Development Plan for OSS World Challenge 2012

Registration No.	2012- ※Registration No. need not be written.
Program title	The Open Motion Planning Library

※ Please be sure to fill in all of the requested information.

1. Program Overview

1) Development goals (background, goals etc.)

The goal of the Open Motion Planning Library (OMPL, http://ompl.kavrakilab.org) is to provide efficient, extensible, and freely available implementations of sampling-based motion planning algorithms. Sampling-based algorithms are commonly used to plan motions for robot manipulators and other systems with a large number of degrees of freedom or complex constraints. The library contains both reference implementations of classic algorithms as well as some of the current state-of-the-art algorithms. We are working with several research groups to create implementations of their algorithms in OMPL. OMPL has been integrated in the Robotic Operating System (ROS, http://www.ros.org) (although it can also be used without it).

The software is targeted at three different audiences: motion planning researchers, robotics industry, and education. Researchers benefit from the common building blocks that make it easy to develop new algorithms, compare against other algorithms, and test on a growing set of benchmark problems. Due to its integration with ROS, the software is easily deployable on industrial and research robotic systems. This provides an easy pathway from motion research idea to something that is deployed on real hardware. Finally, we have developed a teaching module on motion planning that includes programming assignments centered around OMPL.

2) System configuration

(I assume this refers to the build system.) OMPL uses CMake (http://www.cmake.org) for its build system, which makes it easy to check for all dependencies, and facilitates porting OMPL to other operating systems. The core OMPL library relies only on Boost (http://www.boost.org). Additional functionality is available if the Open Dynamics Engine (http://ode.org) is installed.

3) Menus

OMPL is primarily a library, but we have developed an additional software layer on top of OMPL called OMPL.app that integrates collision checking, mesh loading, as well as a GUI (all using other Open Source libraries). The GUI is developed in Python using either PyQt (http://www.riverbankcomputing.co.uk/software/pyqt) or PySide (http://www.pyside.org) for the menus and other widgets.

4) Language used for development

The main library is written in C++. There is additional Python code to automatically generate Python bindings using Py++ (http://sourceforge.net/projects/pygccxml). Py++ is a code generation program that relies on Boost.Python to create Python bindings for C++ code. Almost all C++ functionality is accessible from Python.

5) Systems used

Development is primarily done on Linux and OS X, but OMPL runs on MS Windows as well. We have started to set up a Continuous Integration server (http://teamcity.kavrakilab.org) so that we can check wether code builds and passes tests on a variety of system configurations (including several Linux distributions).

6) Plan for each stage of development

Each developer has his own self-contained project that builds on the core of OMPL, which is stable at this point. We have regular research meetings and group meetings at Rice. The team leader, consultants and core developers have biweekly Skype meetings to discuss progress and future directions. Independently, other researchers are already using OMPL in their own research. In some cases such we are working with them to include their extensions in OMPL, while with others it makes more sense to leave the extension a separately distributed package. Some extensions we have received so far include:

- An implementation of two algorithms that converge to optimal paths, written by Alejandro Perez and Sertac Karaman from MIT, two of the original authors. These algorithms were published as recent as 2011 and were originally implemented in Matlab and only applicable to simple toy examples. Thanks to the rich infrastructure in OMPL these algorithms can now be run on any robot that runs OMPL / ROS.
- A generalization of PRM (the first sampling-based motion planning algorithm), written by James Marble at the University of Nevada at Reno.
- A generalization of RRT (another popular motion planning algorithm), written by Jennifer

Barry at MIT.

An example of software that builds on OMPL but is not part of it is the Lightning planning framework (http://sourceforge.net/p/lightningros) by Dimitry Berenson and Cameron Lee at the University of California at Berkeley. This framework is designed to make a robot learn how to plan using past experience and uses OMPL to find new plans.

7) Number of personnel input and work assignment

There are currently 6 people involved in the development of OMPL. The work assignment is done based on research interest.

2. Long-term prospects of the program developed (No specific form is required.)

In the long term we hope to establish OMPL as the de facto implementation of sampling-based motion planning algorithms that new algorithms will be compared against. Ioan Sucan, one of the developers, is working on a project to systematically compare performance of planning algorithms (including ones from other libraries) on a number of benchmark problems (both synthetic problems as well as real-world data). This will be published on an ongoing basis on a public web site, a prototype of which can be found at http://plannerarena.org.

We are also in close contact with the organizers of ROS-Industrial, a group that aims to certify ROS software components for industrial use. OMPL can have a significant impact in industry where many manipulators are still programmed “by hand”.

Finally, we aspire to change the way samplings-based motion planning algorithms are taught. We provide a clear implementation of algorithms as described in the motion planning literature that maps important concepts directly to C++ classes, so that students can get a more in-depth understanding of the material. Rather than spending significant time writing code, students can immediately use the building blocks that OMPL provides and spend more time on reflecting on what they have learnt. The freely available education material will enhance what the students around the world can learn.