Com S 476/576 Lab 7

Introduction to Open Motion Planning Library (OMPL)

Yuechuan Xue
yuechuan@iastate.edu

Com S 476/576, Spring 2021

Installation

- http://ompl.kavrakilab.org/installation.html
- Ubuntu
  - Download the OMPL installation script:
    http://ompl.kavrakilab.org/install-ompl-ubuntu.sh
  - `chmod u+x install-ompl-ubuntu.sh`
  - Next, `./install-ompl-ubuntu.sh --app`
    will install the latest release of OMPL.app with Python bindings
  - Allocate at least 6GB RAM for your virtual machine
  - On my computer, the installation took 2 hours

Reference/Acknowledgement

- Ioan A. Șucan, Mark Moll, Lydia E. Kavraki, **The Open Motion Planning Library**, IEEE Robotics & Automation Magazine, December 2012.

- Slides of **Mark Moll: Research Scientist Rice University**
  I have copied most of the content from his presentation.

Sampling-based planning algorithms

Roadmaps:

- **PRM [Kavraki, Svestka, Latombe, Overmars '96]**
- **Obstacle based PRM [Amato, Bayazit, Dale '98]**
- Medial Axis PRM [Wilmarth, Amato, Stiller '98]
- **Gaussian PRM [Boor, Overmars, van der Stappen '01]**
- Bridge Building Planner [Hsu, Jiang, Reif, Sun '03]
- Hierarchical PRM [Collins, Agarwal, Harer '03]
- Improving PRM Roadmaps [Morales, Rodriguez, Amato '03]
- Entropy guided Path-planning [Burns, Brendan, Brock '04]
- RESAMPL [Rodriguez, Thomas, Pearce, Amato '06]
- Probab. foundations of PRM [Hsu, Latombe, Kurniawati '06]
- Adaptive PRM [Kurniawati et al. '08]
- Multi-model planning [Hauser et al. '10]
- Small-tree PRM [Lanteigne et al. '11]
- Rapidly-exploring Random Roadmap [Alterovitz et al. '11]
- ...

bold = included with OMPL

Sampling-based planning algorithms

Trees:

- **EST [Hsu et al. '97, '00]**
- **RRT [Kuffner, LaValle '98]**
- **RRT-Connect [Kuffner, LaValle '00]**
- **SBL [Sanchez, Latombe '01]**
- RRF [Li, Shie '02]
- Guided EST [Phillips et al. '03]
- PDRRT [Ranganathan, Koenig '04]
- SRT [Plaku et al. '05]
- DDRRT [Yershova et al. '05]
- ADDRRT [Jaillet et al. '05]
- RRT-Blossom [Kalisiak, van Panne '06]
- **PDST [Ladd, Kavraki '06]**
- **KPIECE [Șucan, Kavraki '08]**
- **T-RRT [Jaillet et al. '10]**
- **SyCLoP [Plaku et al. '10]**
- **RRT* [Karaman et al, '10]**
- RRG [Karaman et al, '10]
- **PRM* [Karaman et al, '10]**
- Bi-RRT* [Akgun et al. '11]
- SR-RRT [Lee et al. '12]
- RRT# [Arslan et al. '13]
- **STRIDE [Gipson et al. '13]**
- **SPARS [Bekris et al. '13]**
- ...

bold = included with OMPL

OMPL in a nutshell

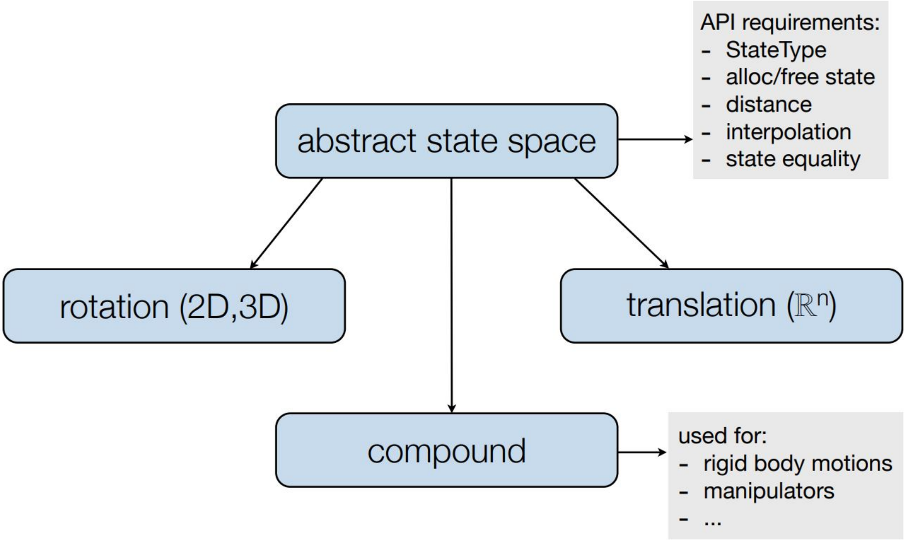
- Common core for **sampling-based motion planners**
- Includes commonly-used heuristics
- Takes care of many low-level details often skipped in corresponding papers
- Intended for use in
  - Education
  - Research
  - Industry

Abstract interface to core sampling-based motion planning concepts

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

Except robot & workspace ...

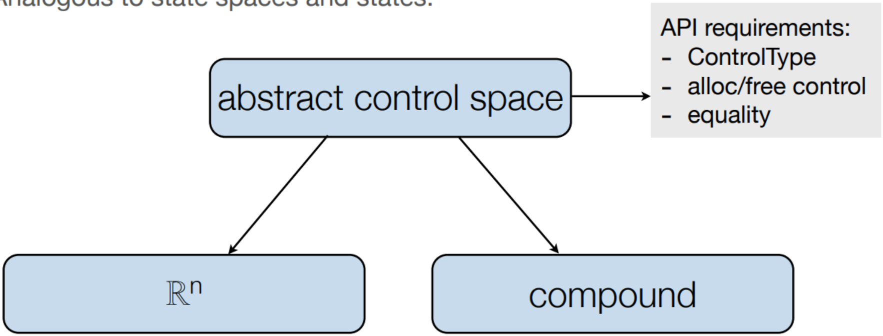
State & State Space



Control & Control Space

- Needed only for control-based planning

- Analogous to state spaces and states:



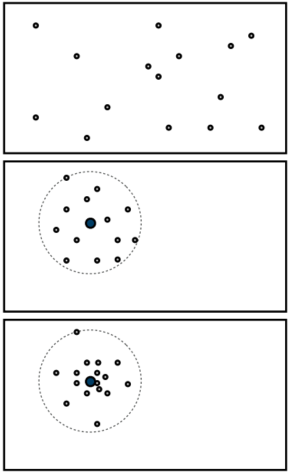
State Validators

- Problem-specific; **must** be defined by user or defined by layer on top of OMPL core $\rightarrow$ **MoveIt!**
- Checks whether state is collision-free, joint angles and velocities are within bounds, etc.
- **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!

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
  - sample from Gaussian centered at given state



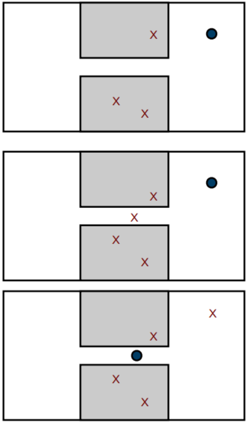
Valid State Sampler

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

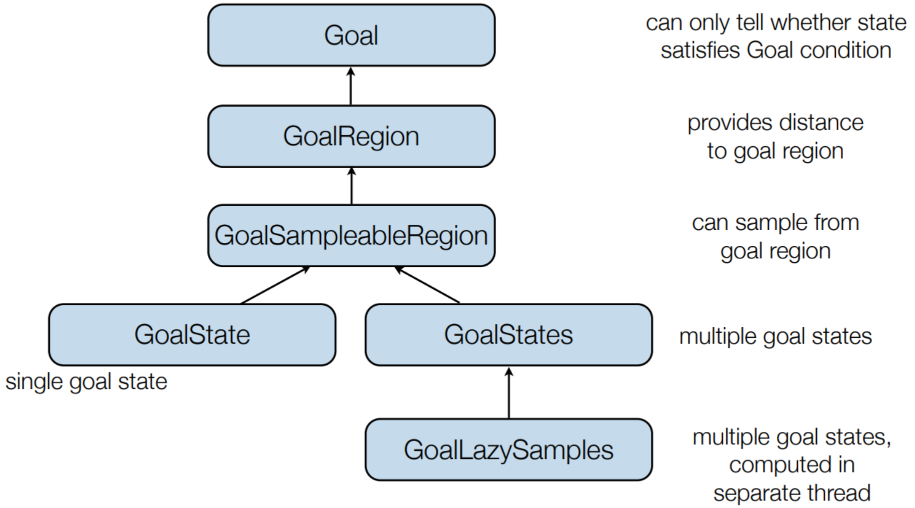
- Simplest form: sample at most n times to get valid state or else return failure

- Try to find samples with a large clearance

- Try to find samples near obstacles (more dense sampling in/near narrow passages)



Goals


Goal
can only tell whether state satisfies Goal condition
GoalRegion
provides distance to goal region
GoalSampleableRegion
can sample from goal region
GoalState
single goal state
GoalStates
multiple goal states
GoalLazySamples
multiple goal states, computed in separate thread

OMPL Planning Algorithms

- 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

Demo - SE3 Rigid Body Planning

```
task = oa.SE3RigidBodyPlanning()

# load the robot and the environment
task.setRobotMesh('3D/cubicles_robot.dae')
task.setEnvironmentMesh('3D/cubicles_env.dae')

start = ob.State(task.getSpaceInformation())
start().setX(-4.96) # setY, setZ, rotation().setIdentity() ...
goal = ob.State(task.getSpaceInformation())
goal().setX(200.49)
task.setStartAndGoalStates(start, goal)

# setting collision checking resolution to 1% of the space extent
task.getSpaceInformation().setStateValidityCheckingResolution(0.01)
task.setup()

if task.solve(10):
    task.simplifySolution()
    path = task.getSolutionPath()
    path.interpolate(10)
    print(path.printAsMatrix())
```