Inverse Kinematics Module Proposal

Fabrizzio Coronado[*] and Ishan Kharat[†]
Maryland Applied Graduate Engineering
University of Maryland
(Dated: October 18, 2023)

I. OVERVIEW

A. Purpose

Motion is an integral component of robotic manipulators which requires concepts such as forward and inverse kinematics. Inverse kinematics is often more useful for practical applications because the user of the manipulator often knows what position or trajectory they want for the end effector. Due to its usefulness, creating a inverse kinematics module is a high priority for Acme. Furthermore, this module is necessary for performing a hardware test to verify whether it matches with software tests. Once this module is integrated with other modules such as a control or perception module, the physical robot use all 3 modules together to perform tasks.

B. Objectives

The proposed inverse kinematics module will be designed with the intention of providing a motion functionality, more specifically an inverse kinematics solver, to the Acme Robotics product. This module will be able to return the joint angles required once it is given the arm characteristics such as: end effector coordinates and euler angles.

C. Scope

This module will be able to calculate the joint angle velocities required in order to have the end effector be in the user-defined position. This module will not be able to calculate the end effector position based on joint angles defined by the user since it only focuses on inverse kinematics. Furthermore, this module specifically focuses only on velocity inverse kinematics. The algorithm used will be using dh parameters with the jacobian to map from end effector to joint angles.

D. Assumptions

This module will assume the user is using the Franka Emika Panda manipulator. Additionally, we assume the user has the latest version of C++ installed.

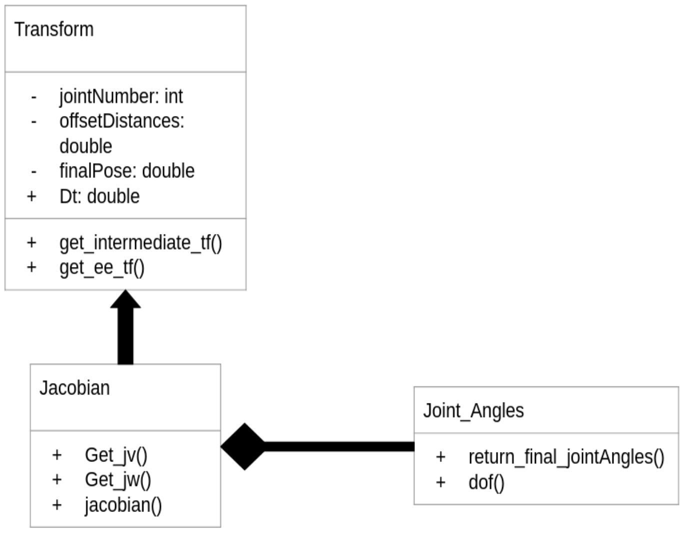

FIG. 1. UML Class Diagram

E. Constraints

A constraint to this module is it will not be able to return joint angles for any end effector positions outside of its workspace nor any positions where the arm is interfering with itself or the ground. Furthermore, since this module assumes the Franka Emika Panda arm will be used, it is constrained to the workspace of the panda arm.

F. Deliverables

At the end of this module development, all of the following will have been submitted to Acme:
1. Inverse Kinematics Module
2. Proposal
3. UML diagram and possibly additional diagrams such as activity diagram and CRC cards.
4. Code stubs and unit tests already integrated with Github CI and code coverage with CodeCov
5. Doxygen documentation
6. Github repository containing all the items listed above
The initial proposal and UML Class diagrams which are subject to change will be submitted during the first phase of this project.

[*] fcoronad@umd.edu — 11310065
[†] ishanmk@umd.edu — 12042714

G. Summary

In conclusion, we are proposing to a velocity inverse kinematics module to Acme while following the AIP process to ensure a high quality module is developed. This module is split into 3 phases, with this proposal being Phase 0, Phase 1 being focused on planning and developing classes and lastly implementation and delivery in Phase 2. To follow AIP, sprints will focus on items in the product and iteration backlog. At the end of each daily sprint, we will gather to discuss and push the updates made in the sprint, to our module repository. This module assumes the end user is working with the Panda manipulator and as a result is constrained to its workspace.

II. PROCESS

Throughout the design, integration and test of this module, we will be following an agile iterative process and have the development be test driven to ensure a quality product is delivered to Acme. To follow AIP, there will be a driver and navigator. Before pushing significant commits to the repository, the navigator will revise the work done by the driver. When both are in agreement with the changes, the commit is released. The two pairs will review the changes during the iteration meeting then fill out the product and iteration backlog. Each individual will be responsible for filling out the time spent completing tasks in the time log.

The development will be split into 3 phases spanning 4 weeks. Phase 0 is the current phase where we develop this proposal. Phase 1 will consist of sprints where the driver and navigator get together to plan what work needs to be done for the day. At the end of the day/sprint, they gather and discuss the progress made, and fill out the product, time and iteration logs. Furthermore in this phase the UML diagram will be made, module development will commence, any revisions necessary to the outline will be submitted, the unit tests and test coverage will be completed successfully and doxygen documentation generated. Phase 2 will be a continuation of Phase 1 plus refactoring, verification testing, finalizing the module design, repository and more.

A. Organization

While AIP typically consists of 5-10 people, we are a team of 2 and will have each member playing switching roles as drive and navigator. We will not be fulfilling the roles of a product nor process manager because we do not have enough people for it and this project is specifically focused on the technical aspect of the AIP model, not the business side.

B. Evolution and Range

The current plan so far only extends to Phase 1 since it is during Phase 1 where the plan outlined may be revised due to uncertainties with development such as successfully completing tasks on time, verifying full functionality, compiling and running without errors, libraries used, etc. Since there are many aspects of this project that can go wrong, the current plan only extends to Phase 1. The final phase, phase 2, will largely depend on how phase 1 goes.

C. Management

The progress of the project is going to be managed by filling out the product + iteration backlog and time log following each sprint. We will then be discussing the progress using these logs at the iteration meeting daily and planning ahead to make sure we are on track.

III. INITIAL PRODUCT BACKLOG

The initial product backlog was created during this Phase which consists of tasks assigned as following: Fabrizzio - Proposal outline, UML class diagram ... Ishan - Video explanation, quad chart, UML class diagram. A considerable amount of time was spent on the proposal and the class diagram took longer than expected due to revisions, which may happen again in future phases. This log is where we will add the desired requirements we intend to deliver to Acme.

IV. TECHNOLOGIES

This project will be written in C++ using the following libraries: eigen, iostream, etc. CMake will be used to build the module on Ubuntu 22.04 and github is where the module will reside and be updated. Code coverage will be generated using Github CI along with CodeCov for automatically running unit tests and generating the reports. The unit tests will be conducted using the GoogleTest module. Additionally, formatting and bug checking will be performed using cppcheck and cpplint packages. We will be using the MIT license to disclaim liability and keep the module open source. Lastly, doxygen will be used for documenting all class members. Reference materials for the MIT license, GoogleTest, Codecov, the panda arm, doxygen etc. can be found online and will be mentioned in the final version of the README or another document within the module.