Autonomous and Mobile Robotics

Prof. Giuseppe Oriolo

Motion planning in practice: An introduction to Kite

Massimo Cefalo

Dipartimento di Ingegneria informatica automatica e gestionale Antonio Ruberti

Sapienza Università di Roma

Outline

Introduction to Kite

- Kineo Kite, or simply Kite, is a commercial software for solving motion planning problems and implementing new planning algorithms
- based on software originally developed by Kineo CAM, a French company founded in 2000 by researchers of LAAS-CNRS
- Kineo CAM released KineoWorks, a platform for automatic motion planning and path planning
- in 2012 Kineo CAM was acquired by Siemens PLM Software, that currently sells Kineo Kite as an end-user application based on KineoWorks

Basic features

- fully modular architecture (specialized modules are available for specific applications)
- cross-platform (Linux/Windows)
- static and interactive collision detections
- automated collision-free path computation, analyzer and editor
- includes many path planners
- path optimization
- 3D animation and movie generation (avi format)

Compatibility features

- 3D engine based on OpenGL
- 3D graphical path and kinematics editor
- VRML 2 importer
- XML Parser for saving and loading scenes in KXML format
- compatibility with other applications supported by optional modules: for instance, using CGR Data Importer it is possible to import CATIA™ files (a CAD design and prototyping software used in many industrial applications)

An example: automatic assembly and disassembly in production processes

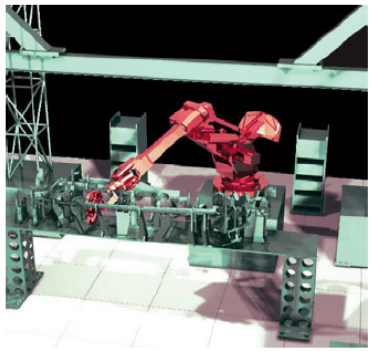
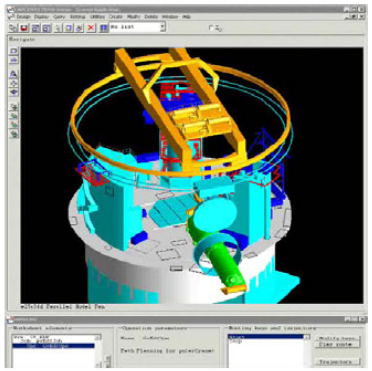
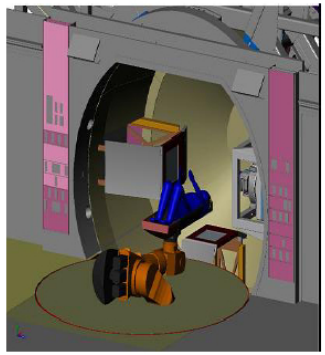
Kineo Collision Detection module (KCD)

Advanced features

- possibility to add new modules developing C++ libraries (addons)
- Python scripting for rapid prototyping and testing
- optional Kineo CAM addons provides further features
  - example: Kite Wrapping computes swept volumes of multiple objects along a path and exports volumes to VRML files

Kite for research

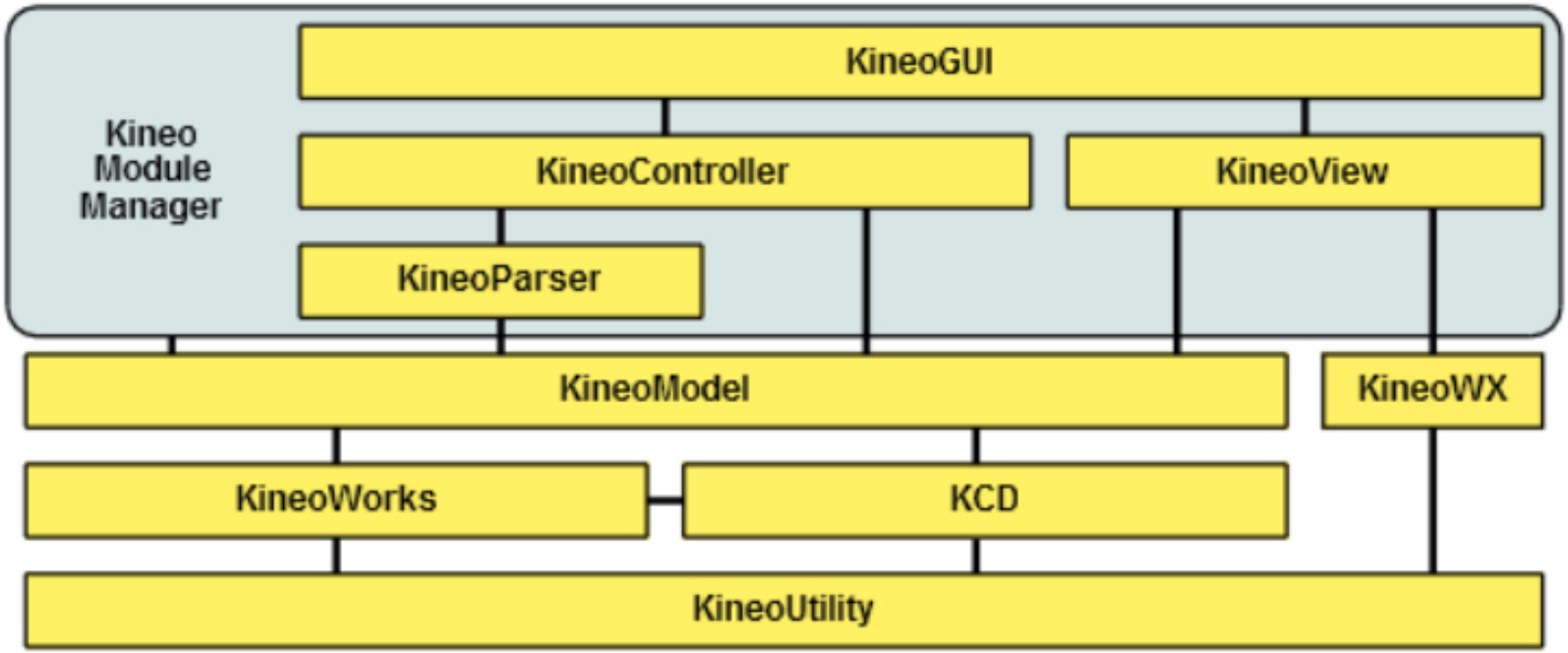
- Kite is distributed under commercial license, but allows academic and no-profit organizations to do research through the Application Programming Interfaces (API)
- Kite and its APIs together are called Kite SDK

Kite SDK architecture (1/2)

- Kite SDK is built on top of KCD (the Kineo Collision Detector) and KineoWorks (the Kineo path-planning library)
- Kite SDK uses a few design principles that make it quite scalable:
  - object-oriented design
  - layered model-view-controller (MVC) architecture
  - smart pointers for safe and easy memory management
- the Kite SDK Graphical User Interface is built on top of WxWidgets, the cross-platform GUI library
- the 3D scene is realized in OpenGL

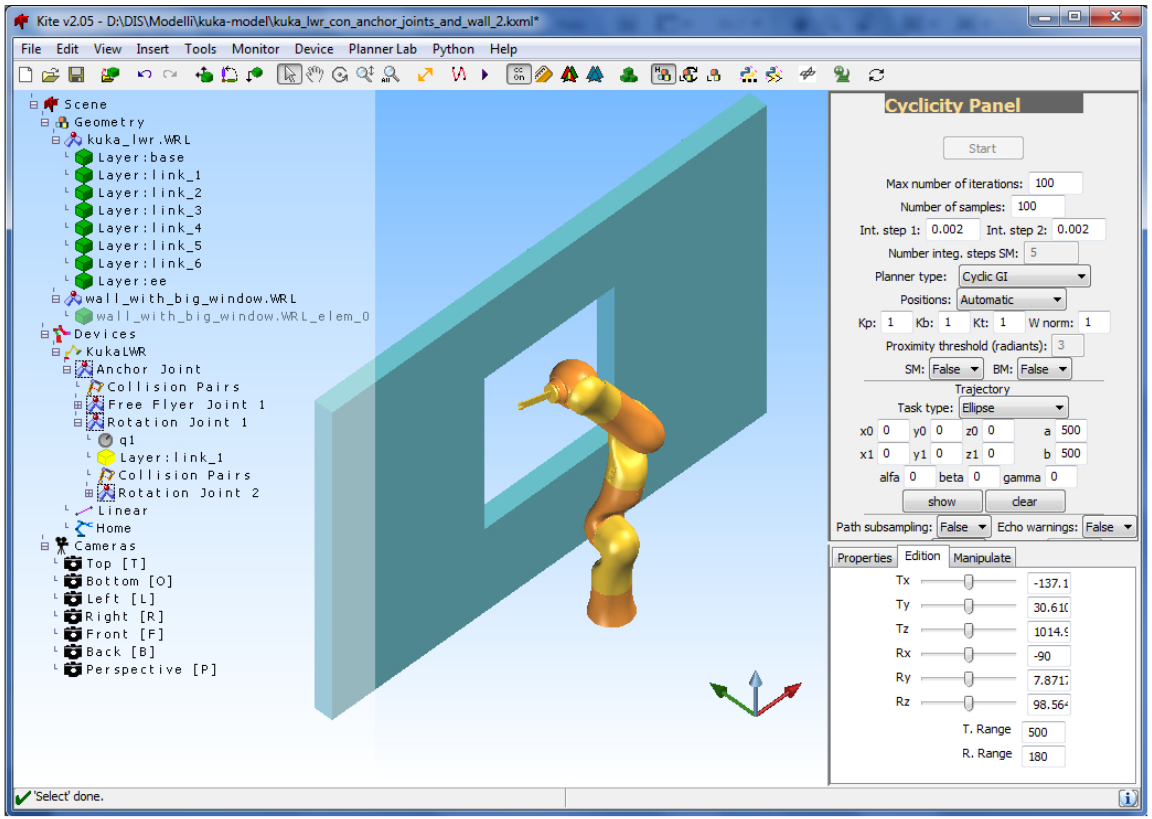
Kite SDK architecture (2/2)



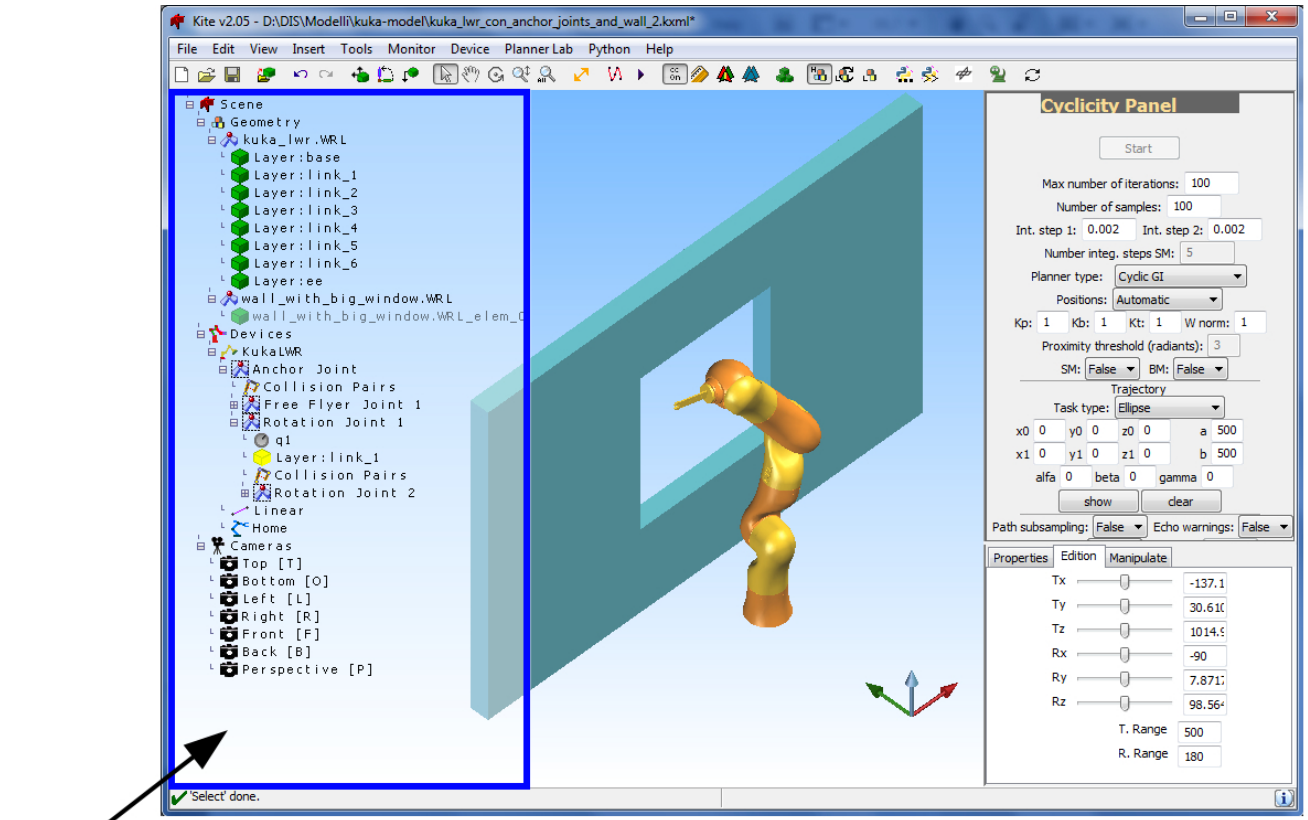
The most important modules are:

- KineoWorks: the core Kineo path planning library
- KCD: the core Kineo collision detection library

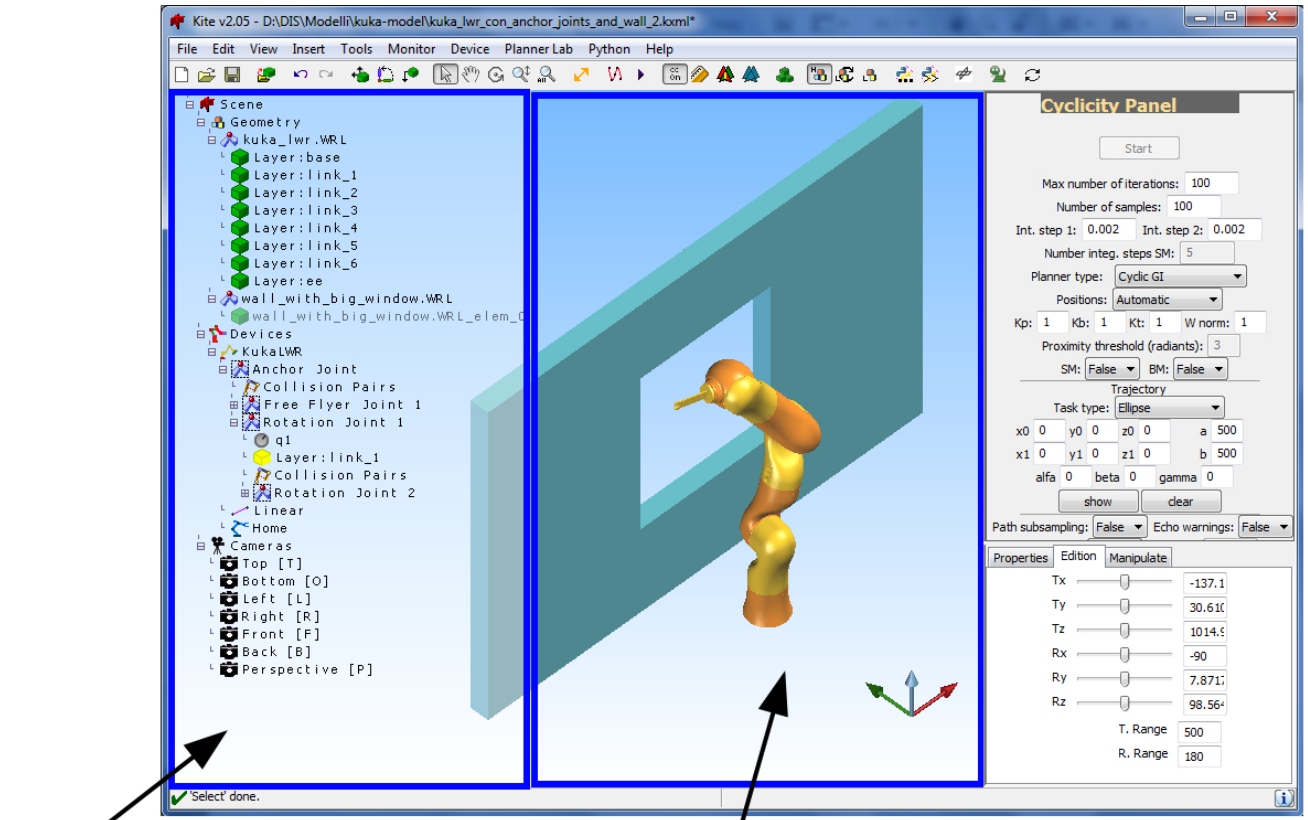
The GUI: a brief description



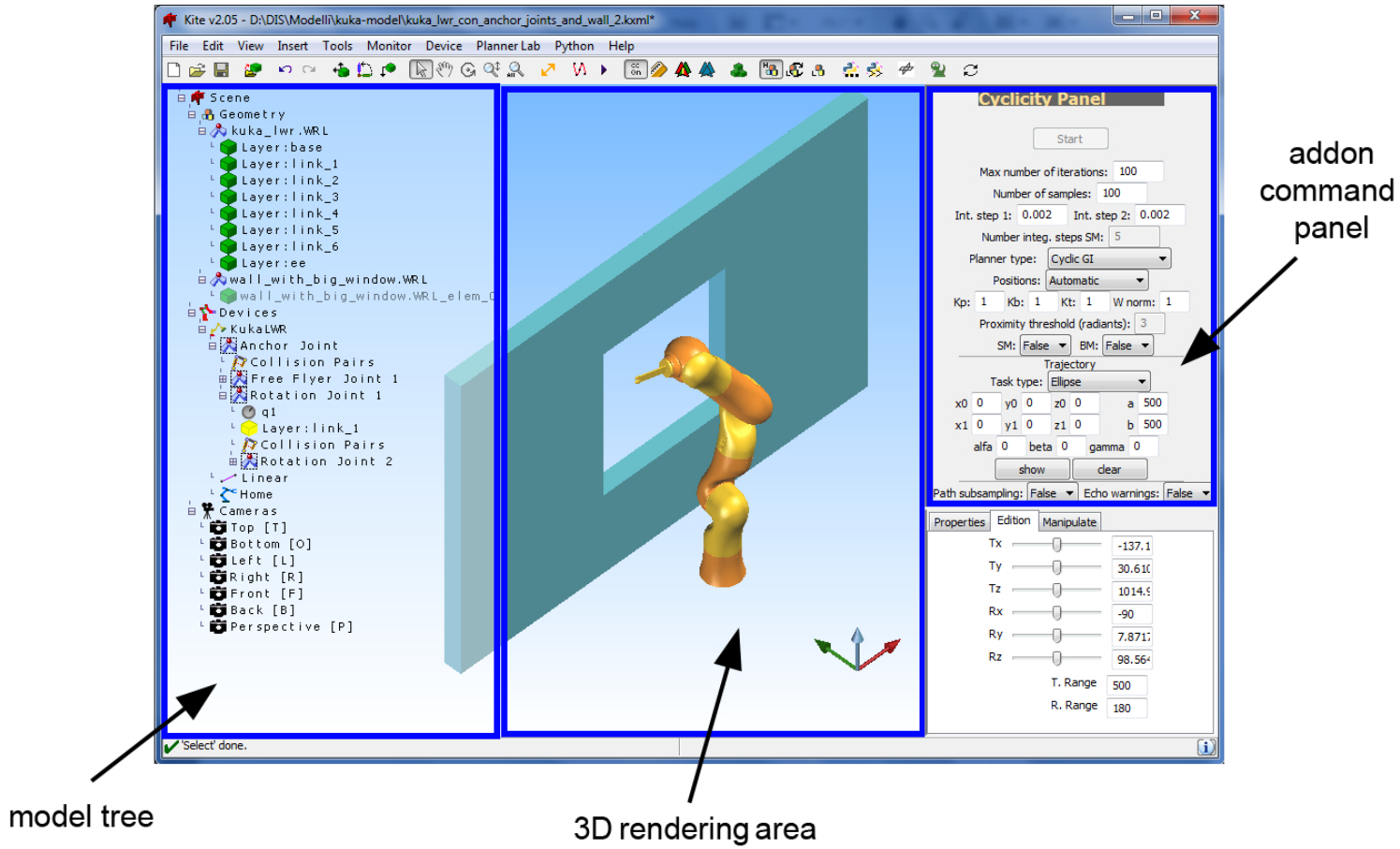
The GUI: a brief description



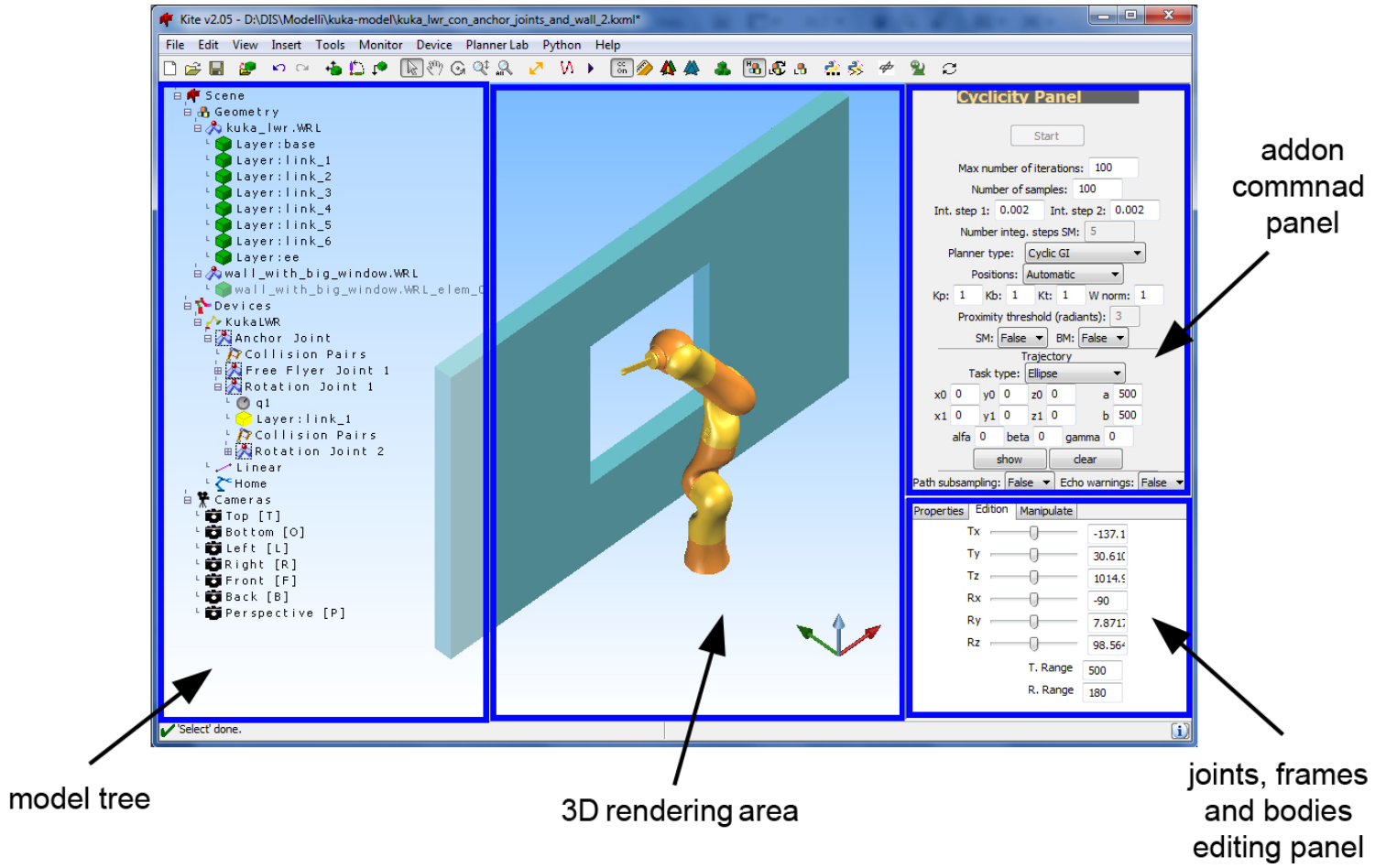
The GUI: a brief description



The GUI: a brief description



The GUI: a brief description



Demo 1: motion planning considering moving obstacle

Demo 2: motion planning considering dynamics

Kite: pros and cons

+ Kite offers a powerful engine for 3D simulations
+ the default distribution provides several path planners
+ can be extended by developing new addons to add numerical solvers and path planners
+ it provides a very efficient collision detection library
+ it is robust and reliable

- the documentation is in some point lacking
- does not cover physics (hence no dynamics, but a new version which manages sensors is being released)
- does not include a numerical solver (but this is typical for kinematics-only frameworks)
- it is not open source: details on the implementation are not known
- no modules to communicate with the physical world
- in Windows, it requires Visual Studio 2005 for developing new addons

Some interesting free alternatives to Kite (1/2)

- BioMove3D (from LAAS-CNRS; http://www.openrobots.org/wiki/move3d). The configuration space representation is the core feature of this library. It divides the world between movable articulated objects and static objects
- MPK: Motion Planning Kit (http://ai.stanford.edu/ mitul/mpk/). It is a C++ library and toolkit for developing single and multi robot motion planners
- Motion Strategy Library: a C++ library for motion planning free for both academic and commercial use (http://msl.cs.uiuc.edu/msl/)
- OMPL: the Open Motion Planning Library (http://www.kavrakilab.org/software). A C++ library for sampling-based motion planning. OMPL is also integrated in ROS and will be available as a ROS package

Some interesting free alternatives to Kite (2/2)

- OOPSMP: an Object-Oriented Programming System for Motion Planning (http://www.kavrakilab.org/software). It is still available for download, but is no longer further developed
- Simox: a C++ toolbox containing three libraries for 3D simulation of robot systems, sampling based motion planning and grasp planning (http://simox.sourceforge.net/)
- OpenRave: provides an environment for testing, developing, and deploying motion planning algorithms in real-world robotics applications (http://openrave.programmingvision.com/)
- **V-REP**: free, open source and multi-platform; simulates physics and provides ROS connectivity (http://www.coppeliarobotics.com/)

V-REP, in particular...

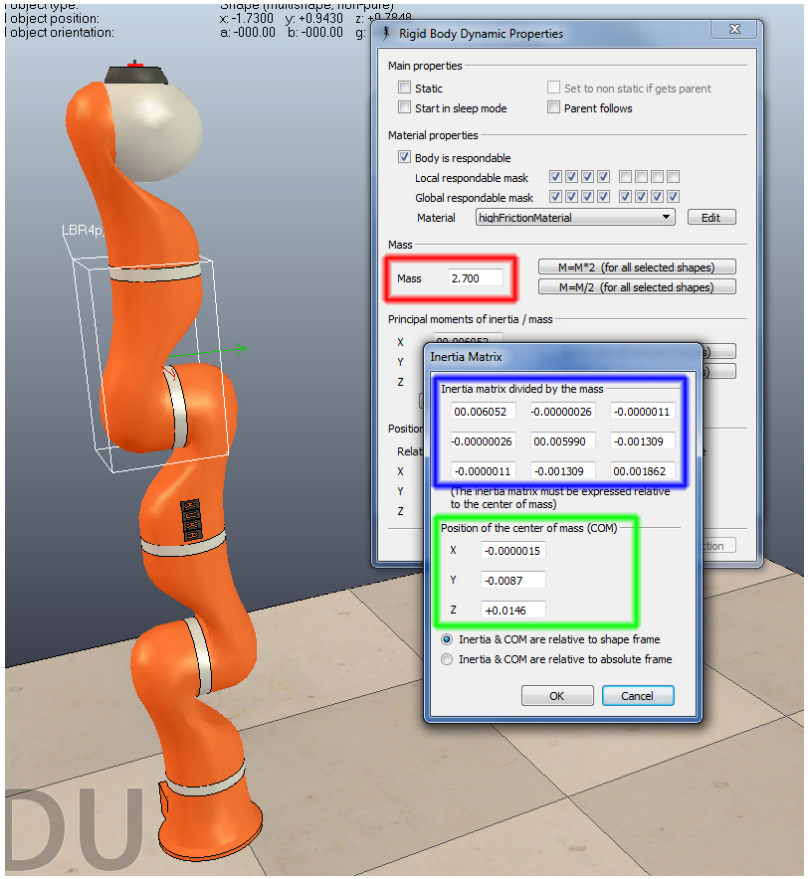
V-REP is a robotic simulator: a software environment aimed at generic robotic applications. With respect to Kite:

+ it is free and open source
+ it provides engines for dynamic/physics simulations
+ it allows the simulation of sensors
+ many robot models are available in the standard distribution
+ its functionalities can be extended using many programming languages (C/C++, Python, Java, Lua, MATLAB, Octave, Urbi) and programming approaches (remote clients, plugins, ROS nodes...)
- its collision detection library is less efficient than KCD
- does not provide tools for motion planning : it is a general purpose software environment, not specifically oriented to motion planning

V-REP promises to become a reference platform for prototyping and developing robotics applications.

Vrep official demo video 2014

Dynamic modeling in V-REP

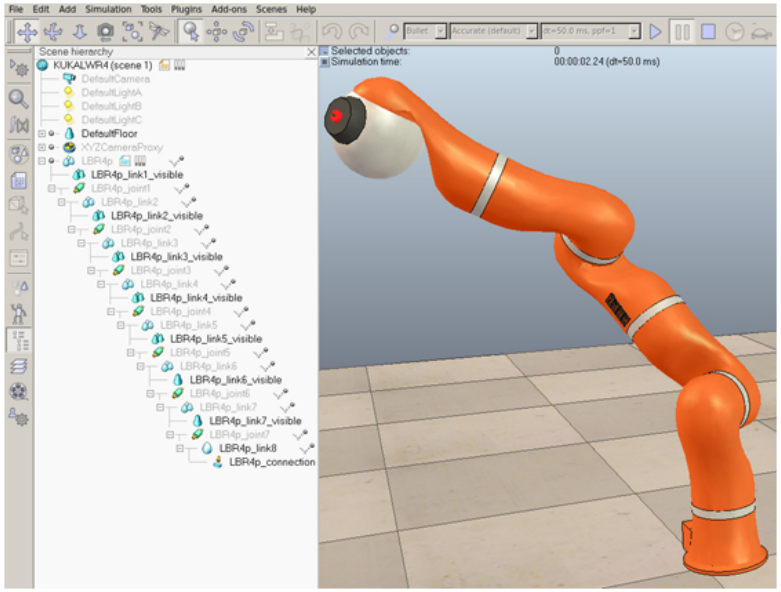


Building a dynamic model in V-REP is very easy: no equations are needed

It requires a few simple steps:

1. import a CAD model of the robot
2. associate to each body of the robot its dynamic parameters: mass, center of mass, inertia matrix

Dynamic modeling in V-REP



3 build a model tree: a tree that represents all hierarchical information of the kinematic chains. It contains bodies and joints

V-REP and MATLAB