Acme Robotics Project Proposal

Project Component: Manipulator Arm Path Planner (IK Solver)

Introduction

The Manipulator Arm Path Planner is crucial for the robot's efficiency and versatility. It allows the robot to plan and execute precise arm movements, opening up opportunities for a wide range of applications. The module ensures that the robot can interact with its environment, objects, and other components with precision. This is particularly valuable in industries such as manufacturing, healthcare, and logistics, where precise manipulation is vital.

Purpose

The Purpose of this project is to develop a C++ software library with Inverse Kinematic solver for the motion planning of a Robotic Manipulator ARM for Acme Robotics. The library computes inverse velocity kinematics, that is joint angles, based on the end effector velocities.
This software module can be used for the motion planning (IK solver) of any n-degrees of freedom manipulator.

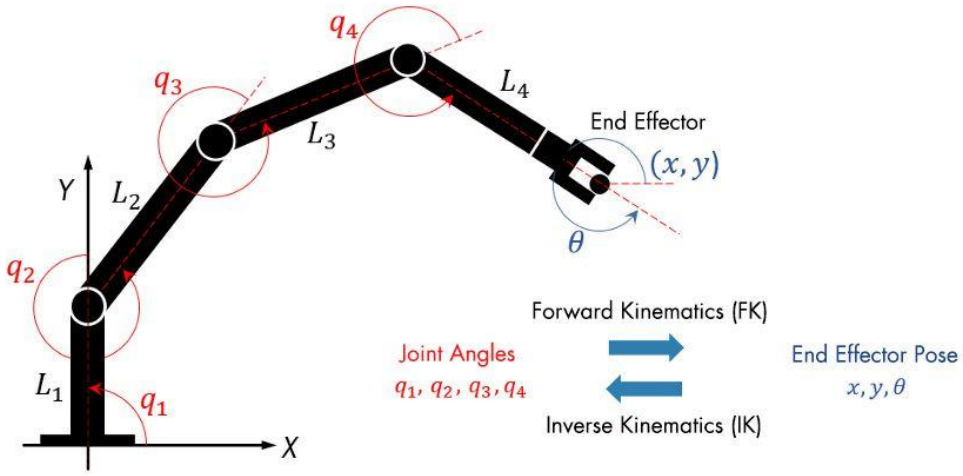

Image Credits: https://www.mathworks.com/discovery/inverse-kinematics.html

Algorithm

We are using the DH parameter technique to solve the Inverse Kinematics problem of a manipulator.
At each instant the algorithm computes the Jacobian of end effector velocities w.r.t to joint angles. Then inverse inverse of this Jacobian matrix is used to compute joint angles from the end effector velocities using the below formula

JointVelocities = J-1 * EndEffectorVelocities

We have decided to demonstrate the solution by plotting the end effector location using computed joint angles for the given trajectory.

Technologies and Libraries

The module will be developed using the following technologies:

Programming Language: C++11/14
Libraries: Eigen *(MPL2-licensed)*, Matplotlibcpp *(PSF license based)*
Build System: CMake
Testing Framework: Google Test
Static Code Analysis: cppcheck

Project Assumptions

The following assumptions are made for developing the project:

- The robotic arm is assumed to be a standard serial manipulator.
- The velocity trajectory of the end effector is known in advance
- The speed of computation depends on the systems being used for computation. The module does not provide any algorithms to compute inverse position kinematics.

Quality Assurance

We will be using Standard software design principles following the AIP process. The quality of the software will be ensured through TDD approach with pair programming where we are dividing the 3 member team into a Driver, Navigator and Designer.

Risks and Mitigation

Inverse Kinematics Challenges: If IK problems become challenging for complex manipulator arms, we will explore alternative solvers and consult experts to validate their performance. The resulting IK solutions for a given trajectory path depends on multiple parameters including joint angle limits, manipulator velocity limits and algorithm convergence accuracy. We are planning to mitigate some of these problems by providing guidelines on the probable solvable trajectories the library supports for a given robot.

Final Deliverables

Our final deliverables to Acme Robotics will include:

- UML diagrams (class diagram, dependency graph).
- C++11/14 compliant code with resolved compiler warnings.
- Unit tests using the Google Test framework.
- Code coverage reports with CodeCov (targeting 90% or higher coverage).
- Static code analysis using cppcheck.
- CMake files including external dependencies.
- Developer-level documentation in Doxygen format.
- README with comprehensive instructions and explanations.
- A well-organized GitHub repository with an informative commit history.
- GitHub CI widgets/badges for test passing, code coverage, and license information

Team Members

Design Keeper : Abraruddin Syed
Driver : Krishna Hundekari
Navigator : Tej Kiran Reddy P V