PICKNIK

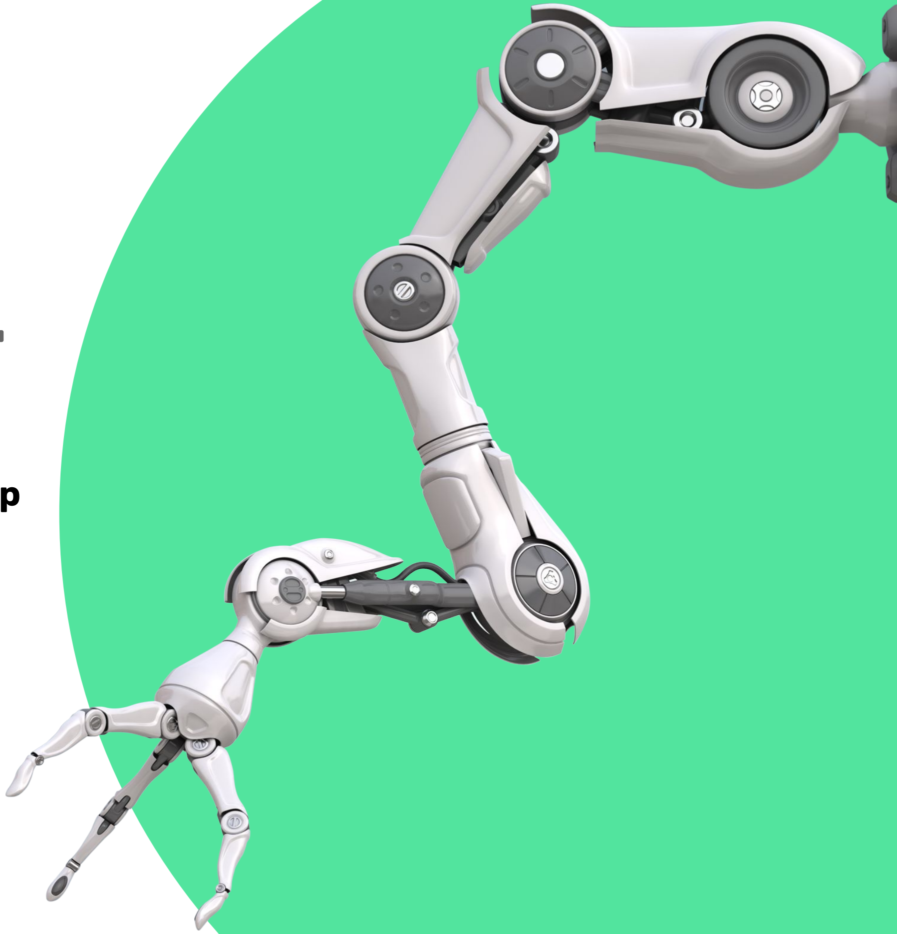
MoveIt Task Constructor

High-Level Task and Motion Planning using MTC

ROSWorld October 2021 - Mobile Manipulation Workshop

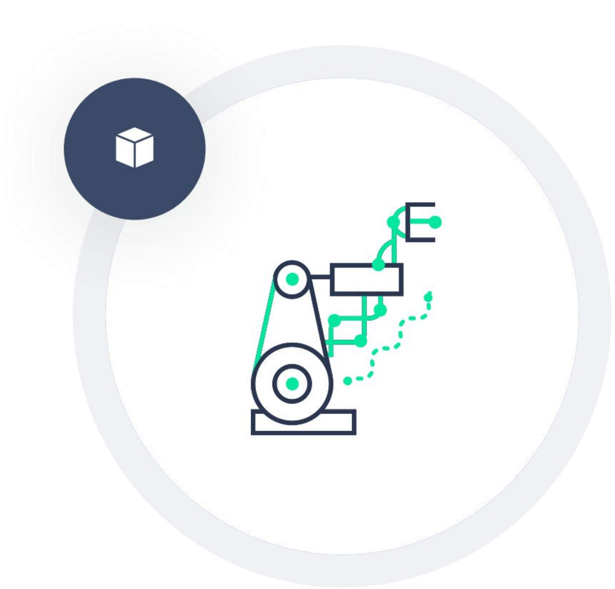
Henning Kayser

henningkayser



Outline

1. Motivation
2. MTC Core Concepts
3. Example Task
4. Default Stages
5. Key Properties
6. Hands-on MTC Demo



Motivation

Goal

- Provide a generic method to solve complex multi-step tasks
- Make code more reusable, maintainable, portable, configurable and robust
- Separate high-level behavior from low-level implementation
- Improve debugging and result introspection

Method

- Encapsulate task steps in composable subproblems
- Generic solvers and interfaces for certain problem types
- Structure for arranging solvers in sequence and hierarchies
- Forwarding of parameters and results between stages
- Inheritance of solver classes

MTC Core Concepts

Task

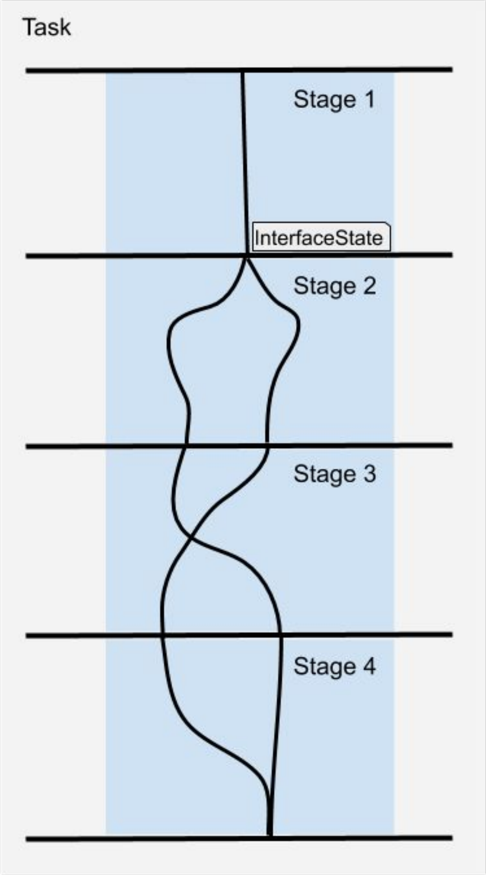
- Specifies a complex planning problem
- Consists of stages that form a sequence of high-level steps

Stage

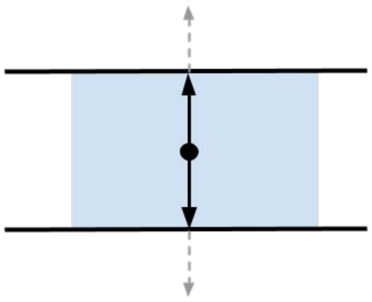
- Low-level implementation of high-level planning steps
- Computes SubSolutions that **connect**, **propagate** or **generate** InterfaceStates

InterfaceState

- Snapshot of planning scene, robot state and properties
- Connection between compatible SubSolutions



Stage Types

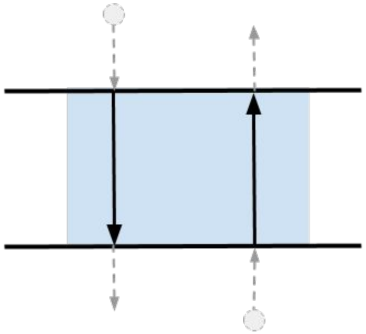


Generator Stage (↕)

- Produces and propagates InterfaceStates to adjacent Stages

Examples:

- Pose sampler (+ IK solver)
- Fixed waypoint state
- Output/Filter of current state

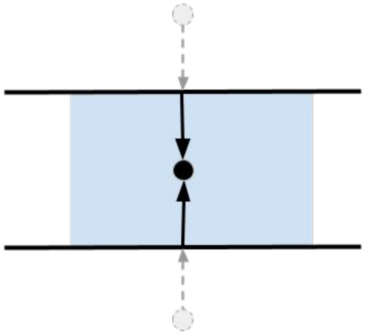


Propagator Stage (⇅ / ↑ / ↓)

- Receives an input InterfaceState, solves a problem and propagates the solution state
- Forward, backward or both

Examples:

- (Relative) cartesian motions (approach/lift when grasping)
- Scene manipulations (attach/detach objects, ACM)
- Filter/Validator of input states



Connector Stage (॥)

- Connects InterfaceStates of both adjacent stages

Example:

- Free-motion plan between start and goal states

Stage Containers and Hierarchies

Serial Container

- Combines multiple sequential stages
- i.e. approach, grasp, lift retreat

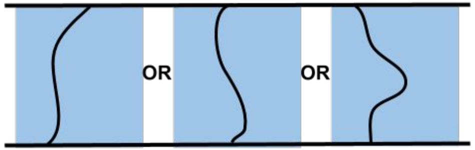
Wrapper Container

- Filter or modify solutions of a subordinate stage
- I.e. wrap a pose generator with an IK solver

Parallel Container

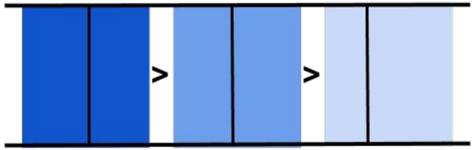
Alternative Stages

- Optional solutions, only one needed
- i.e. pick with left or right hand



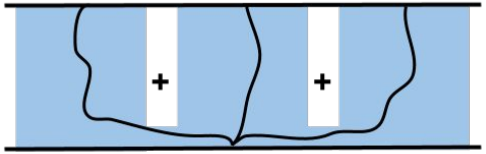
Fallback Stages

- Solve stages in order if higher stages fail
- i.e. default planner and fallback options

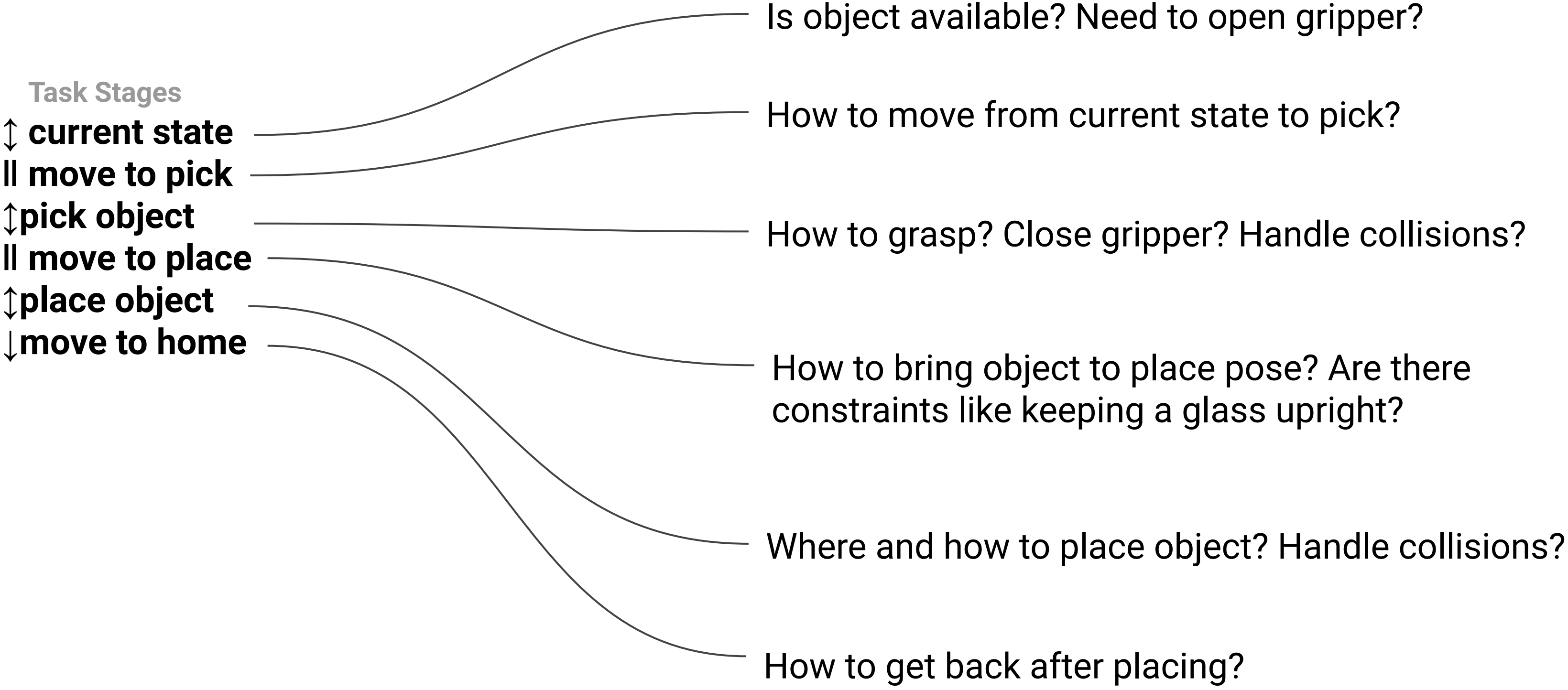


Merger Stages

- Combine multiple distinct problems
- i.e. open gripper while arm moves



Example: Stage Sequence Flow



Task Stages

↕ current state
‖ move to pick
↕pick object
‖ move to place
↕place object
↓move to home

Is object available? Need to open gripper?

How to move from current state to pick?

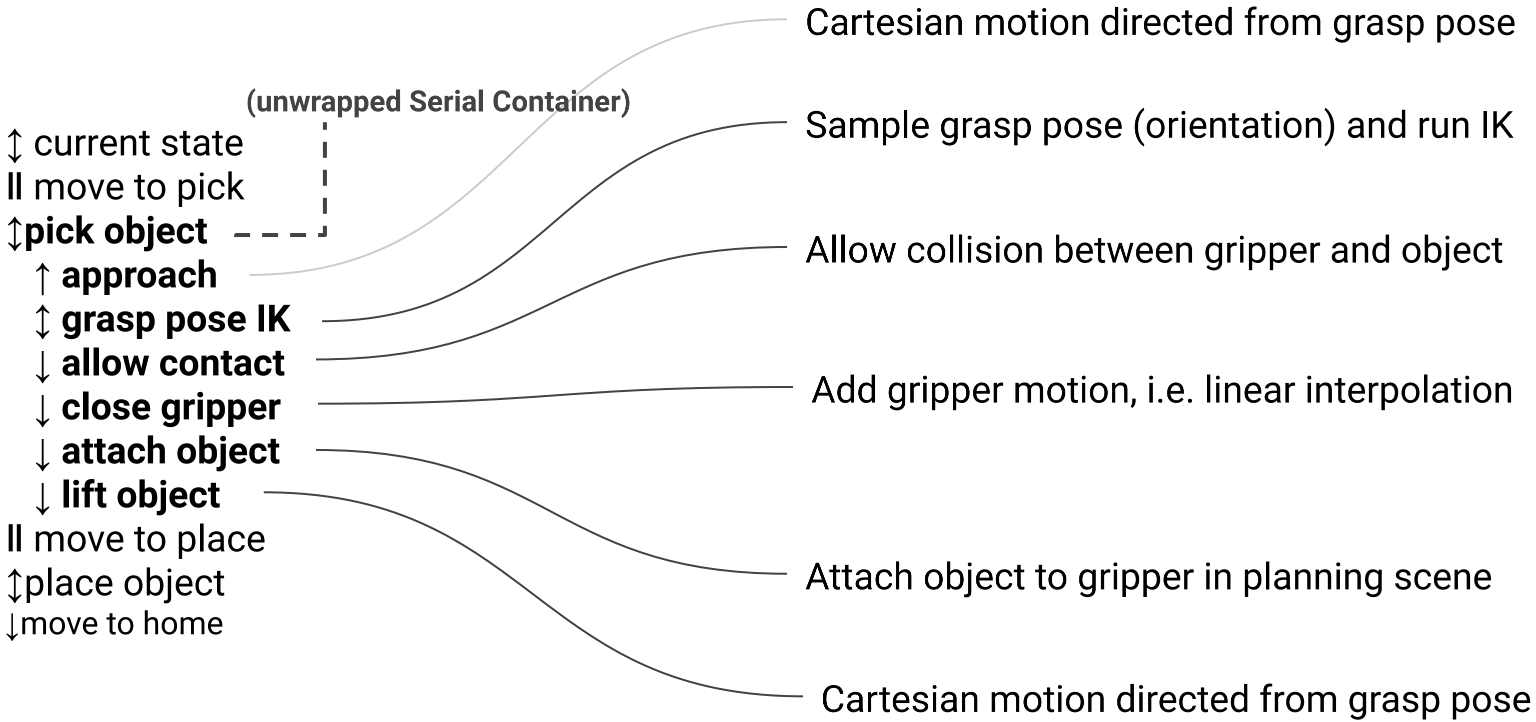
How to grasp? Close gripper? Handle collisions?

How to bring object to place pose? Are there constraints like keeping a glass upright?

Where and how to place object? Handle collisions?

How to get back after placing?

Example: Stage Sequence Flow



(unwrapped Serial Container)

↕ current state
‖ move to pick
↕**pick object**
  ↑ **approach**
  ↕ **grasp pose IK**
  ↓ **allow contact**
  ↓ **close gripper**
  ↓ **attach object**
  ↓ **lift object**
‖ move to place
↕place object
↓move to home

Cartesian motion directed from grasp pose

Sample grasp pose (orientation) and run IK

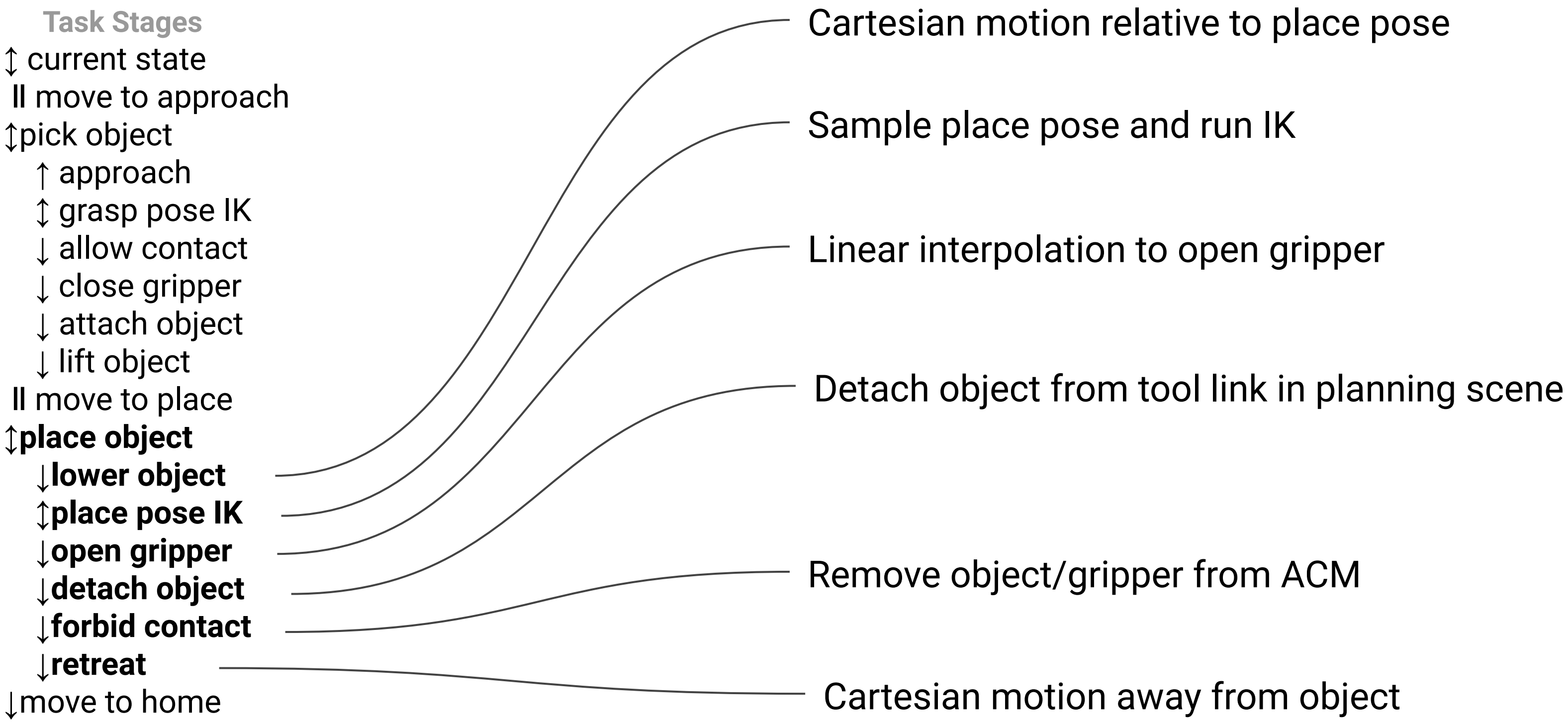
Allow collision between gripper and object

Add gripper motion, i.e. linear interpolation

Attach object to gripper in planning scene

Cartesian motion directed from grasp pose

Example: Stage Sequence Flow



Task Stages

↕ current state

‖ move to approach

↕pick object

↑ approach

↕ grasp pose IK

↓ allow contact

↓ close gripper

↓ attach object

↓ lift object

‖ move to place

↕place object

↓lower object

↕place pose IK

↓open gripper

↓detach object

↓forbid contact

↓retreat

↓move to home

Cartesian motion relative to place pose

Sample place pose and run IK

Linear interpolation to open gripper

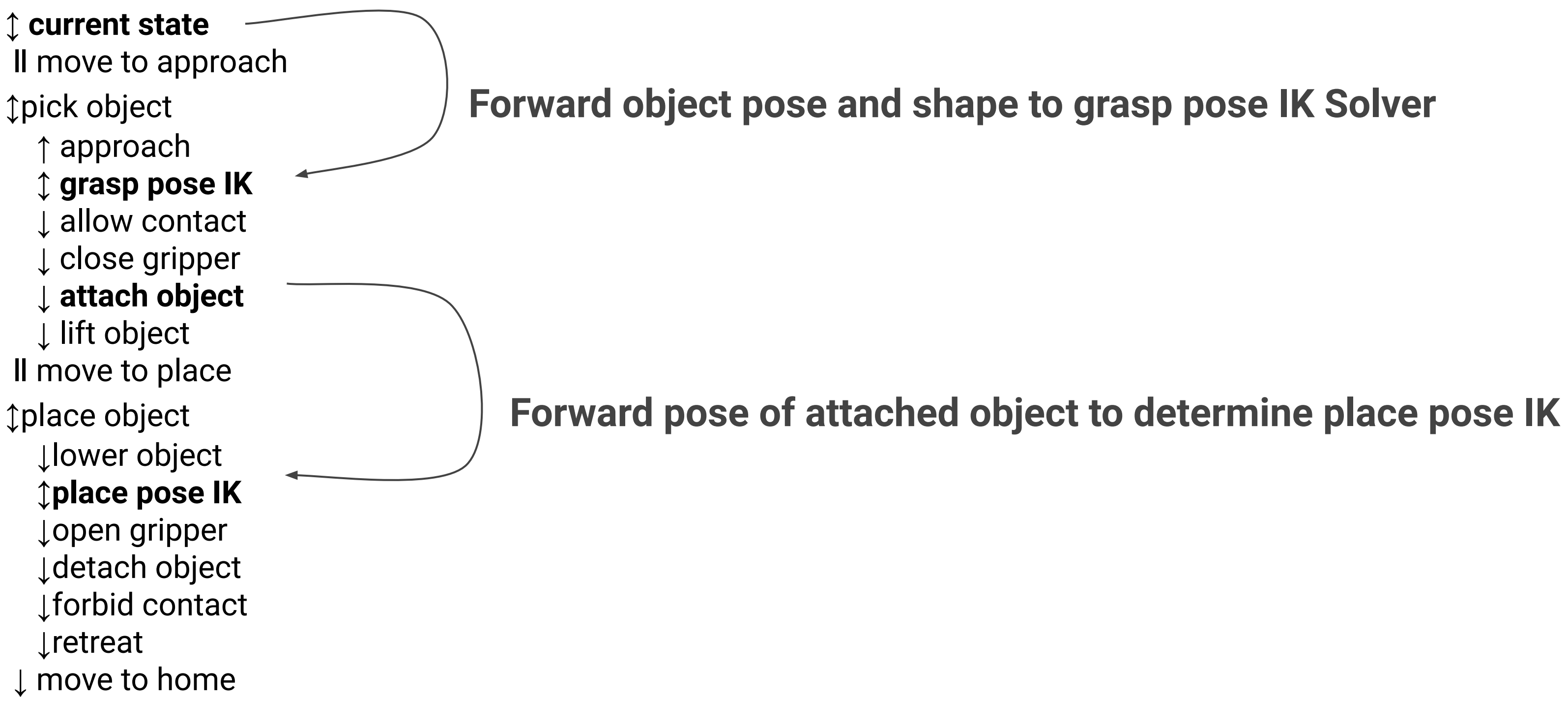
Detach object from tool link in planning scene

Remove object/gripper from ACM

Cartesian motion away from object

Monitoring Generator

...are stages that hook into remote stages for accessing solutions.



↕ **current state**

‖ move to approach

↕pick object

↑ approach

↕ **grasp pose IK**

↓ allow contact

↓ close gripper

↓ **attach object**

↓ lift object

‖ move to place

↕place object

↓lower object

↕**place pose IK**

↓open gripper

↓detach object

↓forbid contact

↓retreat

↓ move to home

Forward object pose and shape to grasp pose IK Solver

Forward pose of attached object to determine place pose IK

Default Stage Classes

18 Steps

- ↕ current state
- ‖ move to approach
- ↕pick object
  - ↑ approach
  - ↕ grasp pose IK
  - ↓ allow contact
  - ↓ close gripper
  - ↓ attach object
  - ↓ lift object
- ‖ move to place
- ↕place object
  - ↓lower object
  - ↕place pose IK
  - ↓open gripper
  - ↓detach object
  - ↓forbid contact
  - ↓retreat
- ↓ move to home

9 Stage Classes

- CurrentState
- Connect
- SerialContainer
  - MoveRelative
  - ComputeIK { GenerateGraspPose }
  - ModifyPlanningScene
  - MoveTo
  - ModifyPlanningScene
  - MoveRleative
- Connect
- SerialContainer
  - MoveRelative
  - ComputeIK { GeneratePlacePose }
  - MoveTo
  - ModifyPlanningScene
  - ModifyPlanningScene
  - MoveRelative
- MoveTo

7 Primitive Stage Classes

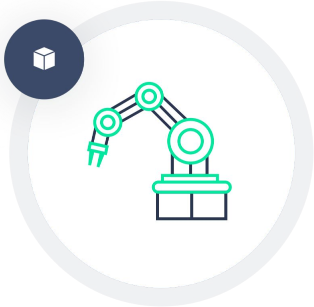
… provided with the MTC library!

CurrentState (Generator)
Connect (Connector)
MoveRelative (Propagator)
ComputeIK (Generator)
ModifyPlanningScene (Propagator)
MoveTo (Propagator)
GeneratePose (Generator)

Key Properties

Advantages

+ Abstraction from setup/robot
+ Code reusability
+ End-to-end manipulation planning
+ Alternative/optional solution paths
+ Visual debugging (limited)
+ Integration with higher level control architectures
+ Solution robustness
+ Testability and maintainability

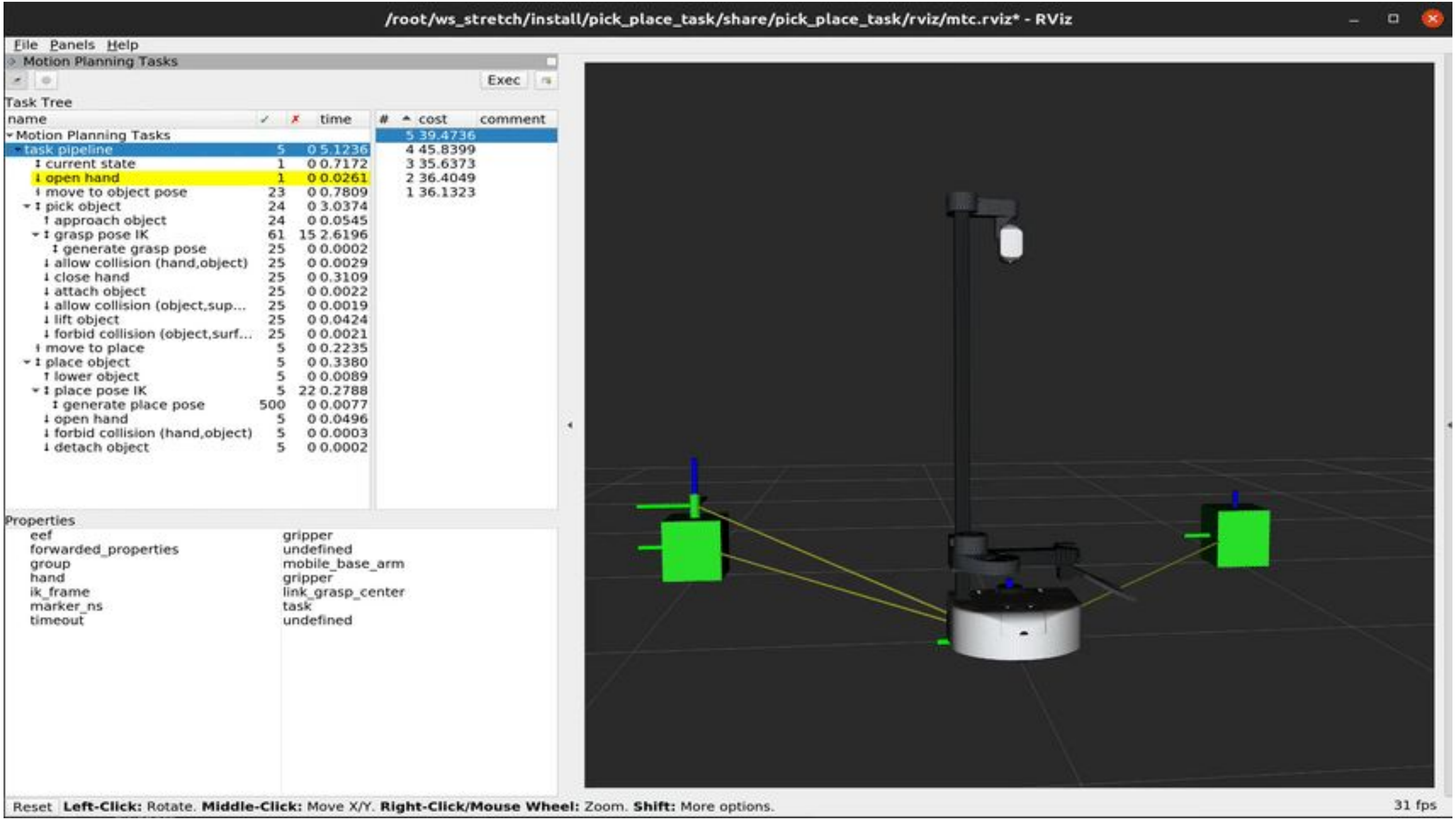


Drawbacks

- New methodology
  -> steep learning curve
- Unintuitive backward+forward directions
- Not possible to adapt running tasks to environment
- Graph complexity can increase planning times exponentially

Hands-on MTC Demo

Runtime Demo



/root/ws_stretch/install/pick_place_task/share/pick_place_task/rviz/mtc.rviz* - RViz

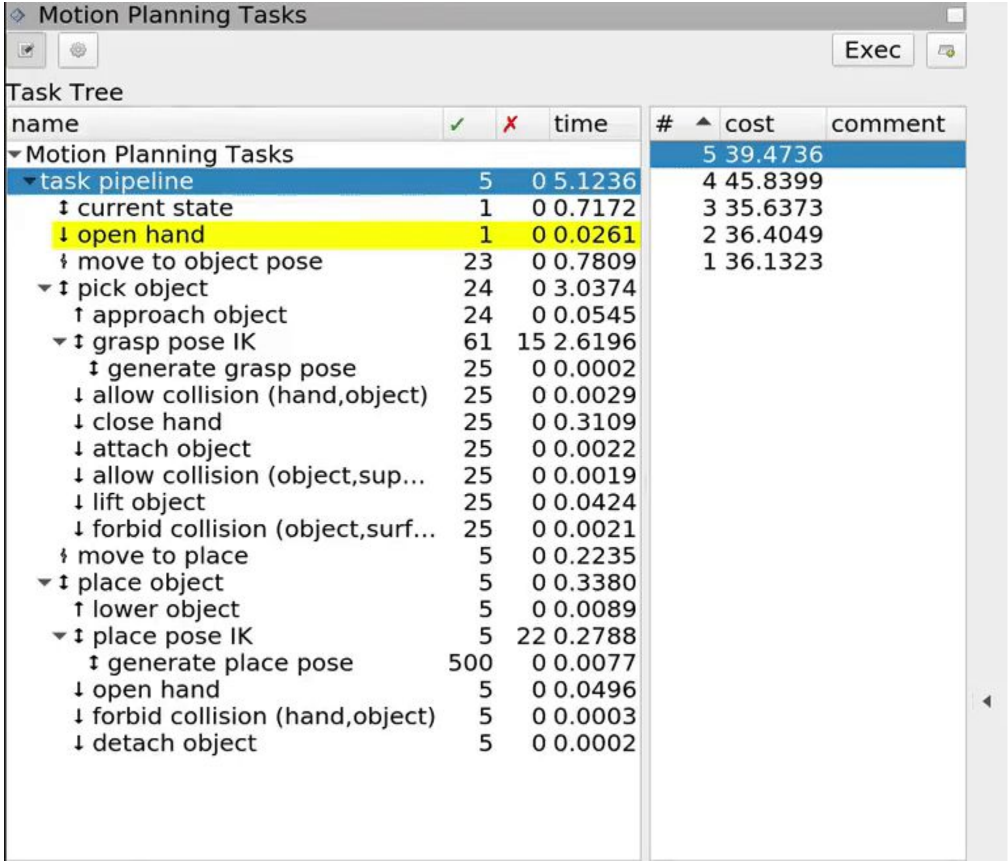
name	✓	✗	time
Motion Planning Tasks			
task pipeline	5	0	5.1236
current state	1	0	0.7172
open hand	1	0	0.0261
move to object pose	23	0	0.7809
pick object	24	0	3.0374
approach object	24	0	0.0545
grasp pose IK	61	15	2.6196
generate grasp pose	25	0	0.0002
allow collision (hand,object)	25	0	0.0029
close hand	25	0	0.3109
attach object	25	0	0.0022
allow collision (object,sup...	25	0	0.0019
lift object	25	0	0.0424
forbid collision (object,surf...	25	0	0.0021
move to place	5	0	0.2235
place object	5	0	0.3380
lower object	5	0	0.0089
place pose IK	5	22	0.2788
generate place pose	500	0	0.0077
open hand	5	0	0.0496
forbid collision (hand,object)	5	0	0.0003
detach object	5	0	0.0002

#	cost	comment
5	39.4736	
4	45.8399	
3	35.6373	
2	36.4049	
1	36.1323	

eef	gripper
forwarded_properties	undefined
group	mobile_base_arm
hand	gripper
ik_frame	link_grasp_center
marker_ns	task
timeout	undefined

Reset **Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:** Zoom. **Shift:** More options. 31 fps

Runtime Demo: MTC Panel



Task Tree

name	✓	✗	time
▾ Motion Planning Tasks			
▾ task pipeline	5	0	5.1236
↕ current state	1	0	0.7172
↓ open hand	1	0	0.0261
⇅ move to object pose	23	0	0.7809
▾ ↕ pick object	24	0	3.0374
↑ approach object	24	0	0.0545
▾ ↕ grasp pose IK	61	15	2.6196
↕ generate grasp pose	25	0	0.0002
↓ allow collision (hand,object)	25	0	0.0029
↓ close hand	25	0	0.3109
↓ attach object	25	0	0.0022
↓ allow collision (object,sup...	25	0	0.0019
↓ lift object	25	0	0.0424
↓ forbid collision (object,surf...	25	0	0.0021
⇅ move to place	5	0	0.2235
▾ ↕ place object	5	0	0.3380
↑ lower object	5	0	0.0089
▾ ↕ place pose IK	5	22	0.2788
↕ generate place pose	500	0	0.0077
↓ open hand	5	0	0.0496
↓ forbid collision (hand,object)	5	0	0.0003
↓ detach object	5	0	0.0002

#	▲ cost	comment
5	39.4736	
4	45.8399	
3	35.6373	
2	36.4049	
1	36.1323	

Runtime Demo: Pipeline Initialization

```
using namespace moveit::task_constructor;
RCLCPP_INFO(LOGGER, "Initializing task pipeline");
task_ = std::make_unique<Task>(); // pick_place_task
task_->loadRobotModel(node);

task_->setProperty("group", parameters.mobile_base_arm_group_name);
task_->setProperty("eef", parameters.end_effector_name);
task_->setProperty("hand", parameters.hand_group_name);
task_->setProperty("ik_frame", parameters.hand_frame);

auto sampling_planner = std::make_shared<solvers::PipelinePlanner>(node);
auto cartesian_planner = std::make_shared<solvers::CartesianPath>();

…   /** Populate Task Stages **/

task_->enableIntrospection(); // Enable RViz panel
task_->plan(5 /* max_solutions */);
if (task_->numSolutions() > 0)
  task_->execute(*task_->solutions().front());
```

Runtime Demo: Stage Implementation

```
/** Open Hand **/
 {
  auto stage =
    std::make_unique<stages::MoveTo>("open hand", sampling_planner);
  stage->setGroup(parameters.hand_group_name);
  stage->setGoal(parameters.hand_open_pose);
  task_->add(std::move(stage));  // Populate Task
 }
```

Runtime Demo: Monitoring Stage

PICKNIK

```
// Forward current_state on to grasp pose generator
Stage *current_state_ptr = nullptr;

/** Current State **/
{
  auto current_state =
      std::make_unique<stages::CurrentState>("current state");
  current_state_ptr = current_state.get();
  task_->add(std::move(current_state));

}


…
```

```
/** Generate Grasp Pose **/
{
  // Sample grasp pose
  auto stage = std::make_unique<stages::GenerateGraspPose>("generate grasp pose");
  stage->properties().configureInitFrom(Stage::PARENT);
  stage->properties().set("marker_ns", "grasp_pose");
  stage->setPreGraspPose(parameters.hand_open_pose);
  stage->setObject(parameters.object_name);
  stage->setAngleDelta(M_PI / 12);
  stage->setMonitoredStage(current_state_ptr); // Hook into current state

  // Compute IK
  auto wrapper = std::make_unique<stages::ComputeIK>("grasp pose IK", std::move(stage));
  wrapper->setMaxIKSolutions(8);
  wrapper->setMinSolutionDistance(1.0);
  wrapper->setIKFrame(parameters.hand_frame);
  wrapper->properties().configureInitFrom(Stage::PARENT, {"eef", "group"});
  wrapper->properties().configureInitFrom(Stage::INTERFACE, {"target_pose"});
  grasp->insert(std::move(wrapper));
}
```

Have fun!