OMPL: The Open Motion Planning Library

Mhequb Hayat
Department of ECE
Center for Advanced Studies in Engineering (CASE).
Islamabad, Pakistan.

Muddasir Khan and Talha Manzoor
Graduate Students-LUMS
Reference/Acknowledgment


• Slides of Mark Moll: Research Scientist Rice University
  I have copied most of the content from his presentation.
BSD License

• Anyone can modify the software
Pre-requisites for using OMPL

- Familiar with C++ programming and compiling code in a Unix environment
- Basic knowledge of Sampling-based Motion Planning
Intended Use

Education:
Class Project from COMP 450 on Path Optimization: Fall 2010
Rice University.

WFAR3 Nov. 2012
Intended Use

Motion Planning Research
Intended Use

Industry:
Intended Use

**Example:** Real-time footstep planning for humanoid robots
Benefits

- A repository of **planners**: choose the right planner and right parameters for that planner.
- Compare **new planners** to existing ones.
- Develop significantly more complex **specialized planners**.
- Enable challenging **research**.
- Support **education** of new scientists.

WFAR3 Nov. 2012
Benefits

ROS Motion Planner uses OMPL
ROS+OMPL=Motion Planning Solution
Benefits

ROS Motion Planner
Other Motion Planning Software

- **MPK**, Schwarzer, Saha, Latombe
- **MSL**, LaValle et al.
- **OpenRAVE**, Diankov & Kuffner
- **KineoWorks**, Laumond et al.
- **OOPSMP**, Plaku et al.
Other Related Robotics Software

• ROS
• Player/Stage, Player/Gazebo
• Webots
• MORSE
• Microsoft Robotics Developer Studio
Open Motion Planning Library
OMPL.app GUI

- OMPL.app a GUI build upon core OMPL
- Uses specific representation of Robot and environment
- Provides collision checking mechanisms
OMPL

• Core OMPL does not explicitly represent Robot and Environment
• There is no default collision checker in OMPL
• For flexibility and applicability to wide variety of Robotics systems
• User must select
  – a computational representation of Robot and Environment
  – Collision detection method
OMPL

Example:
Robot: PR2
Environment
Collision Detection:
Point Cloud from Laser, Stereo etc
Main Features of OMPL
OMPL in a Nutshell

• Common core for sampling-based motion planners
• Includes commonly-used heuristics
• Takes care of many low-level details often skipped in corresponding papers

WFAR3 Nov. 2012
Abstract Interface to all core Motion Planning Concepts

- state space / control space
- state validator (e.g., collision checker)
- sampler
- goal (problem definition)
- planner
- ...
- except robot & workspace...

WFAR3 Nov. 2012
States & state spaces

Abstract state space

- rotation (2D, 3D)
- translation ($\mathbb{R}^n$)
- compound

API requirements
- StateType
- alloc/free state
- distance
- interpolation
- state equality

used for:
- rigid body motions
- manipulators
- ...

WFAR3 Nov. 2012
Control spaces & controls

• Needed only for control-based planning
• Analogous to state spaces and states:

Abstract control space

$\mathbb{R}^n$

compound

API requirements:
- ControlType
- alloc/free control
- equality
State validators

• Problem-specific: must be defined by user
• Checks whether state is collision-free, joint angles and velocities are within bounds, etc.
State validators

• **Optionally,** specific state validator implementations can return
  – distance to nearest invalid state (i.e., nearest obstacle)
  – gradient of distance

*Can be exploited by planners / samplers!*
Most common state validator: collision checker

Several options:

• Implemented in ROS on top of sensor-derived world model
• Implemented in OMPL.app for triangle meshes using PQP library

Need to define specific world representation to implement collision checking
Samplers

- For every **state space** there needs to be a **state sampler**
- State samplers need to support the following:
  - sample uniform
  - sample uniform near given state
Samplers

- sample from Gaussian centered at given state
Valid state samplers

- **Valid state samplers** combine low-level state samplers with the validity checker

- Simplest form: sample at most $n$ times to get valid state or else return failure
Valid state samplers

• Other sampling strategies:
  – Try to find samples with a large clearance
  – Try to find samples near obstacles
Goals

- Goal
  - Goal Region
    - GoalSampleableRegion
      - GoalState
        - single goal state
      - GoalStates
        - multiple goal states
      - GoalLazySamples
        - multiple goal states, computed in separate thread

Goal can only tell whether state satisfies Goal condition
Goal Region provides distance to goal region
GoalSampleableRegion can sample from goal region
Planners

• Take as input a **problem definition**: object with one or more **start states and a goal object**

• Planners need to implement two methods:

  • **solve**:
    – takes `PlannerTerminationCondition` object as argument
    – termination can be based on timer, external events, ...

  • **clear**:
    – clear internal data structures, free memory, ready to run solve again
Many planners available in OMPL

- PRM
- SBL
- LBKPIECE
- RRT
- KPIECE
- LazyRRT
- EST
- BKPIECE
- RRTConnet

geometric planning

planning with controls

WFAR3 Nov. 2012
API overview

only when planning with differential constraints

- ControlSampler
- ControlSpace
- StateSpace
- StateSampler
- ValidStateSampler
- SpaceInformation
- StateValidityChecker
- MotionValidator
- Planner
- SimpleSetup
- ProblemDefinition
- Goal
- UserCode

WFAR3 Nov. 2012
Minimal code example

```cpp
    StateSpacePtr space(new SE3StateSpace());
    // set the bounds (code omitted)

    SimpleSetup ss(space);
    // "isStateValid" is a user-supplied function
    ss.setStateValidityChecker(isStateValid);

    ScopedState<SE3StateSpace> start(space);
    ScopedState<SE3StateSpace> goal(space);
    // set the start & goal states to some values
    // (code omitted)

    ss.setStartAndGoalStates(start, goal);
    bool solved = ss.solve(1.0);
    if (solved)
        setup.getSolutionPath().print(std::cout);
```
OMPL.app GUI
Define Problem
Planner
Bounding Box
Visualization of Solution
OMPL.app GUI
Example
OMPL online

- Web site: http://ompl.kavrakilab.org
- Mailing lists:
  Developers: ompl-devel@lists.sourceforge.net
  Users: ompl-users@lists.sourceforge.net
- Public Mercurial repository:

WFAR3 Nov. 2012
OMPL Tutorials

• Step-by-step walkthroughs for:
  – geometric planning for rigid body in 3D
  – working with states and state spaces
  – representing goals
  – Benchmarking
  – creating new planning algorithms

WFAR3 Nov. 2012
Competition for Students

Solve Motion Planning Problem using OMPL.app GUI. Optimize performance of the planner by changing the parameters.

WFAR3 Nov. 2012
That’s All

Thank you ...

Any Questions?