Add Support for MoveIt Task Constructor to MoveIt 2

Google Summer of Code 2020

Rajendra Singh
Palakkad, Kerala, India

Email  :  singh.raj1997@gmail.com
GitHub :  https://github.com/iamrajee
Web    :  https://iamrajee.github.io/

Overview

- **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. This project is about porting it to MoveIt2 which would also include porting MoveIt 1’s MoveGroup interface to MoveIt 2.
- **Programming language Involved**: C++
- **Mentors**: Robert Haschke, Henning Kayser
- **Duration**: June 2020 - August 2020
About Me

I’m currently in the final year of my Bachelor of Technology degree (2016-2020) 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 relating to SLAM, Reinforcement learning-based controller for the drone, Swarm of two drones, Multi-arm manipulation task and many more (see here).

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. Then I came across a multi-arm demonstration (see here) using moveit2 by Acutronic Robotics and I too wanted to do something similar with a panda arm but bit more complex (thus needed MTC). 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 (eg. #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 (eg. #122, #131, #138 and many more), followed their discussions (eg. #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 motivated to port the same.
  ○ **Eloquent**: Once I became more comfortable with ros2 I decided to jump to newly released 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 the MTC packages 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 built-in 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 a 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
    - e.g  
      ```
      #include <moveit_msgs/srv/GetPlanningScene.h>
      #include <moveit_msgs/srv/get_planning_scene.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 the 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 stage i.e 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 in libraries.
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 GenerateGraspPose class.
J. Added the 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 the beginning and end inference but later encountered an issue(#180).
M. See more steps here in 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

Timezone: GMT +5:30
Planned Timings: 0900-1500 IST or 1900-0100 IST (Will work in other slots if unavailable in first one)

1) Before Community Bonding

A. I would reiterate through this report, adapt to befitting changes in plan (rough draft of which is presented below) and free myself from other responsibilities to focus on this project.
B. I would keep exploring MTC and moveit2 with priority given to complex issues along with searching for bugs.

2) Community Bonding

A. In this one month duration, I would interact and learn more about the PickNik community.
B. 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.
C. I would raise and clear some of my doubts relating to MTC design choice, commit history and tutorials.
D. Keep the discussion open on the MTC for suggestions/feedback related to the project.
E. Finalize the design/features by discussing with the community especially @rhaschke and @v4n.
F. Set up a blog for posting weekly progress along with different ideas, challenges, etc.

3) Week 1: Getting Started!

In the first week, after a thorough discussion with mentors and intend to:
A. I would set up the environment and start coding!
B. However, I would still be open to any feedback on project plan e.g. order of porting packages, setting deadlines, any additional features etc.
C. In the first week, I would be experimenting a lot! Use different code practises, work hours, communication mode and finalize the best practices to stick on later.
D. Also, I would be observing changes and taking frequent feedback on coding style practice.
E. Give final thought on what needs to be accomplished in given duration and what is out of current scope.

4) **Week 2 : Porting Moveit2 Packages**  
   **June 8, 2020 - June 14, 2020**

   A. Firstly, I would port move_group and move_group_interface as [this](#) and [this](#). And adapting to necessary modification.
   B. Then I would port the remaining required and not yet ported moveit2 packages. e.g planning_scene_interface which would also be required in later stages.

5) **Week 3-4 : Started Porting MTC Cores Package**  
   **June 15, 2020 - June 28, 2020**

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

   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 the task.h, storage.h, stages.h etc. and then moving to port stages i.e current_state, moveTo, moveRelative etc.
   C. Then I would porting planner_interface.cpp and simple solvers like 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.

6) **Week 5 : Phase I evaluation**  
   **June 29, 2020 - July 3, 2020**

   Here I intend to document the work done so far. Also, prepare deliverables for Phase I evaluation i.e create a simple demo(e.g cartesian) and demonstrate its working.

7) **Week 6 : Finish Porting MTC Cores Package**  
   **July 6, 2020 - July 12, 2020**
During this time period, I would finish porting the remaining 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 same time, I would also be porting some test files to test the working of the newly ported stage or container.

8) **Week 7-8 : Porting MTC Supporting Package**  
   **July 13, 2020 - July 26, 2020**

   A. During this time period, I would take my time to port moveit_task_constructor_visualization. I would modify and port the corresponding rviz related introspection code.

   B. Then I would port the remaining other supporting packages i.e rviz_marker_tools, capabilities etc. Of Course, some part of them would have already been ported in previous stages as and when required.

   C. I also expect to spend some more time understanding this well.

9) **Week 9 : Phase II evaluation**  
   **July 27, 2020 - July 31, 2020**

   Here I intend to document the work so far. Also, prepare deliverables for Phase II evaluation i.e preparing more matured demo than the last time. It would involve pick and place, collision checks, and planning scene visualisation on rviz.

10) **Week 10 : Creating Demos**  
    **August 3, 2020 - August 9, 2020**

    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?).

11) Week 11 : Finish Any Leftover Work  

A. During this period I would revisit the MTC core and other dependent packages and port the remaining file or portion of code which was commented out earlier (if any) owing to its dependencies on later steps.
B. I'm also keep this intentionally as a buffer time, to meet unexpected delays in previous deadlines.

12) Week 12 : Testing and Documentation  

A. Create and merge any leftover PRs.
B. Ask for feedback from communities, friends etc and refine the work according to the response.
C. I'll wrap up the work, so we can have MTC ready to be released (Need feedback: More information on a release timeline for MTC and would it be dependent on moveit2 releases).
D. 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) E

13) Week 13 : Final evaluation  

Here 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 if required. Re-ensure all deliverables working perfectly. I would also create a final blog to summarize this work. Finally, we have everything ready to submit!

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, I will continue this post-GSoC. (Need feedback: Any suggestions or advice you have?)
Questions

1. **Why are you the right person for this task?**

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 valuable to the open-source community.

I have already started porting MTC to ROS2 and I have a sound understanding of the build history of MTC. Also, I have worked with MTC in my previous projects. Thus, I believe I would be the ideal candidate for this task.

Also for some reasons, if you decide to not to go ahead with me still I would much appreciate your guidance in this regard as I’m really interested to work on it.

2. **Describe your development experience. Include all the technologies you have used for projects.**

   Experienced and proficient in C/C++, Python, CMake, XML
   Favourite Language is C++
   Have worked with projects of various levels of difficulty. E.g Compilers, Parallel Programming etc.
   Good knowledge of version control systems. Have experience with git.
   Comfortable with Linux and currently using Ubuntu 18.04 bionic.
3. Describe your interactions with our community so far.

I have come to know most of the community members through last year MoveIt Workshop. I most interacted with Robert Haschke(@rhaschke) and Michael Görner(@v4n) over Github. I very recently interacted with Jafar Abdi(@JafarAbdi) regarding MTC porting related errors. Also, I interacted with Mark Moll(@mamoll), Felix(@felixvd), Henning Kayser(@henningkayser) through issue/PR on Github.

I interacted with Dave Coleman(@davetcoleman) through his discourse post, tweets and very recently in MoveIt WG meeting(March 26, 2020). Also here meet Martin Pecka, Tyler Weaver, Simon Schmeisser and Omid.

4. What is your preferred method of contact and how should we reach you with it?

Email : singh.raj1997@gmail.com
LinkedIn : https://www.linkedin.com/in/rajendra-singh-6b0b3a13a/
Skype : singh.raj1997(Name), +91 7073091997(Caller ID)
Discourse : https://discourse.ros.org/u/rajendra_singh/summary

Any mode of contact is fine, but I prefer contact through email.

5. Please tell us about your availability for this project? Include work hours per week. Also include commitments during the program(Include any and all holidays, vacations, travel, exams, classes, etc.)

I reviewed the whole GSoC timeline and I would be able to devote approximately 40-50 hours every week to GSOC. I will be devoting my full time to this project. I have no obligations after April and would be devoting all my time to GSOC. I will be reporting my work over daily scrum mails on the mailing list. I would strive to be regular, and sincere with my scrum and daily updates as I understand that selection in this project will require a serious commitment and 100% devotion from my side.
6. What are your expectations from GSoC? How would it help you in future endeavours?

GSoC is an amazing opportunity for students like me to work on a high-level project that they are most interested in. I will get a chance to interact with an awesome moveIt community which will include lots of learning and fun.

According to me, open-source development can be life-changing that will inspire exponential growth, a lot of learning and constant development. Its results could be remarkable, profound and enduring. GSoC programs have the capacity to transform and connect individual groups, organizations and communities. It will be very fulfilling to know that I’ve directly contributed to an organization like PickNik’s growth and development. Thus, I see participating this year in GSoC with MoveIt as a great opportunity to make use of my good technical, leadership and communication skills to develop a next-generation project and experience the guidance of mentors from MoveIt. ROS Development has always been my plus points and I have received great recognition for the same. I will be privileged to receive mentorship from MoveIt maintainers so that I can improve myself as a developer as well as learn some important skills and tips from the honourable mentors of MoveIt.

7. Provide any references who have known you working with MoveIt and can comment on your skills.

- **Dr Santhakumar Mohan**, Robotics and Control, Professor @IIT Palakkad
- **Dr Chandra Shekar Lakshminarayan**, AI, Professor @IIT Palakkad
- **Mr Girish Kumar**, Director, Innovation Labs, GadgEon Smart Systems, Kochi, India
- **Mr Ashok Nair**, Director, Innovation Labs, UST Global, Thiruvananthapuram, India

Thanks for your time. Hoping for a great project by the end of GSoC!