Task-Level Motion Planning for Multi-Manipulator System

(Rajendra Singh, final Year, Computer Science and Engineering)

28 January 2020

BTP PRESENTATION

IIT PALAKKAD

Indian Institute of Technology Palakkad
भारतीय प्रौद्योगिकी संस्थान पालक्काड
Under Ministry of Human Resource Development, Govt. of India
मानव संसाधन विकास मंत्रालय के अधीन, भारत सरकार

GadgEon
Engineering Smartness

TABLE OF CONTENT

Motivation

Nasa's Robonaut Mission

Reference: https://robonaut.jsc.nasa.gov/R2/

DARPA Robotics Challenge(DRC)

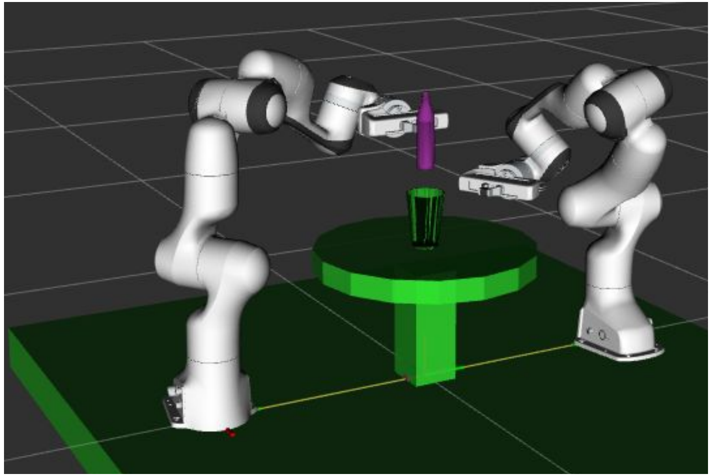
Reference: https://www.darpa.mil/program/darpa-robotics-challenge

Problem Statement

Problem: **Perform complex manipulation task like pick and place, building structures and pouring in multi-manipulator system.**

Subtask/Workflow :

1. Simple **Joint** space planning(**move group**)
2. Simple **Cartesian** space planning(move group)
3. **Pick Place** Task (move group)
4. Simple Joint space planning(MoveIt Task Constructor - **MTC**)
5. Simple Cartesian space planning(MTC)
6. Pick Place Task(MTC)
7. **Multi arm** simple Joint space planning
8. Multi arm simple Cartesian space planning
9. Multi arm Simple Pick Place Task(own work)
10. Multi arm Complex Pick Place Task(IIT)
11. Multi arm planning using Serial container
12. Multi arm planning using Parallel container
    12.1 Alternative
    12.2 Fallback
    12.3 Merger
13. Multiple task
14. Single arm pouring task
15. Complex Multi arm pouring
16. Complex Multi arm pouring task with stages intermixing
17. **Complex pouring task using multiple arm with orientation constraint imposed**

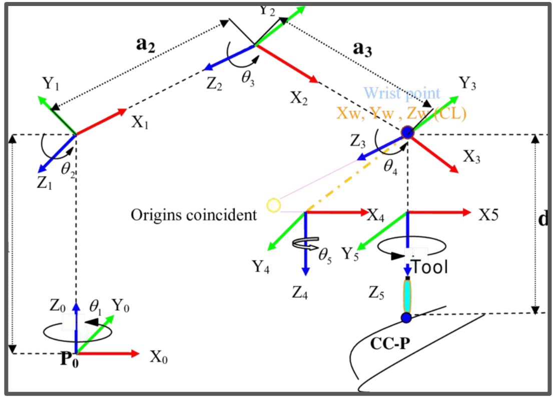


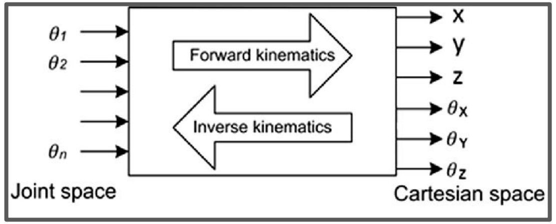
Tools:

1. Panda arm(7 dof arm)
2. Robot operating System(ROS)
3. Motion Planning framework, moveit
4. Moveit_task_constructor(MTC)
5. Open Motion Planning Library(OMPL)

Relevant Work

Inverse Kinematics (Newton methods)

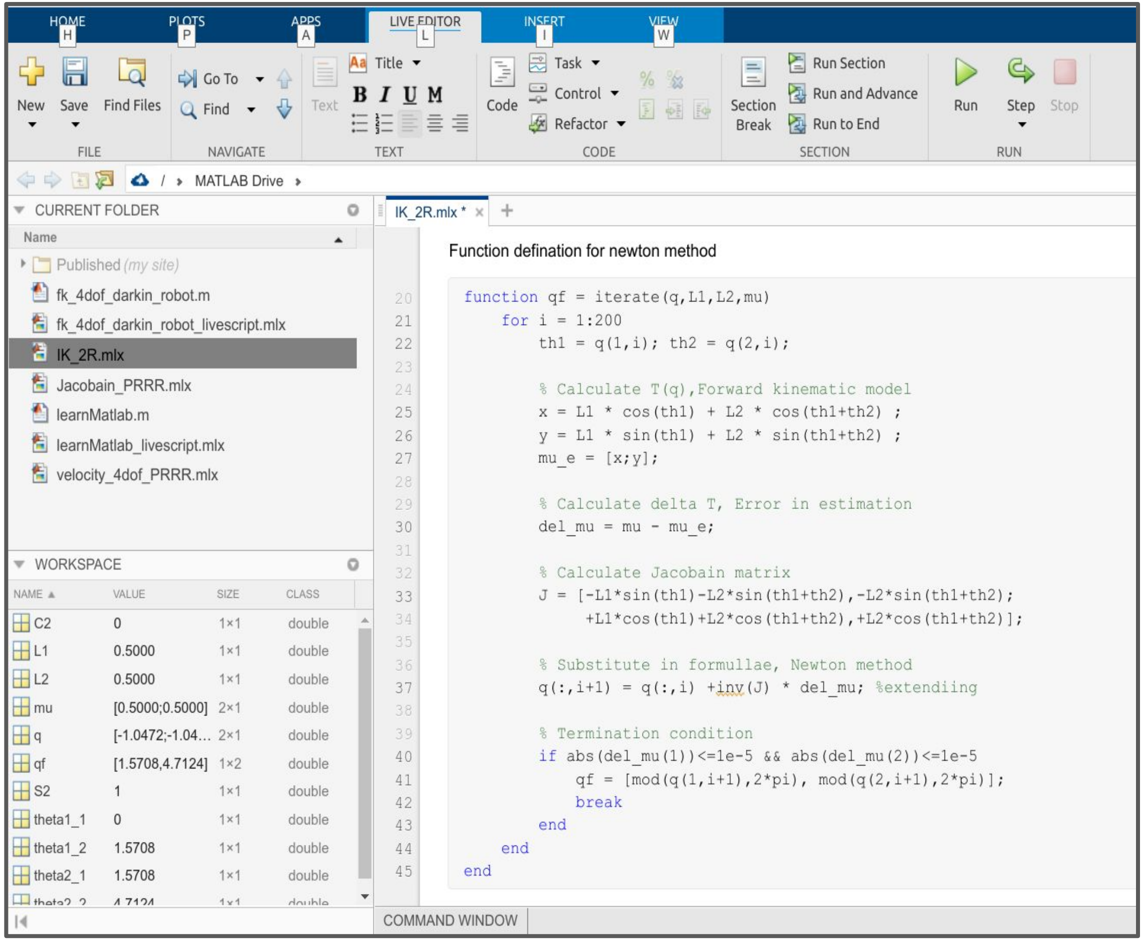




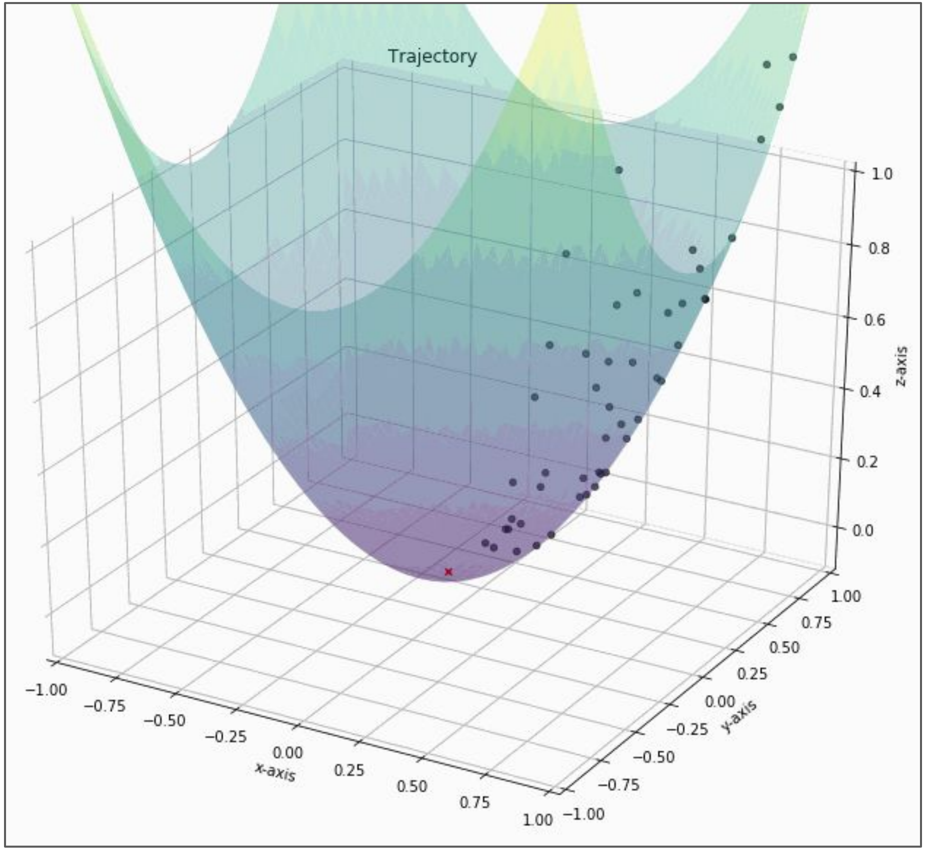
$$\cos q_2 = \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1 a_2}$$

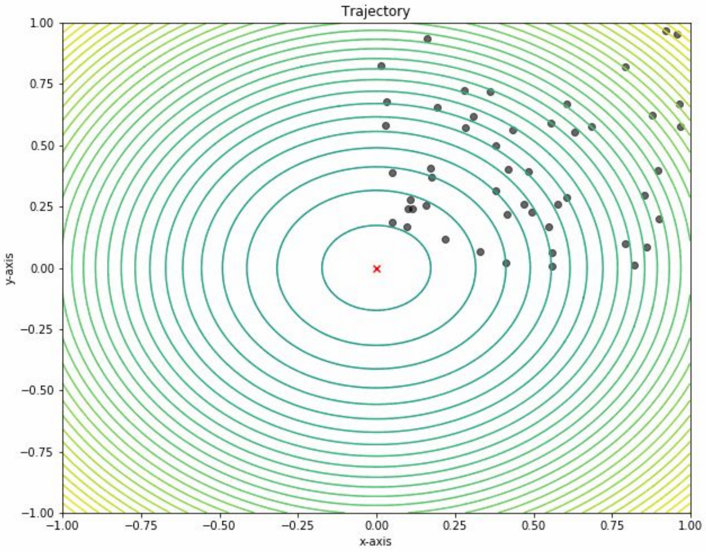
$$q_2 = \cos^{-1} \frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1 a_2}$$

$$q_1 = \tan^{-1} \frac{y}{x} - \tan^{-1} \frac{a_2 \sin q_2}{a_1 + a_2 \cos q_2}$$



Particle swarm optimization (PSO)





velocity of particle i at time k+1 →

$$\mathbf{v}_{k+1}^{i} = \underbrace{w\,\mathbf{v}_k^i}_{\text{current motion}} + \underbrace{c_1\, rand\, \frac{\left(\mathbf{p}^i - \mathbf{x}_k^i\right)}{\Delta t}}_{\text{particle memory influence}} + \underbrace{c_2\, rand\, \frac{\left(\mathbf{p}_k^g - \mathbf{x}_k^i\right)}{\Delta t}}_{\text{swarm influence}}$$

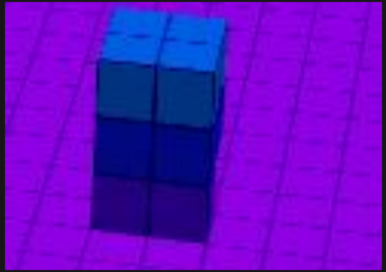
inertia factor range: 0.4 to 1.4

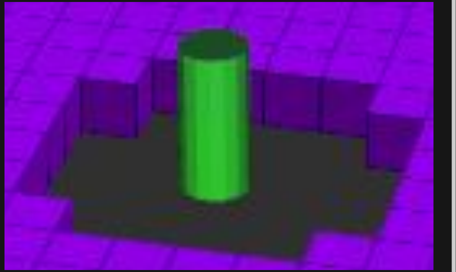
self confidence range: 1.5 to 2

swarm confidence range: 2 to 2.5

Perception

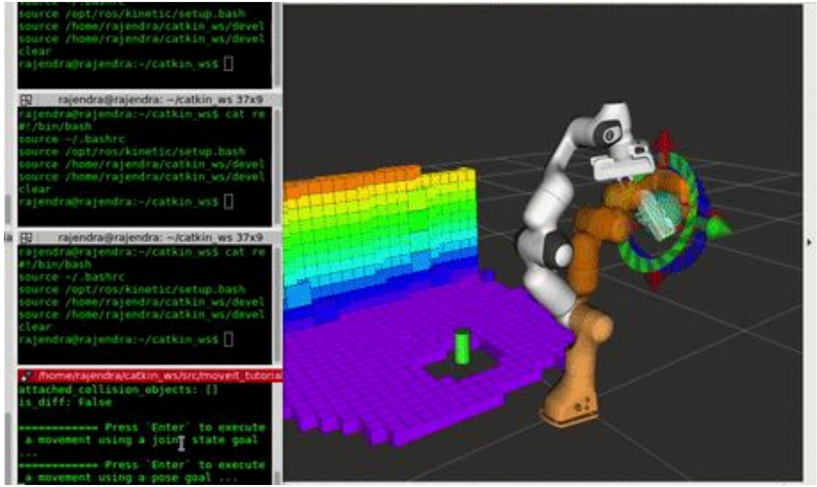
- Converting pointcloud to pcl:PointXYZRGB
- PassThroughFilter
- Compute the point normals
- Detect and eliminate the plane
- Extracting plane normals
- Extract the cylinder
- Compute cylinder_params





Pick and place stack

- Add the collision object and cloud
- Declare the gripper and arm group
- Declare the pre-grasp, grasp and post-grasp approaches
- Chose the planner
- Plan and execute the trajectory



High level planning with Moveit on real robot

Planning group

- arm
  - *Joints*
    - Joint1 - Revolute
    - Joint2 - Revolute
    - Joint3 - Revolute
  - *Links*
    - Link1
    - Link2
    - Base
    - Link3
    - Link8
  - *Chain*
    - Base -> Link3
  - *Subgroups*

Pointcloud

Point Cloud

Point Cloud

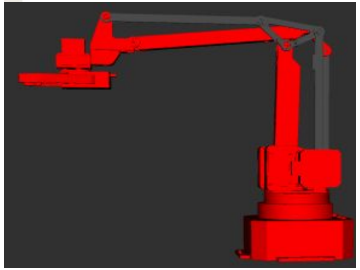
Point Cloud Topic:	/camera/depth/color/points
Max Range:	2.0
Point Subsample:	1
Padding Offset:	0.1
Padding Scale:	1.0
Filtered Cloud Topic:	filtered_cloud
Max Update Rate:	1.0

Collision matrix

	Base	Link1	Link2	Link3	Link8	Link9	Link4	Link5	Link6	Link7
Base		✓	☐	☐	☐	☐	☐	☐	☐	☐
Link1	✓		✓	☐	☐	☐	✓	✓	✓	☐
Link2	☐	✓		✓	☐	☐	✓	☐	☐	☐
Link3	☐	☐	✓		✓	✓	✓	✓	☐	✓
Link8	☐	☐	☐	✓		✓	☐	☐	☐	☐
Link9	☐	☐	☐	✓	✓		✓	☐	✓	✓
Link4	☐	✓	✓	✓	☐	✓		✓	✓	✓
Link5	☐	✓	☐	✓	☐	☐	✓		✓	✓
Link6	☐	✓	☐	☐	☐	✓	✓	✓		✓
Link7	☐	☐	☐	✓	☐	✓	✓	✓	✓	

Known pose

	Pose Name	Group Name
1	home	arm
2	rest	arm



Virtual Joint

Virtual Joint Name:	virtual joint
Child Link:	Base
Parent Frame Name:	world
Joint Type:	fixed

Passive joint

	Joint Names
1	Joint8
2	Joint9
3	Joint4
4	Joint5
5	Joint6
6	Joint7

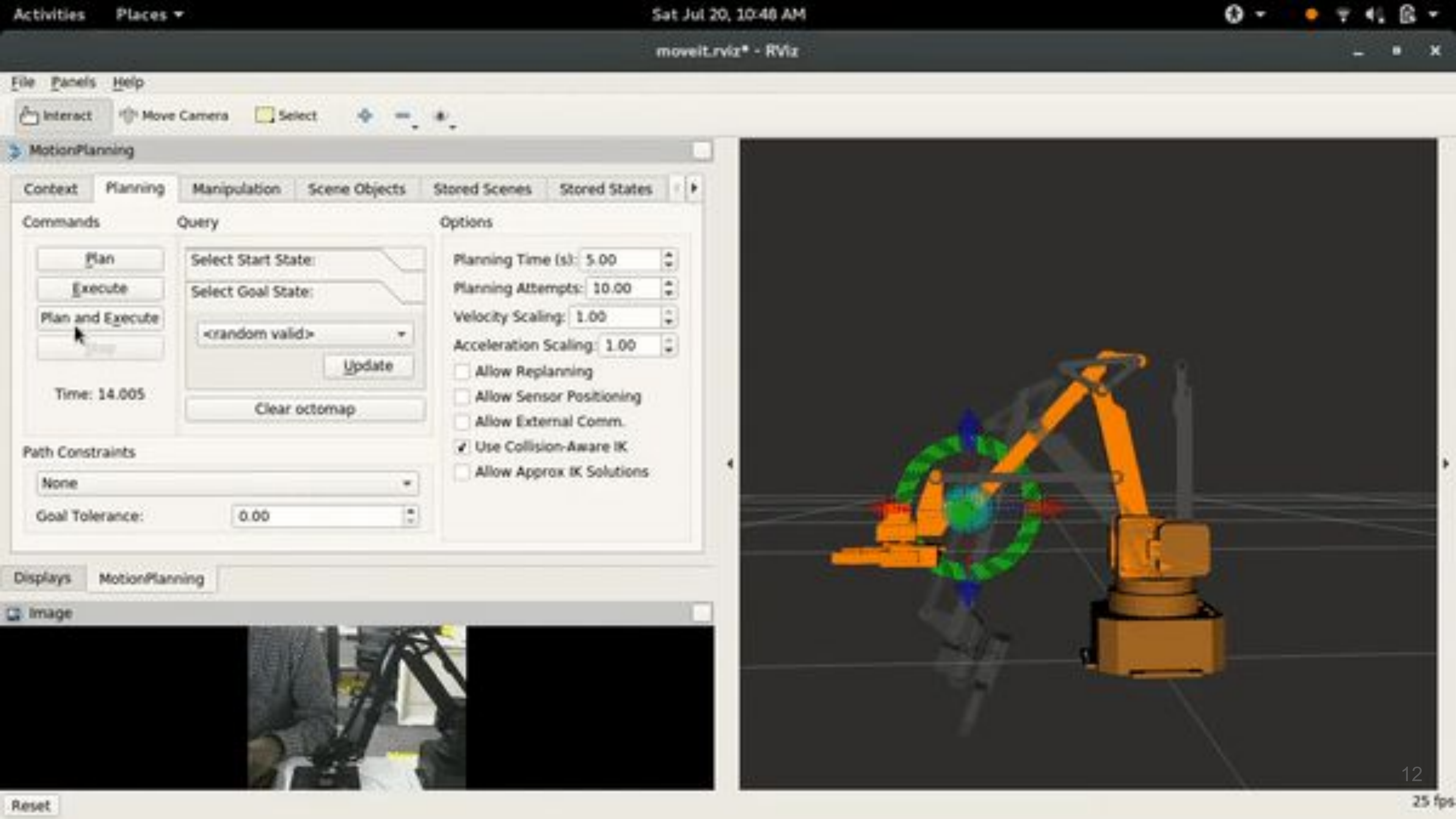
Inverse Kinematics

Kinematics

Group Name:	arm
Kinematic Solver:	trac_ik_kinematics_plugin/TRAC_IKKinematicsPlugin
Kin. Search Resolution:	0.005
Kin. Search Timeout (sec):	0.05
Kin. Solver Attempts:	3

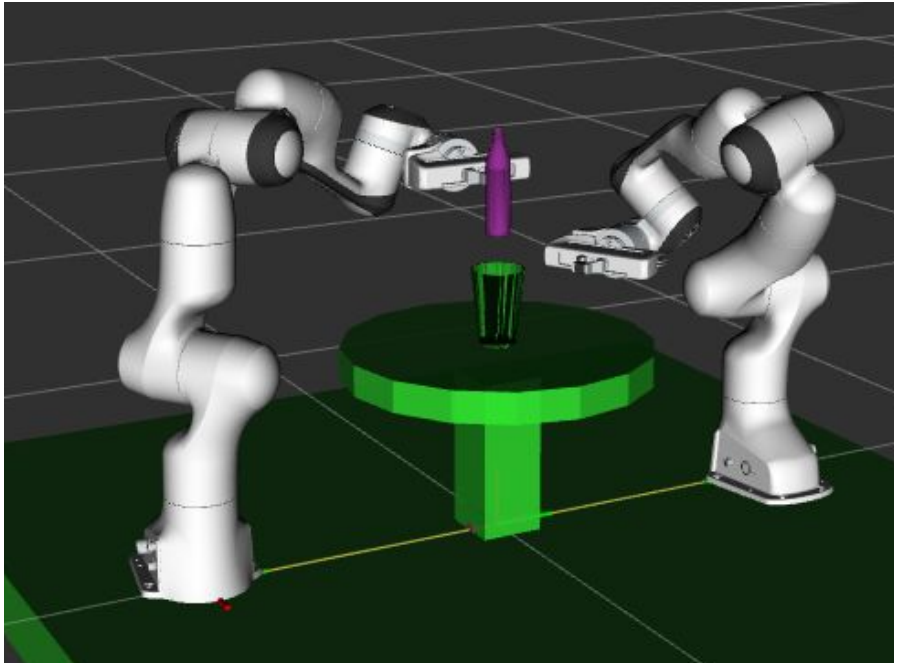
OMPL Planning

Group Default Planner:	RRTstar

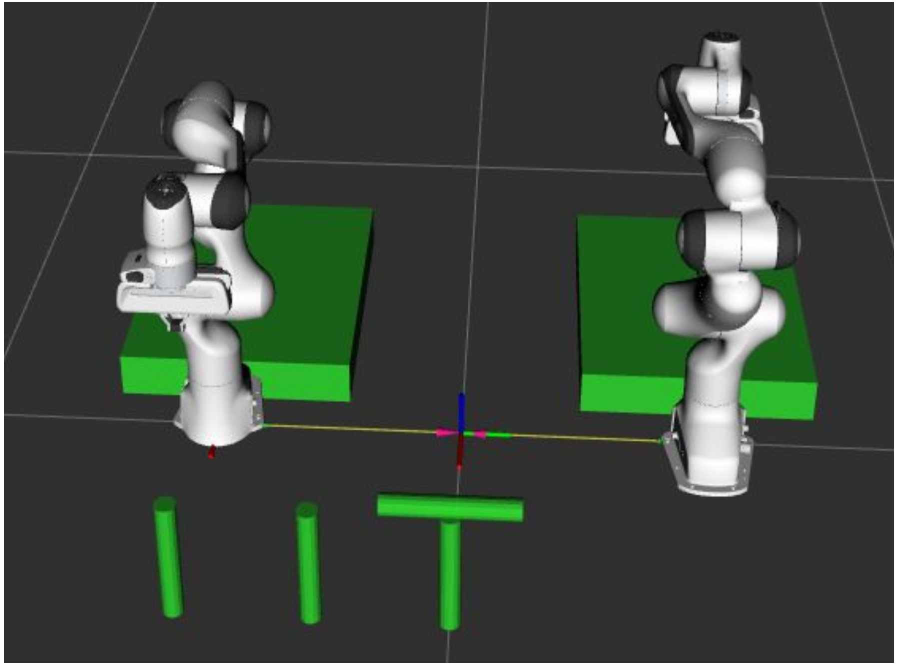

Activities
Places
Sat Jul 20, 10:48 AM
moveit.rviz* - RViz
File
Panels
Help
Interact
Move Camera
Select
MotionPlanning
Context
Planning
Manipulation
Scene Objects
Stored Scenes
Stored States
Commands
Query
Options
Plan
Execute
Plan and Execute
Select Start State:
Select Goal State:
<random valid>
Update
Time: 14.005
Clear octomap
Planning Time (s): 5.00
Planning Attempts: 10.00
Velocity Scaling: 1.00
Acceleration Scaling: 1.00
Allow Replanning
Allow Sensor Positioning
Allow External Comm.
Use Collision-Aware IK
Allow Approx IK Solutions
Path Constraints
None
Goal Tolerance:
0.00
Displays
MotionPlanning
Image
Reset
25 fps

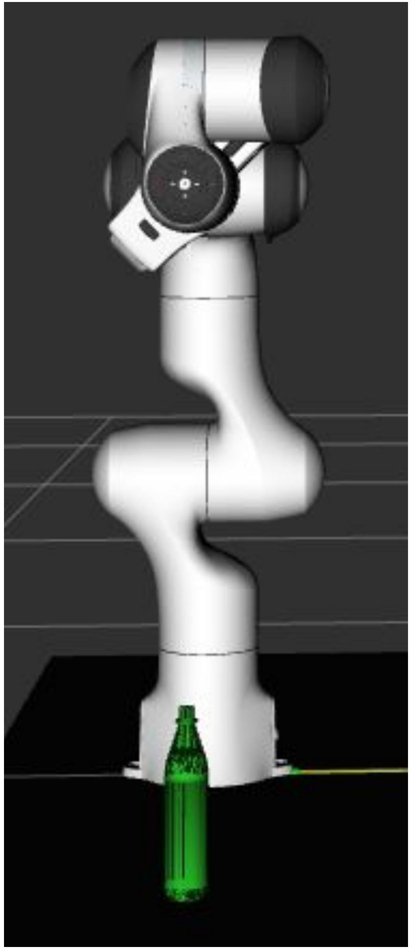
Implementation

Pouring Task

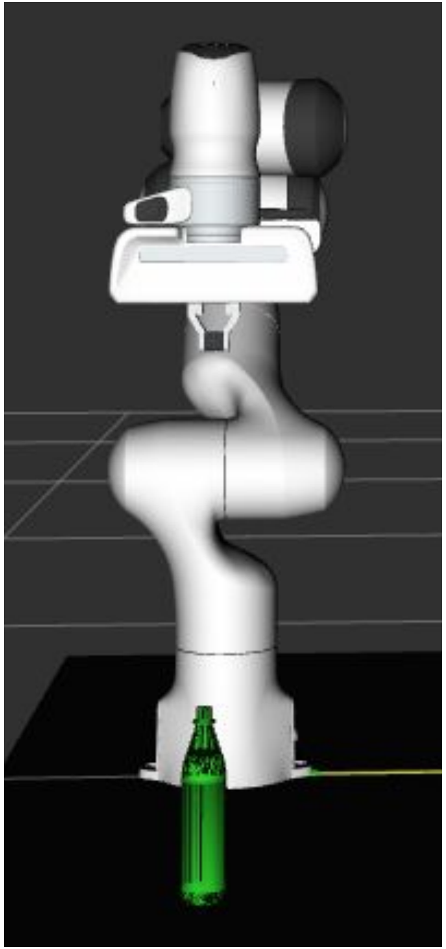


Creating Structure

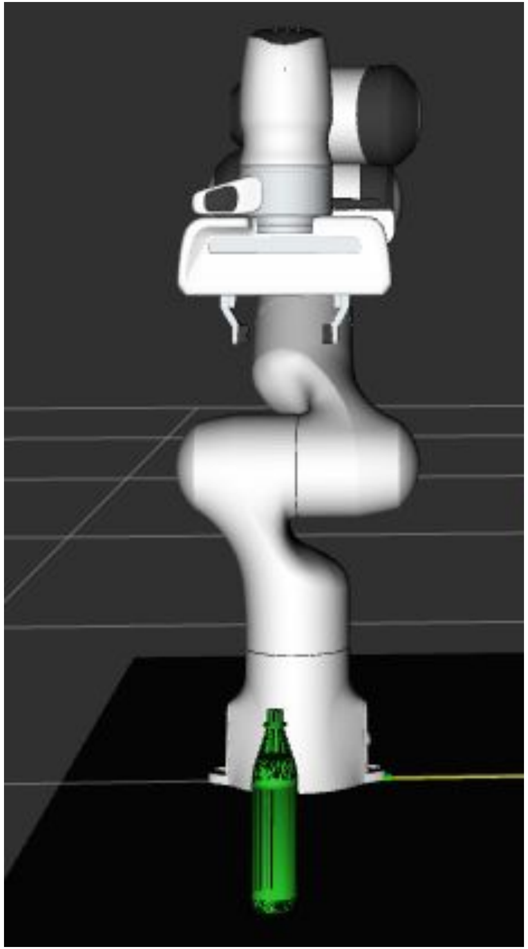




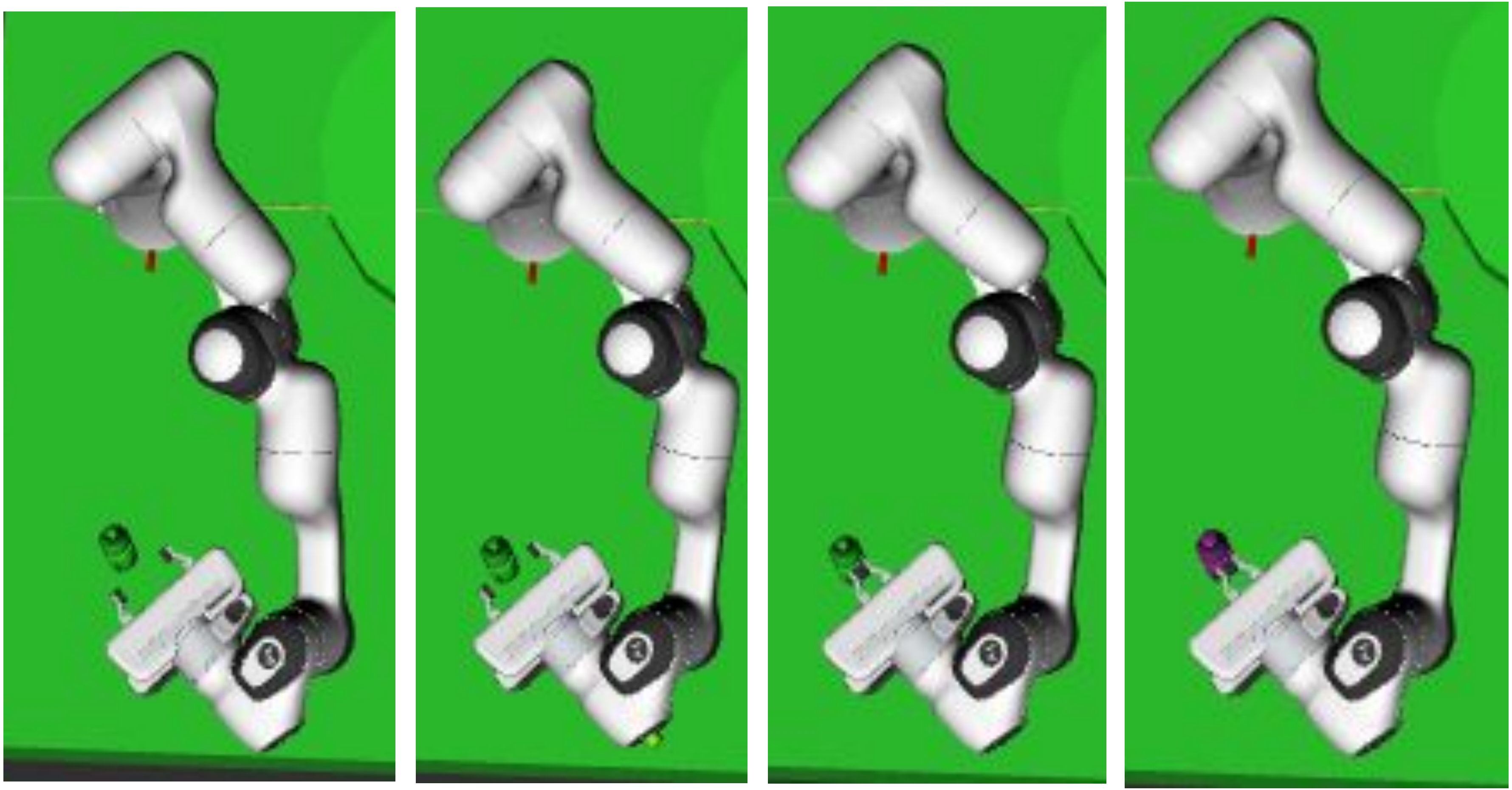
1. Current State



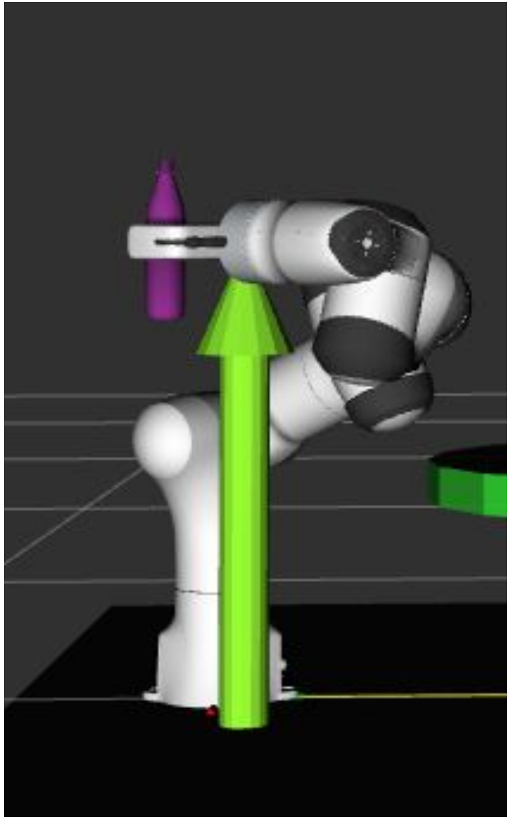
2. MoveTo Home



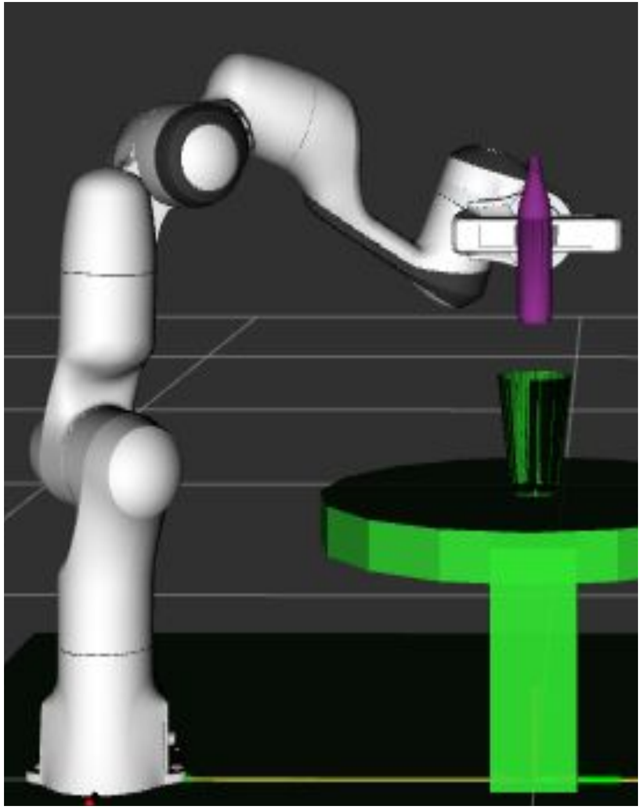
3. Open Hand



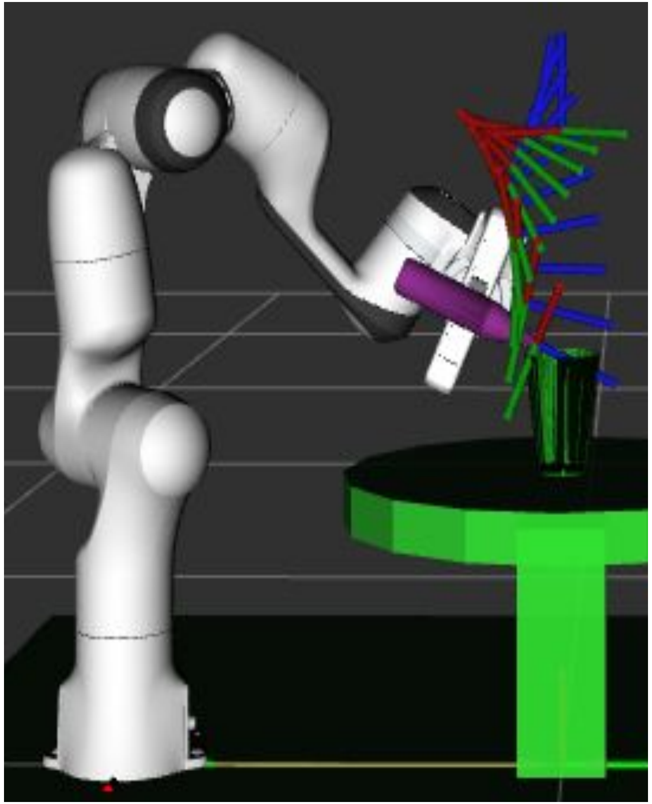
4. MoveTo Pick 5. Approach 6. Grasp 7. Attach



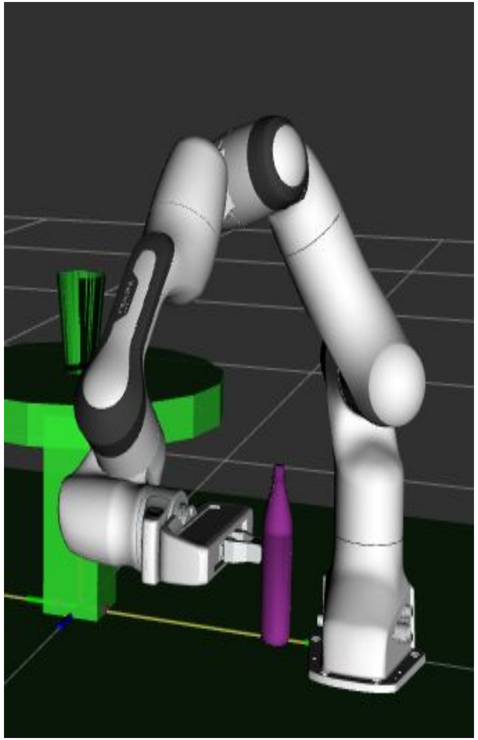
8. Lift

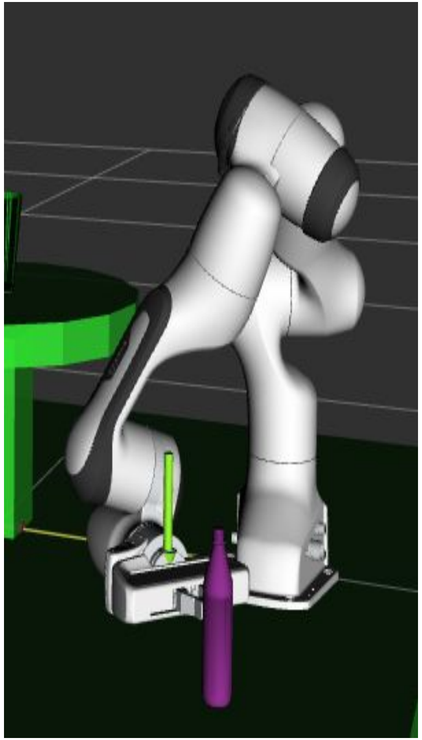


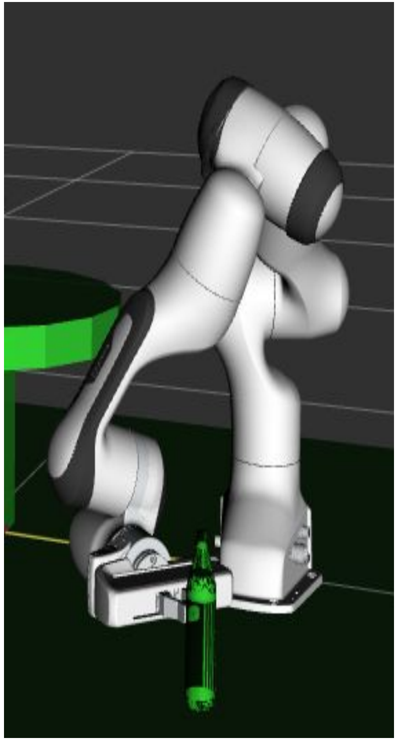
9. MoveTo Pre-Pour

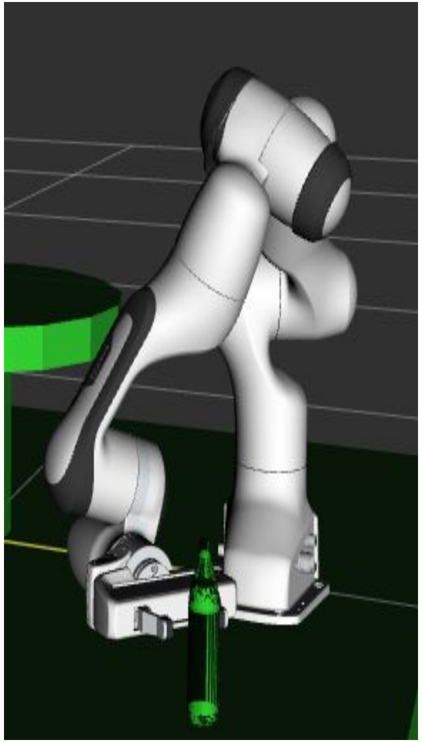


10. Pouring

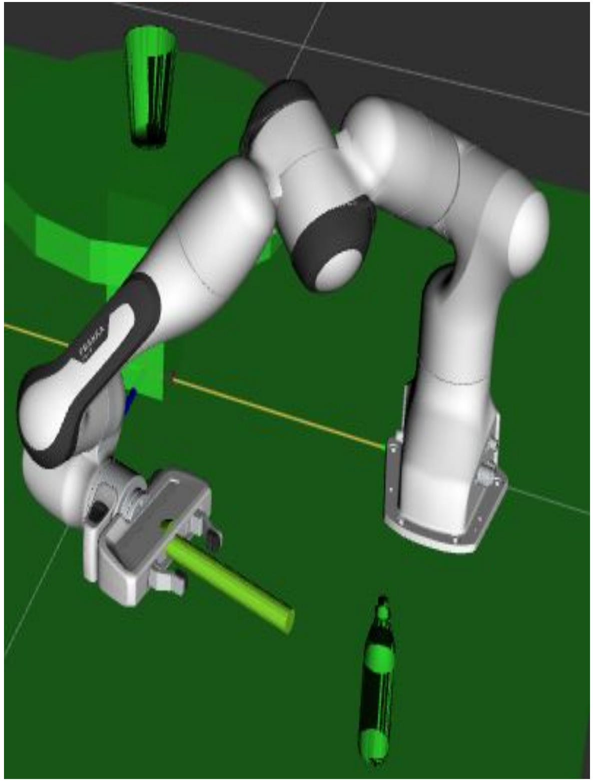




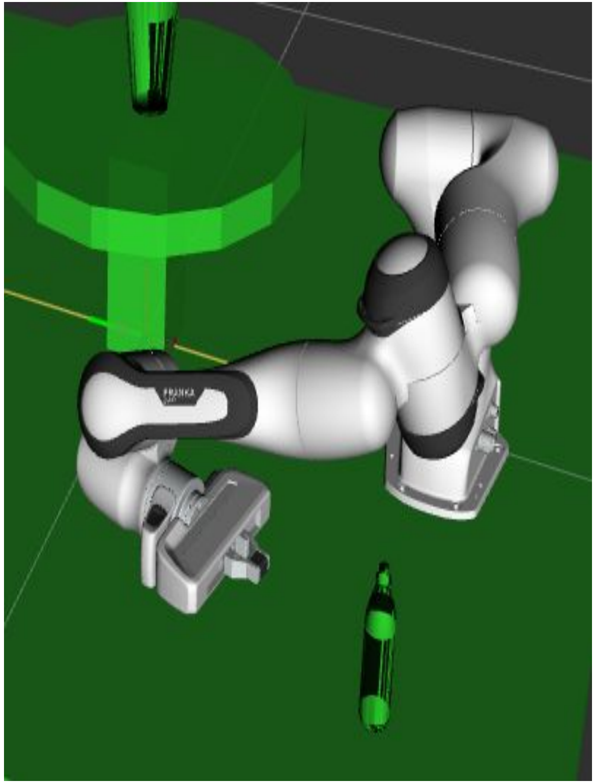




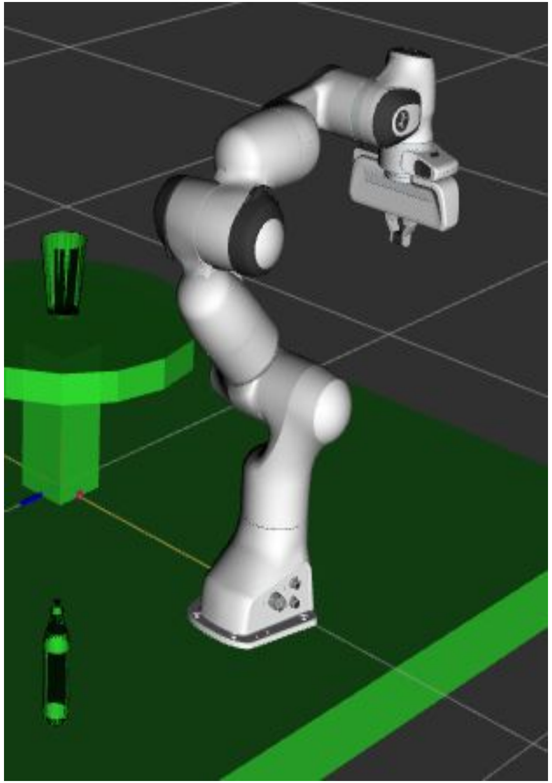
11. MoveTo Place **12. Lower** **13. Detach** **14. Open Hand**



15. Retreat



16. Close Hand



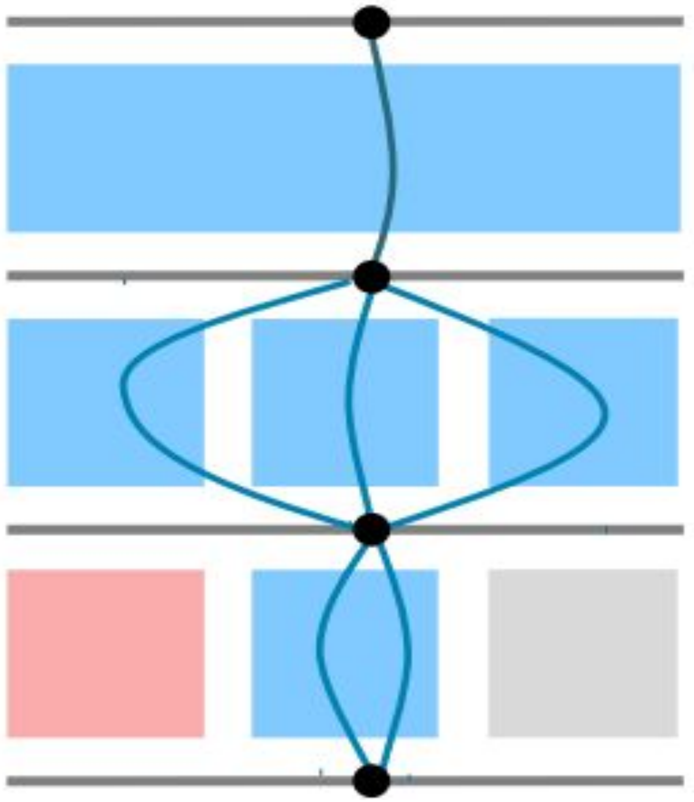
17. MoveTo Home back

Motion Planning Tasks		
pick_place_task	8	0
applicability test	1	0
current state	1	0
move home	1	0
open hand	1	0
move to pick	17	0
pick object	18	0
approach object	21	26
grasp pose IK	47	11
generate grasp pose	25	0
allow collision (hand,object)	47	0
close hand	47	0
attach object	47	0
allow collision (object,support)	47	0
lift object	19	28
forbid collision (object,surface)	19	0
move to place	10	0
place object	9	0
allow collision (object,support)	9	0
lower object	17	47
place pose IK	64	45
generate place pose	47	0
detach object	64	0
open hand	61	3
forbid collision (hand,object)	61	0
retreat after place	9	52
close hand	9	0
move home2	9	0

move home2	6	0
open hand2	6	0
move to pick2	2	0
pick object2	3	0
approach object2	3	5
grasp pose IK2	45	3
generate grasp pose2	150	0
allow collision (hand2,object2)2	9	0
close hand2	9	0
attach object2	9	0
allow collision (object2,support)2	9	0
lift object2	3	6
forbid collision (object2,surface)2	3	0
move to pre-pour pose2	8	0
pre-pour pose2	46	0
pose above glass2	9	0
pouring2	6	3
move to place2	3	0
place object2	3	0
allow collision (object2,support)2	3	0
lower object2	5	3
place pose IK2	18	0
generate place pose2	9	0
detach object2	9	0
open hand2	9	0
forbid collision (hand2,object2)2	9	0
retreat after place2	4	5
close hand2	4	0
move home	3	0
move home2	3	0

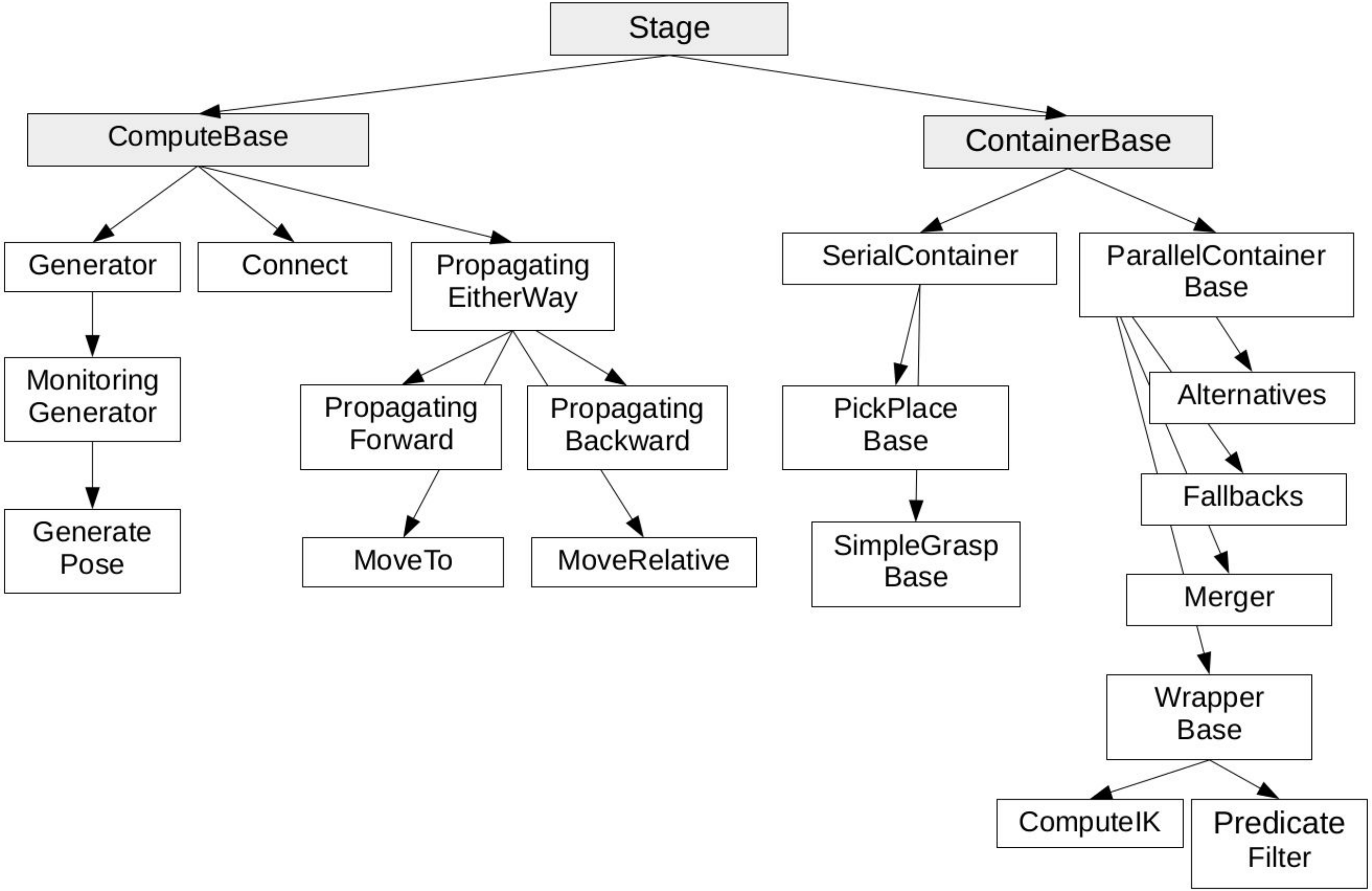
Total of 60 stages

Task level planning using MTC



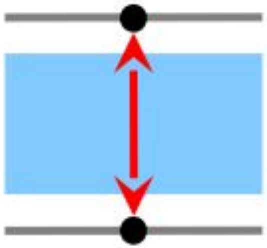
Reference: https://ros-planning.github.io/moveit_tutorials/doc/moveit_task_constructor/moveit_task_constructor_tutorial.html

Hierarchical Structuring



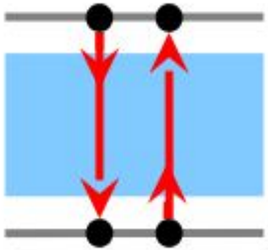
Reference: https://ros-planning.github.io/moveit_tutorials/doc/moveit_task_constructor/moveit_task_constructor_tutorial.html

Generator



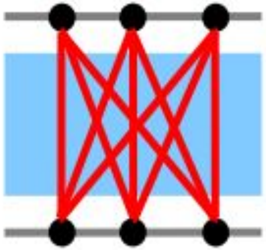
- Produces and propagates InterfaceStates to adjacent Stages
- E.g. IK generator

Propagator



- Receives an input InterfaceState, solves a problem and propagates the solution state
- E.g. MoveTo, MoveRelative

Connector



- Connects InterfaceStates of both adjacent stages
- E.g. Free-motion plan between start and goal states

Reference: https://ros-planning.github.io/moveit_tutorials/doc/moveit_task_constructor/moveit_task_constructor_tutorial.html

Contributions

Focusing

Rajendra Singh
iamrajee

Edit profile

Robotics | ROS 1&2, Moveit, MTC, Gazebo, OpenAI gym | CS@IIT Palakkad
UST Global, India
Rajsamand, Rajasthan
singh.raj1997@gmail.com
https://iamrajee.github.io/

- 18+ issues/Bugs
- 8+ Pull request

Overview | Repositories 70 | Projects 2 | Packages 0 | Stars 24 | Followers 11 | Following 73

Pinned — Customize your pins

iamrajee.github.io
Portfolio website
HTML

ros_catkin_ws
ros directory of raspberry pi(no. 38) raspbian
C++

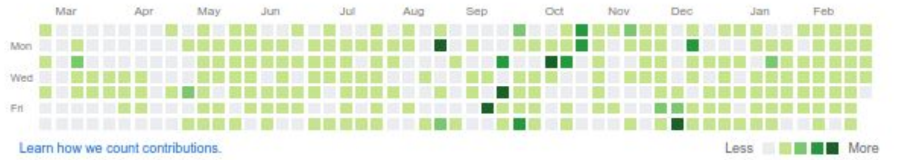
moveit_task_constructor
Forked from ros-planning/moveit_task_constructor
A hierarchical, multi-stage manipulation planner and state machine with user interfaces
C++

ros2_ws_moveit
ros2 workspace for moveit
CMake

src
Robot operating system
C++

ros2/ros2
The Robot Operating System, is a meta operating system for robots.
1.4k 269

927 contributions in the last year — Contribution settings

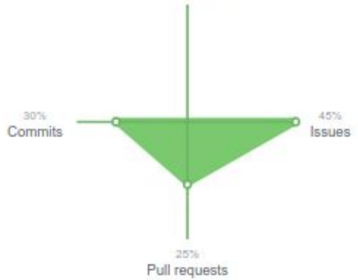


2020
2019
2018
2017

@ros-planning @TAMS-Group @ros2 More

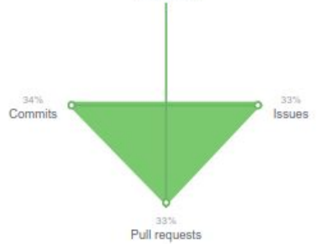
Activity in **ROS Planning**

Contributed to moveit_task_constructor, moveit.ros.org, moveit_tutorials



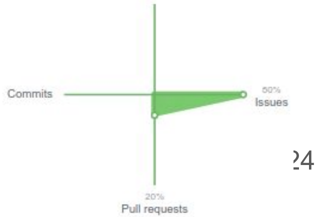
Activity in **TAMS, Dep. Informatics, Univ. Hamburg**

Contributed to mtc_pour



Activity in **ROS 2**

Contributed to ros2, ros2_documentation



rhaschke commented 10 days ago Collaborator

It's not possible to merge trajectories generated by a serial container.
Consider some substages in the serial container that just modify the planning scene (e.g. attaching/detaching an object, or modifying the ACM). How should we merge such a modification into another motion trajectory?

v4hn commented 2 days ago Member

> On Tue, Feb 25, 2020 at 12:27:02AM -0800, Rajendra Singh wrote:
> > To call your two execute_helpers ...
> Thank you I understood. Can we change this preempt behaviour of action goal?

This is a matter of changing the ExecuteTaskSolution capability, at least, to a general `ActionServer`.
This requires additional bookkeeping, probably a similar transition in general plan execution in MoveIt
and would basically "only" add support for your current use-case where you want to execute
independent controllers.

Of course, you're welcome to provide a pull-request that achieves this behavior, but the more
reasonable
solution for yourself might be to run two independent `PlanExecution` classes locally, or even execute
the subtrajectories of the solutions yourself
by sending them to the correct `FollowJointTrajectory` actions. This is of course not very elegant,
though...

rhaschke commented 3 days ago Collaborator

To call your two execute_helpers independently, you can just use two threads directly.
But, even if you manage this, I don't think, a single move_group node can handle two execution requests in parallel. As the corresponding capability relies on a `SimpleActionServer` the following doc applies:

> only one goal can have an active status at a time, new goals preempt previous goals based on the stamp in their GoalID field (later goals preempt earlier ones)

3. Conclusion

Progress Report

Single arm ✓✓	Multi-arm ✓✓	Parallelising Task ?	Satellite repair project ✘✘

Done:
Joint state goals
Cartesian goals
Pick Place task
Pouring task

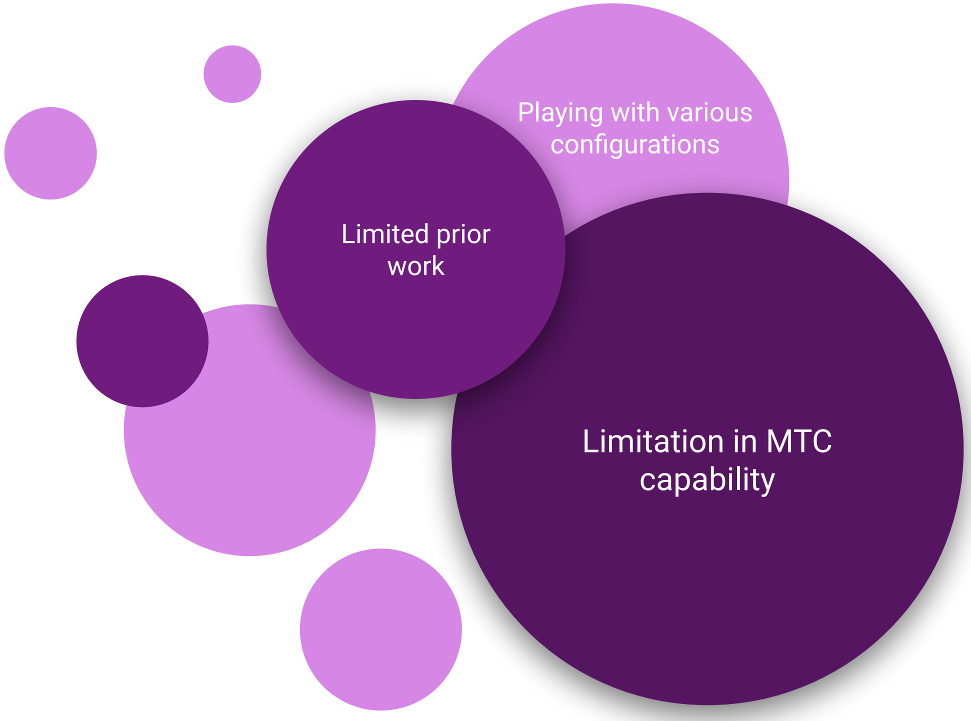
Done:
Joint state goals
Cartesian goals
Pick Place task
Pouring task

Done:
Merger, Alternative, Fallout, Multi task planning

In Progress:
Multi move_group, non preemptable goals

Future Work:
Four arm planning in more constrained environment, More complex task

Challenges


Playing with various configurations
Limited prior work
Limitation in MTC capability

THANK YOU!

Any Question?

```cpp
// --------------------- 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(hand_open_pose_);
    stage->setObject(object);
    stage->setAngleDelta(M_PI / 12);
    stage->setMonitoredStage(current_state);  // 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(grasp_frame_transform_, hand_frame_);
    wrapper->properties().configureInitFrom(Stage::PARENT, { "eef", "group" });
    wrapper->properties().configureInitFrom(Stage::INTERFACE, { "target_pose" });
    grasp->insert(std::move(wrapper));
}
```