Bridging the Gap between the Open-source Task-Space Constraint-Based Control Framework and Real-World Human-Robot Interaction Applications

Anastasia Bolotnikova[1,2]

***Abstract*— Real-world robotics applications require the use of affordable, mass-produced robots. At the same time, robust performance in real-world settings is still a research problem, which requires cutting-edge developments from academia. We present the open-source software toolkit that bridges the gap between advanced constraint-based Quadratic Programming task-space motion control framework, developed in academia, and the affordable and mass-produced robots widely used in a variety of real-world applications, produced by SoftBank Robotics Europe. We describe the developed tools and outline how they facilitate development of controllers for real-world applications focusing on assistive human-robot interaction.**

I. Introduction

An advanced and powerful Quadratic Programming (QP) task-space control framework [1], called `mc_rtc`, is now available in opensource[1]. It is developed in the research community and oriented towards cutting-edge technologies in constraint-based QP task-space robot(s) motion control. Originally, `mc_rtc` was mainly used with high-cost research platforms. As the framework gained maturity, it became evident that other robotics platforms can benefit from its functionalities; now the control framework is robot-independent.

However, to enable `mc_rtc` framework to compose QP objectives and constraints for the motion control of any new type of robot, a robot description package needs to be provided and a robot module software component needs to be developed. These components allow to take full advantage of the `mc_rtc` framework functionality for the new type of robot in simulation. In order to enable framework users to control a real robot of a particular type, an interface must be developed to allow communication between the `mc_rtc` and the robot's low-level controllers, sensors and devices.

The aim of this work is to present the open-source software interface, called `mc_naoqi`[2] (Sec. II), for executing `mc_rtc` controllers on widely used SoftBank Robotics Europe (SBRE) robots [2], [3]. We also open-source robot description packages and modules (Sec. III) and sample controllers (Sec. IV). Finally, we demonstrate and discuss how the developed tools facilitate development of controllers for real-world human-robot interaction (HRI) applications (Sec. V). Fig. 1 shows an overview of the developed tools and their interconnections, described concisely in this document and in more detail in our recently submitted work [4].

[1]SoftBank Robotics Europe, Paris, France
[2]University of Montpellier–CNRS LIRMM, Interactive Digital Humans, Montpellier, France

[1]`https://jrl-umi3218.github.io/mc_rtc`
[2]`https://github.com/jrl-umi3218/mc_naoqi`

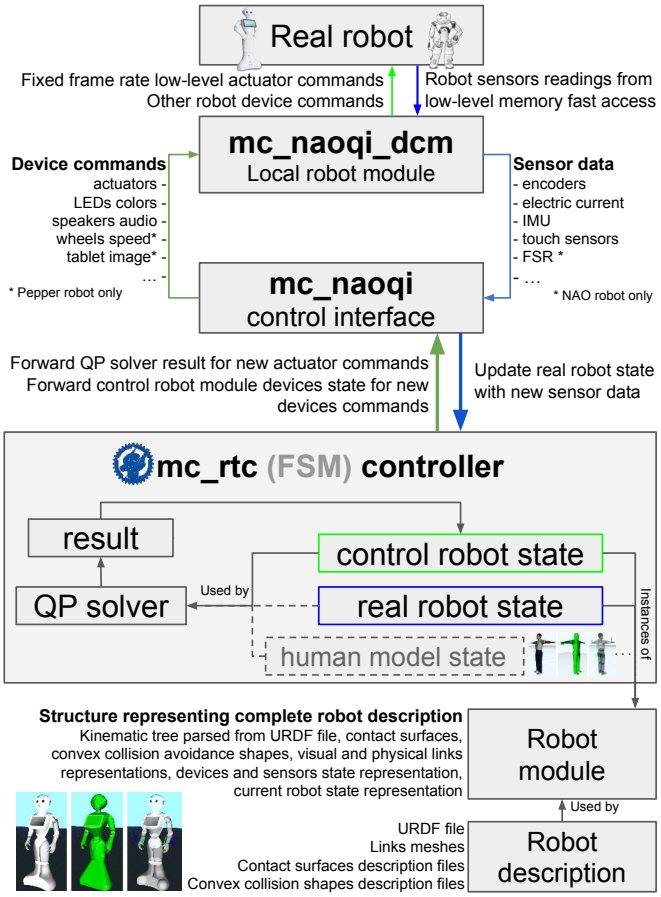


Fig. 1: `mc_naoqi` interface enables communication between SBRE humanoid robots and `mc_rtc` control framework. It can be used to steer the robot behaviour in HRI applications.

II. Control Interface

Fig. 1 illustrates the role of the `mc_naoqi` interface as a communication layer between `mc_rtc` control framework and NAOqi operating system running onboard SBRE robots.

The `mc_naoqi` interface is forwarding fixed frame rate control commands from the QP solver of `mc_rtc` controller to the onboard low-level robot actuators control. For user-friendly and interacting humanoid robots, it is highly beneficial to endow `mc_rtc` controller with the functionality to also forward other device commands, such as sentence to play from the speakers, desired tablet screen image or eye led color, from the `mc_rtc` controller to the robot devices via `mc_naoqi` interface. This functionality allows `mc_rtc` framework users to develop controllers which can provide a richer interaction experience for HRI applications.

The mc_naoqi interface is also responsible for getting the most up-to-date sensor readings from a robot low-level memory in real-time and forwarding sensor measurements in a suitable form to the mc_rtc controller real robot state representation as a feedback. This way, the task-space QP controller keeps track of the real robot state and can use it to perform closed-loop QP control computations.

Besides the encoder values, force sensor and IMU measurements, mc_naoqi interface also forwards to the mc_rtc controller an electric motor current and touch sensor readings. For HRI applications, the touch sensor readings are especially beneficial to be forwarded to the controller, as they allow to detect when a human touches the robot. Such a signal can be used inside the mc_rtc controller logic to trigger an appropriate reaction of the robot to the touch.

A customized local robot low-level module, called mc_naoqi_dcm[3], is cross-compiled for NAOqi OS and is set to run onboard the robot to read sensor values and set device commands via Device Communication Manager (DCM) every 12 ms. A fast access to the low-level robot memory is initialized when mc_naoqi_dcm starts to run on the robot. This allows to read a predefined set of sensor values from robot memory in the fastest way.

III. Robot Description and Module

To control any robot with mc_rtc framework, a basic description of this robot needs to be provided. Such robot description includes robot kinematic tree, dynamic properties of the links, description of robot contact surfaces (i.e. covers), and convex anti-collision shapes. With this work, we release the robot descriptions for NAO[4] and Pepper[5] robots.

As shown in Fig. 1, a robot description is used by a *robot module* to create a structure that provides a complete description of the robot: kinematic tree parsed from URDF files, visual and physical representations of robot links, surfaces attached to the robot bodies, sensors and other devices, strictly convex hulls and primitive shapes for collision avoidance, etc. The instance of this structure is used by the mc_rtc framework, as control robot state representation, to formulate the QP objectives and constraints. We make the robot modules for NAO[6] and Pepper[7] publicly available with this work. For fast prototyping and experiments, the robot description and module can easily be augmented with any new robot hardware elements, e.g. new onboard camera.

The Pepper robot module exploits a generic robot devices feature of mc_rtc, which allows to add any kind of custom device representation as part of the robot module. Currently implemented devices in the Pepper robot module are loudspeaker, visual display and touch sensor. An example of robot specific tasks and constraints are also present in the released robot module and can serve as an example of implementation of such custom elements.

[3] https://github.com/jrl-umi3218/mc_naoqi_dcm
[4] https://github.com/jrl-umi3218/nao_description
[5] https://github.com/jrl-umi3218/pepper_description
[6] https://github.com/jrl-umi3218/mc_nao
[7] https://github.com/jrl-umi3218/mc_pepper

IV. Sample Controllers

To facilitate of writing a new mc_rtc controller, especially for new potential users of the framework, we provide a basic sample *PepperFSMController*[8] (Fig. 2). It can be used as a starting point for new controller development or as an example of how similar projects should be implemented.

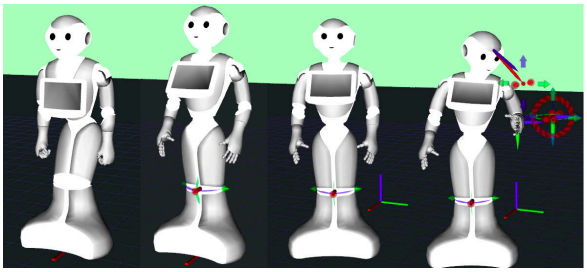


Fig. 2: RViz scenes of sample FSM Pepper controller states.

On the topic/withHumanModel branch of this project, a multi-robot QP (MQP) feature is exploited by adding a human model and its state as part of the controller (recall Fig. 1). A human model is integrated into mc_rtc exactly the same way as any other robot model, by providing a description[9] and implementing a corresponding software module[10]. This MQP controller can be used to develop and simulate a wide variety of HRI scenarios (e.g. Fig. 3).

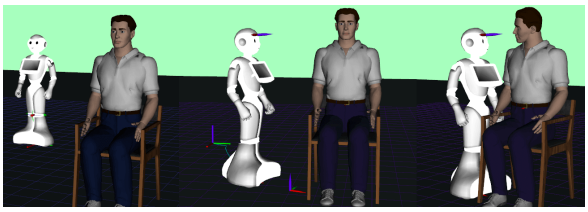


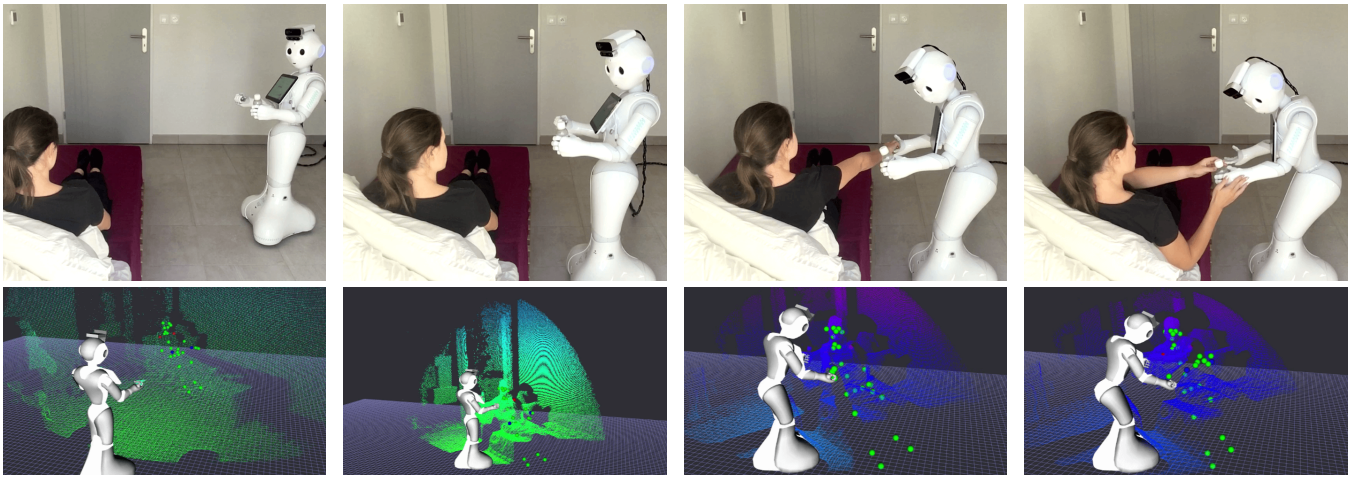
Fig. 3: Simulated *NavigateToHuman* state of a sample MQP mc_rtc FSM controller with a human model included.

V. Real-world applications: assistive HRI

In our recent work, we have showcased how the developed tools, described in the current work, can be used to enable Pepper robot to perform autonomous initiation of human physical assistance [5]. The controller code is publicly available and it shows how our developed software components allow to efficiently create complex controllers for rich, intuitive and efficient sensor-based HRI. This includes closed-loop navigation toward human, verbal, visual and body language communication, and physical interaction.

Furthermore, parts of developed controllers can easily be reused, which allows for rapid development of controllers for new HRI applications. For instance, we demonstrate in a new demo how a Pepper robot performs autonomous medicine delivery (Fig. 4). For this new controller, the code for the closed-loop navigation toward a human was directly reused from the previous work. Development of other controller parts only took a few days for an experienced user.

[8] https://github.com/jrl-umi3218/pepper-fsm-controller
[9] https://github.com/jrl-umi3218/human_description
[10] https://github.com/jrl-umi3218/mc_human



(a) Navigation to the human (b) Verbal communication (c) Human takes the pills (d) Human takes the bottle

Fig. 4: Experiment screenshots (top), perceived robot state, Azure Kinect point cloud and human body markers (bottom).

Fig. 4 shows excerpts screenshots from the experiment video. In the bottom row images, the scenes are visualized in RViz. Note, that additional hardware, namely Azure Kinect, which is used for human state feedback, and RealSense camera, which is included in the robot prototype, but not used in this experiment, are easily included in the robot description (Fig. 5) and processed by the robot module, and therefore are also included in the QP problem formulation (e.g. for more accurate robot center of mass computation). Full experiment can be seen in the accompanying video.

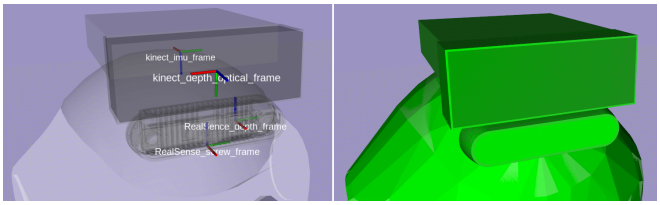


Fig. 5: Extra hardware included in the prototype Pepper robot

Once the robot reaches a position nearby the person, it communicates verbally its intention to pass the medicine (Fig. 4b). Then it proceeds to open its right gripper for the person to take the pills (Fig. 4c). The passing of the bottle with liquid is arranged with the help of the robot module feature, described in Sec. III, that allows to forward robot tactile sensor data to the `mc_rtc` controller, which then triggers robot left hand gripper to open slightly to allow the person to get out the bottle more easily (Fig. 4d).

Fig. 6 shows the progress of the closed-loop PBVS task, that uses Azure Kinect body tracking for feedback to make the robot navigate to the person (Fig. 4a). Task errors eventually converge near zero, although in a not very smooth way. This is due to the low quality of human detection (that prohibits constant usage in a continuous closed-loop way), and the low detection frame rate and latency issues (that limits the speed in reaching the person).

We encourage interested readers to see the video presentation associated with this work. The video demonstrates the autonomous medicine delivery HRI application experiment and describes the scheme from Fig. 1 in detail.

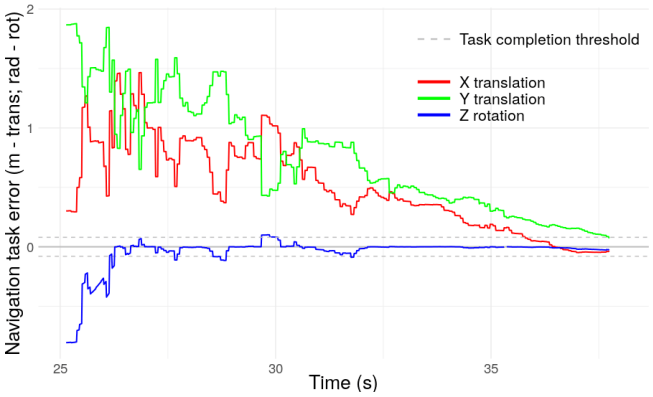


Fig. 6: Evolution of Position Based Visual Servoing closed-loop Pepper mobile base navigation to human task errors

VI. Future work

Next, we intend to showcase the advantage of use of our software tools in real-world assistive HRI settings: interaction with real patients in hospitals and retirement homes.

References

[1] K. Bouyarmane, K. Chappellet, J. Vaillant, and A. Kheddar, "Quadratic programming for multirobot and task-space force control," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 64–77, 2019.

[2] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier, "Mechatronic design of nao humanoid," in *IEEE International Conference on Robotics and Automation*, pp. 769–774, IEEE, 2009.

[3] A. Pandey and R. Gelin, "A mass-produced sociable humanoid robot: pepper: the first machine of its kind," *IEEE Robotics & Automation Magazine*, vol. 25, no. 3, pp. 40–48, 2018.

[4] A. Bolotnikova, P. Gergondet, A. Tanguy, S. Courtois, and A. Kheddar, "Task-space control interface for softbank humanoid robots and its human-robot interaction applications." submitted to IEEE/SICE International Symposium on System Integration (SII 2021), August 2020.

[5] A. Bolotnikova, S. Courtois, and A. Kheddar, "Autonomous initiation of human physical assistance by a humanoid," in *IEEE International Conference on Robot and Human Interactive Communication*, (Naples, Italy), 31 August–4 September 2020.