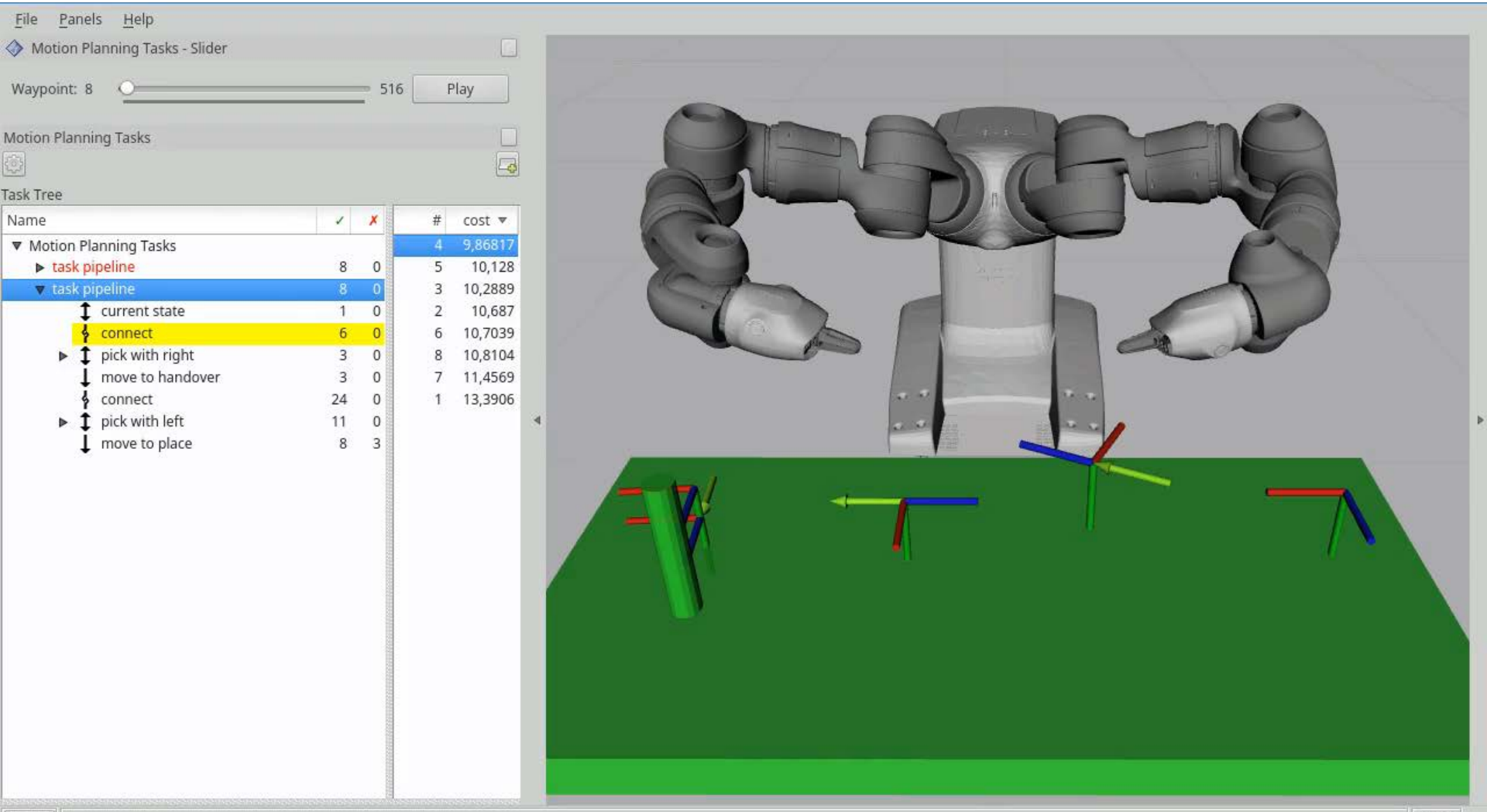
MoveIt! Task Constructor
A framework for planning task sequences

Robert Haschke[1], Michael Görner[2]
[1] *Center of Excellence Cognitive Interaction Technology (CITEC), Bielefeld University, Germany*
[2] *TAMS Group, Hamburg University, Germany*

Motivation


File
Panels
Help
Motion Planning Tasks - Slider
Waypoint: 8
516
Play
Motion Planning Tasks
Task Tree
Name
✓
✗
cost
Motion Planning Tasks
task pipeline 8 0
task pipeline 8 0
current state 1 0
connect 6 0
pick with right 3 0
move to handover 3 0
connect 24 0
pick with left 11 0
move to place 8 3
4 9,86817
5 10,128
3 10,2889
2 10,687
6 10,7039
8 10,8104
7 11,4569
1 13,3906
Reset
31 fps

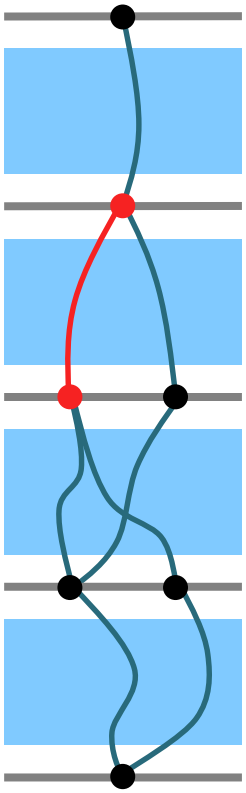
Objectives

- Definition + Planning of non-trivial manipulation sequences
  - Modular
  - Customizable
  - Multiple arms/hands
  - Cost-ranking of alternative solutions

- Replace MoveIt‘s manipulation pipeline
  - Limited to single-arm pick-and-place
  - No introspection

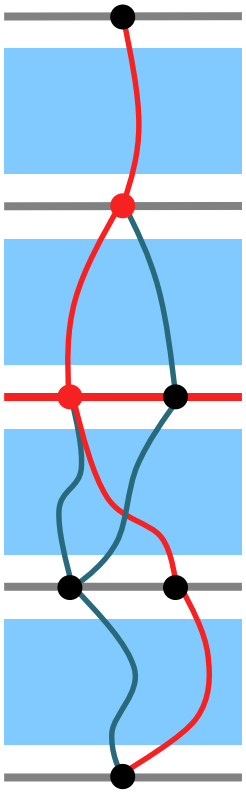
- No Symbolic Task Planning
  - Assuming task structure is known
  - Planning on level of alternative solution paths

Overview



- **Pipeline** composed from **Stages**
- Each stage connects a *start* to an *end* **InterfaceState** via 1...n **SubSolutions**

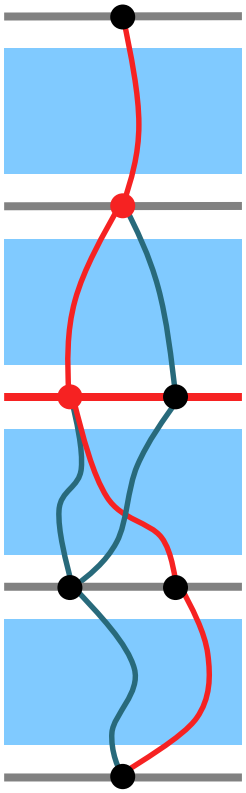
Overview



- **Pipeline** composed from **Stages**
- Each stage connects a *start* to an *end* **InterfaceState** via 1...n **SubSolutions**

- Stages interface each other via *list* of InterfaceStates
- Solution = fully-connected path through pipeline

Overview



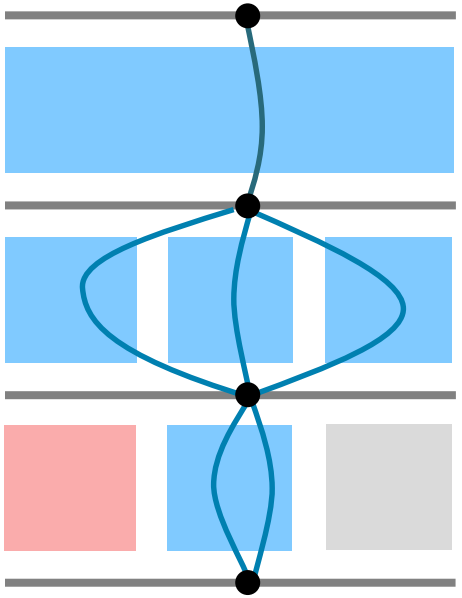
- **Pipeline** composed from **Stages**
- Each stage connects a *start* to an *end* **InterfaceState** via 1...n **SubSolutions**

- Stages interface each other via *list* of InterfaceStates
- Solution = fully-connected path through pipeline

- InterfaceState
  - MoveIt's *PlanningScene*
  - Properties, e.g.
    - grasp type
    - end effector to use for grasping

Hierarchical Structuring

- SerialContainer
  - *Sequential* chaining of sub tasks
- ParallelContainer
  - Alternatives
    - Consider all solutions of children
  - Fallback
    - Consider children one by one
  - Merger
    - Combine solutions of children for parallel execution
    - Example: arm approaching + hand opening
    - Requires extra feasibility check!
- Wrapper
  - Filter / duplicate / modify solutions



Semantic Stage Types

- Planning proceeds non-linearly:
  - generators: seed for planning
  - propagation: advance partial solutions
  - connectors: connect partial solutions

- Example: Pick-n-Place with Handover

  ↕ current state
  ∞ connect
  ↕ pick with right hand
  ↓ move to handover pose
  ∞ connect
  ↕ pick with left hand
  ↓ move to place

Semantic Stage Types

- Planning proceeds non-linearly:
  - generators: seed for planning
  - propagation: advance partial solutions
  - connectors: connect partial solutions

- Example: Pick-n-Place with Handover

  ↕ **current state**
  ∞ connect
  ↕ **pick with right hand**
  ↓ move to handover pose
  ∞ connect
  ↕ **pick with left hand**
  ↓ move to place

Semantic Stage Types

- Planning proceeds non-linearly:
  - generators: seed for planning
  - propagation: advance partial solutions
  - connectors: connect partial solutions

- Example: Pick-n-Place with Handover

  ↕ **current state**
  ∞ connect
  ↕ **pick with right hand**
  ↓ **move to handover pose**
  ∞ connect
  ↕ **pick with left hand**
  ↓ **move to place**

Semantic Stage Types

- Planning proceeds non-linearly:
  - generators: seed for planning
  - propagation: advance partial solutions
  - connectors: connect partial solutions

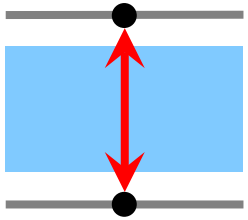
- Example: Pick-n-Place with Handover

↕ current state
∞ connect
↕ pick with right hand
↓ move to handover pose
∞ connect
↕ pick with left hand
↓ move to place

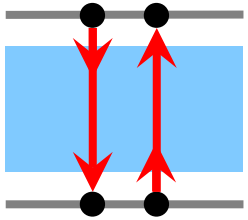
Stage Types by Interface

- Type determined by what is read from / written to interfaces

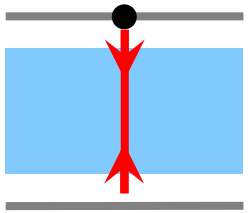
- Generator
  - No reading, Write to both interfaces
  - Examples: CurrentState, FixedState, GraspGenerator



- Propagator
  - Read from one, write to opposite interface
  - Examples: Approach, Lift



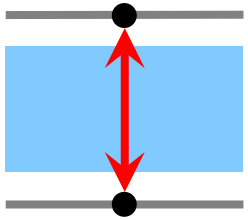
- Connector
  - Read both interfaces
  - Combinatorial explosion
  - Check compatibility of states



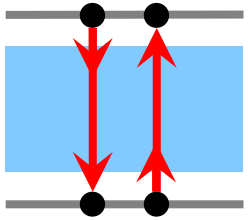
Stage Types by Interface

- Type determined by what is read from / written to interfaces

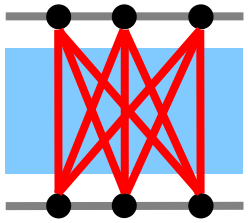
- Generator
  - No reading, Write to both interfaces
  - Examples: CurrentState, FixedState, GraspGenerator



- Propagator
  - Read from one, write to opposite interface
  - Examples: Approach, Lift



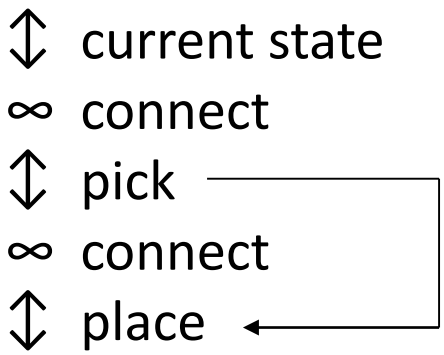
- Connector
  - Read both interfaces
  - Combinatorial explosion
  - Check compatibility of states



MonitoringGenerator

- Generator might need input
  from a *remote* stage
  - Grasp/Place an object
    at the current position

- MonitoringGenerators
  hook into solutions of another stage



Available Primitive Stages

- Generators
  - Fetch current Planning Scene from move_group
  - Cartesian pose generator / sampler
  - ComputeIK
  - Simple grasp generator
- Propagator
  - MoveTo: plan towards absolute goal
  - MoveRelative: plan relative motion
  - Manipulate Planning Scene
    - Attach / Detach objects
    - Modify ACM
- Connect

Joint space or
Cartesian space

Providing Custom Stages

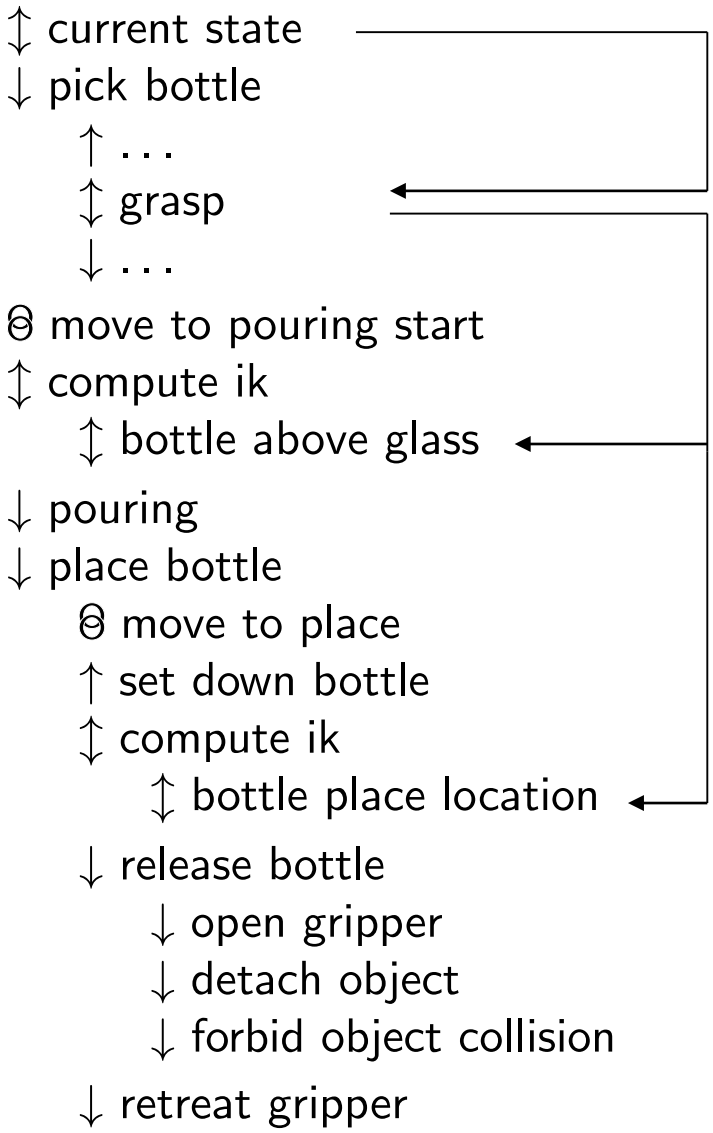
```
class MyStage : public PropagatingForward {
public:
   MyStage(string name);

   void computeForward(const InterfaceState& from) override
   {
      ...
      SubTrajectory solution(trajectory, cost, comment);
      solution.markers().push_back(marker);
      sendForward(from, move(end_scene), move(solution));
   };
};
```

Task Code Example

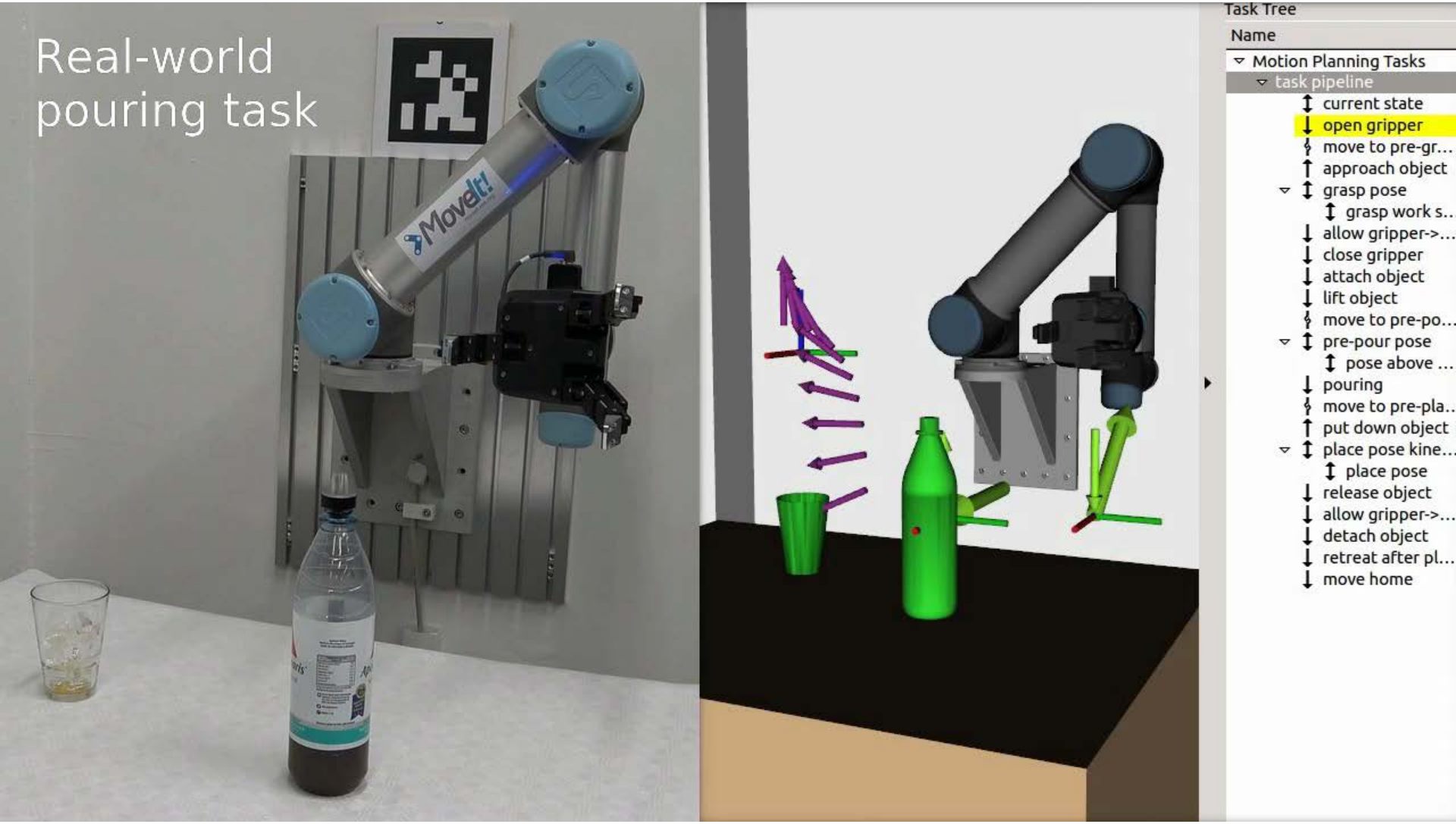
```
Task task;

auto initial = make_unique<CurrentState>();
task.add(move(initial));
...
auto grip = make_unique<MoveTo>("grip", planner);
grip->setGroup("gripper");
grip->setGoal("closed");
task.add(move(grip));
...
if(task.plan())
   execute(task.solutions()[0]);
```

MoveIt pipeline
Straight-line
  Cartesian space
  Joint space

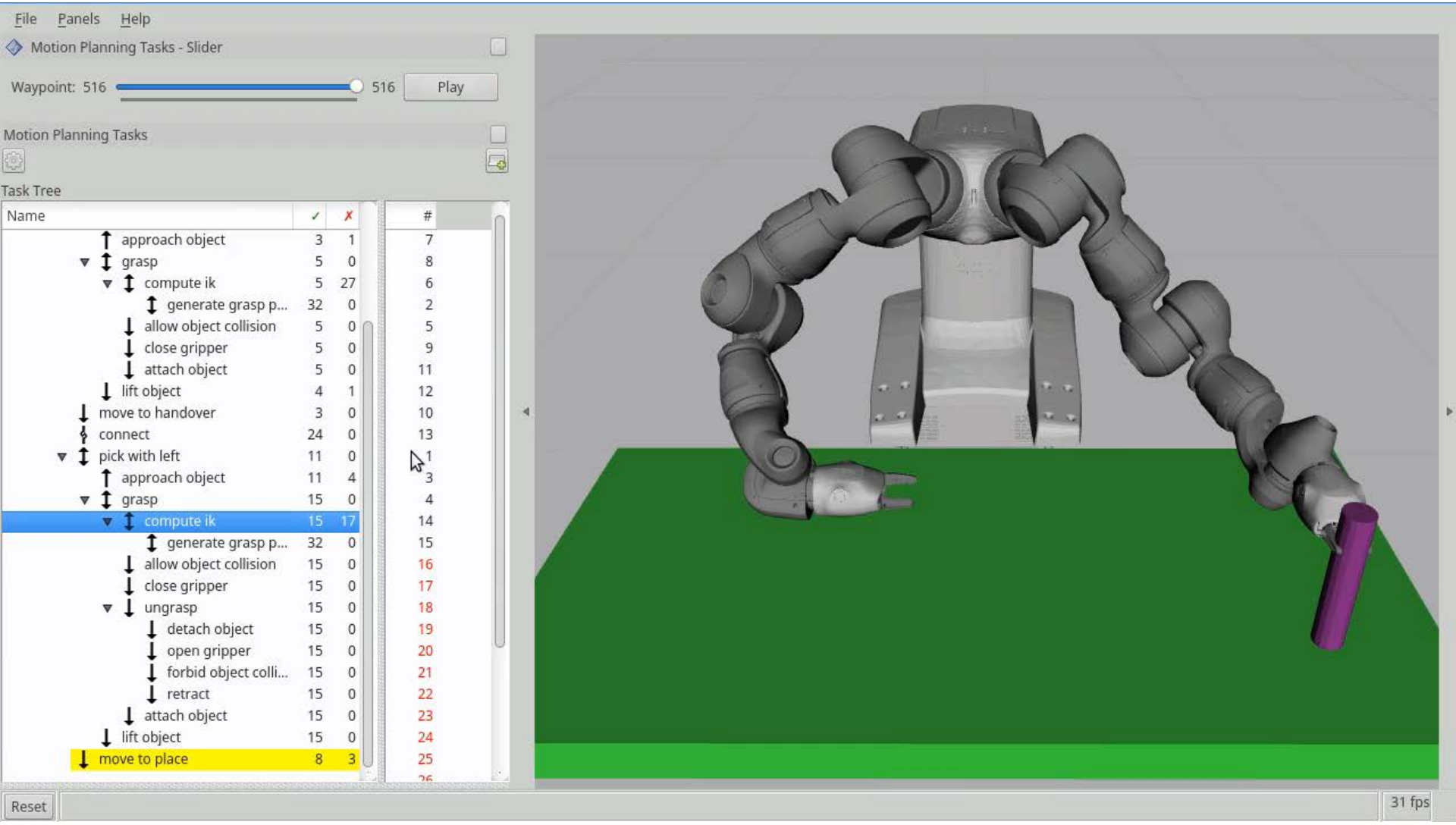
A more complex example: Pouring



A more complex example: Pouring



Introspection



Outlook: Envisioned Features

- Drop-In replacement for MoveIt‘s Pick+Place capability
- Interactive GUI
  - Configure + validate task pipeline in rviz
    - Save / load YAML
    - C++ / python code generation
- Execution Handling
  - Premature execution of planned sub tasks
  - Choose controllers for sub tasks (force control, servoing, …)
  - Failure handling
    - Replan from current situation
    - Revert to previous stage

Scheduling

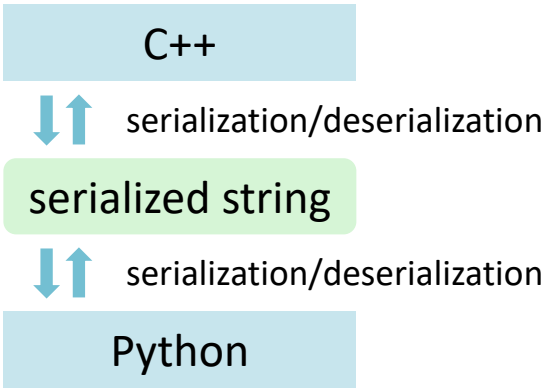
- Find „good“ solutions fast!

- Priority queues @ different levels
  1. InterfaceState: remember best solution only
  2. InterfaceStateList: sort by length and acc. cost of partial solution
  3. Stage scheduling (TODO)
     - Interface type
     - success rate
     - estimated computation time

- Compute stages in parallel threads

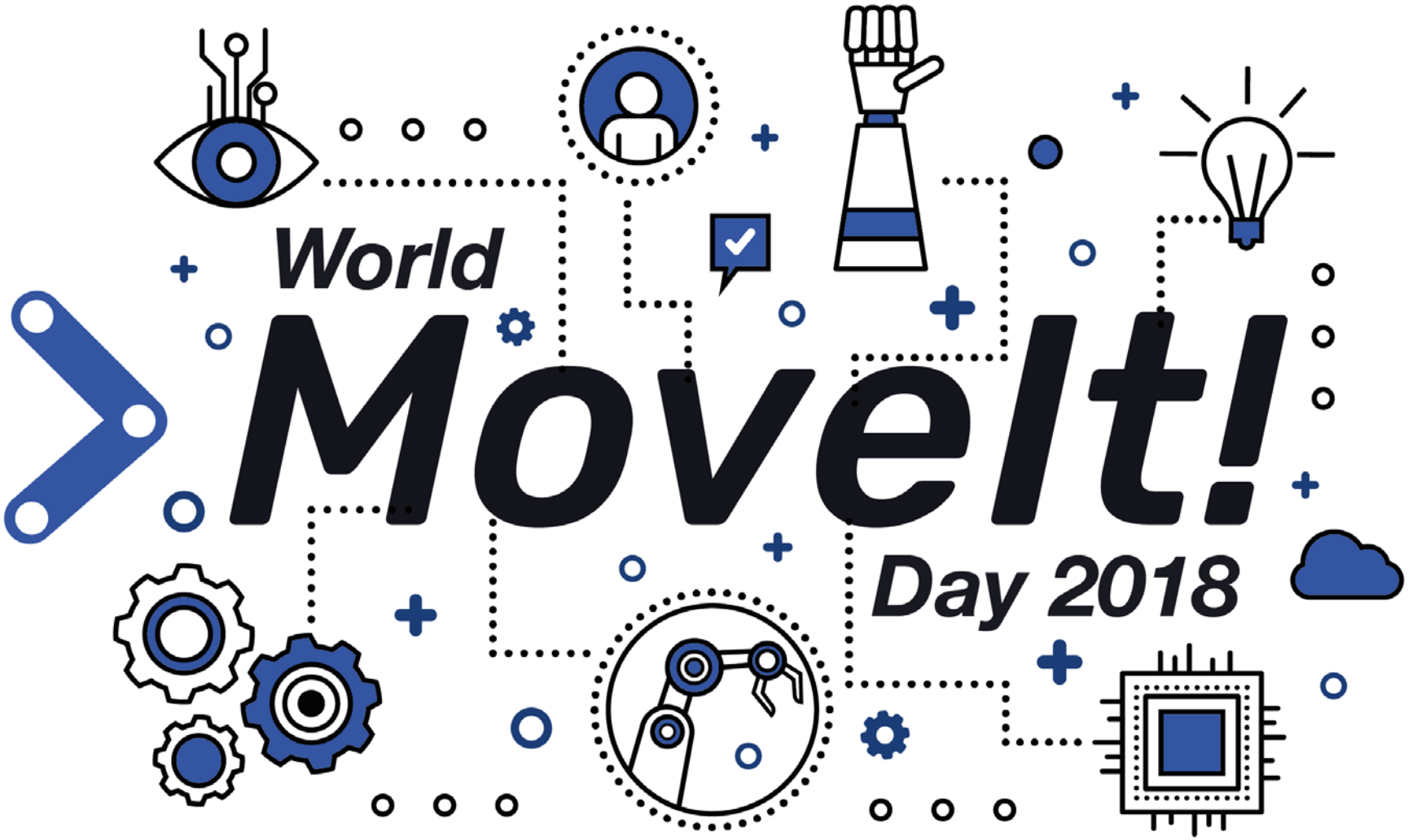
Python Wrapper (wip)

- Using Boost Python
- Data transfer via:
  - ROS msgs
  - serialized strings
  - **Boost Python‘s type conversion magic**




```
task = core.Task()
task.add(CurrentState("current"))
move = MoveTo("move", PipelinePlanner())
move.group = "arm"
goal = RobotState()
…
move.setGoal(goal)
task.add(move)

if task.plan():
    task.execute(task.solutions[0])
```

World MoveIt Day: October 25 2018



- https://github.com/ros-planning/moveit_task_constructor
- https://github.com/ubi-agni/mtc_demos
- https://github.com/tams-group/mtc_pour