OMPL: The Open Motion Planning Library

Mark Moll and Lydia E. Kavraki
Department of Computer Science
Rice University
Houston, TX
USA

Motion planning problems are hard

PROBLEM	COMPLEXITY
Geometric Constraints:	
Sofa Mover (3DOF)	$O(n^{2+\varepsilon})$ - not implemented [HS96]
Piano Mover (6DOF)	Polynomial – no practical algorithm [SS83]
n Disks in the Plane	NP-Hard [SS83]
n Link Chain in 3D	PSPACE-Complete [HSS87]
Generalized Mover	PSPACE-Complete [Canny88]
Dynamics Constraints:	
Point with Newtonian Dynamics	NP-Hard [DXCR93]
Polygon Dubin's Car (Linear)	Decidable [CPK08]
Nonlinear	Unknown, probably undecidable
Discrete Transitions and Dynamics Constraints:	
Hybrid Systems	Undecidable [Alur et. al 95]

Exact, approximate, and probabilistically complete algorithms

Method	Advantage	Disadvantage
exact	theoretically insightful	impractical
cell decomposition	easy	does not scale easily
control-based	online, very robust	requires good trajectory
potential fields	online, easy	slow or fail
sampling-based	**fast and effective**	**cannot recognize impossible query**

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]
and many others

Trees:

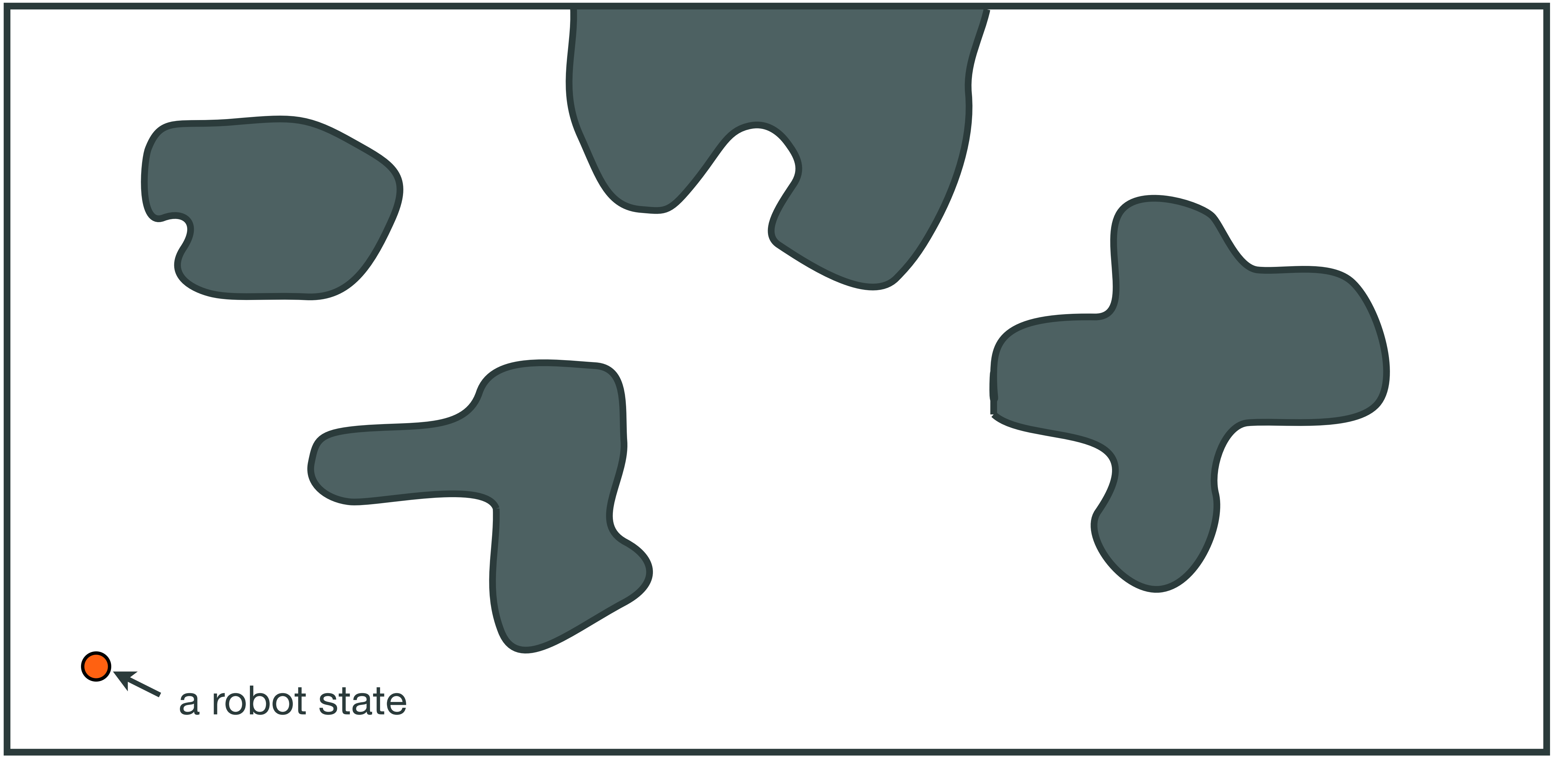
EST [Hsu et al. '97, '00]
RRT [Kuffner, LaValle '98]
RRT-Connect [Kuffner, LaValle '00]
SBL [Sanchez, Latombe '01]
RRF [Li, Shie '02]
Guided EST [Phillips et al. '03]
PDRRT [Ranganathan, Koenig '04]
SRT [Plaku et al. '05]
DDRRT [Yershova et al. '05]
ADDRRT [Jaillet et al. '05]
RRT-Blossom [Kalisiak, van Panne '06]
PDST [Ladd, Kavraki '06]

Trees (continued):

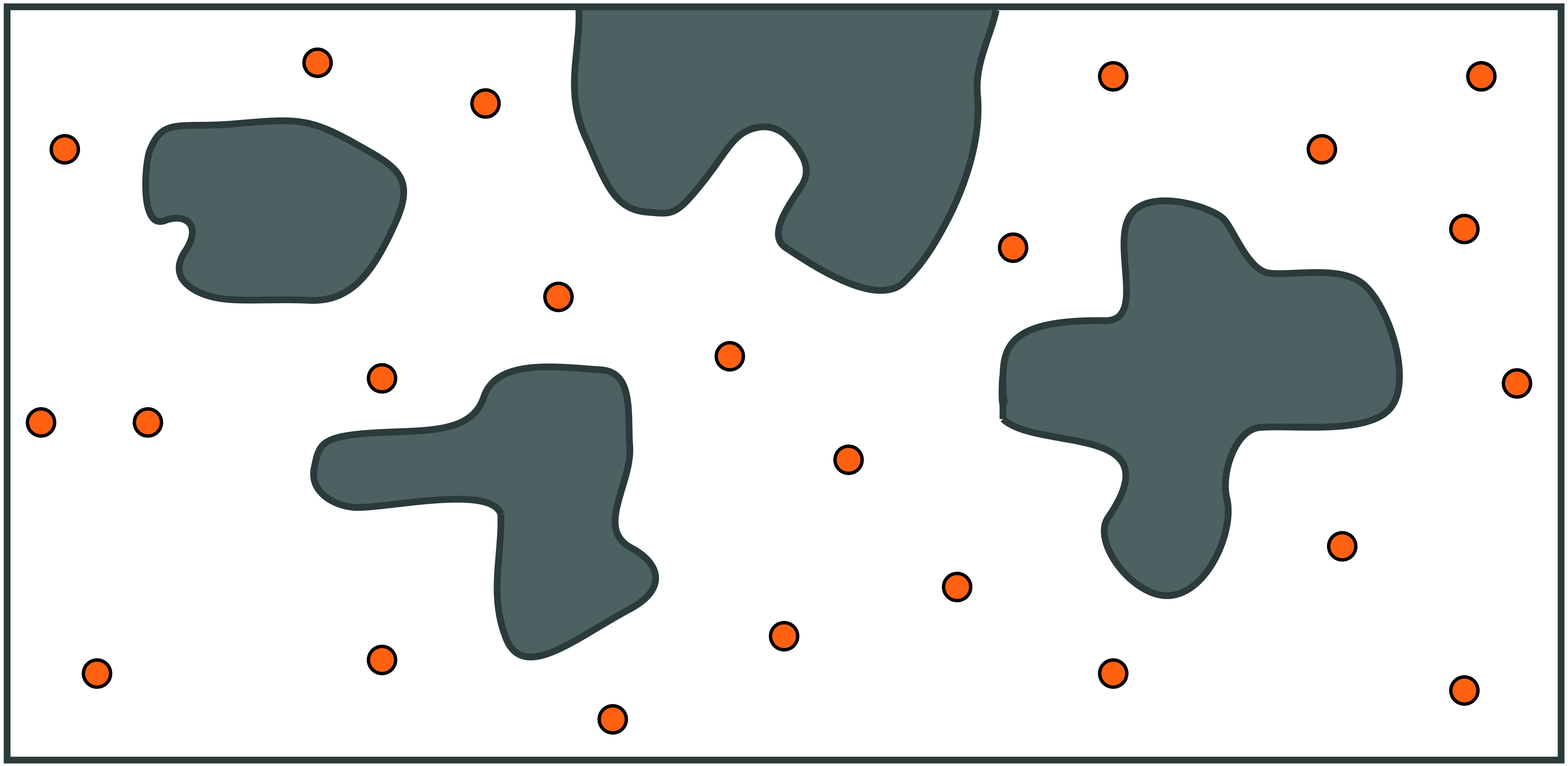
Utility RRT [Burns, Brock '07]
GRIP [Bekris, Kavraki '07]
Multiparticle RRT [Zucker et al. '07]
TC-RRT [Stillman et al. '07]
RRT-JT [Vande Wege et al. '07]
DSLX [Plaku, Kavraki, Vardi '08]
KPIECE [Şucan, Kavraki '08]
RPDST [Tsianos, Kavraki '08]
BiSpace [Diankov et al. '08]
GRRT [Chakravorty, Kumar '09]
IKBiRRT [Berenson et al. '09]
CBiRRT [Berenson et al. '09]
J+RRT [Vahrenkamp '09]
RG-RRT [Shkolnik et al. '09]
PCA-RRT [Li, Bekris '10]
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]
and many others

bold = included with OMPL

Point robot in 2-D

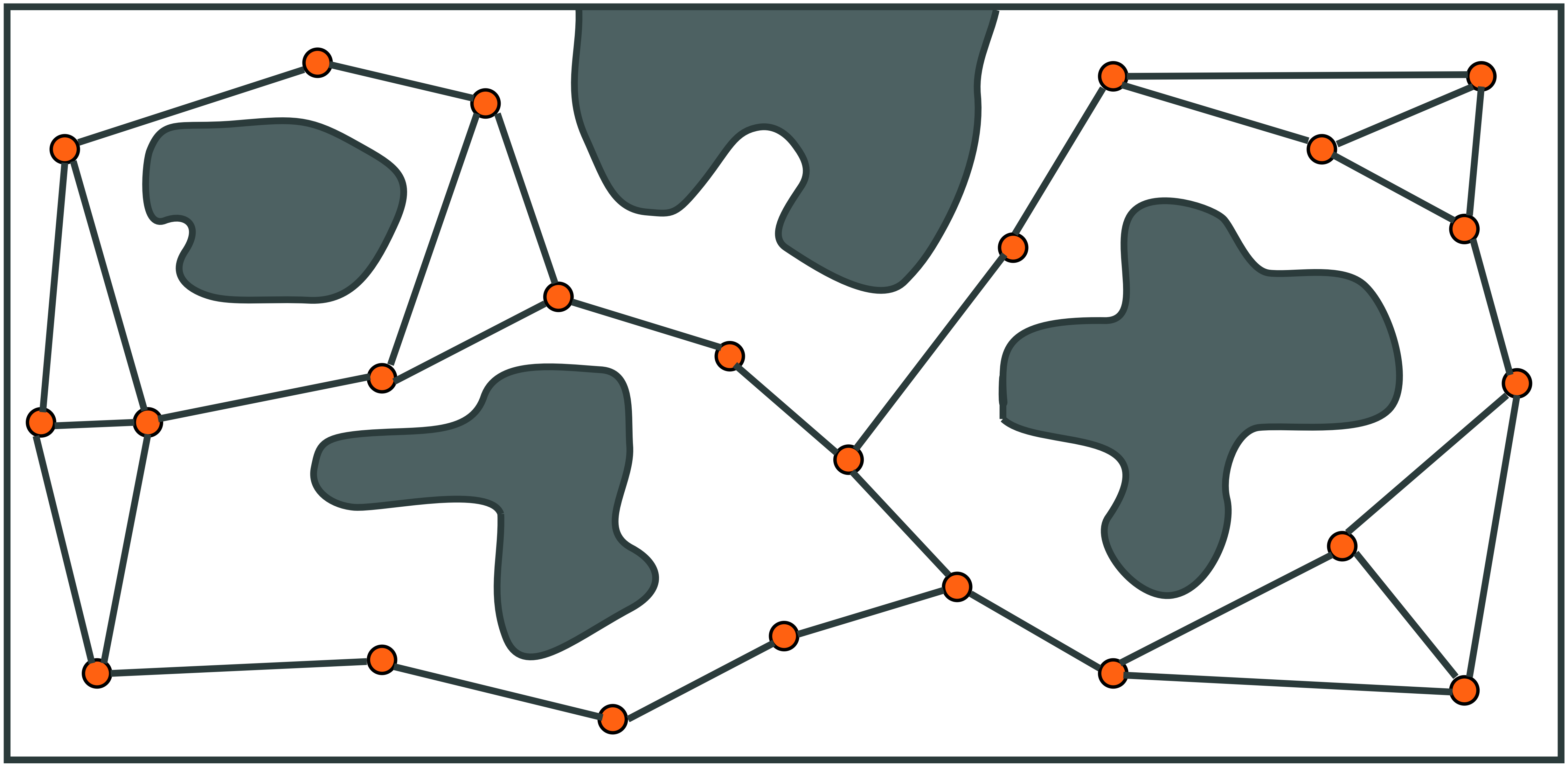


Operation of PRM



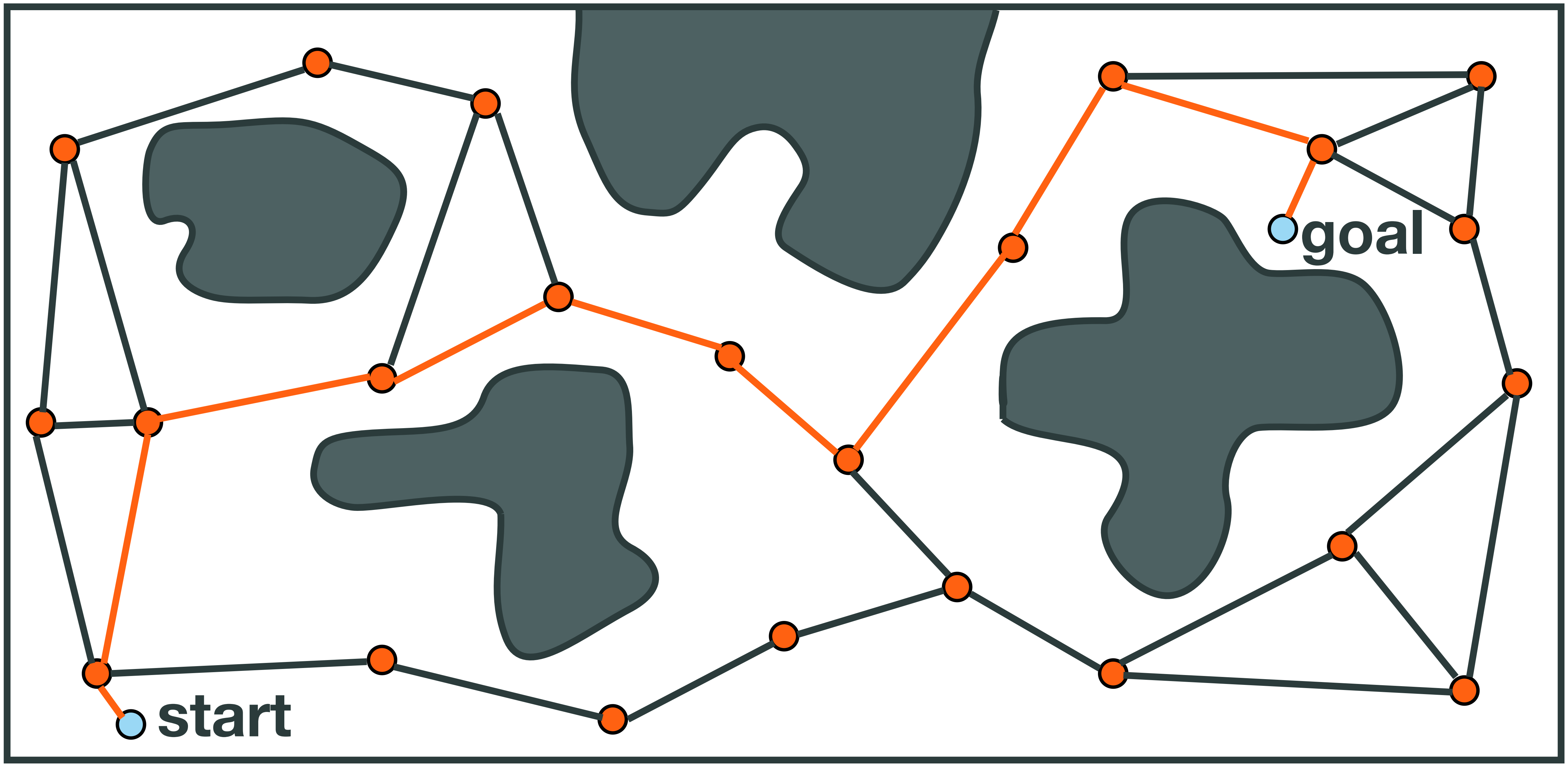
: **nodes,** random states

Operation of PRM



:edges, paths computed by local planner

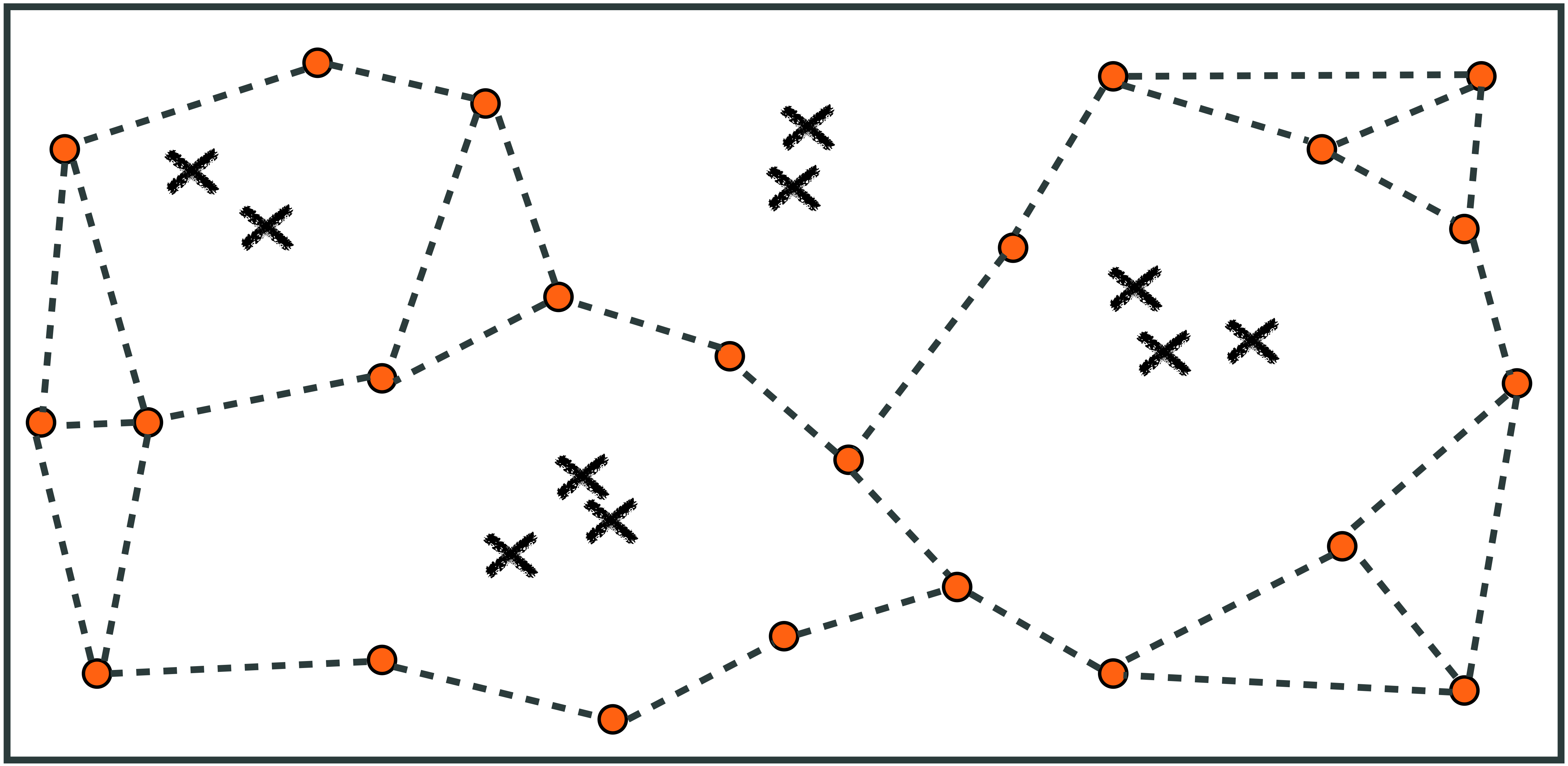
Answering queries



plan a path:

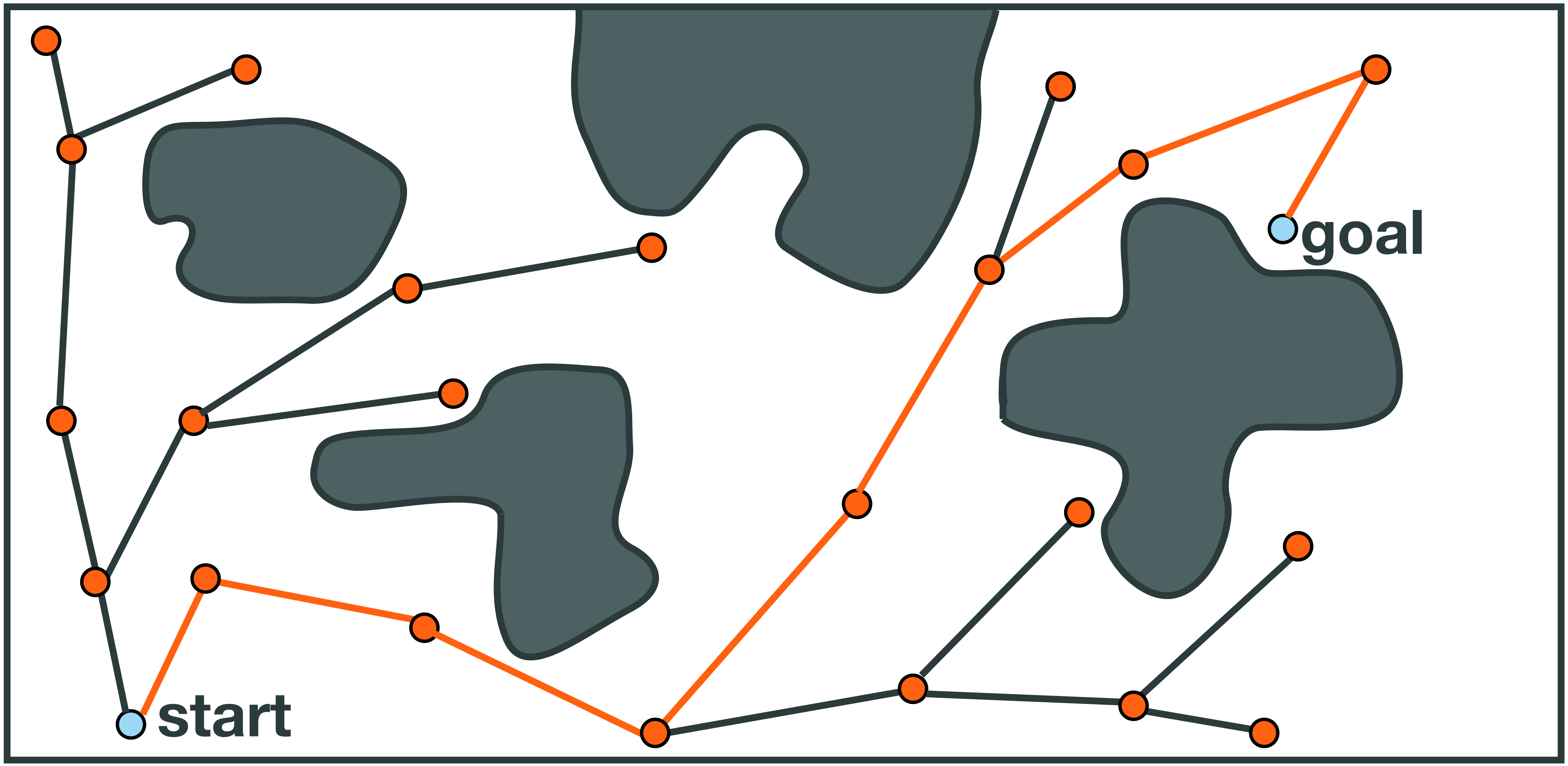
1. connect start & goal to roadmap
2. perform graph search

Operation of PRM



- - - - feasible path computed by local planner

Sampling-based tree planner operation



- Repeat until **goal** is connected to tree.
- Bi-directional trees are possible when considering only geometric constraints.

Main features of OMPL

OMPL in a nutshell

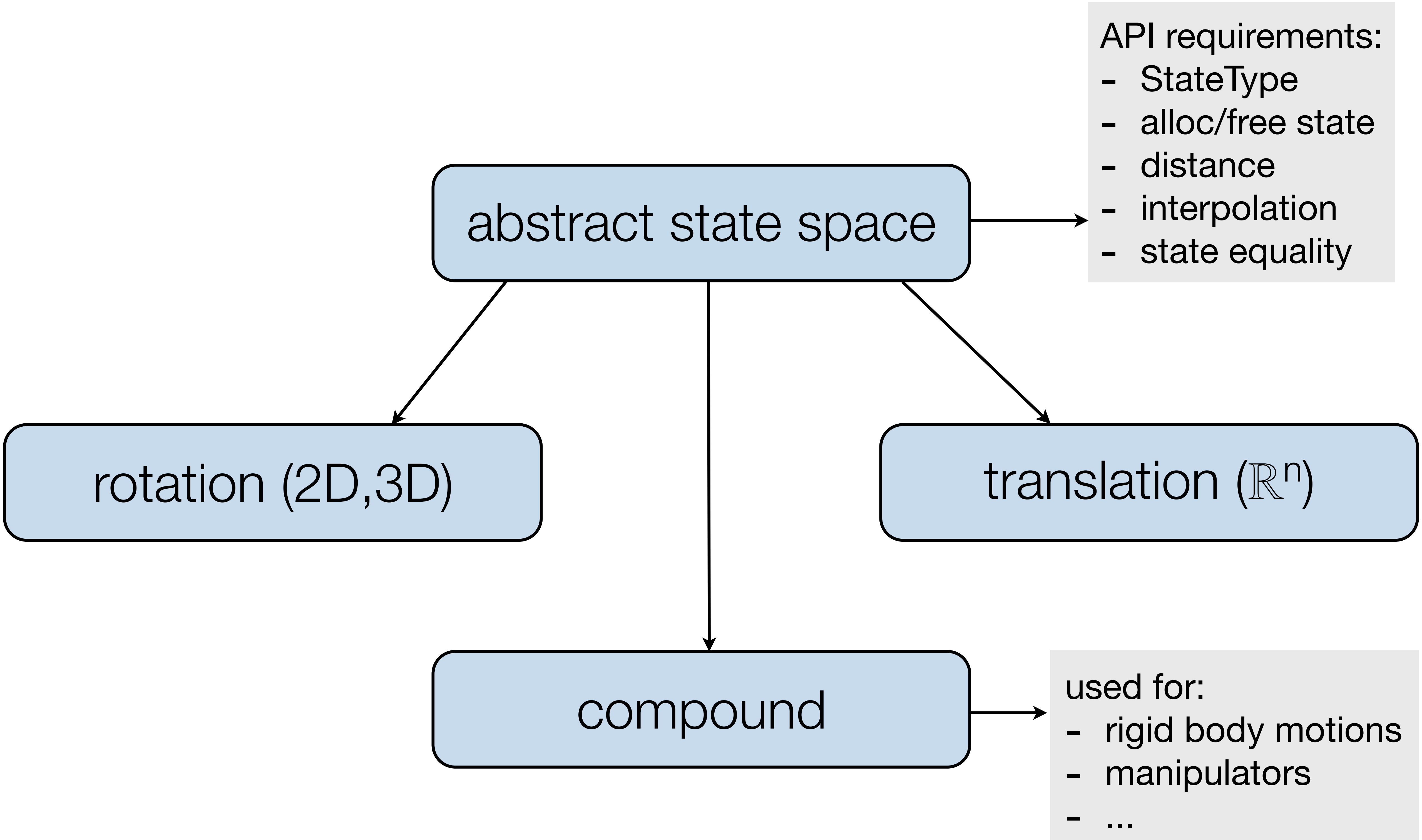
- Common core for sampling-based motion planners
- Includes commonly-used heuristics
- Takes care of many low-level details often skipped in corresponding papers
- Intended for use in:
  - Education
  - Research
  - Industry

Abstract interface to core sampling-based motion planning concepts

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

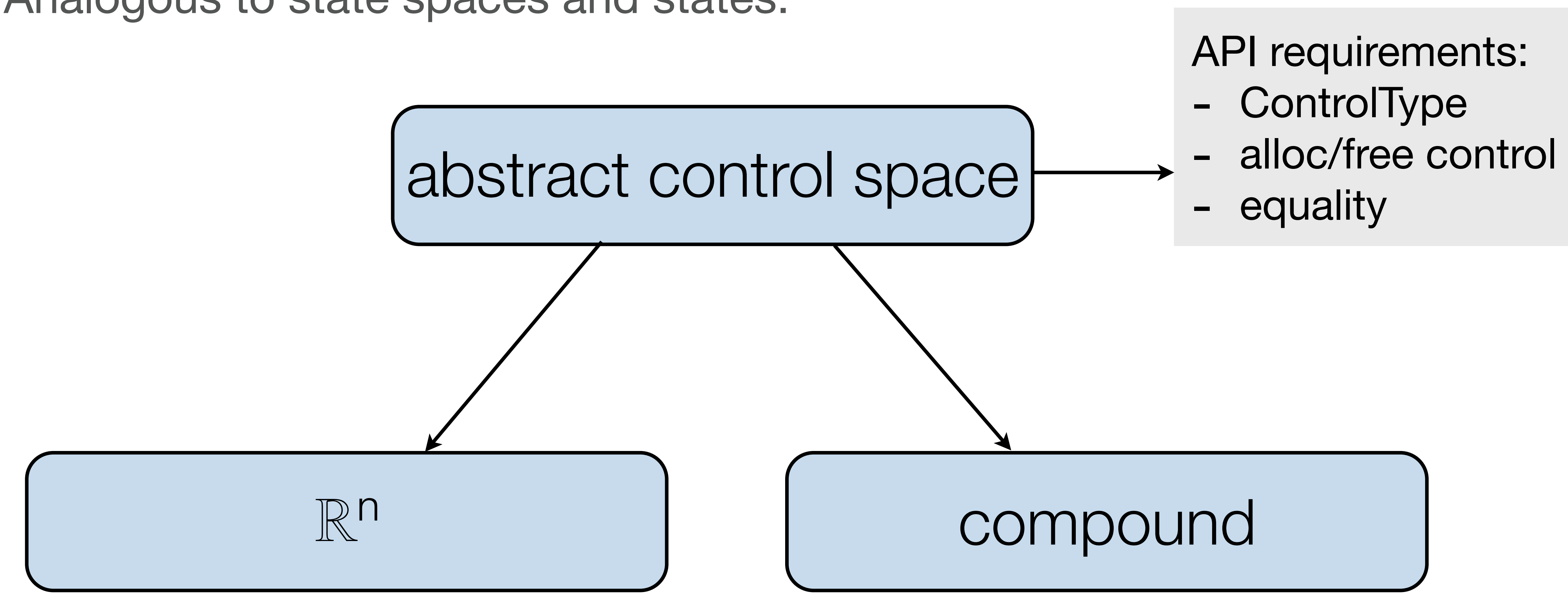
except robot & workspace...

States & state spaces



Control spaces & controls

- 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 → **MoveIt!**

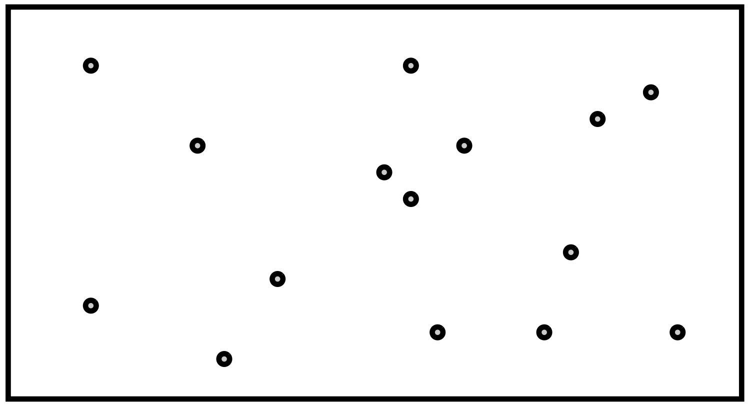
- Checks whether state is collision-free, joint angles and velocities are within bounds, etc.

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

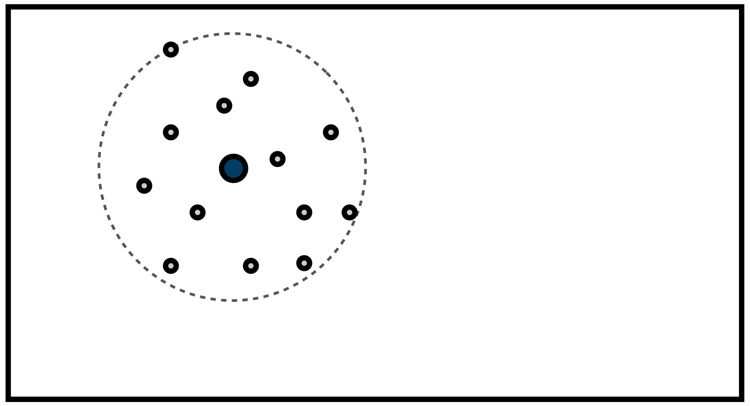
  Can be exploited by planners / samplers!

Samplers

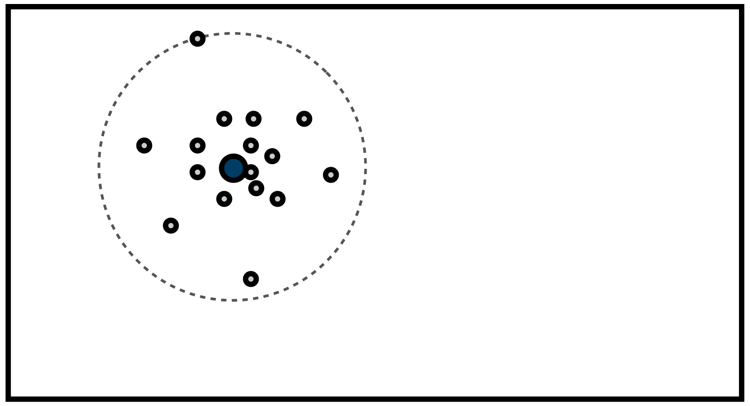
- For every **state space** there needs to be a **state sampler**
- State samplers need to support the following:
  - sample uniform



  - sample uniform near given state



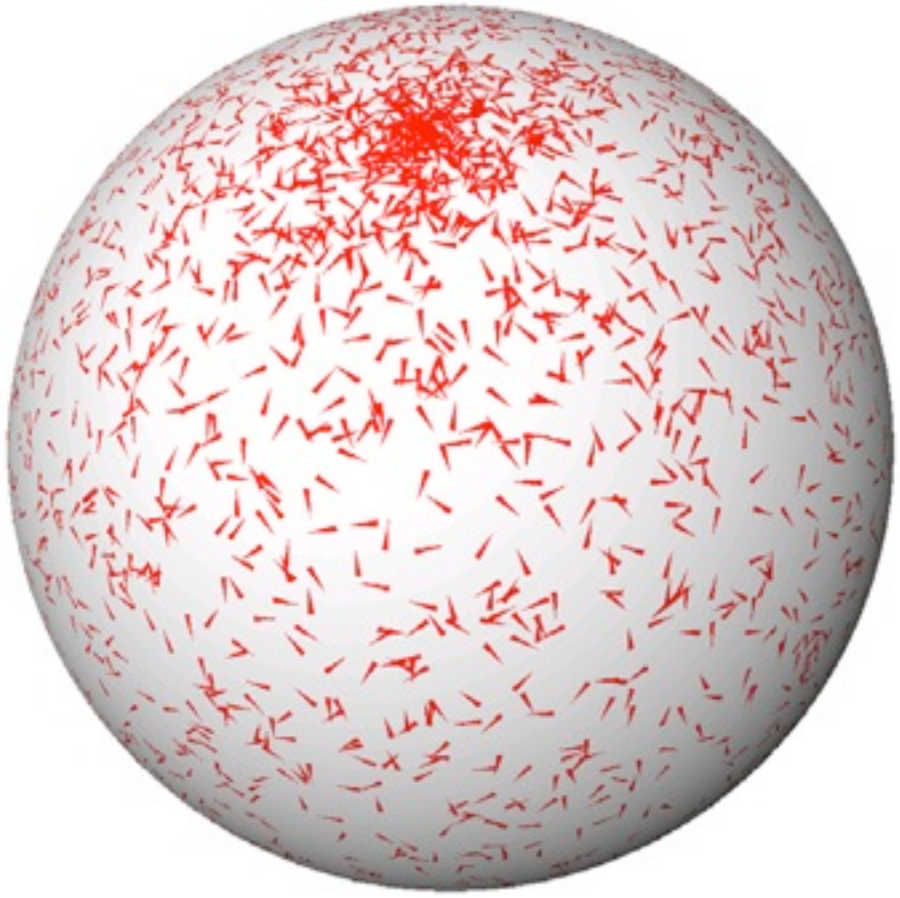
  - sample from Gaussian centered at given state



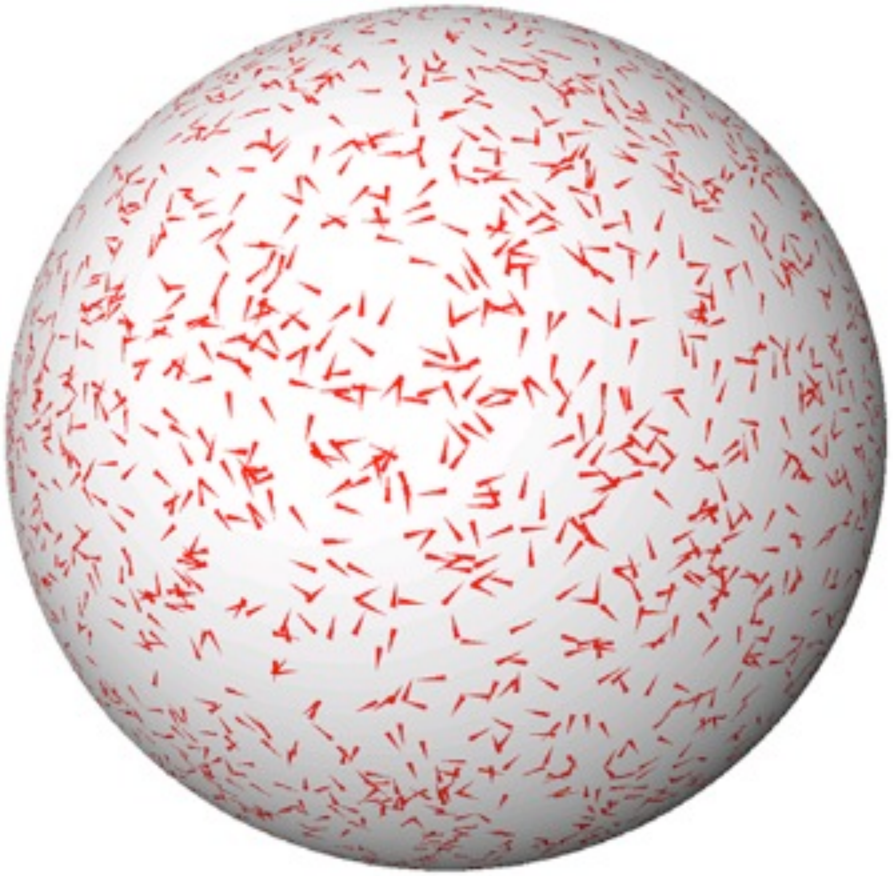
Many ways to get sampling wrong

Example: uniformly sampling 3D orientations

naïve & wrong:



correct:



Images from Kuffner, ICRA '04

Similar issues occur for nearest neighbors

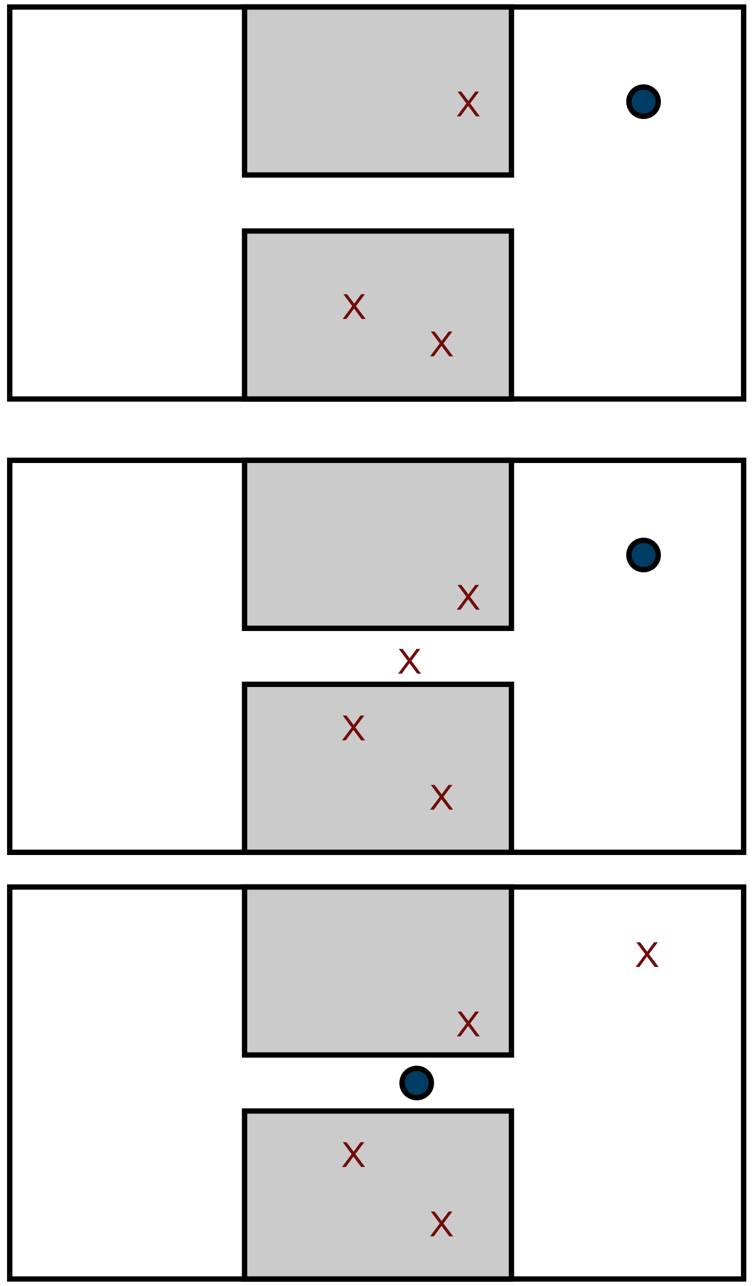
- *k* nearest neighbors can be computed efficiently with *k*d-trees in **low-dimensional, Euclidean** spaces.

- In high-dimensional spaces **approximate** nearest neighbors much better

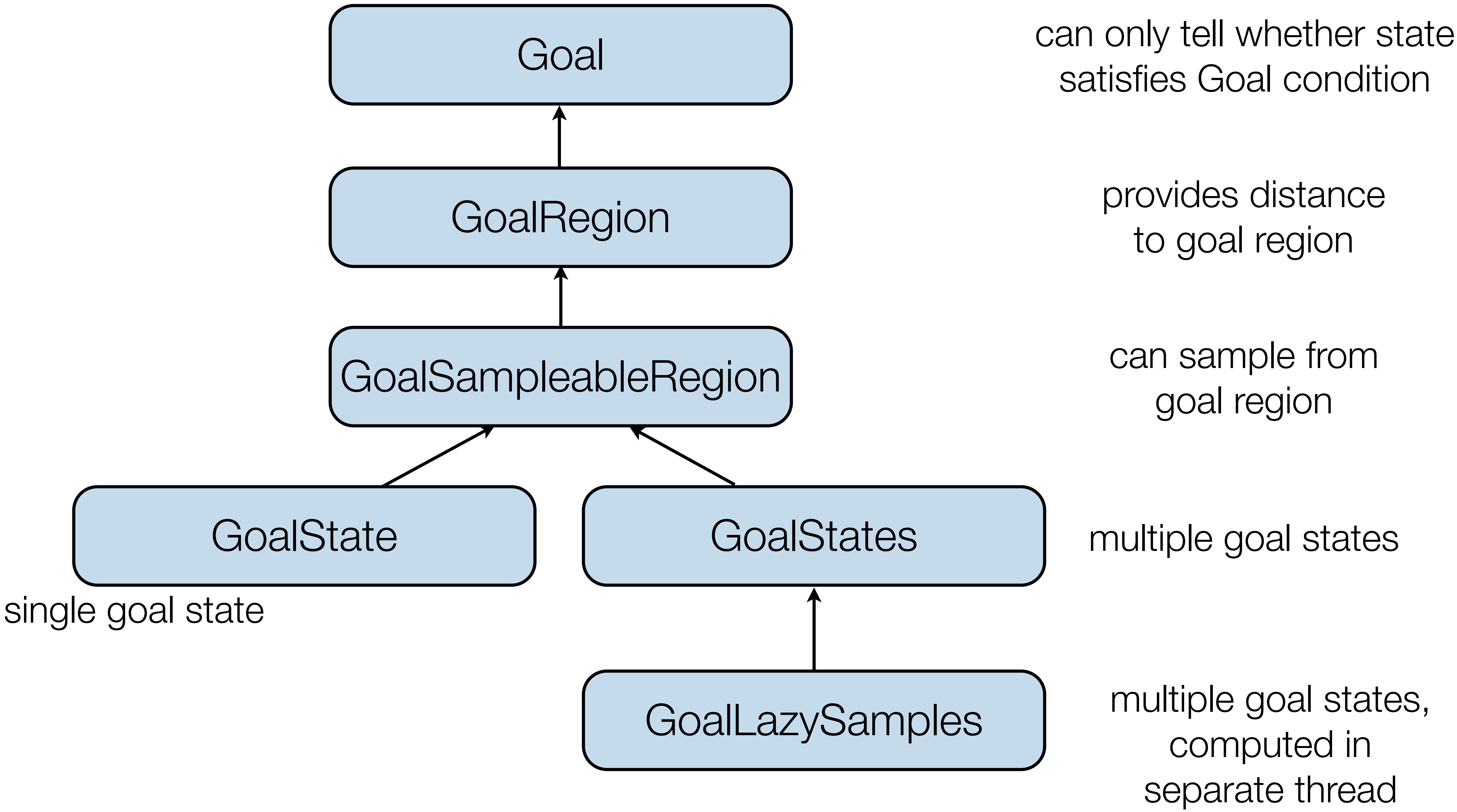
- In **non-Euclidean** spaces (e.g., any space that includes **rotations**), other data structures are necessary

***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
- Other sampling strategies:
  - 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

Many planners available in OMPL

Planner

geometric planning	*planning with controls*
KPIECE, BKPIECE, LBKPIECE	KPIECE
PRM, **LazyPRM**	RRT
RRT, RRTConnect, LazyRRT	EST
EST, SBL	Syclop
PDST	**PDST**
STRIDE	

Optimizing planners:

PRM*

RRT*, BallTreeRRT*

T-RRT

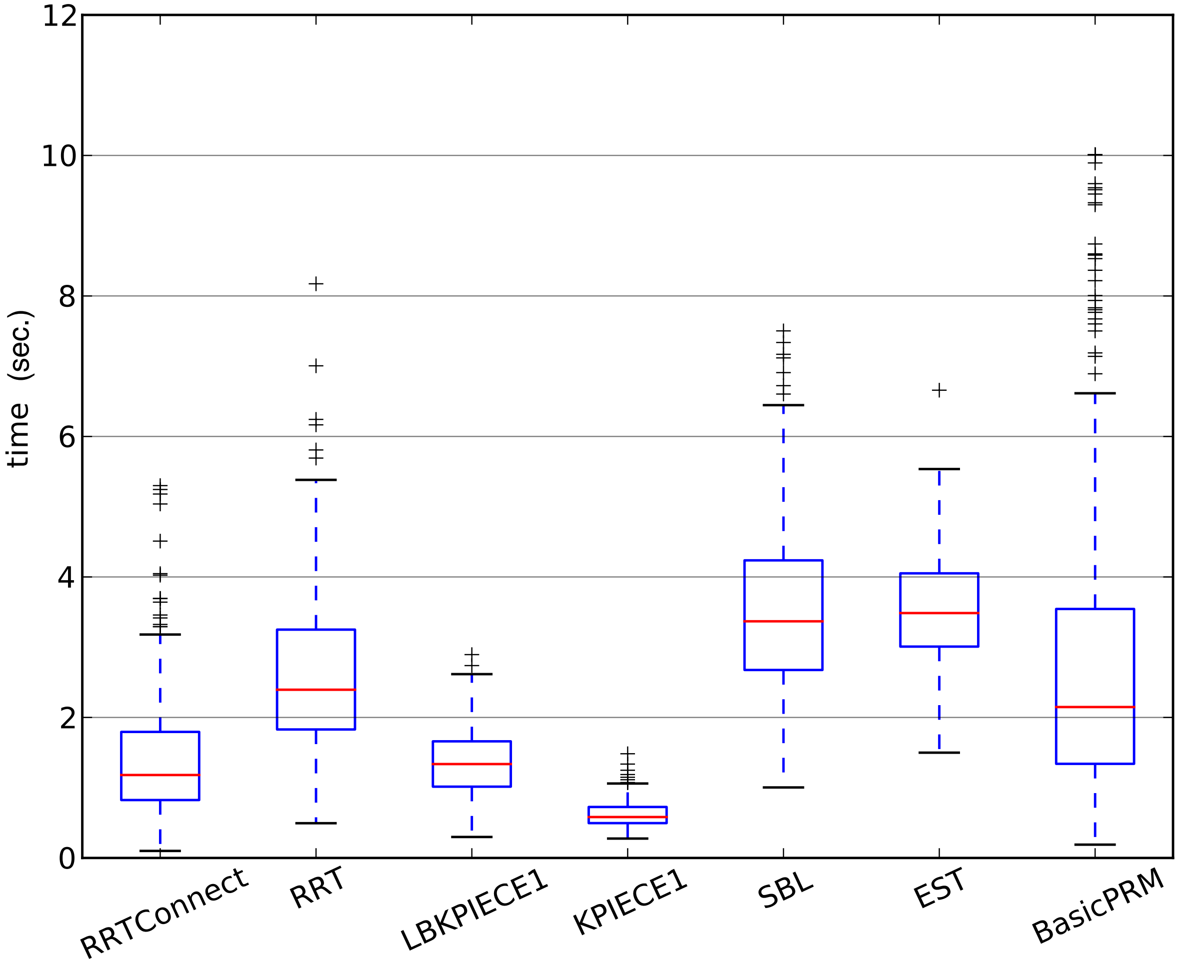
SPARS, SPARS-2

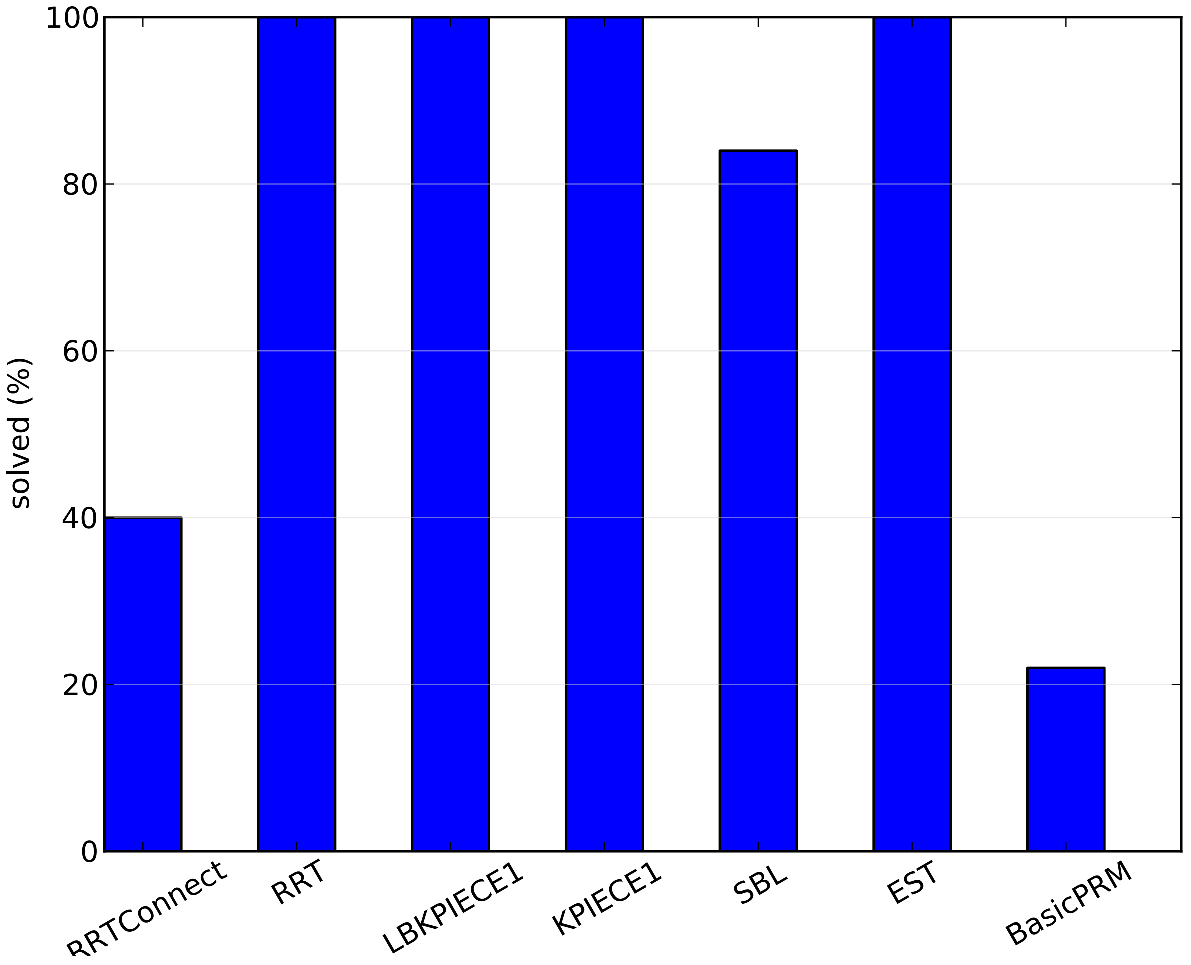
■ = *available soon!*

Minimal code example

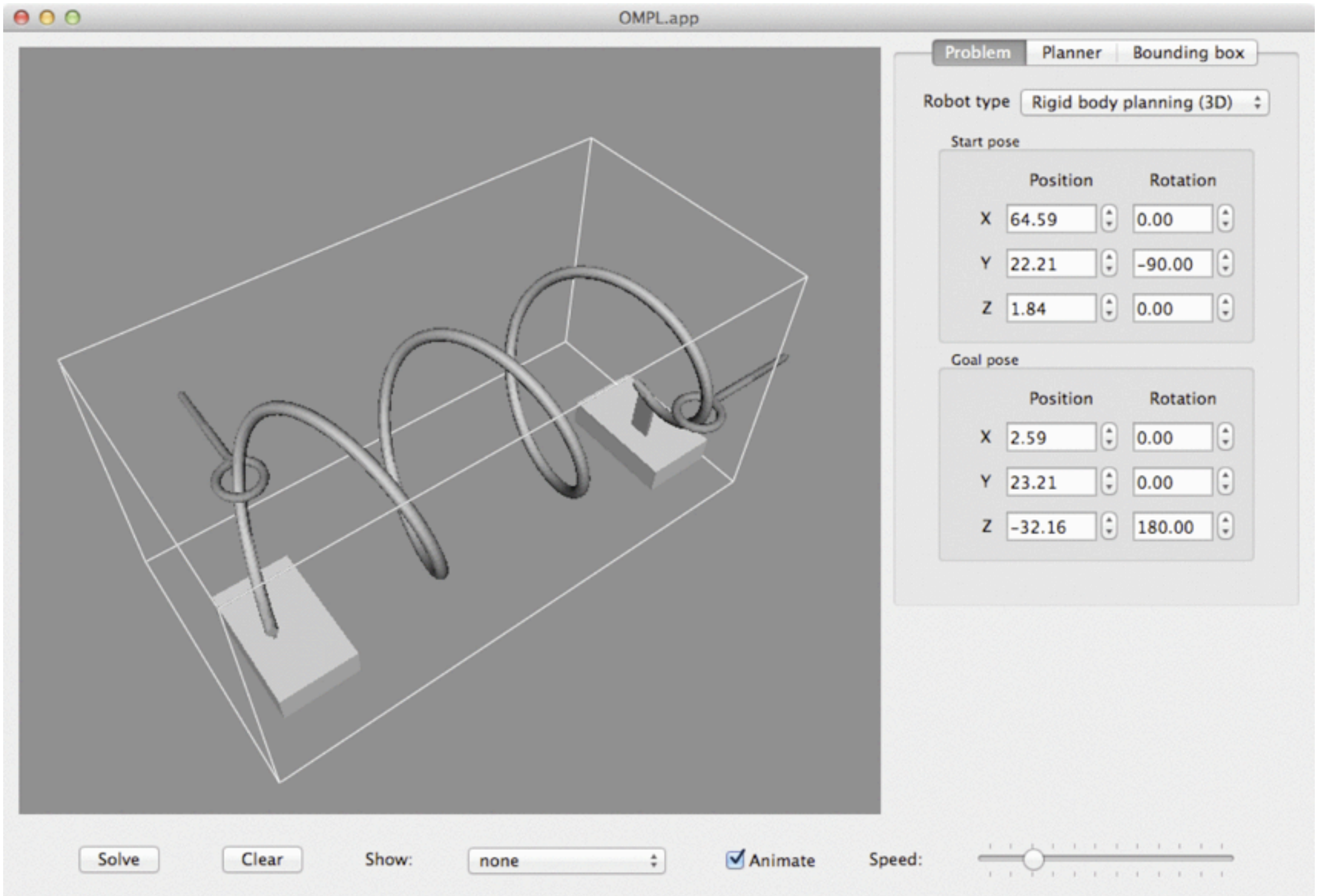
```
space = SE3StateSpace()
# set the bounds (code omitted)

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()
```

Benchmarking

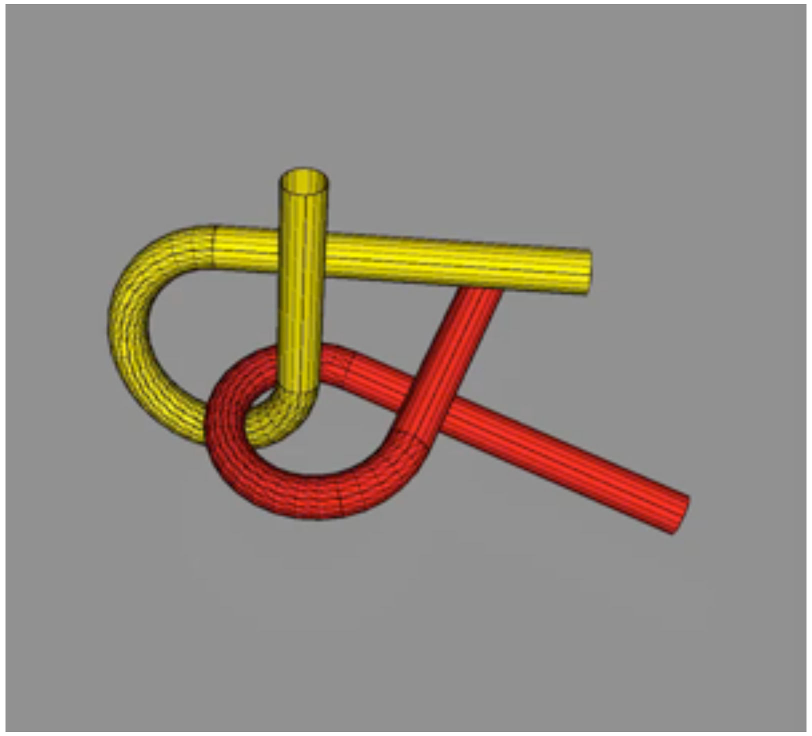


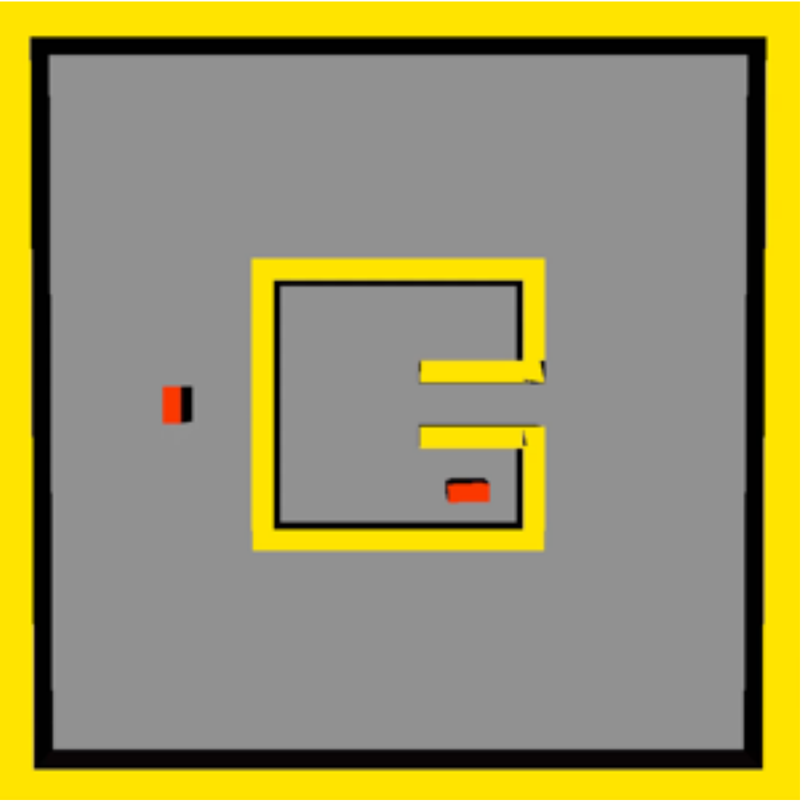


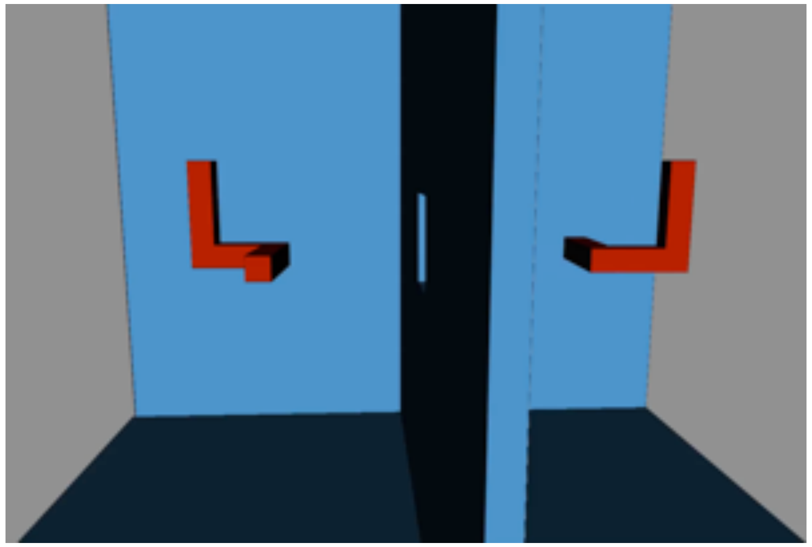
OMPL.app

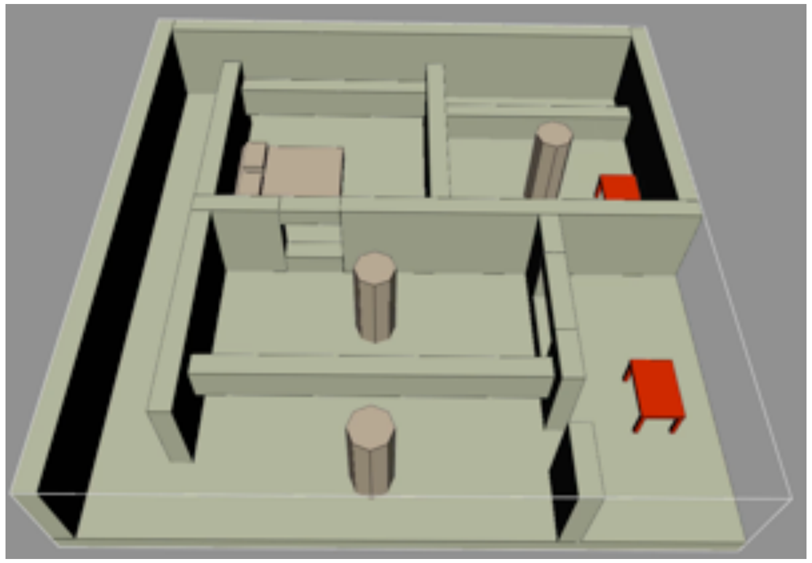

OMPL.app
Problem
Planner
Bounding box
Robot type
Rigid body planning (3D)
Start pose
Position
Rotation
X
64.59
0.00
Y
22.21
-90.00
Z
1.84
0.00
Goal pose
Position
Rotation
X
2.59
0.00
Y
23.21
0.00
Z
-32.16
180.00
Solve
Clear
Show:
none
Animate
Speed:

Sample OMPL.app problems









Resources to get started with OMPL

The Open Motion Planning Library

ompl.kavrakilab.org/index.html

OMPL | Overview | Download | Documentation | Code | Issues | Community | About | Blog | Search API

The Open Motion Planning Library

OMPL, the Open Motion Planning Library, consists of many state-of-the-art sampling-based motion planning algorithms. OMPL itself does not contain any code related to, e.g., collision checking or visualization. This is a deliberate design choice, so that OMPL is not tied to a particular collision checker or visualization front end.

OMPL.app, the front-end for OMPL, contains a lightweight wrapper for the FCL and PQP collision checkers and a simple GUI based on PyQt / PySide. The graphical front-end can be used for planning motions for rigid bodies and a few vehicle types (first-order and second-order cars, a blimp, and a quadrotor). It relies on the Assimp library to import a large variety of mesh formats that can be used to represent the robot and its environment.

Current version: 0.12.2
Released: Jan 22, 2013

Click for citation,
if you use OMPL in your work

Like 29 Send

Contents of This Library

- OMPL contains implementations of many sampling-based algorithms such as PRM, RRT, EST, SBL, KPIECE, SyCLOP, and several variants of these planners. See available planners for a complete list.
- All these planners operate on very abstractly defined state spaces. Many commonly used state spaces are already implemented (e.g., SE2, SE3, R^n, etc.).
- For any state space, different state samplers can be used (e.g., uniform, Gaussian, obstacle based, etc.).
- API overview
- Documentation for just the OMPL core library (i.e., without the "app" layer).

Getting Started

- The OMPL primer prov… sampling-based motio… OMPL.
- Download and install…
- Learn how to use the …
- Demos and tutorials
- Frequently Asked Qu…
- Familiarize yourself wi… throughout OMPL.
- Learn how to integrate… build system.
- If interested in using P… documentation for th…

Other Resources

News & Events

- OMPL has been accepted as a mentoring organization for the 2013 Google Summ…
- OMPL has won the 2012 Open Source Software World Grand Challenge!
- An article about OMPL has been accepted for publication in IEEE's Robotics & Automa…
- At ROSCON, Sachin Chitta and Ioan Şucan gave a talk about MoveIt!, the new moti… (including OMPL). It will eventually replace the arm navigation stack.
- IROS 2011 Tutorial on Motion Planning for Real Robots. This hands-on tutorial des… motion planning.

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

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

Public repositories at:
https://bitbucket.org/ompl

Physical and Biological Computing Group • Department of Computer Science • Rice University
Generated on Tue Jan 22 2013 11:06:06 by doxygen 1.8.3

OMPL for education

- Programming assignments centered around OMPL, available upon request.
- Ongoing educational assessment.
- Already in use in several robotics / motion planning classes.

Happy OMPL users: students in the Algorithmic Robotics class at Rice, Fall 2010

Discussion

- OMPL actively developed, but ready for general use
- Can easily implement new algorithms from many reusable components
- Simple high-level interface:
  - Can treat motion planner almost as a black box
  - Easy enough that non-experts can use it
- Interface generic enough to be extensible in many ways

We want your contributions!

Acknowledgements

Rice University:

Lydia Kavraki

Ryan Luna

Matt Maly

Bryant Gipson

Devin Grady

Amit Bhatia

Willow Garage:

Ioan Şucan

Sachin Chitta

Funding from:

NSF CCLI grant #0920721

NSF IIS grant #0713623

Willow Garage