Search-based Planning Library SBPL

Maxim Likhachev

Robotics Institute

Carnegie Mellon University

Carnegie Mellon
THE ROBOTICS INSTITUTE

sbpl

Outline

- Overview
- Few SBPL-based planners in details
  - 3D (x,y,θ) lattice-based planning for navigation
    (available as ROS node or standalone within SBPL)
  - single and dual 7DOF arm motion planning using manipulation lattice
    (available as ROS node)
- Pros/Cons

Outline

- Overview
- Few SBPL-based planners in details
  - 3D (x,y,θ) lattice-based planning for navigation (available as ROS node or standalone within SBPL)
  - single and dual 7DOF arm motion planning using manipulation lattice (available as ROS node)
- Pros/Cons

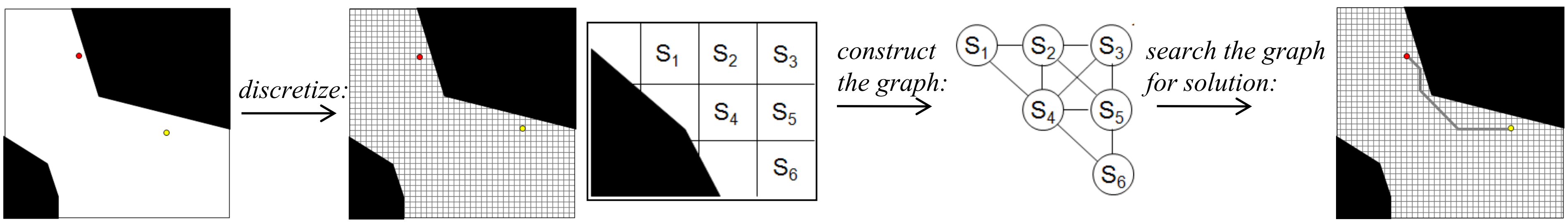
Search-based Planning Library (SBPL)

- A library for planning with heuristic search (e.g., A* search and its variants)
- Standalone library and integrated into ROS
- Compiles under linux and windows
- http://www.sbpl.net/software or http://www.ros.org/wiki/sbpl

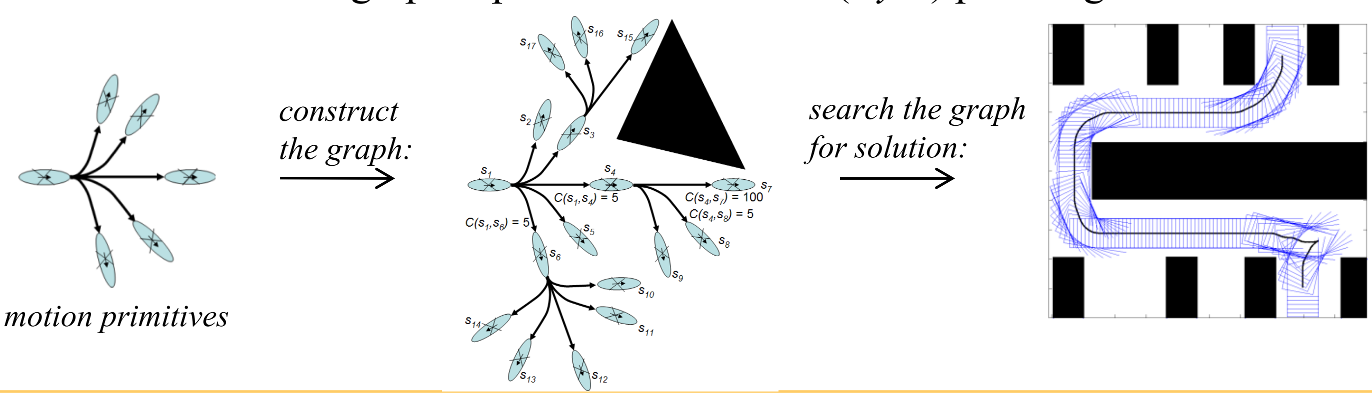
Planning with Heuristic Search

- generate a systematic graph representation of the planning problem
- search the graph for a solution with a heuristic search
- **typically the construction of the graph is interleaved with the search** (i.e., only create the states/edges that search explores)

2D grid-based graph representation for 2D (*x,y*) search-based planning:



lattice-based graph representation for 3D (*x,y,θ*) planning:



- Typical components of a Search-based Planner

  - Graph construction (given a state what are its successor states)
  - Cost function (a cost associated with every transition in the graph)
  - Heuristic function (estimates of cost-to-goal)
  - Graph search algorithm (for example, A* search)

Planning with Heuristic Search

- Typical components of a Search-based Planner
  - Graph construction (given a state what are its successor states)
  - Cost function (a cost associated with every transition in the graph)
  - Heuristic function (estimates of cost-to-goal)
  - Graph search algorithm (for example, A* search)

domain dependent

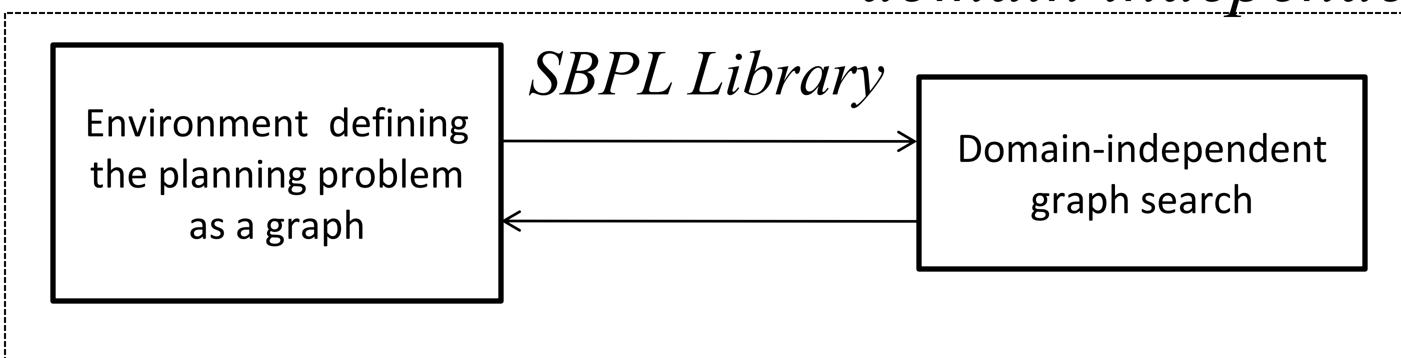
domain independent

Planning with Heuristic Search

- Typical components of a Search-based Planner
  - Graph construction (given a state what are its successor states)
  - Cost function (a cost associated with every transition in the graph)
  - Heuristic function (estimates of cost-to-goal)
  - Graph search algorithm (for example, A* search)

domain dependent

domain independent



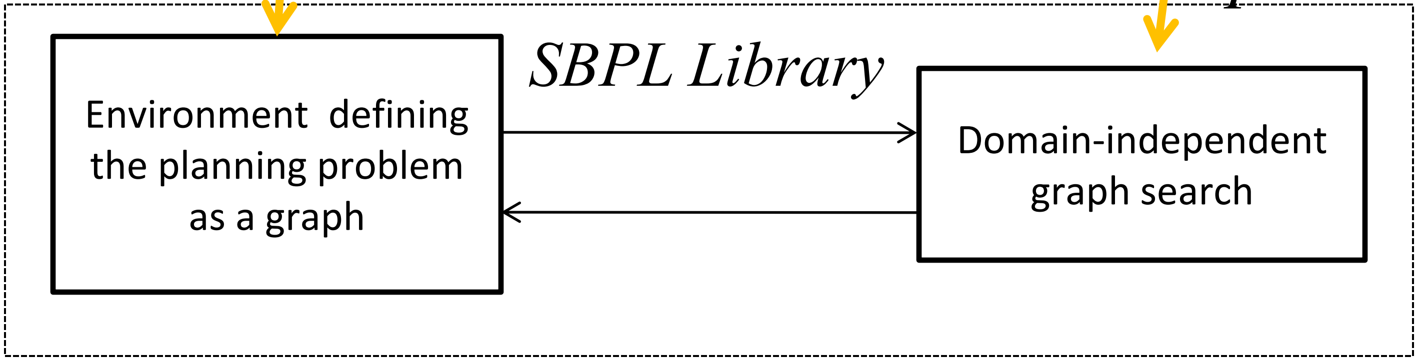
Implements:
Successors/Predecessors of a state;
Transition cost; State heuristic

Implements:
Graph search
(e.g, A, D*, ARA*, etc.)*

- Graph construction (given a state what are its successor states)
- Cost function (a cost associated with every transition in the graph)
- Heuristic function (estimates of cost-to-goal)
- Graph search algorithm (for example, A* search)

...main dependent

domain independent



Implements:
Successors/Predecessors of a state;
Transition cost; State heuristic

Implements:
Graph search
(e.g, A, D*, ARA*, etc.)*

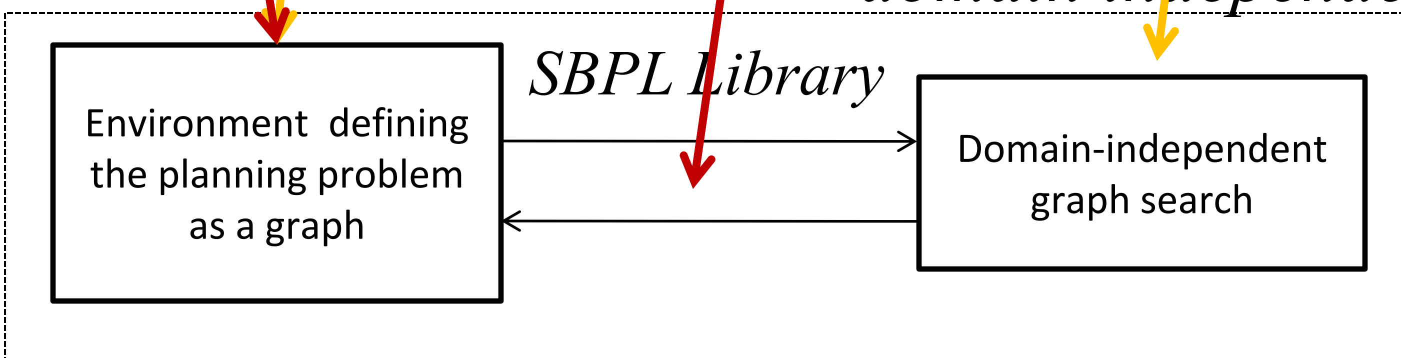
- Gra… a state what are its successor states)
- … every transition in the graph)
- … of cost-to-goal)
- Graph search alg…

…main dependent

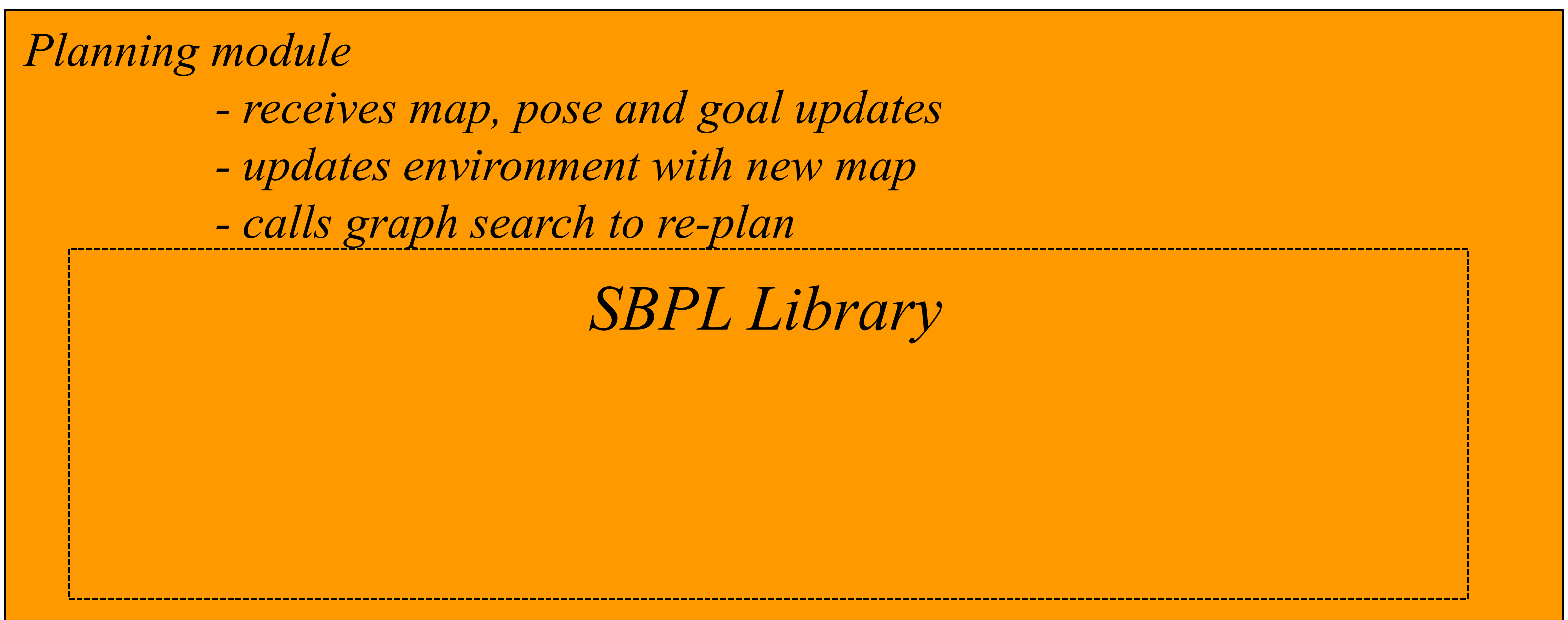
Memory allocated on-the-fly only for states visited by search

All communications happen via state IDs (no domain information)

domain independent



- Usage of SBPL:
  - build a planner using existing components to run on a robot
  - plugin and test your own graph search
  - develop and plugin an environment for your specific planning problem "representable" as a graph search problem



Search-based Planning Library (SBPL)

- Currently implemented graph searches within SBPL:
  - ARA* - anytime version of A*
  - ANA* - anytime non-parametric version of A*
  - Anytime D* - anytime incremental version of A*
  - R* - a randomized version of A* (hybrid between deterministic searches and sampling-based planning)

- Currently implemented environments (planning problems) within SBPL:
  - 2D (x,y) grid-based planning problem
  - 3D (x,y,θ) lattice-based planning problem
  - 3D (x,y,θ) lattice-based planning problem with full-body collision checking
  - N-DOF planar robot arm planning problem

- ROS packages that use SBPL:
  - SBPL lattice planner for (x,y,θ) planning for navigation
  - SBPL lattice planner for (x,y,θ) planning for navigation with full-body collision checking
  - SBPL cart planner for PR2 navigating with a cart
  - SBPL motion planner for PR2 single- and dual-arm motions
  - default move_base invokes SBPL lattice planner as part of escape behavior
  - SBPL door planning module for PR2 opening and moving through doors
  - SBPL footstep planner for humanoids (by Armin Hornung at Univ. of Freiburg)

Search-based Planning Library (SBPL)

- Main.cpp shows simple examples for how to use SBPL:

```
EnvironmentNAVXYTHETALAT environment_navxythetalat;
if(!environment_navxythetalat.InitializeEnv(argv[1], perimeterptsV, NULL))
{
        SBPL_ERROR("ERROR: InitializeEnv failed\n");
        throw new SBPL_Exception();
}
if(!environment_navxythetalat.InitializeMDPCfg(&MDPCfg))
{
        SBPL_ERROR("ERROR: InitializeMDPCfg failed\n");
        throw new SBPL_Exception();
}
//plan a path
vector<int> solution_stateIDs_V;
bool bforwardsearch = false;
ADPlanner planner(&environment_navxythetalat, bforwardsearch);
if(planner.set_start(MDPCfg.startstateid) == 0)
{
        SBPL_ERROR("ERROR: failed to set start state\n");
        throw new SBPL_Exception();
}
if(planner.set_goal(MDPCfg.goalstateid) == 0)
{
        SBPL_ERROR("ERROR: failed to set goal state\n");
        throw new SBPL_Exception();
}
planner.set_initialsolution_eps(3.0);

bRet = planner.replan(allocated_time_secs, &solution_stateIDs_V);
SBPL_PRINTF("size of solution=%d\n",(unsigned int)solution_stateIDs_V.size());
```

Outline

- Overview
- Few SBPL-based planners in details
  - 3D (x,y,θ) lattice-based planning for navigation
    (available as ROS node or standalone within SBPL)
  - single and dual 7DOF arm motion planning using manipulation lattice
    (available as ROS node)
- Pros/Cons

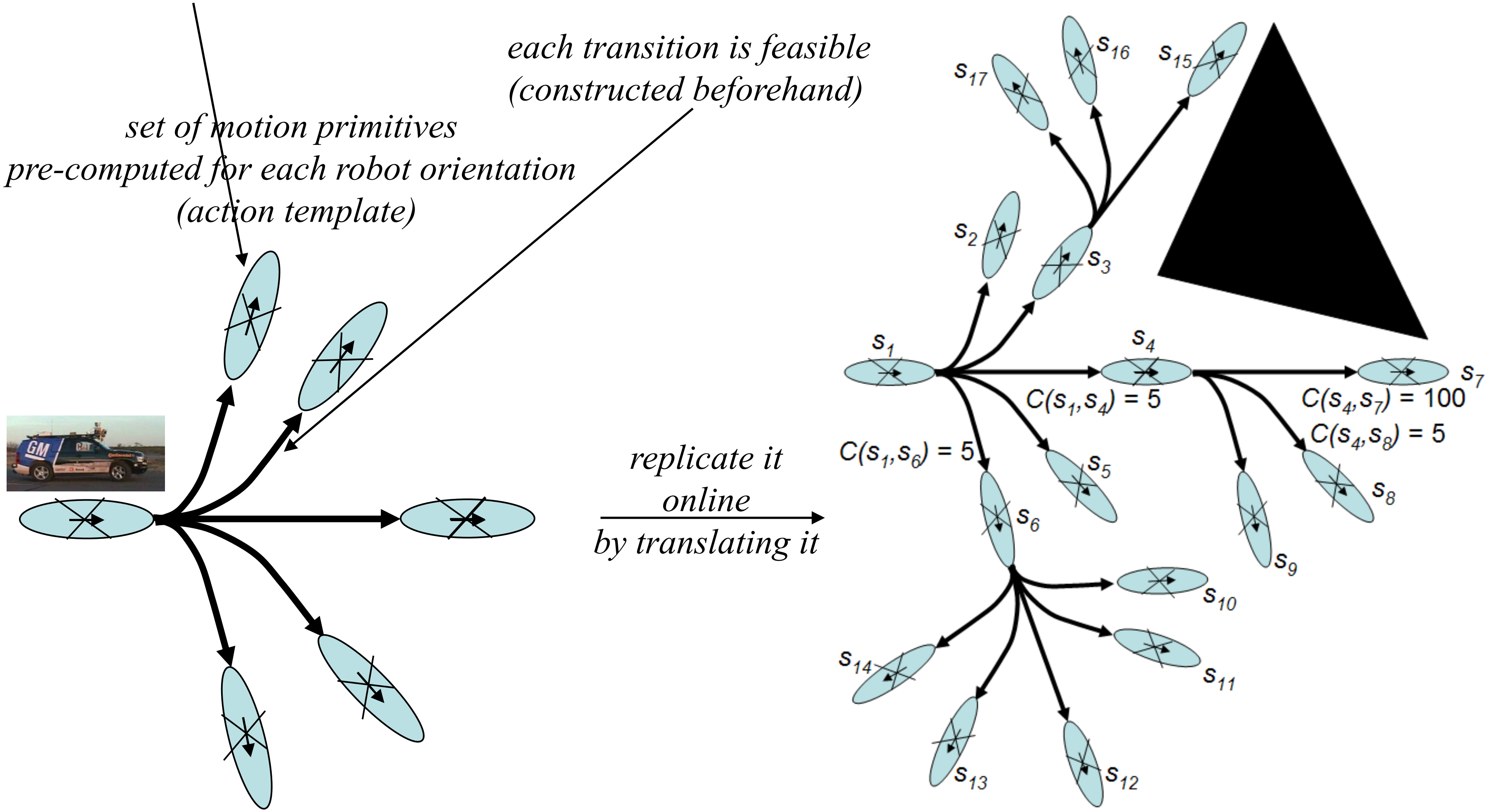
3D (x,y,θ) Planning for Navigation

sbpl_lattice_planner *in ROS*

3D (x,y,θ) Planning for Navigation

- Environment:
  - graph constructed using motion primitives [Pivtoraiko & Kelly, IROS'05]

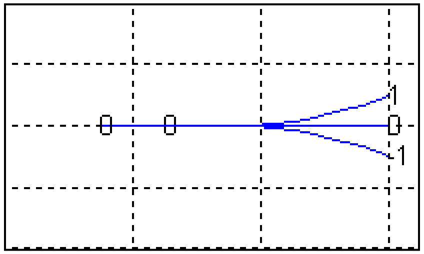
outcome state is the center of the corresponding cell in the underlying (x,y,θ,…) cell

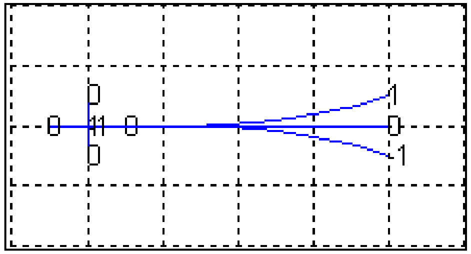


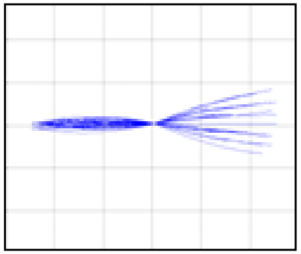
3D (x,y,θ) Planning for Navigation

- Environment:
  - graph constructed using motion primitives [Pivtoraiko & Kelly, IROS‘05]
  - takes set of motion primitives as input (.mprim files generated within matlab/mprim directory using corresponding matlab scripts):

unicycle model or *unicycle with sideways motions* or …

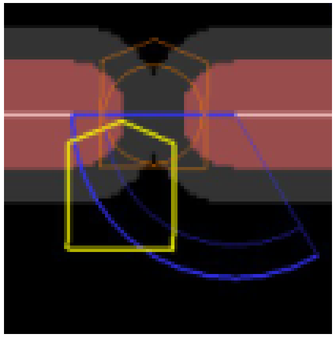






3D (x,y,θ) Planning for Navigation

- Environment:
  - graph constructed using motion primitives [Pivtoraiko & Kelly, ‘05]
  - takes set of motion primitives as input (.mprim files generated within matlab/mprim directory using corresponding matlab scripts)
  - takes the footprint of the robot defined as a polygon as input



3D (x, y, θ) Planning for Navigation

- Graph search:
  - typically ARA* (anytime version of A*) or Anytime D* (anytime incremental version of A*)

sbpl_lattice_planner *in ROS*

3D (x, y, θ) Planning for Navigation

- Planning with full-body collision checking (**3d_navigation** node in ROS)

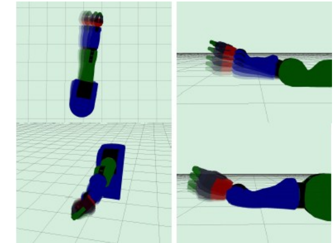
Hornung et al., ICRA '12

Outline

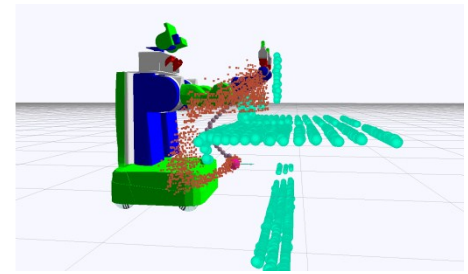
- Overview
- Few SBPL-based planners in details
  - 3D (x,y,θ) lattice-based planning for navigation
    (available as ROS node or standalone within SBPL)
  - single and dual 7DOF arm motion planning using manipulation lattice
    (available as ROS node)
- Pros/Cons

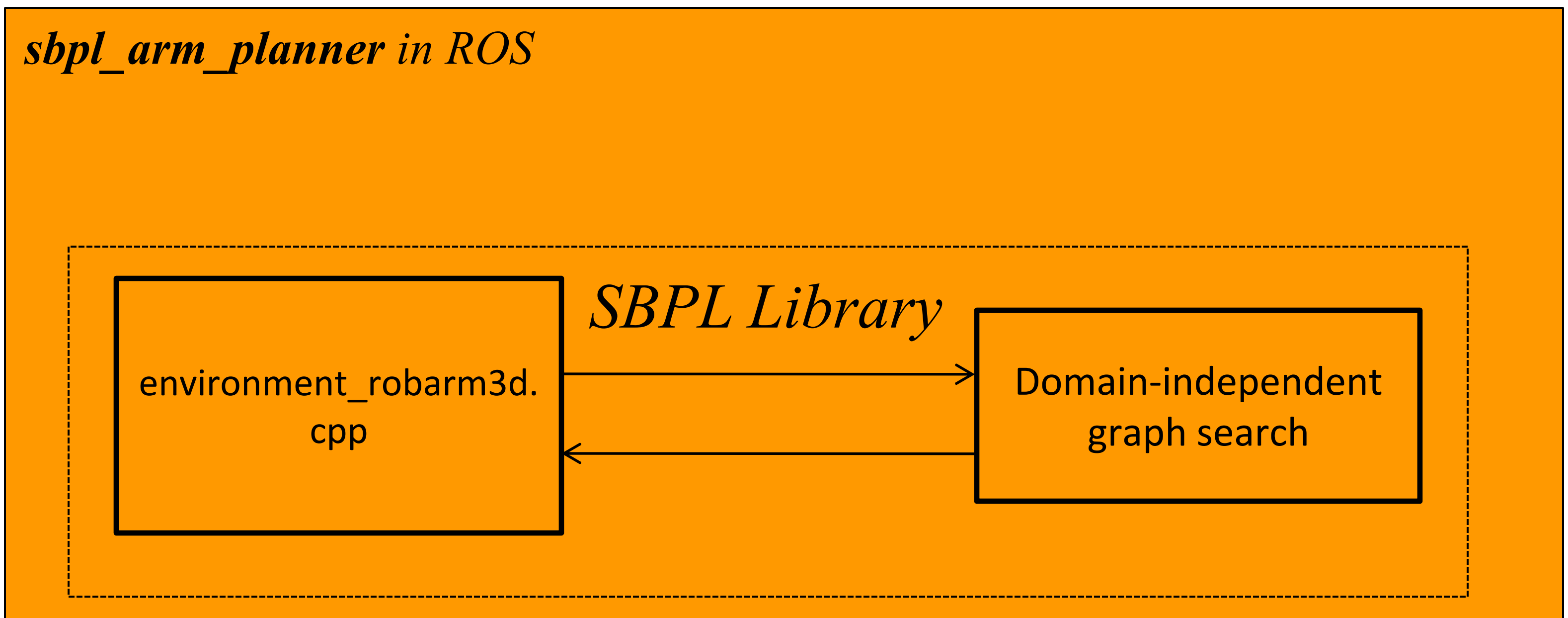
Single- and Dual-arm Motion Planning

- Environment:
  - graph constructed using static & adaptive motion primitives for the arm(s) [Cohen et al., ICRA‘11]

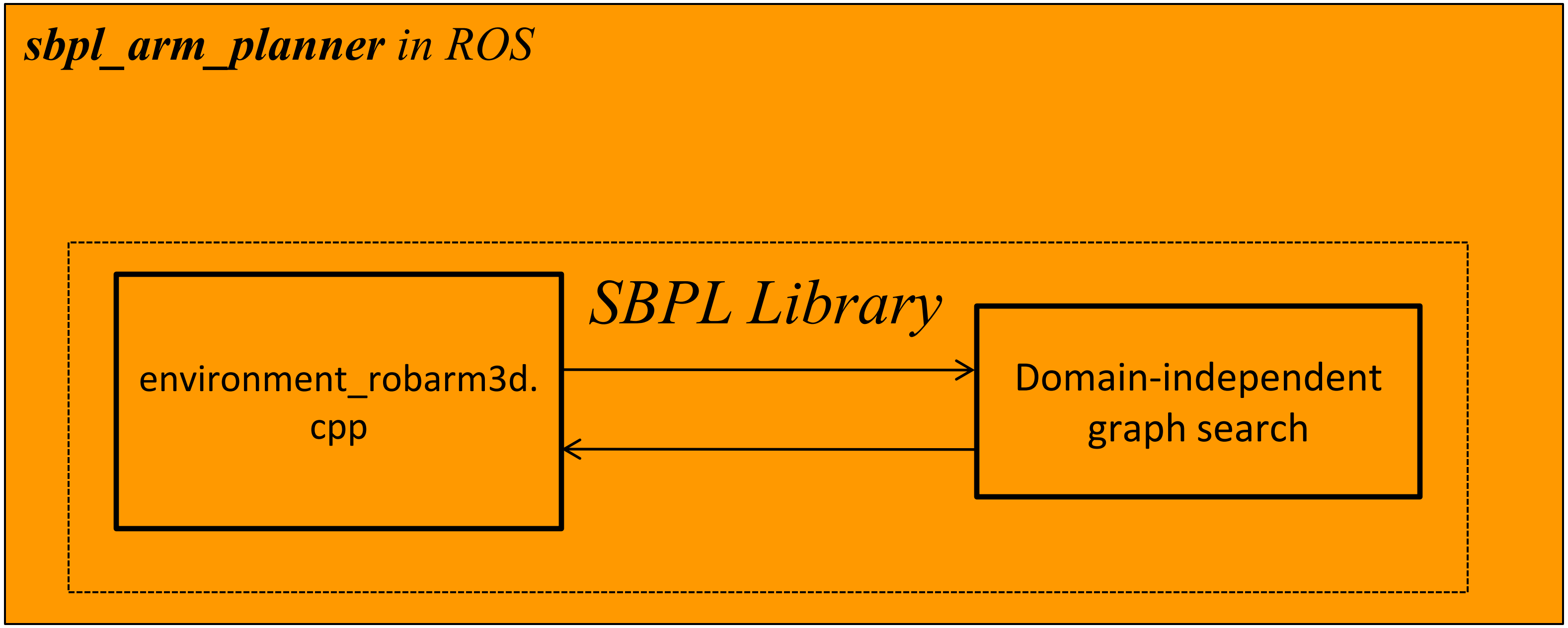


  - heuristic for any state is 3D distance for end-effector accounting for obstacles (computed as 3D BFS) [Cohen et al., ICRA‘11]



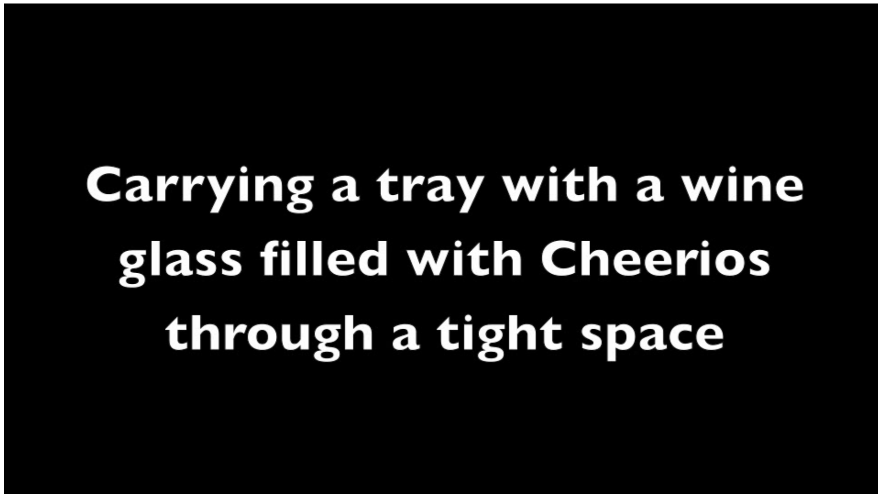


- Graph search:
  - typically ARA* (anytime version of A* search)



Single- and Dual-arm Motion Planning

- Planning for PR2 and KUKA arms



[Cohen et al., ICRA'12]

work led by Cohen & Cowley; joint work with CJ Taylor

joint work Stentz, Herman, Galati, Kelly, Meyhofer, etc. at NREC/CMU

Outline

- Overview
- Few SBPL-based planners in details
  - 3D (x,y,θ) lattice-based planning for navigation (available as ROS node or standalone within SBPL)
  - single and dual 7DOF arm motion planning using manipulation lattice (available as ROS node)
- Pros/Cons

Motion Planning with Heuristic Search

- Pros
  - typically good cost minimization
  - consistent motions
  - handle discrete transitions naturally

- Cons
  - can be slow if heuristic function has deep local minima
  - designing a “good” but fast-to-compute heuristic function is important
  - designing and coding up a compact graph representation can be non-trivial

http://www.sbpl.net/software or http://www.ros.org/wiki/sbpl

Thanks to Willow Garage for their support of SBPL!

Collaborators on SBPL: S. Chitta, B. Cohen, M. Phillips