools

1. Programming Language (C++)

QtCreator



sampleset.cpp - Kautham2 - Qt Creator

Build Settings

Edit build configuration: debug

CMake

Reconfigure project: Run cmake

Build directory: /home/jan.rosell/Kautham/trunk/build/debug

Build Steps

Make: make

Clean Steps

Make: make clean

Build Environment

Use **System Environment** and Set **LANG** to **C**

CMake Wizard

Run CMake

Refreshing cbp file in /home/jan.rosell/Kautham/trunk/build/debug.

Arguments: -DCMAKE_BUILD_TYPE=Debug

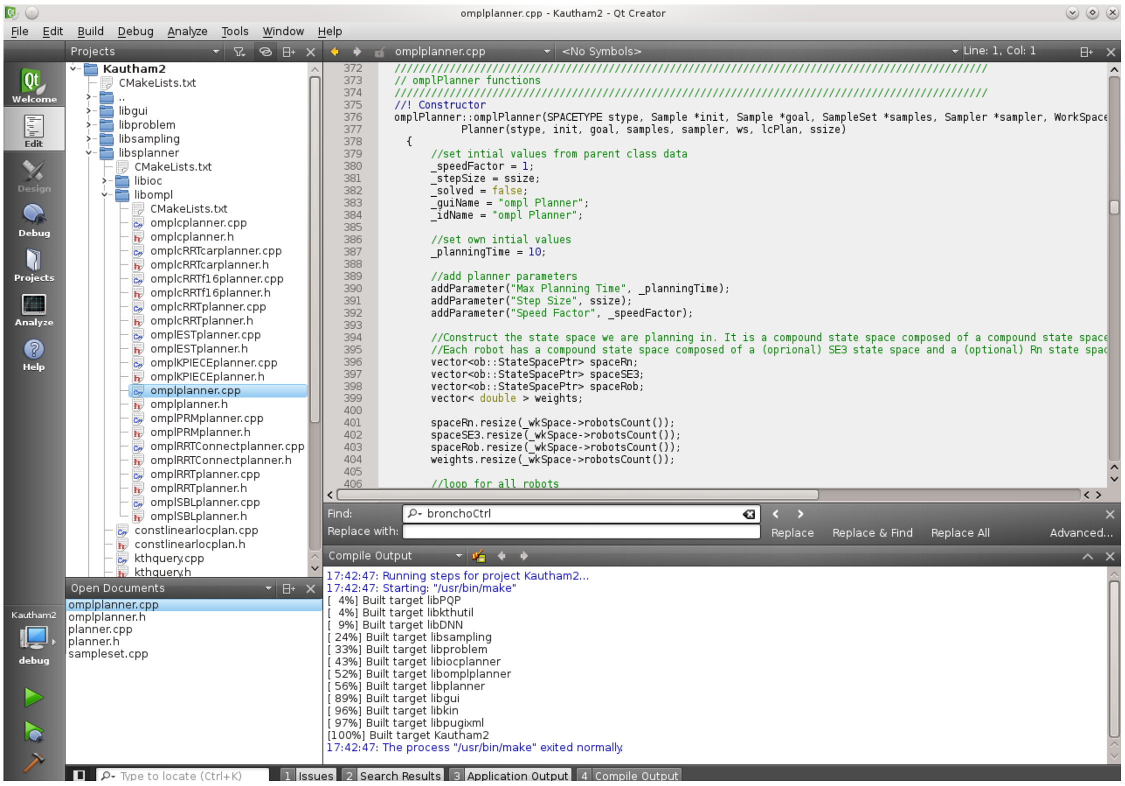
Generator: Unix Generator (GCC (x86 64bit))

```
-- flags: ON OFF ON ON OFF
-- prepared to make KAUTHAM GUI
-- prepared to include OMPL
-- Use latest OMPL version (from GIT): setting -DKAUTHAM_USE_OMPL_LATEST flag
-- Use IOC planners: setting -DKAUTHAM_USE_IOC flag
-- Boost version: 1.49.0
-- Found the following Boost libraries:
--   system
--   serialization
--   thread
-- Boost library found at first attempt.
-- Looking for Coin3D headers -- found /usr/include/Inventor/So.h
-- Looking for Coin3D library -- found /usr/lib/libCoin.so
-- Looking for Coin3D library with debug symbols -- not found
-- OMPL Package FOUND: setting -DKAUTHAM_USE_OMPL flag
-- OMPL lib is: /usr/lib/libompl.so
-- OMPL include is: /usr/include
-- OMPL VERSION is: 0.13.0
-- Found GNU Scientific Library 1.15: /usr/include/gsl
/usr/lib/libgsl.so;/usr/lib/libgslcblas.so
```

3. Tools

1. Programming Language (C++)

QtCreator



```
Kautham2
  CMakeLists.txt
  ..
  libgui
  libproblem
  libsampling
  libsplanner
    CMakeLists.txt
    libioc
    libompl
      CMakeLists.txt
      omplcplanner.cpp
      omplcplanner.h
      omplcRRTcarplanner.cpp
      omplcRRTcarplanner.h
      omplcRRTf16planner.cpp
      omplcRRTf16planner.h
      omplcRRTplanner.cpp
      omplcRRTplanner.h
      omplESTplanner.cpp
      omplESTplanner.h
      omplKPIECEplanner.cpp
      omplKPIECEplanner.h
      omplplanner.cpp
      omplplanner.h
      omplPRMplanner.cpp
      omplPRMplanner.h
      omplRRTConnectplanner.cpp
      omplRRTConnectplanner.h
      omplRRTplanner.cpp
      omplRRTplanner.h
      omplSBLplanner.cpp
      omplSBLplanner.h
    constlinearlocplan.cpp
    constlinearlocplan.h
    kthquery.cpp
    kthquery.h
```

```
omplplanner.cpp
omplplanner.h
planner.cpp
planner.h
sampleset.cpp
```

```cpp
///////////////////////////////////////////////////////////////////////////////////////////
// omplPlanner functions
///////////////////////////////////////////////////////////////////////////////////////////
//! Constructor
omplPlanner::omplPlanner(SPACETYPE stype, Sample *init, Sample *goal, SampleSet *samples, Sampler *sampler, WorkSpace
          Planner(stype, init, goal, samples, sampler, ws, lcPlan, ssize)
  {
      //set intial values from parent class data
      _speedFactor = 1;
      _stepSize = ssize;
      _solved = false;
      _guiName = "ompl Planner";
      _idName = "ompl Planner";

      //set own intial values
      _planningTime = 10;

      //add planner parameters
      addParameter("Max Planning Time", _planningTime);
      addParameter("Step Size", ssize);
      addParameter("Speed Factor", _speedFactor);

      //Construct the state space we are planning in. It is a compound state space composed of a compound state space
      //Each robot has a compound state space composed of a (oprional) SE3 state space and a (optional) Rn state spac
      vector<ob::StateSpacePtr> spaceRn;
      vector<ob::StateSpacePtr> spaceSE3;
      vector<ob::StateSpacePtr> spaceRob;
      vector< double > weights;

      spaceRn.resize(_wkSpace->robotsCount());
      spaceSE3.resize(_wkSpace->robotsCount());
      spaceRob.resize(_wkSpace->robotsCount());
      weights.resize(_wkSpace->robotsCount());

      //loop for all robots
```

Find: bronchoCtrl

```
17:42:47: Running steps for project Kautham2...
17:42:47: Starting: "/usr/bin/make"
[  4%] Built target libPQP
[  4%] Built target libkthutil
[  9%] Built target libDNN
[ 24%] Built target libsampling
[ 33%] Built target libproblem
[ 43%] Built target libiocplanner
[ 52%] Built target libomplplanner
[ 56%] Built target libplanner
[ 89%] Built target libgui
[ 96%] Built target libkin
[ 97%] Built target libpugixml
[100%] Built target Kautham2
17:42:47: The process "/usr/bin/make" exited normally.
```

3. Tools

2. Input data (XML)

XML: Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

pugixml: light-weight C++ XML processing library

```xml
<Robot  name="2D-Robot" DHType="Modified" robType="Chain">

  <Joints size="4">
    <Joint name="Base" ivFile="2DRR/base.iv">
      <DHPars alpha="0.0" a="0.0" theta="0.0" d="0.0"></DHPars>
      <Description rotational="false" movable="false" trunk="true"></Description>
      <Limits Hi="0" Low="0"></Limits>
      <Weight weight="1.0"></Weight>
      <Parent name=""></Parent>
    </Joint>
```

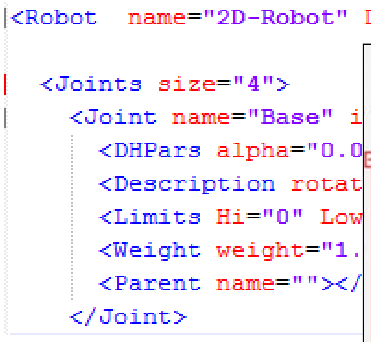
http://code.google.com/p/pugixml/

3. Tools

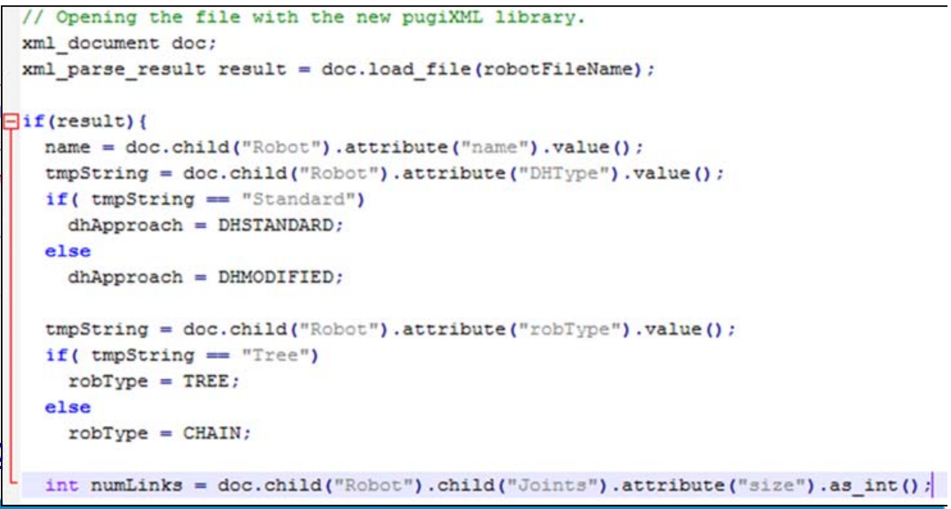
2. Input data (XML)

XML: Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

pugixml: light-weight C++ XML processing library



```
<Robot  name="2D-Robot" DHType="Modified" robType="Chain">

  <Joints size="4">
    <Joint name="Base" i
      <DHPars alpha="0.0
      <Description rotat
      <Limits Hi="0" Low
      <Weight weight="1.
      <Parent name=""></
    </Joint>
```



```
// Opening the file with the new pugiXML library.
xml_document doc;
xml_parse_result result = doc.load_file(robotFileName);

if(result){
  name = doc.child("Robot").attribute("name").value();
  tmpString = doc.child("Robot").attribute("DHType").value();
  if( tmpString == "Standard")
    dhApproach = DHSTANDARD;
  else
    dhApproach = DHMODIFIED;

  tmpString = doc.child("Robot").attribute("robType").value();
  if( tmpString == "Tree")
    robType = TREE;
  else
    robType = CHAIN;

  int numLinks = doc.child("Robot").child("Joints").attribute("size").as_int();
```

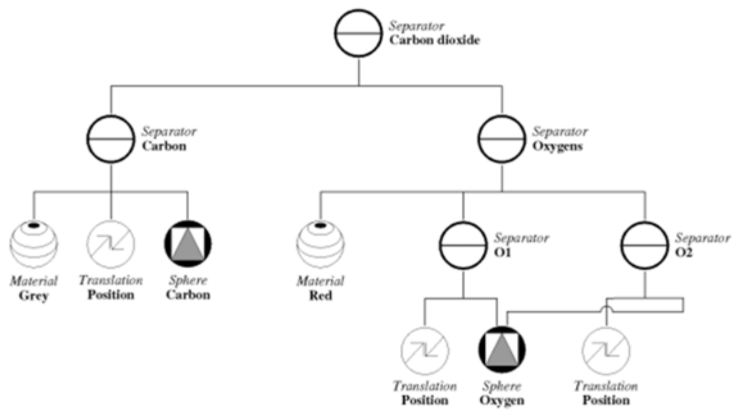
http://code.google.com/p/p

3. Tools

3. Graphics rendering (Coin3D)

VRML (Virtual Reality Modeling Language) is a language to describe 3D scenes represented as hierarchical structures called scene graphs.
It is derived from Open Inventor.

Scene graphs are composed of nodes:

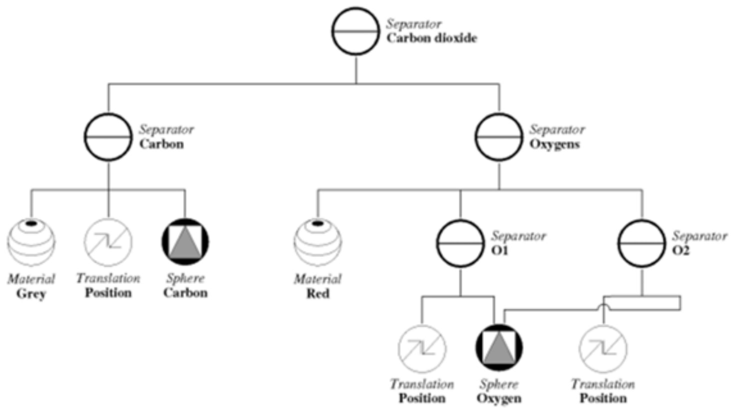


3. Tools

3. Graphics rendering (Coin3D)

VRML (Virtual Reality Modeling Language) is a language to describe 3D scenes represented as hierarchical structures called scene graphs.
It is derived form Open Inventor.

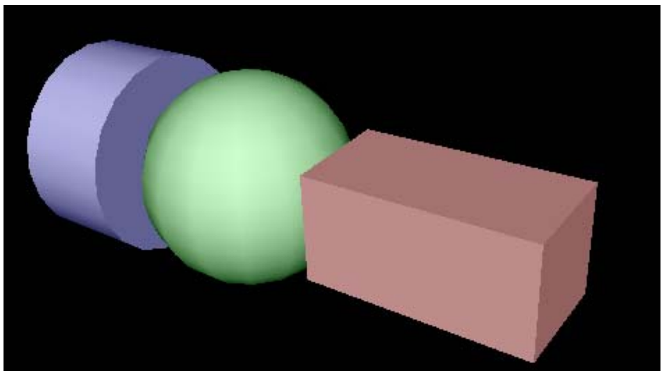
Scene graphs are composed of nodes:



- **Grouping nodes:** Allow collections of nodes to be treated as a single object..
  - Separator
- **Shape nodes**: Define the geometry of objects.
  - Cone,
  - Cube,
  - Cylinder,
  - IndexedFaceSet,
  - IndexedLineSet,…
- **Property nodes:** Affect the way shapes are drawn
  - Coordinate3, Normal
  - Material, Texture2
  - Scale, Rotation ,Translation, Transform

3. Tools

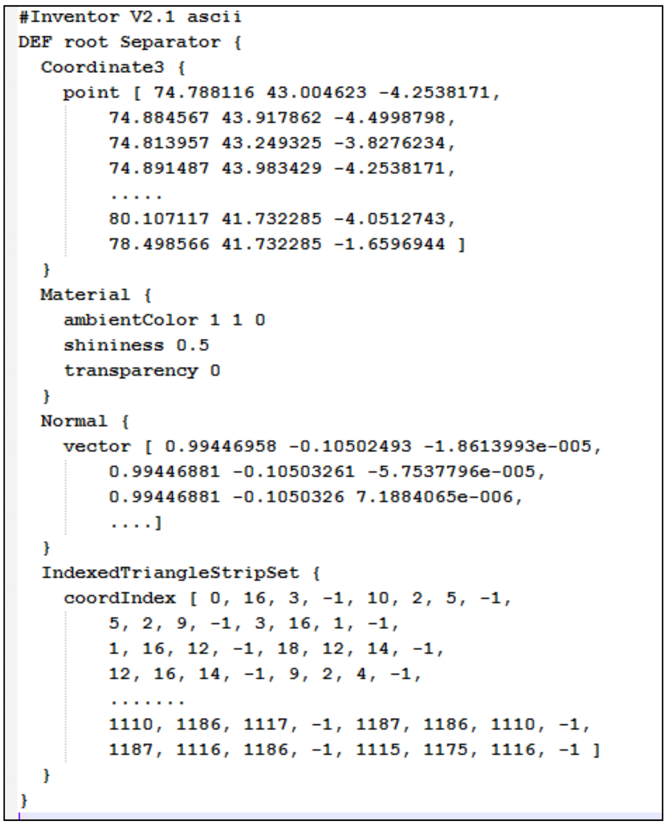
3. Graphics rendering (Coin3D)



```
#Inventor V2.1 ascii
Separator {
  Separator {
    Material {   ambientColor 1 0 0    }
    Translation {  translation 2 0 0    }
   Cube {
      width 2
      height 1
      depth 1
    }
  }
  Separator {
    Material {   ambientColor 0 1 0  }
    Sphere {
      radius 1
    }
  }
  Separator {
    Material {    ambientColor 0 0 1   }
    Transform {
      translation -1.5 0 0
      rotation 0 0 1  1.57
    }
    Cylinder {
      radius 1
      height 1
    }
  }
}
```

3. Tools

3. Graphics rendering (Coin3D)



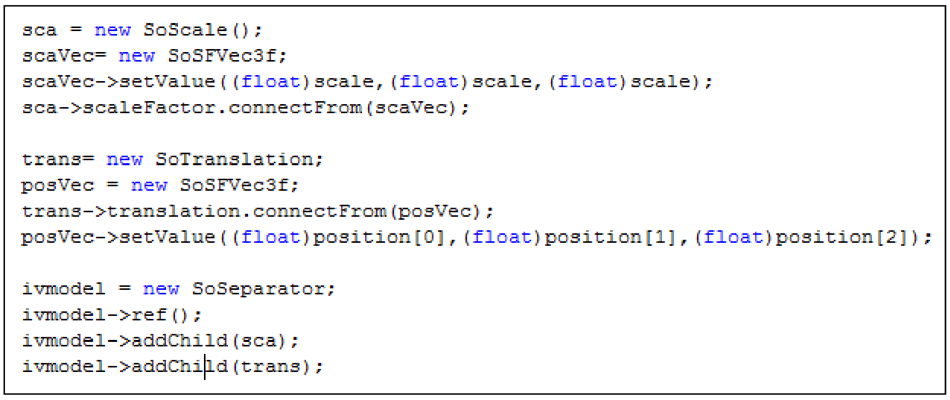
```
#Inventor V2.1 ascii
DEF root Separator {
  Coordinate3 {
    point [ 74.788116 43.004623 -4.2538171,
        74.884567 43.917862 -4.4998798,
        74.813957 43.249325 -3.8276234,
        74.891487 43.983429 -4.2538171,
        .....
        80.107117 41.732285 -4.0512743,
        78.498566 41.732285 -1.6596944 ]
  }
  Material {
    ambientColor 1 1 0
    shininess 0.5
    transparency 0
  }
  Normal {
    vector [ 0.99446958 -0.10502493 -1.8613993e-005,
        0.99446881 -0.10503261 -5.7537796e-005,
        0.99446881 -0.1050326 7.1884065e-006,
        ....]
  }
  IndexedTriangleStripSet {
    coordIndex [ 0, 16, 3, -1, 10, 2, 5, -1,
        5, 2, 9, -1, 3, 16, 1, -1,
        1, 16, 12, -1, 18, 12, 14, -1,
        12, 16, 14, -1, 9, 2, 4, -1,
        .......
        1110, 1186, 1117, -1, 1187, 1186, 1110, -1,
        1187, 1116, 1186, -1, 1115, 1175, 1116, -1 ]
  }
}
```

3. Tools

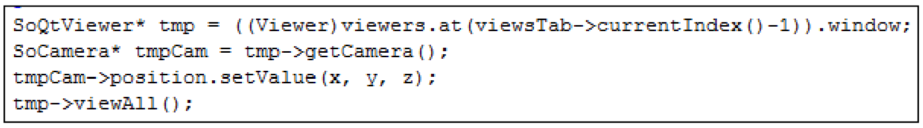
3. Graphics rendering (Coin3D)

Coin3D is an open source implementation of the Open Inventor API.

It allows to read, create and manipulate 3D scenes



```
sca = new SoScale();
scaVec= new SoSFVec3f;
scaVec->setValue((float)scale,(float)scale,(float)scale);
sca->scaleFactor.connectFrom(scaVec);

trans= new SoTranslation;
posVec = new SoSFVec3f;
trans->translation.connectFrom(posVec);
posVec->setValue((float)position[0],(float)position[1],(float)position[2]);

ivmodel = new SoSeparator;
ivmodel->ref();
ivmodel->addChild(sca);
ivmodel->addChild(trans);
```



```
SoQtViewer* tmp = ((Viewer)viewers.at(viewsTab->currentIndex()-1)).window;
SoCamera* tmpCam = tmp->getCamera();
tmpCam->position.setValue(x, y, z);
tmp->viewAll();
```

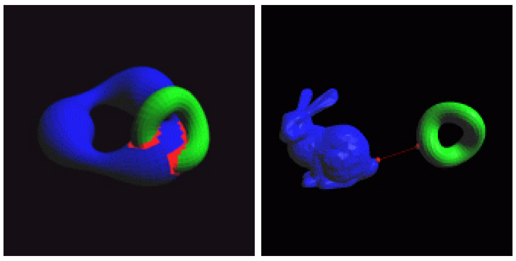
3. Tools

4. Collision detection (PQP)

PQP (**Proximity Query Package**) is a library for performing three types of proximity queries on a pair of geometric models composed of triangles:

- *Collision detection* - detecting whether the two models overlap, and optionally, all of the triangles that overlap.
- *Distance computation* - computing the minimum distance between a pair of models, i.e., the distance between the closest pair of points.
- *Tolerance verification* - determining whether two models are closer or farther than a tolerance distance

http://gamma.cs.unc.edu/SSV/



3. Tools

4. Collision detection (PQP)

```
pqpmodel = new PQP_Model;
SoCallbackAction triAction;
tri_info info(pqpmodel,scale);
pqpmodel->BeginModel();
triAction.addTriangleCallback(
     SoShape::getClassTypeId(),
     triang_CB, (void*)&info);
triAction.apply(ivModel());
pqpmodel->EndModel();
```

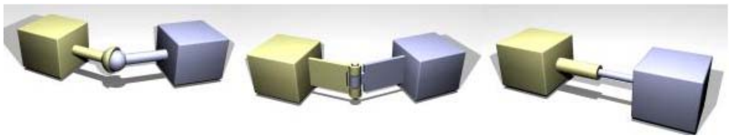
```
PQP_CollideResult pqp_coll_result;
KthReal *pR,*pO;
double posR[3];
double oriR[3][3];
double posO[3];
double oriO[3][3];
SbMatrix matO, matR;
SbRotation vcR;

pR = getPosition();
pO = other->getPosition();
for(int i=0;i<3;i++){
    posR[i]= (double)pR[i];
    posO[i]= (double)pO[i];
}

matR = orientationMatrix();
matO = ((IVPQPElement*)other)->orientationMatrix();
for (int i = 0; i < 3; i++){
    for (int j = 0; j < 3; j++){
        oriR[i][j] = matR[i][j];
        oriO[i][j] = matO[i][j];
    }
}

PQP_Collide(&pqp_coll_result,
  oriR, posR, pqpModel(),
  oriO, posO, ((IVPQPElement*)other)->pqpModel(),
  PQP_FIRST_CONTACT);
return (pqp_coll_result.Colliding()!= 0);
```

3. Tools

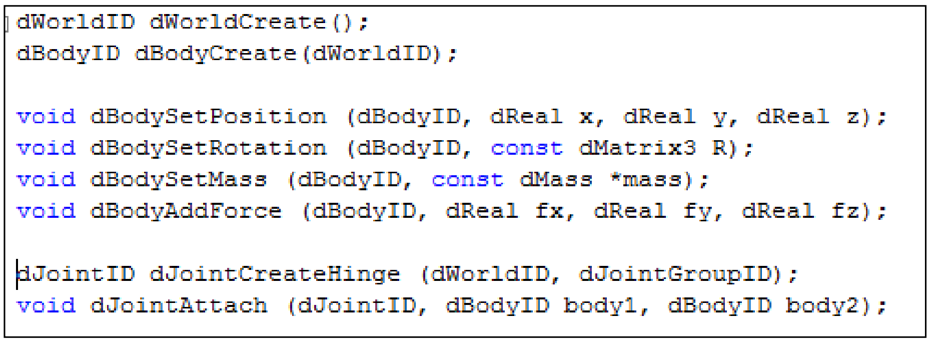
5. ODE (dynamic simulation)

ODE is an open source, high performance library for simulating rigid body dynamics. It is fully featured, stable, mature and platform independent

It has advanced joint types and integrated collision detection with friction.



It has an easy to use C/C++ API, e.g:



```
dWorldID dWorldCreate();
dBodyID dBodyCreate(dWorldID);

void dBodySetPosition (dBodyID, dReal x, dReal y, dReal z);
void dBodySetRotation (dBodyID, const dMatrix3 R);
void dBodySetMass (dBodyID, const dMass *mass);
void dBodyAddForce (dBodyID, dReal fx, dReal fy, dReal fz);

dJointID dJointCreateHinge (dWorldID, dJointGroupID);
void dJointAttach (dJointID, dBodyID body1, dBodyID body2);
```

OPEN DYNAMICS ENGINE™

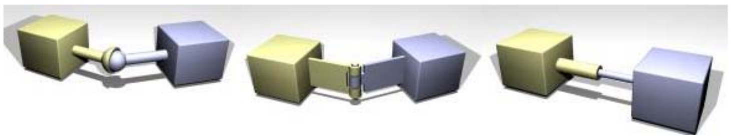
http://www.ode.org/

3. Tools

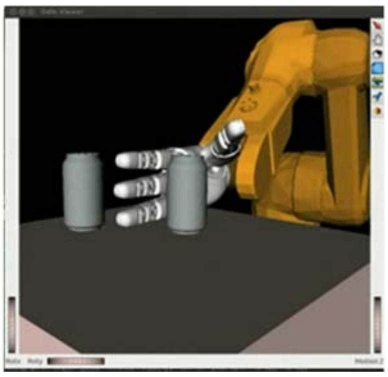
5. ODE (dynamic simulation)

ODE is an open source, high performance library for simulating rigid body dynamics. It is fully featured, stable, mature and platform independent

It has advanced joint types and integrated collision detection with friction.



It has an easy to use C/C++ API, e.g:



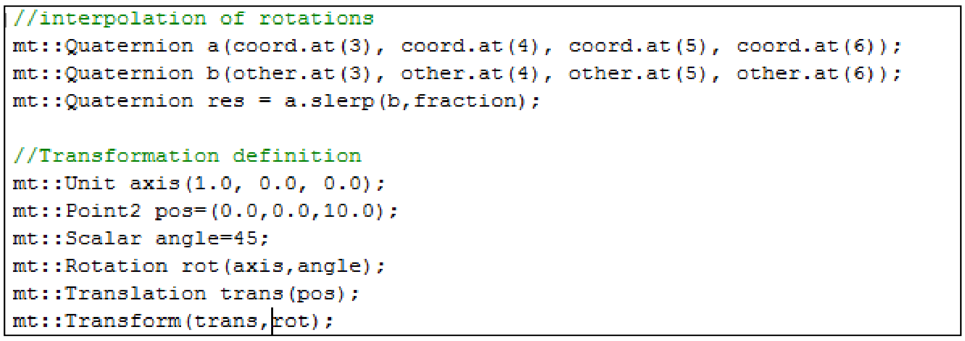
OPEN DYNAMICS ENGINE™

http://www.ode.org/

3. Tools

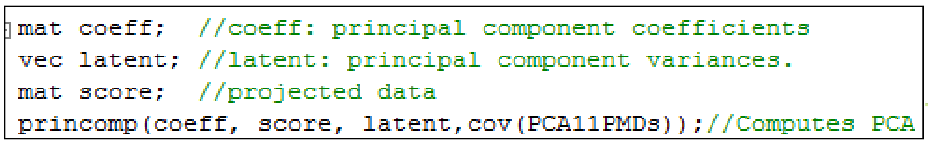
6. Maths Utilities (mt / Armadillo)

mt: The mt library is a set of classes and functions that implement concepts from the domains of geometry and rigid body mechanics.

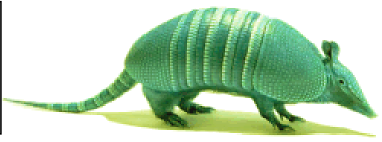


```
//interpolation of rotations
mt::Quaternion a(coord.at(3), coord.at(4), coord.at(5), coord.at(6));
mt::Quaternion b(other.at(3), other.at(4), other.at(5), other.at(6));
mt::Quaternion res = a.slerp(b,fraction);

//Transformation definition
mt::Unit axis(1.0, 0.0, 0.0);
mt::Point2 pos=(0.0,0.0,10.0);
mt::Scalar angle=45;
mt::Rotation rot(axis,angle);
mt::Translation trans(pos);
mt::Transform(trans,rot);
```

Armadillo: is a C++ linear algebra library with a good balance between speed and ease of use. Its API is similar to MATLAB



```
mat coeff;  //coeff: principal component coefficients
vec latent; //latent: principal component variances.
mat score;  //projected data
princomp(coeff, score, latent,cov(PCA11PMDs));//Computes PCA
```

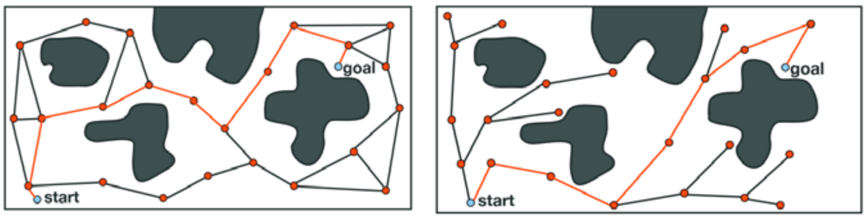


http://arma.sourceforge.net/

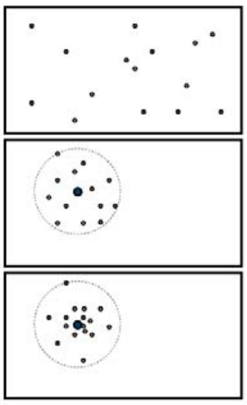
3. Tools

7. Motion Planning (OMPL)

- OMPL contains implementations of many sampling-based algorithms such as PRM, RRT, and several variants of these planners.



- All the planners operate on very abstractly defined state spaces. Many commonly used state spaces are already implemented (e.g. $SE(2), SE(3), \mathbb{R}^n$)



- For any state space, different state samplers can be used (e.g., uniform, Gaussian, obstacle based).

http://ompl.kavrakilab.org/

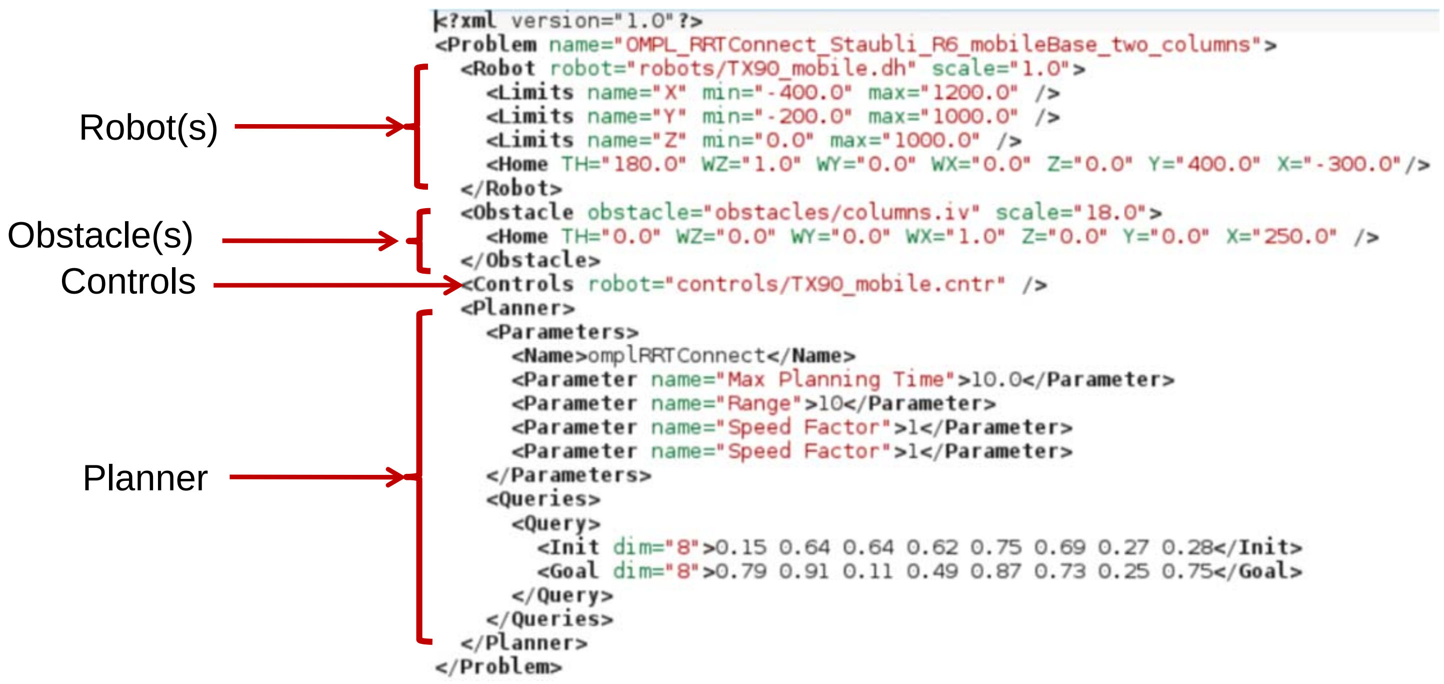
Contents

1. Introduction
2. Results
3. Tools
4. **Basic features**
   1. **Problem description**
   2. **Modelling robots and obstacles**
   3. **Planners**
   4. **Visualization**
5. Advanced features
6. The code
7. Getting started

4. Basic features

1. Problem description

XML file with four basic main sections



```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>
```

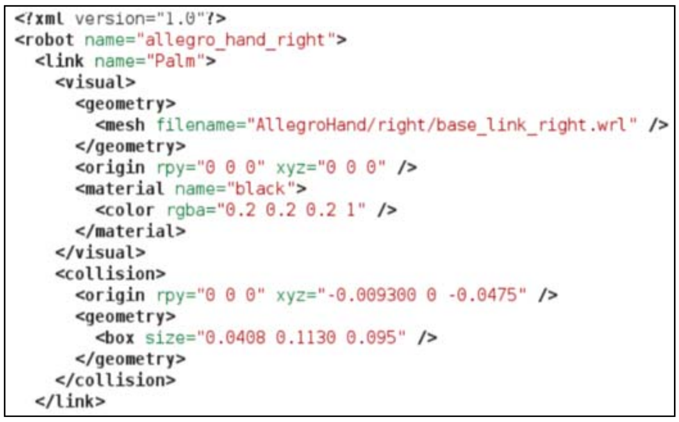
4. Basic features

2. Modelling robots and obstacles

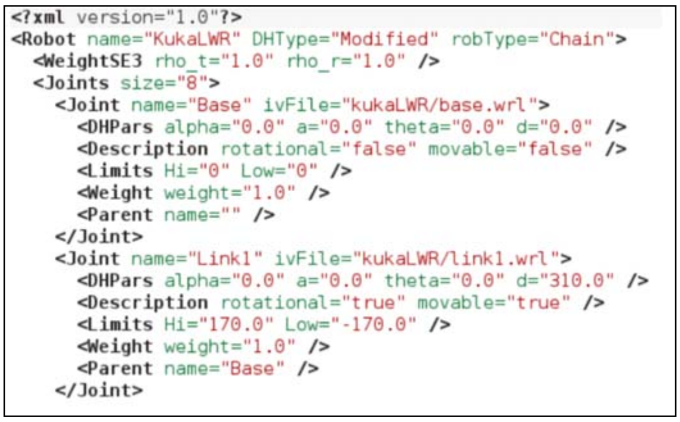
Robots: tree-like kinematic structure with an optional mobile base ($SE3 \times \mathbb{R}^n$) defined using

a) *Universal Robotic Description Language (URDF)*

b) *Denavit-Hartenberg (DH) parameters coded in an 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>
```



```
<?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>
```

Includes: path to a VRML describing the geometry, (optionally) collision models and inverse kinematic

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

4. Basic features

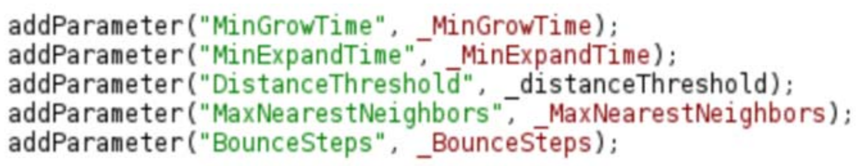
3. Planners

Type of planners:

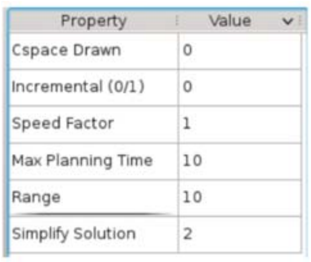
- Potential field-based planners: NF1, HF
- Sampling-based planners (OMPL): PRM, RRT, RRTconnect, RRT*, SBL, EST, KPIECE

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*

4. 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>
```

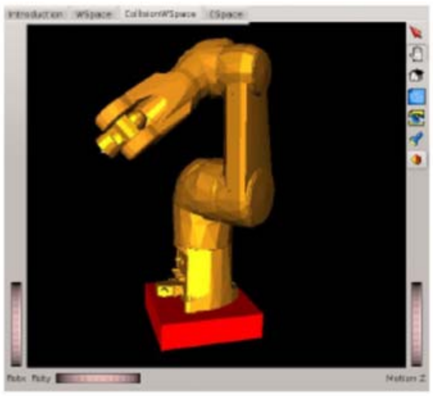
Free-flying robot with two translational d.o.f.

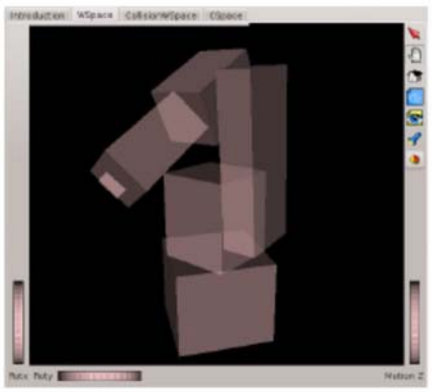
```xml
<?xml version="1.0"?>
<ControlSet>
  <Offset>
    <DOF name="bigcube/X" value="0.5" />
    <DOF name="bigcube/Y" value="0.5" />
  </Offset>
  <Control name="bigcube/x" eigValue="1.0">
    <DOF name="bigcube/X" value="1.0" />
  </Control>
  <Control name="bigcube/y" eigValue="1.0">
    <DOF name="bigcube/Y"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

4. Basic features

4. Visualization

1. Visualization of rendering models and collision models.



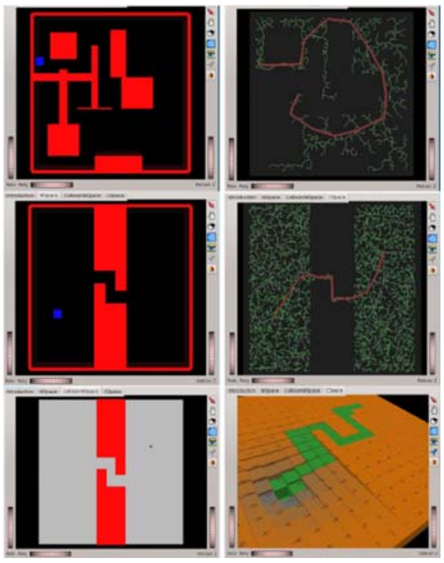


2. Visualization of the workspace and of the configuration space
   - For 2- or 3-dimensional problems: the roadmaps, trees or potential landscapes
   - For higher dimensional problems: the projections of the configuration space onto the first two or three degrees of freedom of each robot .

4. Basic features

4. Visualization

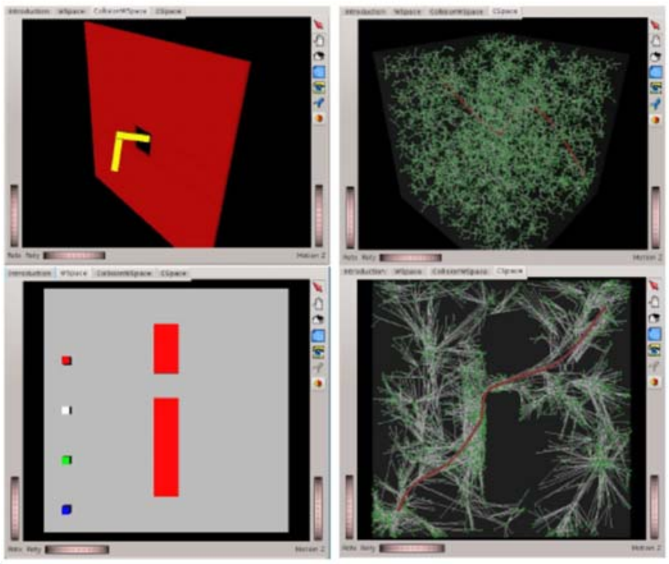
1. Visualization of rendering models and collision models.
2. Visualization of the workspace and of the configuration space
   - *For 2- or 3-dimensional problems*: the roadmaps, trees or potential landscapes
   - *For higher dimensional problems*: the projections of the configuration space onto the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



4. Basic features

4. Visualization

1. Visualization of rendering models and collision models.
2. Visualization of the workspace and of the configuration space
   - *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

1. Introduction
2. Results
3. Tools
4. Basic features
5. **Advanced features**
    1. **Coupling between degrees of freedom**
    2. **Constrained motion planning**
    3. **Dynamic simulation**
    4. **Integration with task planning**
    5. **Benchmarking**
6. The code
7. Getting started

5. 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.

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

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.

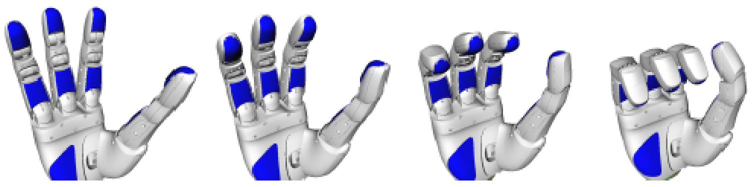
5. Advanced features

1. Coupling between degrees of freedom

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

```xml
<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



5. Advanced features

2. Constrained motion planning

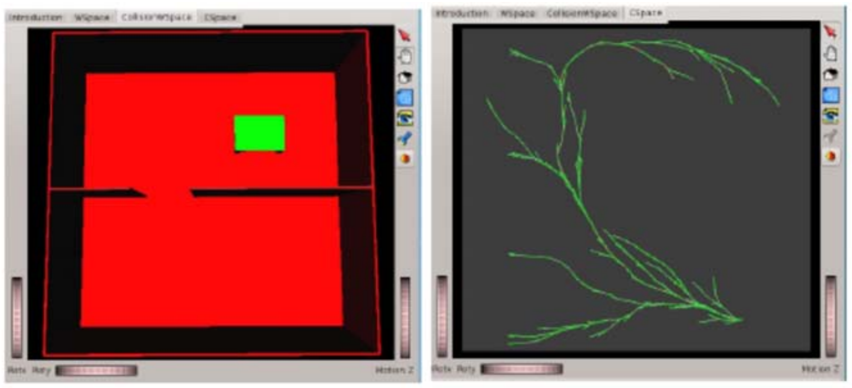
OMPL has two sets of planners: *geometric* and *based on controls*

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

Only the RRT has been currently incorporated



5. 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
- 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

5. Advanced features

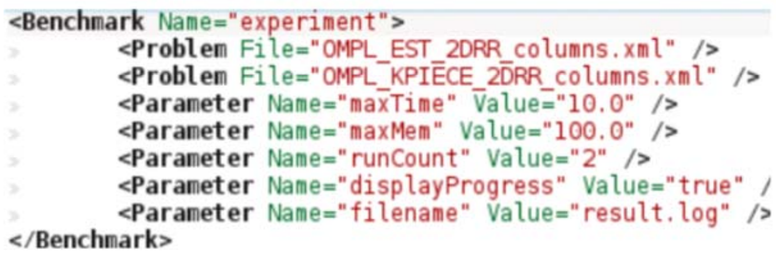
4. Integration with task planning

- To integrate task and motion planning, the ROS middleware is used.
- At the motion planning level, Kautham is encapsulated as a ROS service
- At the task planning level, the Cognitive Robot Abstract Machine (CRAM), behaves as a ROS client

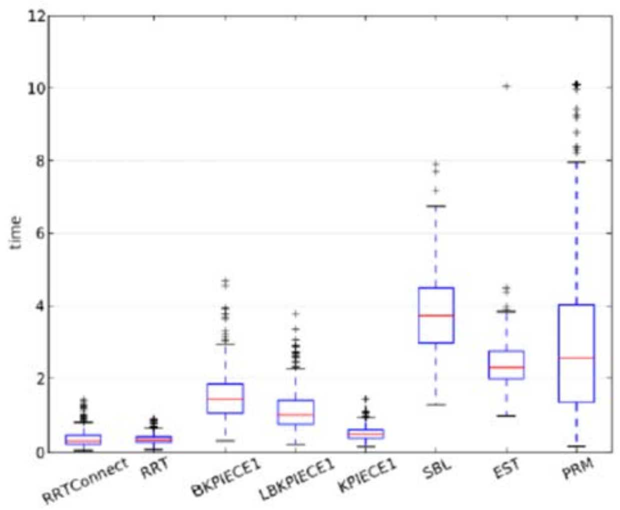
5. Advanced features

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>
```



Contents

1. Introduction
2. Results
3. Tools
4. Basic features
5. Advanced features
6. The code
7. Getting started

6. The code

Structure

demos
- **IOC_demos**
- **OMPL_demos**
- **models**
  - **robots**
  - **obstacles**
  - **controls**

doc
src
- applications
- external
- planner
- problem
- sampling
- util

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

IOC demos: Divided in four folders containing basic planners for 2D examples and for higher dimensional examples, and more elaborate planners for hand-arm robotic sysytems.

OMPL demos: Divided in several folders, one for each problem. Each folder contains the problem file defined to be solved with several planners.

Models: Contains the description of different robots using either dh parameters or the urdf description, and the files with the geometry. The obstacles are described by iv files.

6. The code

Structure

demos
doc
src
- **applications**
  - **console**
  - **graphical**
  - **ros**
- external
- planner
- problem
- sampling
- util

The three Kautham applications: Console, GUI and using ROS.

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

Widget classes: Define the widgets of the GUI
GUI application file: The main function to run the GUI.

ROS node: Provide access, using ROS services, to all the utilities offered by Kautham through the kauthamshell class.

6. The code

Structure

demos
doc
src
    applications
    external
        gdiam
        libann
        pqp
        pugixm
    planner
    problem
    sampling
    util

Third party libraries:

Gdiam: Used to compute bounding volumes for the collision checking.

Libann: Library to compute Approximate Nearest Neighbors. Used for the IOC PRM planner.

PQP: Collision detection library.

Pugixml: Library used to parse input XML files.

6. The code

Structure

demos
doc
src
    applications
    external
    planner
        ioc
        omplc
        omplg
    problem
    sampling
    util

The sources 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), local planners, a general PRM planner and several derived classes devoted to the planning of hand-arm robotic systems using PMDs.

omplc: Contains the planners based on the control OMPL planners.

omplg: Contains the planners based on the geometric OMPL planners.

6. The code

Structure

demos
doc
src
    applications
    external
    planner
    problem
    sampling
    util

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

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

6. The code

Structure

demos
doc
src
    applications
    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.

6. The code

Structure

demos
doc
src
  applications
  external
  planner
  problem
  sampling
  util

Auxiliar classes.

Libmt: library for the management of transformations.

Kthutil: Basic structs and enumerates.

Libkin: inverse kinematics for some robots.

Contents

1. Introduction
2. Results
3. Tools
4. Basic features
5. Advanced features
6. The code
7. Getting started

7. Getting started

0. Activate the ROS path with the menu: *Programari Docencia>ESAII>Activate ROS*

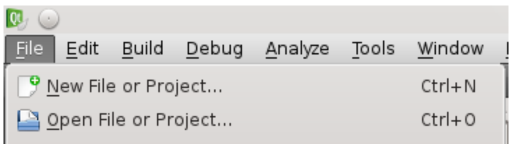
1. Create your workspace
   > mkdir –p catkin_ws/src
   > cd catkin_ws

2. Download the code from ATENEA and decompress it in catkin_ws

3. Build the application
   > catkin_make

4. Open the GUI application
   > cd devel/lib/kautham2
   > ./Kautham2

5. Browse the code
   > qtcreator&



~/catkin_ws/src/CMakeList.txt