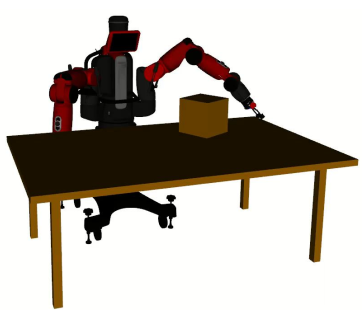
Benchmarking for Robot Motion Planning

Lydia E. Kavraki and Constantinos Chamzas
Department of Computer Science
Rice University

RICE UNIVERSITY

Motion Planning



7-Dof Baxter

8-Dof Fetch

15-Dof R2

OMPL

Benchmarking Tools

Motion Planning

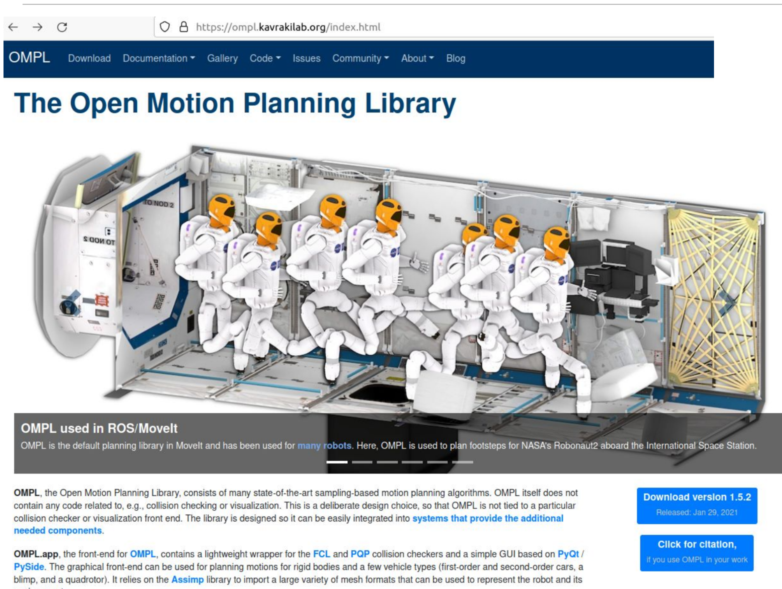
1.OMPL

Environments

Benchmarking

Tuning

The Open Motion Planning Library (OMPL)



Online at:
https://ompl.kavrakilab.org

Public Repository:
https://github.com/ompl

Contact at us:
ompl-devel@lists.sourceforge.net
ompl-devel@lists.sourceforge.net

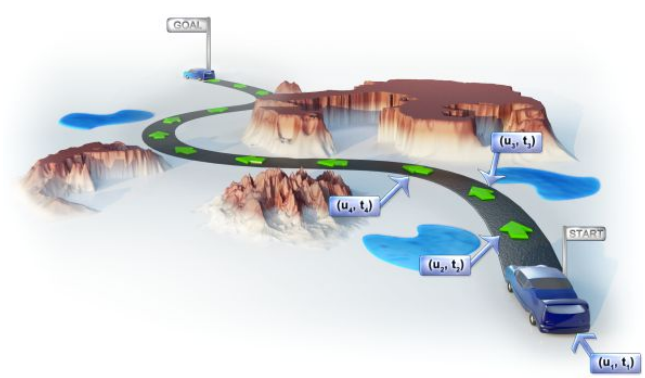
Publication:
"The open motion planning library.", Sucan et al, *RAM,* 2012

OMPL Metrics 2020

- Used in more then 126 robotic platforms
- > 3,000 **registered** users (many more get OMPL from package managers or do not register)
- Available in many package managers for all common operating systems:
  - apt (Ubuntu/Debian)
  - MacPorts/HomeBrew (macOS)
  - vcpkg (Microsoft Windows)
- OMPL web site since January, 2011:
  - 550,259 sessions
  - 221,092 unique visitors
  - 2,121,630 page views, 4 pages/visit, 6 minutes per visit

What is OMPL?

- C++ library (with Python bindings)
- Common core for sampling-based motion planners
- Includes commonly-used heuristics
- Takes care of many low-level details often skipped in corresponding papers
- Implementations of many state-of-the-art motion planners
- Intended for use in:
  - Education
  - Research
  - Industry



OMPL Planning Algorithms – January 2022

Multi-query Planners
- **PRM** (Kavraki et al 1996)
- **LazyPRM** (Bohlin & Kavraki 2000)

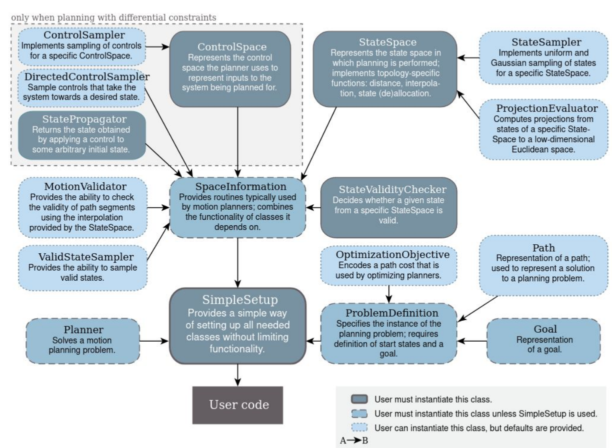
Single-Query Planners
- **RRT** (Kuffner & Lavalle 2000)
- **RRTConnect** (Kuffner & Lavalle 2000)
- **EST** (Hsu et al 1999)
- **SBL** (Sancez & Latombe 2001)
- **KPIECE** (Sucan & Kavraki 2008)
- **BKPIECE** (Sucan & Kavraki 2008)
- **LBKPIECE** (Sucan & Kavraki 2008)
- **STRIDE (**Gibson et al. 2013)
- **PDST** (Ladd & Kavraki 2005)
- **QRT** (Orthey 2019)

Optimizing Planners
- **Transition-Based RRT** (Jaillet et all 2000)
- **PRM*** (Karaman & Frazzoli 2011)
- **SPARS** (Dobson et al. 2012)
- **SPARS2** (Dobson & Bekris 2013)
- **RRT*** (Karamazon &Frazzoli 2011)
- **InformedRRT*** (Gammel et al. 2014)
- **Batch Informed Trees (**Gammel et al. 2015)
- **AIT*** (Strub and Gammel 2021)
- **EIT*** (Strub and Gammel 2021)
- **LBTRRT** (Salzman & Halperin 2013)
- **Sparse Stable RRT** (Li et al. 2016)

OMPL is an Abstract Interface

- State space (e.g, configuration space)
- State validator (e.g., collision checker)
- State sampler / Motion validator
- Start/Goal (problem definition)
- Motion Planner
- and so on...

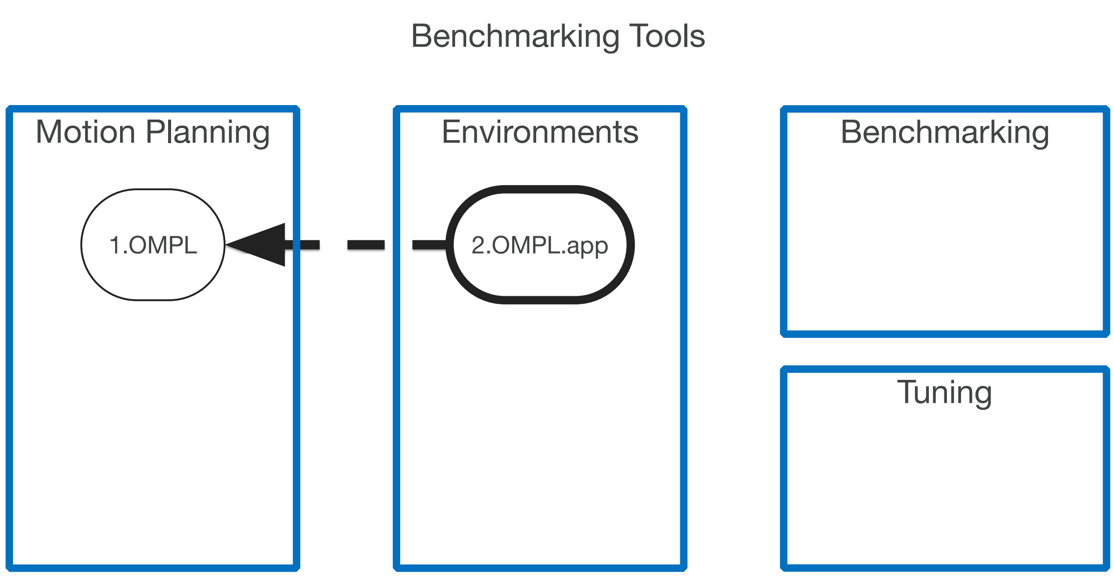
except robot & workspace...



What is OMPL not?

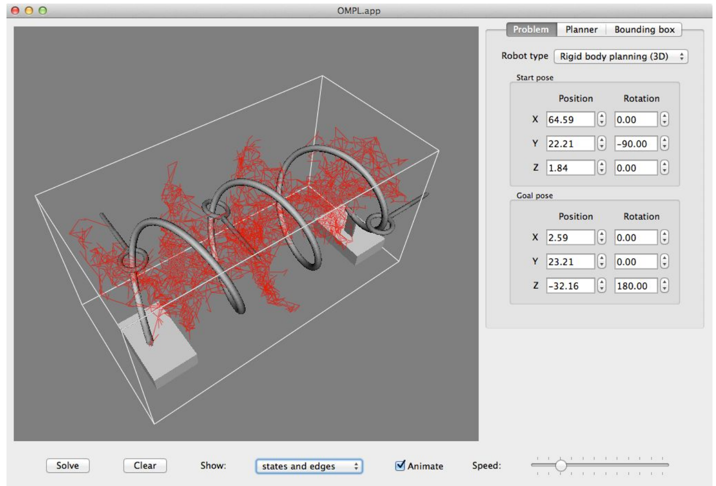
- OMPL does **not**:
  - Represent/Visualize the geometry of the robot or the workspace
  - Provide collision checking routines
  - Simulate robot kinematics or dynamics

Robowflex

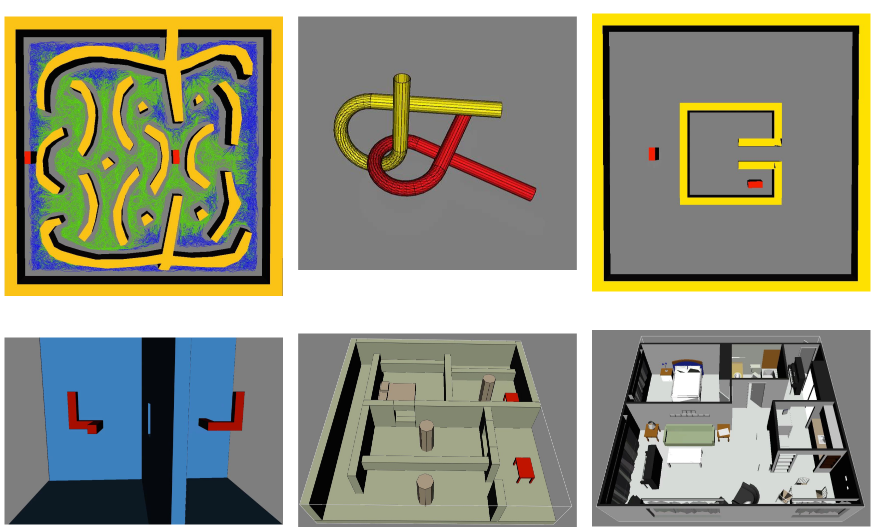


What is OMPL.app?

- Lightweight GUI front-end for OMPL
- Assumes a robot and environment representation (triangle meshes)
- Provides collision checking routines



Sample OMPL.app problems

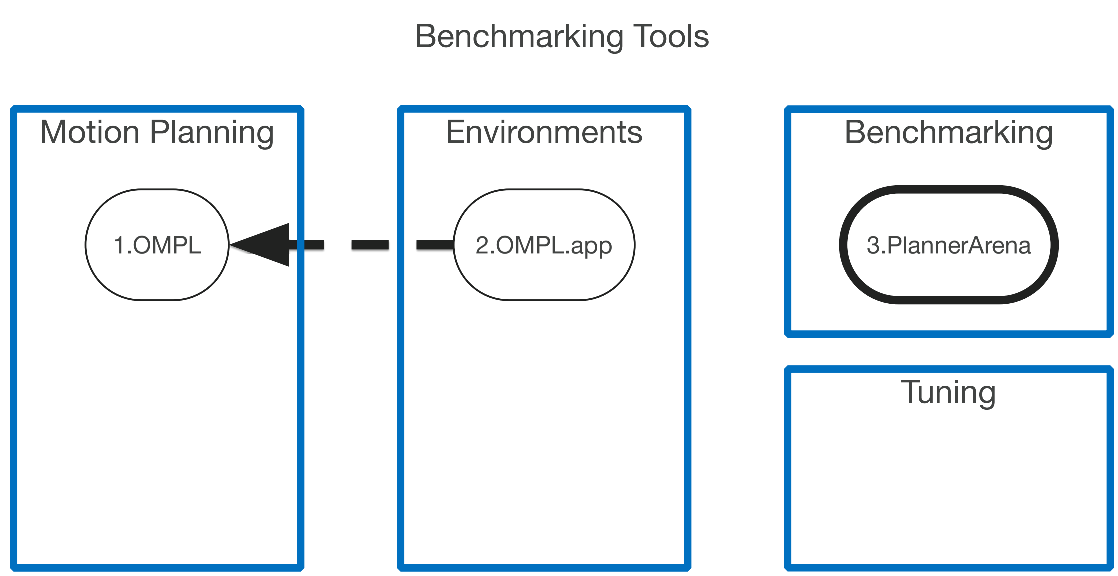


Minimal code example (Python and C++)

```python
space = SE3StateSpace()
# set the bounds (code omitted)

ss = SimpleSetup(space)
# "isStateValid" is a user-supplied function
ss.setStateValidityChecker(isStateValid)

start = State(space)
goal = State(space)
# set the start & goal states to some values
# (code omitted)

ss.setStartAndGoalStates(start, goal)
solved = ss.solve(1.0)
if solved:
    print setup.getSolutionPath()
```

```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);
```

PlannerArena



Planner Arena

Number of missing data points out of the total number of runs per planner

planner	missing	total
BKPIECE1	0	50
EST	0	50
KPIECE1	0	50
LBKPIECE1	0	50
PDST	0	50
SBL	0	50

Online at:
https://plannerarena.org

Publication:
"Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization,“ Moll et al., *RAM*, 2015

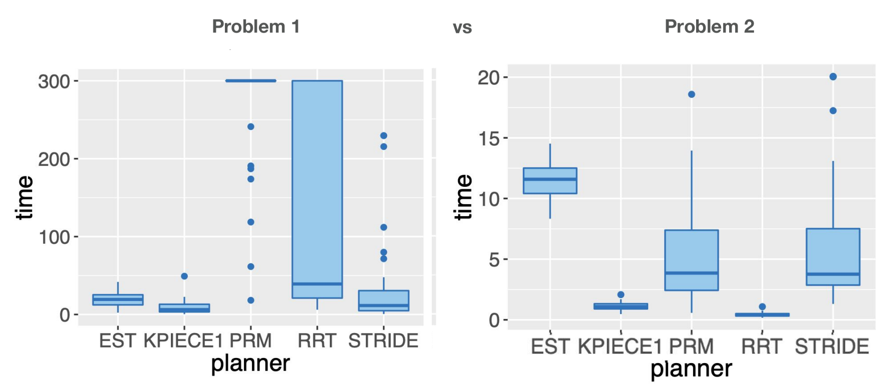
Benchmarking

```
SimpleSetup setup;
// motion planning problem setup code omitted
Benchmark b(setup, “My First Benchmark”);

b.addPlanner(base::PlannerPtr(new geometric::RRT(setup.getSpaceInformation())));
b.addPlanner(base::PlannerPtr(new
geometric::KPIECE1(setup.getSpaceInformation())));
b.addPlanner(base::PlannerPtr(new geometric::SBL(setup.getSpaceInformation())));
b.addPlanner(base::PlannerPtr(new geometric::EST(setup.getSpaceInformation())));
b.addPlanner(base::PlannerPtr(new
geometric::PRM(setup.getSpaceInformation())));

b.benchmark(runtime_limit, memory_limit, run_count, true);
b.saveResultsToFile();
```

Script post-processes benchmark log files to create/update SQLite database and plots

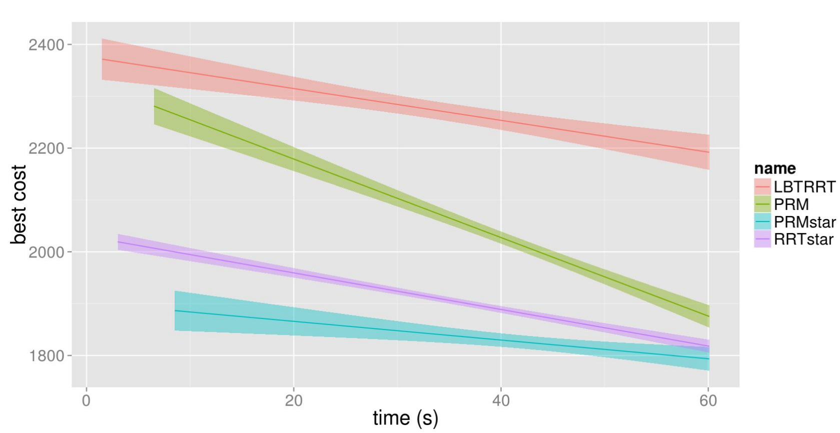
OMPL/PlanerArena Benchmarking

Easily compare performance of planners on different problems:

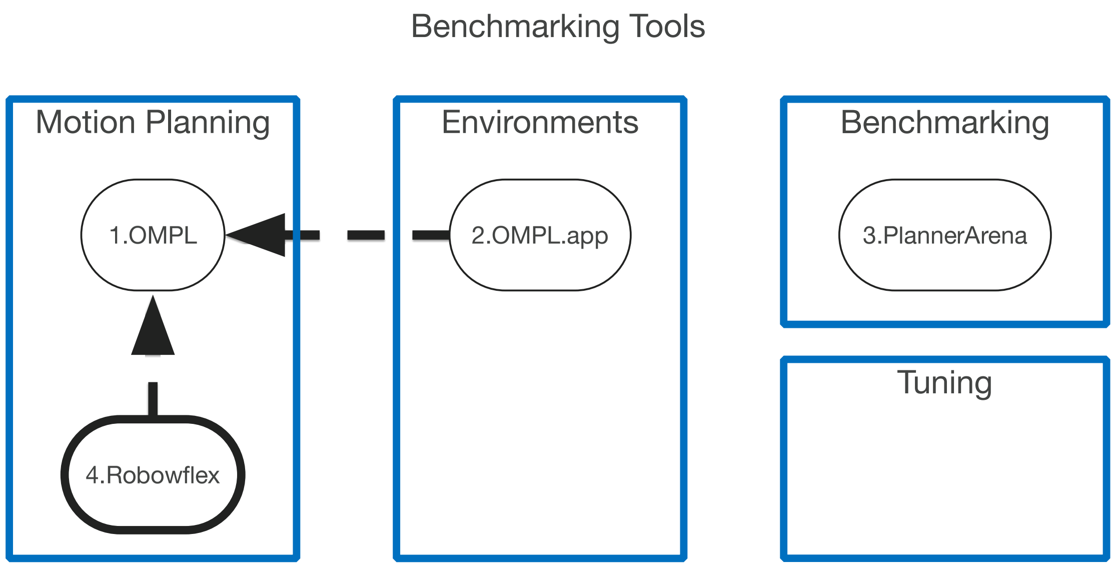


OMPL/PlannerArena Benchmarking

Plot convergence rate (with confidence intervals) for asymptotically (near)-optimal planners:



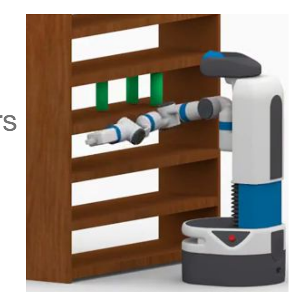
Robowflex



Robowflex

Robowflex is:

- A high-level library that wraps around MoveIt
- Research focused, e.g., for benchmarking, customizing planners
- Open source
- Integrated with other libraries, e.g., DART, Tesseract, etc.



Metrics:

- Used in 9 publications
- 49 stars and 12 forks on Github

Online at:
https://github.com/KavrakiLab/robowflex

Publication:
“Robowflex: Robot Motion Planning with MoveIt Made Easy,” Kingston et al.,*Arxiv* 2021

Scripting

Robowflex simplifies setting up:

- Robot models
- Planning scenes
- Motion Planners

```cpp
// Create a default Fetch robot.
auto fetch = std::make_shared<rx::FetchRobot>();
fetch->initialize();

// Create an empty scene.
auto scene = std::make_shared<rx::Scene>(fetch);

// Create the default planner for the Fetch.
auto planner = //
  std::make_shared< //
    rx::OMPL::FetchOMPLPipelinePlanner>(fetch);
planner->initialize();

// Create a motion planning request.
auto request = //
  std::make_shared<rx::MotionRequestBuilder>( //
    planner, "arm_and_torso");
```

No launch files necessary, can do everything inside scripts

Save scenes and requests to file

```cpp
// Set the start state.
fetch->setGroupState("arm_and_torso",
  {0.05, 1.32, 1.40, -0.2, 1.72, 0, 1.66, 0});
request->setStartConfiguration( //
  fetch->getScratchState());

// Set the goal state.
fetch->setGroupState("arm_and_torso",
  {0.27, 0.5, 1.28, -2.27, 2.24, -2.77, 1, -2});
request->setGoalConfiguration( //
  fetch->getScratchState());

// Set the desired planner.
request->setConfig("RRTConnect");

// Do motion planning!
auto result = planner->plan( //
                scene, request->getRequest());
if (result.error_code_.val //
    != moveit_msgs::MoveItErrorCodes::SUCCESS)
    return 1;

return 0;
```

Research: OMPL Integration / Benchmarking

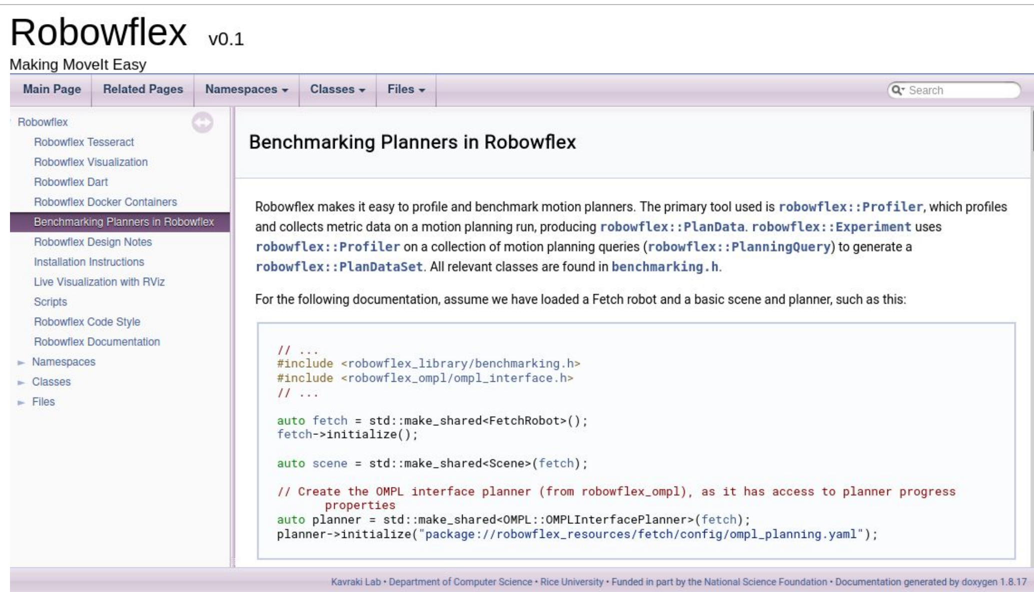
Easy to access underlying OMPL planner to customize behavior with MoveIt

```cpp
// Create an OMPL planner
auto planner = //
  std::make_shared< //
    rx::OMPL::OMPLInterfacePlanner>(fetch);

// Extract underlying OMPL structures
auto context = //
  planner->getPlanningContext(scene, request);
auto ss = context->getOMPLSimpleSetup();

// Customize OMPL planner
ss->setPlanner(…);
ss->getStateSpace()->setStateSamplerAllocator(…);
```

Benchmark scenes and planners and retrieve results in standard formats (e.g., OMPL's for **PlannerArena**)

```cpp
rx::Benchmarker benchmarker;
benchmarker.addBenchmarkingRequest( //
  "example", scene, planner, request);

benchmarker.benchmark({ //
  std::make_shared<rx::OMPLBenchmarkOutputter>( //
    "example/")});
```

Extensive Documentation and Demo Scripts

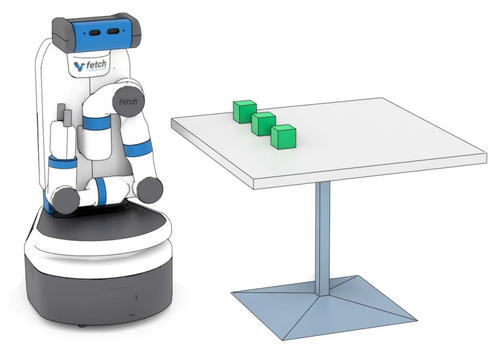


DART Integration

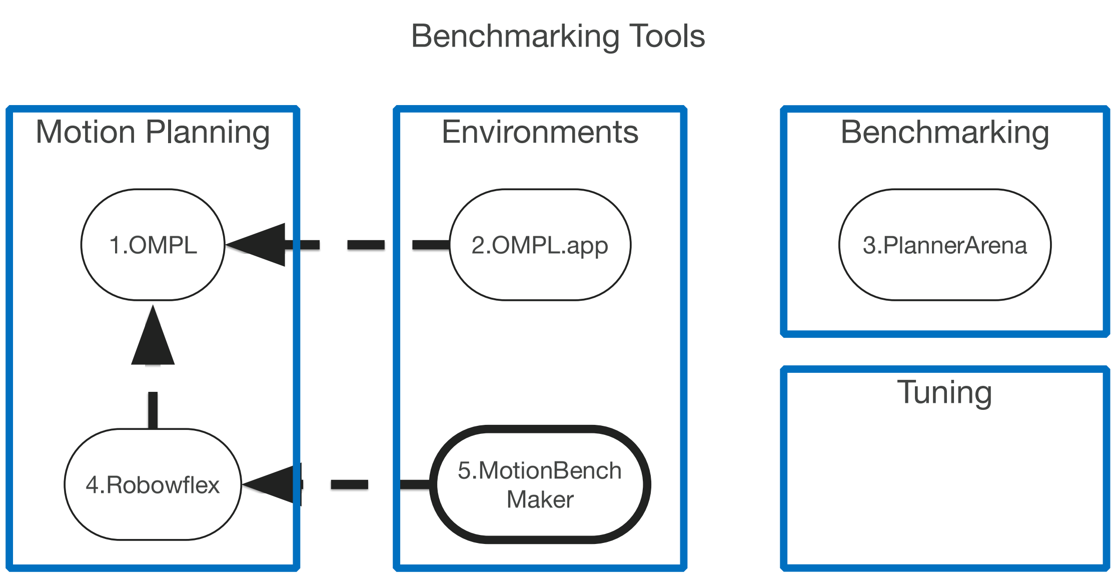
- DART (Dynamic Animation and Robotics Toolkit) is an open source physics simulator

Blender Integration

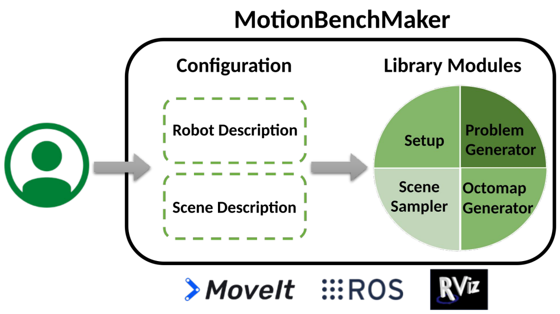
- Simplified pipeline for visualization with Blender



MotionBenchMaker



MotionBenchMaker



Public Repository:
https://github.com/Kavrakilab/motion_bench_maker

Publication:
"MotionBenchMaker: A Tool to Generate and Benchmark Motion Planning Datasets," Chamzas et al., *RAL*, 2022

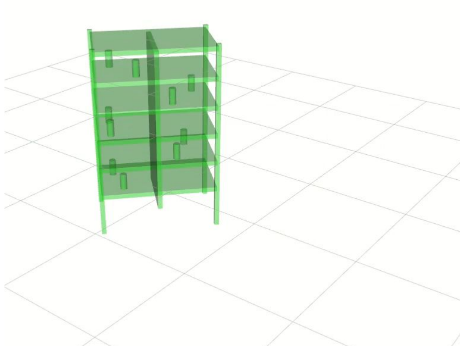
Main Modules of MotionBenchmaker

- **Scene Sampler:** Given a nominal scene it generates "realistic" variations through SE(3) and URDF sampling.

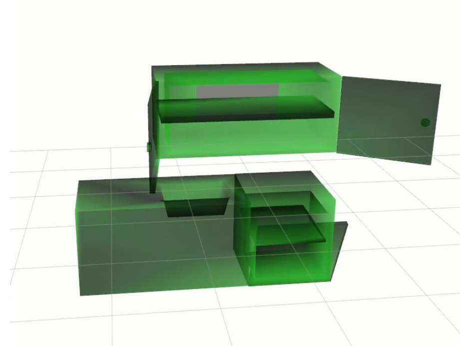
- **Problem Generator:** Given an object-centric relative pose, a nominal scene and a robot description, it generates a full motion planning problem.

- **Octomap Generator:** Converts a geometrically-represented scene to a pointcloud or octomap representation.

MotionBenchMaker: Scene Sampler

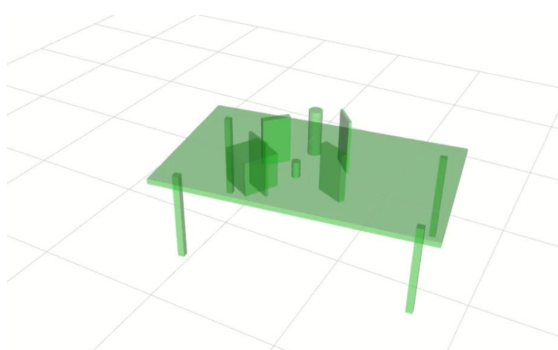


SE(3) Sampling



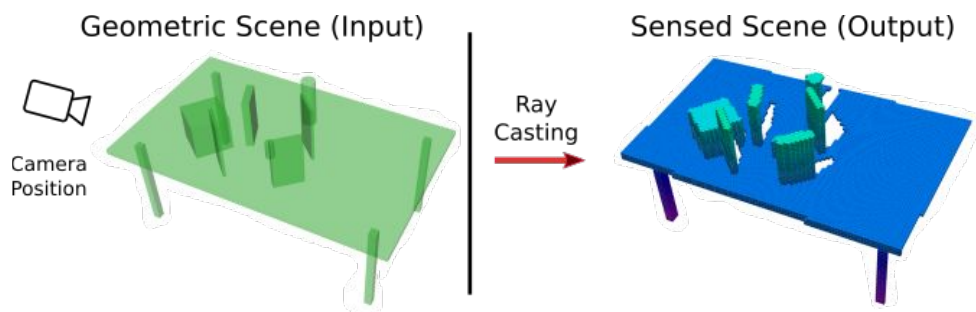
URDF Sampling

MotionBenchMaker: Problem Generator



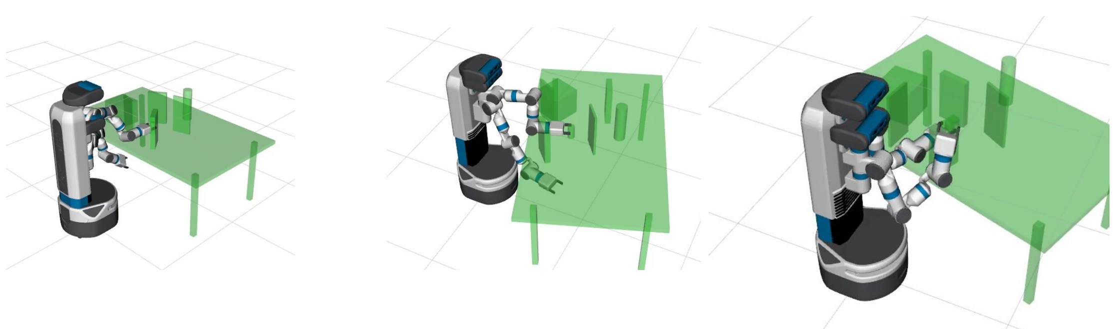
A robot-agnostic scene is resolved to a full motion planning problem by finding valid goal/start configurations.

MotionBenchMaker: Octomap Generator



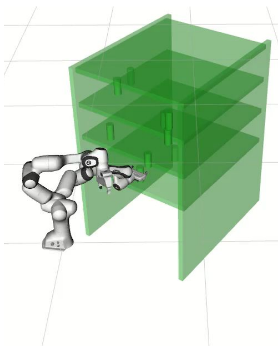
A geometric scene representation is converted to a pointcloud/octomap representation.

Easily generate different problems

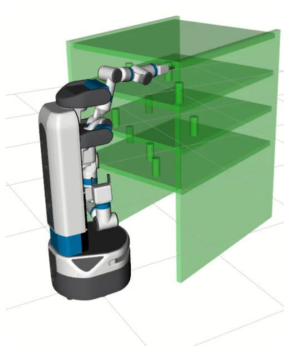


Different variations of the nominal scene create different motion planning problems.

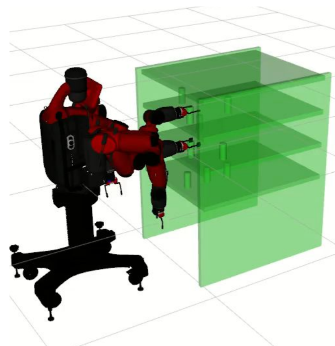
A different robot can easily be used



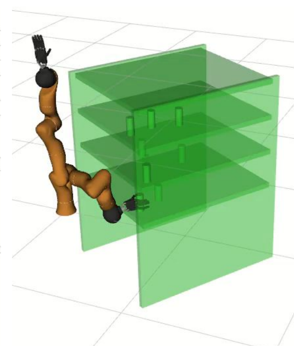
7-Dof Manipulator



8-Dof Manipulator



14-Dof double manipulator



31-Dof multi-tip manipulator

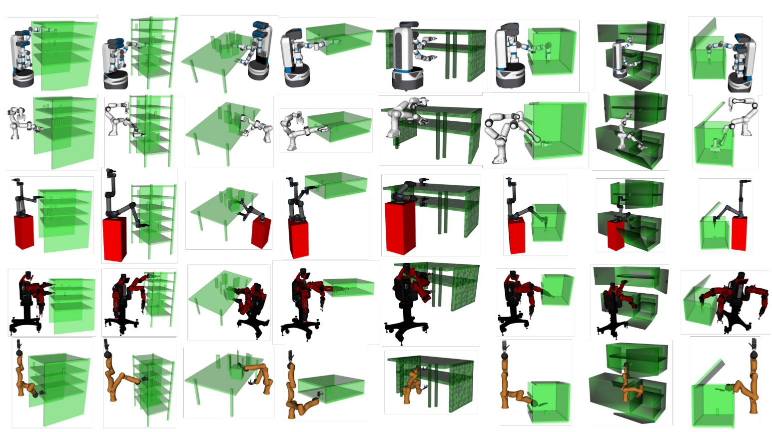
Different robots can easily be used with the same nominal scene.

A precomputed set of benchmarking datasets

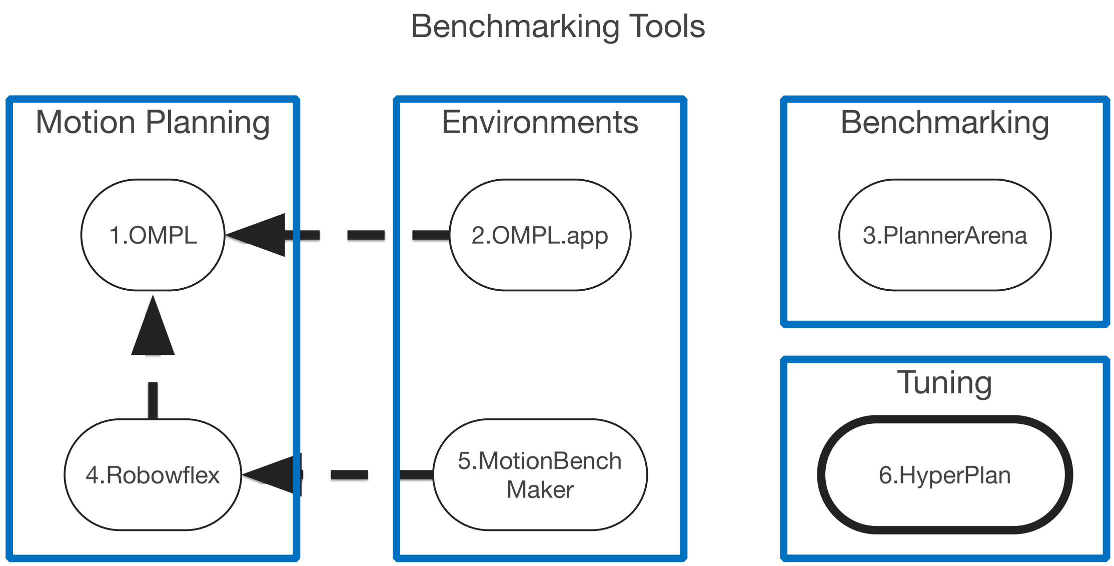
- A set of 40 different motion planning scenarios, with 5 robots and 8 scenes each with a 100 different motion planning queries

- Available for download online from the public github repo

- Can be easily used with any robowflex-supported platform



HyperPlan



HyperPlan

HyperPlan :

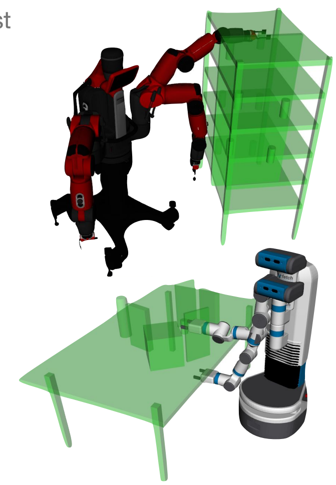
- Is a parameter optimization framework for motion planners
- Includes suitable metrics for different optimization criteria
- Can improve the overall planning performance by up to an order of mangitude

Public Repository:
https://github.com/Kavrakilab/hyperplan
(Coming Soon!)

Publication:
"HyperPlan: A Framework for Motion Planning Algorithm Selection and Parameter Optimization," Moll et al., *IROS*, 2021

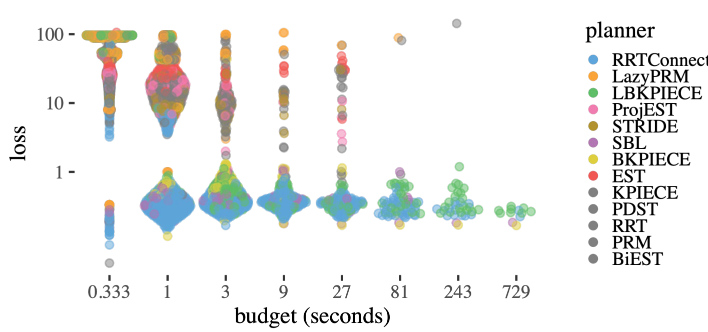
So many planners, so little time

- **Motion planning researchers** need to compare their algorithms against state of the art.
- **Practitioners** need to select and tune a performant algorithm for given applications.
- Key questions in both cases:
  - Which planners to compare against or select for use?
  - How can we intelligently tune their performance?
  - How do we characterize performance?



Hyperparameter optimization to the rescue!

- Motion planning algorithm selection and parameter tuning can be cast as **hyperparameter optimization problem**.
- **Key question**:
  how to design **loss functions** that give estimates of motion planning algorithm performance?



Main results

- Hyperparameter optimization can give **order of magnitude performance improvement** over baseline
- Optimized planner configuration's performance **generalizes broadly**
- Motion planning researchers: use HyperPlan to establish a good baseline for comparison
- Practitioners: use HyperPlan to tune planning performance for given application domain

Overview

Tool	Online at	Publication
OMPL	https://ompl.kavrakilab.org	"The open motion planning library.", Sucan et al, RAM, 2012
PlannerArena	https://plannerarena.org	"Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," Moll et al., RAM, 2015
Robowflex	https://github.com/KavrakiLab/robowflex	"Robowflex: Robot Motion Planning with MoveIt Made Easy," Kingston et al.,Arxiv 2021
MotionBenchMaker	https://github.com/Kavrakilab/motion_bench_maker	"MotionBenchMaker: A Tool to Generate and Benchmark Motion Planning Datasets" Chamzas et al., RAL, 2022
HyperPlan	https://github.com/Kavrakilab/hyperplan (Coming Soon!)	"HyperPlan: A Framework for Motion Planning Algorithm Selection and Parameter Optimization," Moll et al., IROS, 2021