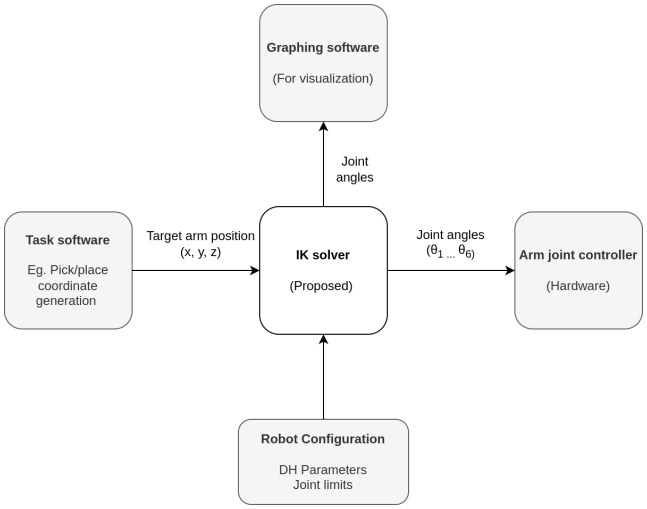
Project Proposal

ENPM808X - Mudit Singal (119262689), Abhishekh Reddy (119399002) and Abhimanyu Saxena (119342763)

Overview

We propose an inverse kinematics solver package for simulating the motion of the 6-DOF articulated arm that will be mounted on the service rover as a part of improving the rover functionality for remote inspection and operation of industrial machinery in environments that are potentially unsafe for humans. Testing the software packages for performing various tasks using the actual arm such as picking and placing objects and operating machine interfaces such as levers, knobs and switches requires verification of the arm motion in a simulation environment. The IK solver package receives a target end-effector position from other software packages responsible for moving the arm to perform tasks. The output will be a stream of joint angles required to move the arm to the target coordinate.



Proposed software diagram

Constraints

A middleware such as the ROS cannot be used which means existing solutions such as KDL cannot be used to solve the inverse kinematics problem. The project is also constrained by time which is only four weeks long.

Assumptions

A 6-DOF articulated arm is considered as the target configuration for this project. The desired joint trajectory is assumed to make the end-effector follow a straight line trajectory. The precision of joint angles from the compute method using standard data types and the Eigen library is

adequate. Arm dynamics are not considered hence, all links are assumed to be rigid and joints are frictionless.

Scope and Range

It is undecided whether a planner is required to generate joint trajectories. If a straight line trajectory is inadequate, a planner class with RRT* algorithm could be implemented. This software package could further be extended to be used on the actual hardware which bridges the gap between simulation and real-world performance for the other software packages being tested.

Process

Design and Development

An initial requirement elicitation is done to create an initial backlog of properties and functionality of the software package. Significant concepts from the backlog are then extracted and used to create a UML diagram consisting of classes and their relations which is further implemented [1]. The first sprint involves creating unit tests and implementing the initial design to pass the unit tests. The results from unit testing combined with new ideas from the team and users are added to the backlog. New unit tests could also be added or the existing ones could be modified based on the backlog and sprints are repeated iteratively until the project deadline or final package, whichever comes first.

The software package will be developed using Pair Programming with driver-navigator roles. To ensure the structure and timely progress of the project, a third person, the design keeper will oversee the project. Meticulous unit tests are prepared to ensure proper functioning of all the components of the software package. Code inspection by multiple people (navigator and design keeper), static code analysis, linting and memory leak checks are performed after every sprint in addition to unit tests to deliver high-quality code.

Technical

Computing inverse kinematics is done analytically using the Jacobian method. This involves inverting the Jacobian to map the position space to joint space [3]. The planned component functionality is to receive a target position for the end-effector in 3D cartesian coordinates and output joint angles trajectory for the arm to achieve desired motion to reach the target coordinate. The user provides configuration the configuration of arm such as DH parameters and joint limits that are necessary for computing inverse kinematics. Resulting joint angles from inverse kinematics solver are fed back into a forward kinematics solver to verify the end-effector reaches the input coordinate. Algorithms from the Eigen library will be used for calculating the inverse of jacobian, matrix multiplication to obtain transformation matrix for forward kinematics.

Technologies

C++ with standard above 11 will be the preferred language to utilize object-oriented approach while having faster execution times to satisfy the real-time performance requirement of this software. CMake build system will be used to build the project, allowing for a modular approach towards maintaining code in different files. CodeCov will be used for monitoring code coverage while performing unit tests to ensure the entire codebase is covered during testing. Since linear algebra is involved in computing the inverse kinematics, Eigen library will be used to leverage functions for matrix operations required for linear algebra [2]. This library is licensed under the Mozilla Public License (MPL).

Deliverables

GitHub repository to the software package will be provided which contains the source code in the main branch with the steps to build and integrate the package for simulation. A separate branch consists of documentation in the form of PDF files. The documentation can also be found in the Wiki page of the project repository.

Organization

The team consists of a pair and a design keeper as discussed in the development process. The pair will switch roles after a sprint to ensure both will receive sufficient time on the keyboard. The roles are as follows:

Person	Role
Abhimanyu Saxena	Navigator
Abhishekh Reddy	Driver
Mudit Singal	Design Keeper

Management

In this four-week long project, only three weeks are available for developing the software. Each sprint will last for one week with 10 hours of dedicated time per week. A total of three sprints will be performed as discussed in the process section to achieve proposed software package. A review meeting is held after every sprint, in which the results from the most recent build, its code coverage report and test suite results are discussed. This also provides an opportunity for the team members/stakeholders to input suggestions and ideas that could be added to the backlog for the next sprint.

Acronyms and Definitions

Acronym	Definition
DOF	Degrees of Freedom
FK	Forward Kinematics
IK	Inverse Kinematics
ROS	Robot Operating System
RRT	Rapidly-exploring Random Trees
KDL	Kinematics and Dynamics Library

Reference Materials

1. Software Engineering - The Current Practice - Textbook
2. Eigen Library - Link
3. Inverse Kinematics - Link