

Rajendra Singh

B. Tech final year, Computer Science and Engineering, IIT Palakkad
Kerala, India

Email : singh.raj1997@gmail.com
GitHub : <https://github.com/iamrajee>

Add Support for MoveIt Task Constructor to MoveIt 2

Google Summer of Code 2020

- **Description:** The MoveIt Task Constructor(MTC) framework provides a flexible and transparent way to define and plan actions that consist of multiple interdependent subtasks. It draws on the planning capabilities of MoveIt to solve individual subproblems in black-box planning stages. Porting this to MoveIt 2 would also include porting MoveIt 1's MoveGroup interface to MoveIt 2.
- **Programming language Involved:** C++
- **List of potential mentors:** Robert Haschke, Henning Kayser
- **Duration:** 3 months (June - August, 2020)

Content

1.	About Me.....	03
2.	Motivation.....	03
3.	Experience with ROS.....	04
4.	Necessary changes while porting MTC to ROS2.....	05
5.	Prior Work.....	06
6.	Milestones.....	07
7.	Why me?.....	10


About Me

I'm currently in the final year of my Bachelor of Technology degree in Computer Science and Engineering from Indian Institute of Technology(IIT), Palakkad. I have a passionate interest in perception and motion planning. I started learning ROS in my second year at college and since then I have used it in my several projects(see [here](#)) relating to SLAM, Reinforcement learning based controller for drone, Swarm of two drones, Multi-arm manipulation task etc. Also, I'm proficient in C++ and have more than 4 years of working experience with it.

Motivation

I have been working on a multi-manipulator system using MTC for quite some time. I had faced some difficulties in concurrently executing the tasks. I had worked on some [solution](#) suggested by **Michael Görner** with limited success. Then I came across a multi-arm demonstration(see [here](#)) using moveit2 by **Acutronic Robotics** and I too wanted to do something similar but more complex(thus needed MTC) with a panda arm. This motivated me to port MTC and the related planning interface to ROS2.

I really liked this project as it matched my interest, and I have been driven to work on this in the next phase of my final year college thesis(see my previous presentation on multi-arm task planning using MTC [here](#)).



Experience with ROS

● Involvement in Open Source Project

- **ros-planning**: moveit, moveit_task_constructor, moveit2, moveit.ros.org, moveit_tutorials (eg. [#124](#), [#125](#), for more see [here](#))
- **ros2**: ros2, ros2_documentation (e.g [#555](#), for more see [here](#))
- **TAMS-Group**: mtc_pour (see [here](#))

Many of these involvements were limited to issues and minor enhancements. However, I would love to spend more time on this, discuss ideas at discourse and be an active part of the open-source community.

● Experience with ROS distros and their relevant projects

- **Kinetic**: I started to learn ROS with kinetic and spent most of my time with it. See projects [here](#).
- **Melodic**: I started learning MTC with melodic and played extensively with its several configurations and demos. I also encounter many issues(e.g. [#122](#), [#131](#), [#138](#) and many more), followed their discussions(e.g. [#1835](#), [#1899](#)) to get involved with the community. See [this](#) repo for relevant projects.
- **Dashing**: I first exposed to ROS2 back in July 2019 with dashing. Here I spent some time figuring out its differences with ROS 1 and went through the ros2 tutorial. Then I explored moveit2 and observed the new changes. As many of the moveit2 packages were not ported by then, I was discouraged to move ahead with it and continued working with moveit and MTC.
- **Eloquent**: Once I became more comfortable with ros2 I decided to jump to newly released non-LTS ros2 eloquent. As ros org tutorials were greatly updated by then, I went through entire ros2 tutorials, Concepts, Troubleshooting, Contributing, Features Status, Roadmap once again. I've also gone through previous year ROS2 related ROSCon presentations(also added a few of them [here](#)). I started experimenting with moveit2 demo_run_cpp. I wanted to do more with it but move_group and related packages were not ported by then. I started to look at previous year GSoC ros2 porting related projects and contacted one project member to learn more about their experience. Then I started porting some of moveit2 packages and MTC, and continuing the same now. See [this](#) repo for relevant projects.

Necessary changes while porting MTC to ROS2

- **Package manifests**

- Changing package.xml from format 1 of the package specification to newer format wherever applicable.
- Changing the dependencies names if it is named differently in ros2.

- **Message, service, and action definitions**

- As some primitive types like duration and time which were builtin types in ROS 1 have been replaced with normal message definitions. Thus we should use them from the builtin_interfaces package instead.

- **Build system**

- Modifying the CMakeLists.txt to be used with ament commands instead of catkin.
- Using rosidl_generate_interfaces to generate msgs, srv and action instead of using generate_messages, add_message_files etc.
- Removing any occurrences of the devel space variables e.g CATKIN_DEVEL_PREFIX.
- Adding gtest and linters accordingly (would also need to update package.xml to add corresponding dependencies).

- **Source code**

- As namespace of ROS 2 messages, services, and actions use a sub-namespace (msg, srv, or action, respectively) after the package name, thus changing the includes and classes accordingly.
- Also changing the included filename from CamelCase to underscore separation and changing include type from *.h to *.hpp
- Changing the initialisation of msgs as Shared pointer types are provided as typedefs within the message structs.

- **Launch files**

- We have to adapt to a lot of changes in the launch file. This would mainly be for MTC/demo package launch files.

Prior Work

Initially I started with following the full commit history of MTC to build it for ros2 eloquent(see [here](#) in repository) from scratch by making appropriate modifications as and when required. This way I thought I could better understand the reason for a particular design choice of MTC. So far this involved:

- A. Creating a rough layout of task and subtask class.
- B. Added first subtask i.e subtask::CurrentState
- C. As the gripper required a MoveGroupInterface class, I then [ported](#) the move_group package.
- D. Later [ported](#) move_group_interface package and solved non-matching function errors([#179](#)) by making appropriate modifications.
- E. Added gripper subtask and their relevant function like compute etc.
- F. Created draft of subtask::GenerateGraspPose
- G. Improved GenerateGraspPose subtask by adding time management, multiple IK solutions, check collisions, angle delta and grasp offset.
- H. Then [ported](#) planning_scene_interface to add objects in the planning scene.
- I. Creating a test_gen_grasp_pose and tested the working of subtask::GenerateGraspPose class.
- J. Added first implementation of cartesian_position_motion
- K. Improved gripper subtask by allowing collisions with grasped objects
- L. Then tried improving cartesian_position_motion by adding beginning and end inference but later encountered an issue([#180](#)).
- M. See more at this [blog](#).

Earlier I had planned to similarly continue building the MTC step by step. However later, following **Michael Görner's suggestion** I would rather focus on porting includes/message types and the rviz related introspection code directly from the present architecture of MTC.

Milestones

1) Buffer time (April, 2020)

In this buffer time I would reiterate through this report, adapt to befitting changes in plan(rough draft of which is presented below), freeing myself from other college responsibilities to focus on this project(In fact I would be totally free after April 25, 2020).

2) Community Bonding (May 4, 2020 - June 1, 2020)

In this one month duration I would interact and learn more about the PickNik community. I would have an in-depth look at MoveIt Code Style Guidelines, Handling Pull Requests Guidelines, Contributing to MoveIt Guidelines and MoveIt Roadmap 2020 etc. I would raise and clear some of my doubts relating to MTC design choice, commit history and tutorials.

3) Week 1

In the first week, after thorough discussion with mentors and intend to:

- A. Lay down a proper plan which I would follow later during porting MTC e.g. order of porting packages, setting deadlines for each step etc.
- B. Decide upon necessary modification in current MTC features and if any additional features need to be introduced.
- C. List down other necessary things I need to take care of later during porting.
- D. Finalise on what needs to be accomplished in given duration and what is out of current scope.

4) Week 2

- A. I would first finish porting required and not yet ported moveit2 packages. i.e move_group and other planning_interface e.g. move_group_interface, planning_scene interface as we would require this in later stages.

5) Week 3-4

During this time period, I intend to start with porting core functionality of MTC, as decided in 3) A

- A. I would start with porting corresponding CMakeList, msgs and srv from msgs package.
- B. I would then port core/includes e.g starting with task.h, storage.h, stages.h etc. and then moving to porting stages i.e current_state, moveTo, moveRelative etc. and some solvers like cartesian_path.
- C. Then would porting planner_interface.cpp and simple planner like solvers cartesian_path etc.
- D. Here I would also start porting minimal cpp nodes(task, stages etc) and create simple tests to validate newly ported simple stages.
- E. As we will be hitting **phase I evaluation** after 4th week, here I intend to create a simple cartesian demo and demonstrate its working.

6) Week 5-6

During this time period, I would finish porting core functionality of MTC, which were left out in the last step.

- A. I would first finish porting remaining stages(e.g. Predicate_filter, simple_grasp, etc) and solvers. Also introducing some new stages like pouring(**Need feedback:** Should I do this or not? If then when would be the right time to do so, now or in later steps?).
- B. Then port remaining src files like container.cpp, merge.cpp etc.
- C. At the sametime I would also be porting some test files to test the working of the newly ported stage or container.

7) Week 7-8

- A. During this time period, I would port the remaining other supporting package of core e.g visualization, rviz_marker_tools and capabilities. Of Course some part of them would have already been ported in previous stages as and when required.

- B. As we would be approaching **phase II evaluation**, I would prepare for more matured demonstration. It would involve pick and place, collision checks, and planning scene visualisation on rviz.

8) Week 9

During this period I would revisit the MTC core and other dependent packages and port the remaining file or portion of code as some of that would have been commented out earlier as that would be dependent on later steps.

9) Week 10

By this time If all of the above packages finish porting then I would port the MTC demo package.

- A. This would involve porting basic cartesian and modular planners.
- B. Porting basic pick and place demo(and corresponding node and header file)
- C. I'm also planning to create a simplistic pouring [demo](#) too using a panda arm.
- D. If time permits and If it's okay to do so, I would like to port and merge some demos from mtc_demos repository to moveit_task_constructor/demo package(**Need feedback:** Should I or not?).

10) Week 11-12

- A. During this period, I'll wrap up the work, so we can have MTC ready to be released(**Need feedback and more information on this**).
- B. I would also try to utilise this time period, to frame new possible improvements, features, and create a roadmap for the future, and work on recognising features beyond the scope of current release by building upon work done in 3) D
- C. As we almost hit the **final evaluation** timeline, I would make sure all my work is well documented. I would recheck all code to see if everything is well commented and add more details as and where required.

I strongly feel that there is a need for a detailed tutorial about MTC on moveit org. However, I doubt if it could be done in this limited time. Thus, if allowed I could continue working on this post GSoC.

Why me?

As I am really passionate and determined to do this(have been seeking to know about this MTC update even before PickNik announced GSoC, see [here](#) and [here](#)) and also I'm working on this as my final year thesis, I believe proper guidance would really help me to contribute something important to the open source community.

I have a sound understanding of the build history of MTC and have worked with MTC in my projects previously. Thus, I believe I would be the ideal candidate to work on the MTC porting project as I have already started working on it.

If you decide to not to go ahead with me for some reason I would still appreciate your guidance in this project as I'm really passionate to work on it.