Individual Lab Report 8 - Progress Review 9

Autonomous Reaming for Total Hip Replacement

HIPSTER | ARTHuR

Anthony Kyu

Team C:

Kaushik Balasundar | Parker Hill | Anthony Kyu
Sundaram Seivur | Gunjan Sethi

October 13th, 2022

Carnegie Mellon University
The Robotics Institute

Contents

1 Individual Progress

The main task completed since the last progress review was implementing the task prioritization framework (with Kaushik) that was created and formalized last progress review (Figure 1).

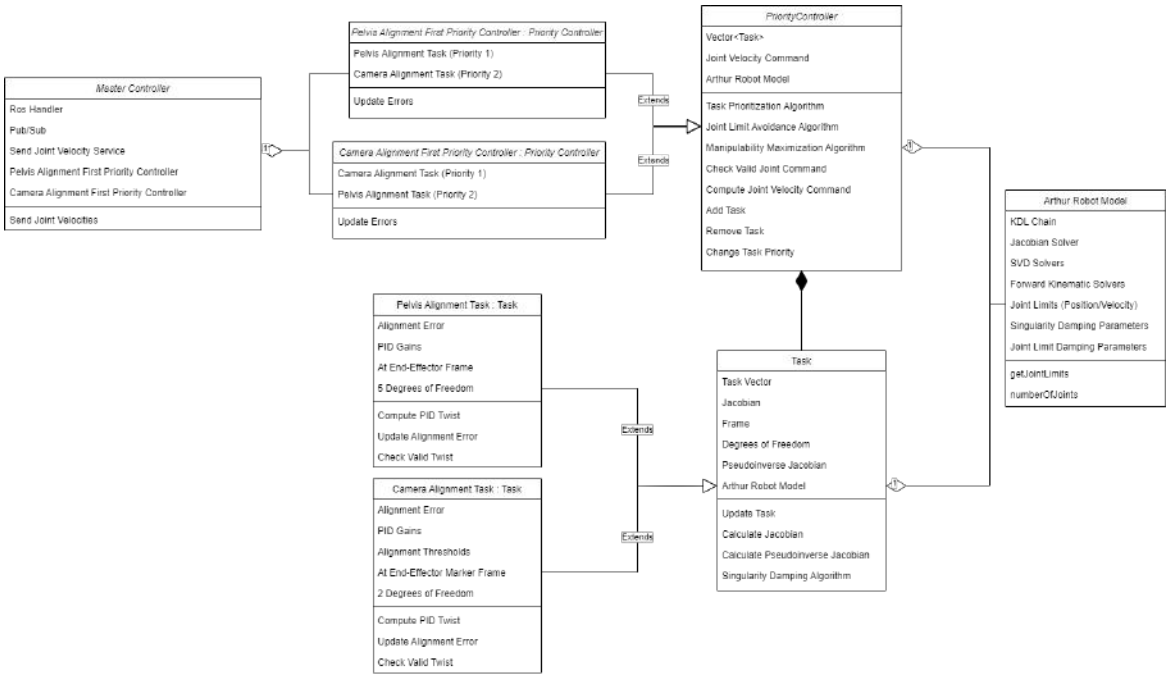

Figure 1: UML Diagram of the Task Prioritization Controller Framework

In our previous work on the joint velocity controller, we already implemented the Task and Arthur Robot Model classes, so algorithms such as pseudoinverse jacobian calculations, singularity damping, and joint limit avoidance were implemented. A few functions within the Robot Model class had to be implemented, which allowed the KDL Chain object to be parsed based on the given frame name. This allowed us to calculate different jacobians for different frames/segments of the URDF.

With this, we then implemented the Priority Controller class, which essentially holds multiple tasks and prioritizes which task is most important, sacrificing degrees of freedom for tasks of lower priority. For instance, we want to track the pelvis as well as track the camera, but we are willing to sacrifice camera tracking accuracy for pelvis tracking accuracy if we do not have enough degrees of freedom in the arm. In this class, we re-implemented joint limit avoidance (Figure 2), and implemented singularity avoidance and task prioritization algorithms. For singularity avoidance, we used gradient estimation for a measure of manipulation and then move in the task null space to maximize manipulability and therefore minimize proximity to singularities. The implementation of this algorithm can be observed in Figure 3.

For task prioritization, we project lower-priority tasks into the null space of higher-priority tasks. For the project, our highest priority task is tracking the pelvis frame, and the secondary task

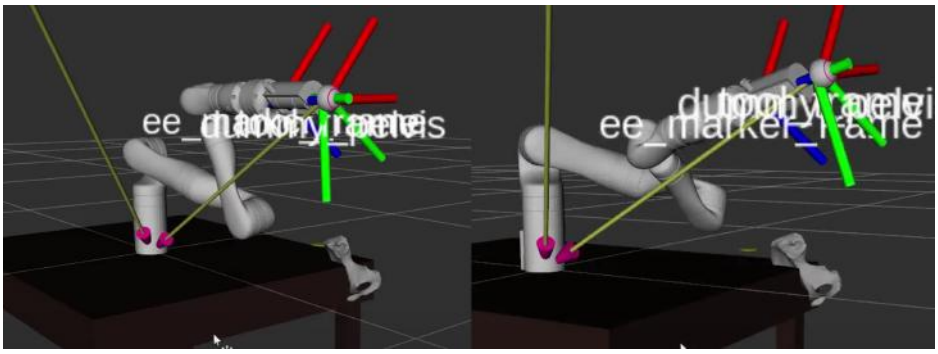

Figure 2: Joint Limit Avoidance Implementation. As Joint 2 reaches its joint limit, the controller stops using Joint 2 and adapts to using other joints to do pelvis alignment.

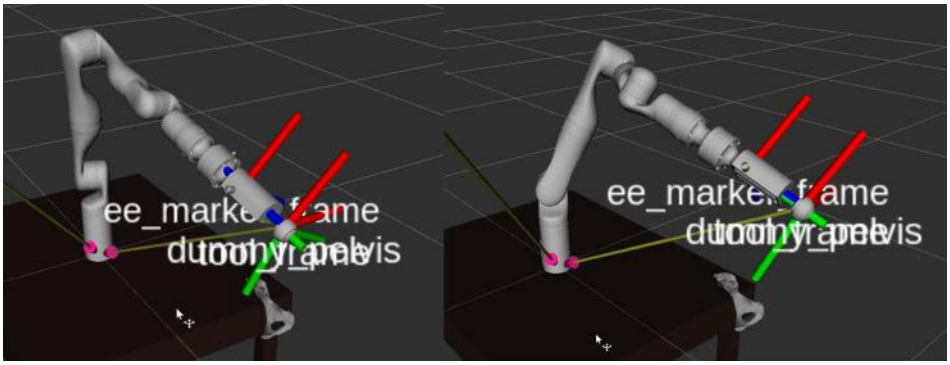

Figure 3: Implementation of Singularity Avoidance. On the left, singularity avoidance is off, so the robot moves through a singularity while tracking a dummy pelvis frame. On the right, singularity avoidance is on, so the arm reconfigures itself in its redundant space to avoid singularity while tracking the pelvis.

is aligning the end-effector markers to the camera. The task prioritization algorithm can be seen in Figure 4, and the implementation can be observed in Figure 5.

We also implemented helper functions to add and remove tasks, as well as to check if joint velocities and future joint positions are valid, reducing joint velocities linearly if they are too high from the given twist command.

Once this was implemented, the Task class was extended into a Pelvis Alignment Task and a Camera Alignment Task which each implemented its own PID controller to output the desired twist command based on the error of alignment with the pelvis or camera. The Master Controller was also reconfigured to use the newly implemented tasks. Once a dummy camera frame was added to the simulation, these classes were tested and debugged.

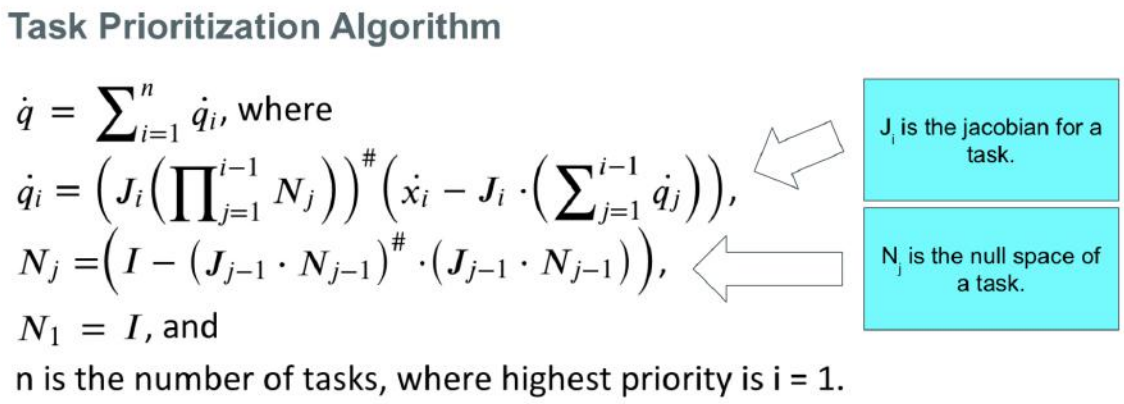

Figure 4: Task Prioritization Framework

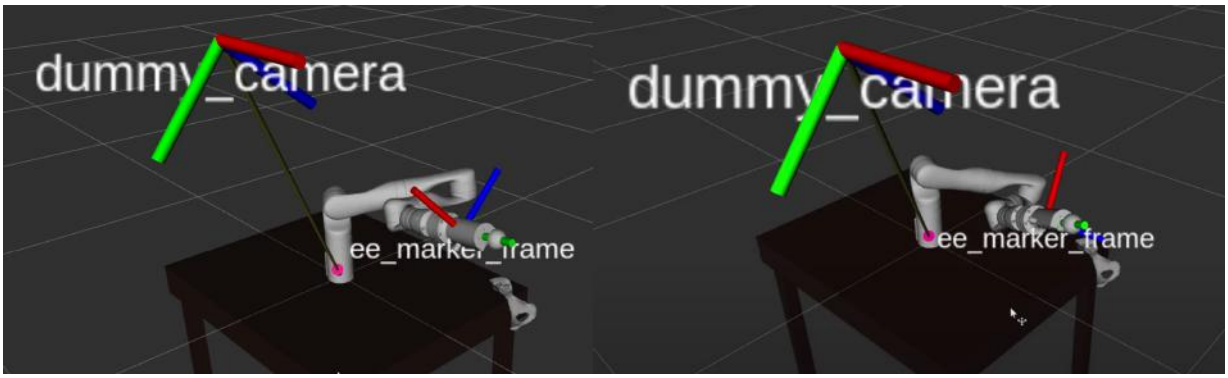

Figure 5: Implementation of Task Prioritization Framework. On the left is just the pelvis alignment task. On the right is the execution of two simultaneous tasks, with Pelvis Alignment as the highest priority and Camera Alignment as the lowest priority. As shown on the right, the end-effector markers align with the dummy camera frame without sacrificing the accuracy of the pelvis alignment task. Since there aren't enough degrees of freedom left to fully align the markers with the camera, degrees of freedom for that lower-priority task are sacrificed.

2 Challenges

The major challenges for this progress review were the immense amount of refactoring of code needed to implement the framework and the unclear API of using the KDL library to calculate jacobians in different frames of the robot.

In the last progress review, we only had a limited number of classes from this framework implemented. Most of the code was written into one file, which was the Master Controller file. However, when swapping over to this framework entirely, this required most of the code to be moved into modular class files, which required an intense amount of refactoring, and consequently caused a lot of bugs that took a chunk of time to debug through testing and code reviews.

The other challenge was rooted in the need to calculate jacobians in different frames of the robot (end-effector frame, and end-effector marker frame). Luckily, the KDL library already has some functions to allow us to do that. However, the documentation for these functions was not entirely clear, and required parsing through the KDL chain to get parameters for jacobian calculations. For some reason, the parameters are not consistent throughout the library, which caused a bug in what frame the jacobian was in, and therefore caused issues when testing task prioritization. After rigorous testing, and reading the computed output jacobians to see if they made sense, this issue was resolved.

3 Team Work

3.1 Anthony Kyu

Anthony worked with Kaushik to set up the task-prioritization framework, creating several new classes based on the software architecture, further setting up the simulation environment, and finally testing the framework in simulation. Anthony also worked with Parker to design the end-effector marker mount, providing feedback on the design, and helping 3D print some parts. Anthony also helped Sundaram to debug some of the Watchdog Module code, providing suggestions for code structure and CMake. And lastly, Anthony helped collect data for reaming on the pelvis encased in ballistics gel.

3.2 Parker Hill

Parker continued working on the end-effector, integrating a new motor plate for indirect force sensing, limit switches, and a marker holder into the design. He 3D-printed these new parts and assembled the end-effector to a functional state. Working with Kaushik, he then setup the electrical system and integrated it with the end-effector, allowing for the end-effector to be controlled by ROS. Finally, he collaborated with Gunjan and Sundaram to determine how to receive information from the watchdog so that it can be displayed in the user interface.

3.3 Sundaram Seivur

Sundaram worked on developing the watchdog module by setting up a ROSCPP node and successfully compiling the CMake file with the necessary dependencies. For this, he worked with

the owners of all the subsystems to finalize the functionality of the watchdog and the features that need to be developed. He made a decision tree that helped with the development of the subsystem and rigorously tested the inputs and perception subsystem working. He also worked on creating the ballistics gel mold for testing the pelvis model. He worked with Kaushik and Anthony to collect data by reaming the pelvis model submerged in the gel and analyzing the results generated. He discussed with Parker the integration of the Watchdog module with the User Interface and assisted him with evaluating the performance of the 3D-printed end-effector.

3.4 Kaushik Balasundar

Kaushik worked with Anthony in setting up the task-prioritization framework and testing it in simulation. He assisted Parker with wiring electronics and programming the reamer end-effector. He assisted Sundaram in setting up the ballistics gel encasing for the pelvis. Finally, he post-processed raw surgery data and conducted frequency analysis of the vibrations during reaming to validate the use of Ballistics gel as a proxy for soft tissue.

3.5 Gunjan Sethi

Gunjan continued development on the UI module. She setup the basic wireframe of the UI on Open3D. She then completed the Image Alignment tool development that is able to display multiple pointclouds and transform the implant pointcloud using UI-based controls. Further, she collaborated with Parker and Sundaram to facilitate the integration of the watchdog module with the UI. Finally. she worked with Kaushik to calibrate the new end-effector marker and test its detection and tracking.

4 Plans

For the next progress review, I plan on translating this task prioritization framework from simulation to reality. This would require some setup, since our physical hardware has also changed in the past few progress reviews. requiring collaboration with Parker to get the stl files of the new mechanical design to put into and modify the URDF. After the URDF is updated, we can then test in reality, and completion of that will open the door for online calibration.

In addition, I will also be working with Sundaram to integrate the controls subsystem with the Watchdog Module, further defining and implementing an interface between the two subsystems. And lastly, I will be working with Parker to start developing a controller for the end-effector. First, we will be collecting data to get a relationship between the force applied and current sent to the motor. After that, we will be implementing an admittance controller for the end-effector using the current-force relationship defined earlier. We may also implement a position controller on top of that. As a stretch goal, I will also be assisting Kaushik with online calibration.