**N-Link DH Inverse kinematics solver using neural networks**

This project aims to train a dense neural network to perform inverse kinematics for rigid body link chains. This isn’t a new idea, as a trained solution to relatively complex nonlinear equations it has been discussed in multiple papers, although most current practical implementations, due to regulations and hardware restrictions for inference in the industry field, still use classical solvers such as BFGS (Broyden–Fletcher–Goldfarb–Shanno algorithm) which is a form of optimization and estimation algorithm (like gradient descent) from a class of algorithms known as quasi-Newton methods. Also, another common solver is Levenberg–Marquardt algorithm (also known as damped least-squares). It is a discipline where there are multiple implementations, both iterative and closed-form solutions. More generally, this is a study of minimizing error using neural networks on arbitrary nonlinear mappings and also to experiment with the most optimal parameters and the most suited activation function (tanh and an attempted implementation of RBF).

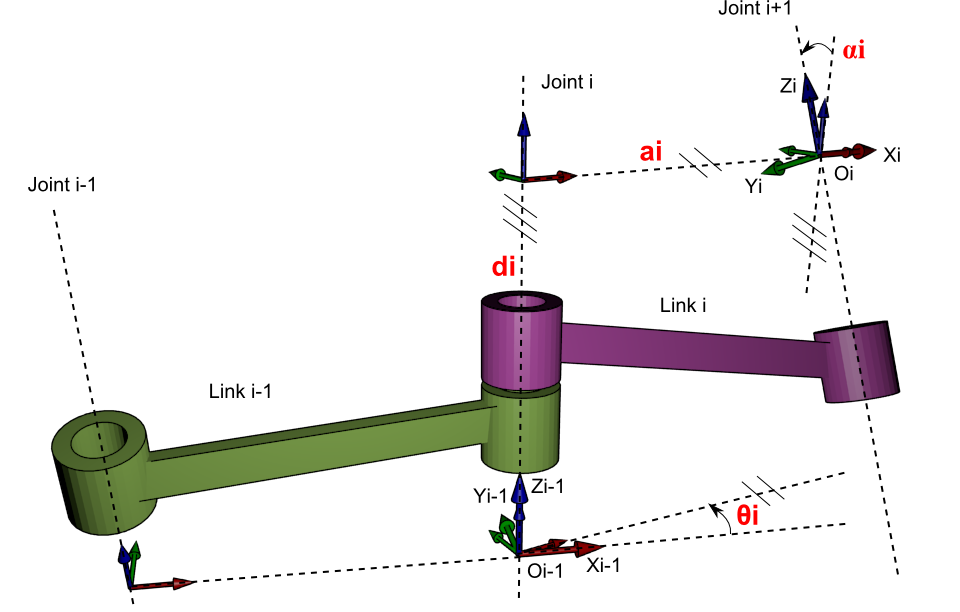
There are three fairly common ways to perform inverse kinematics for rigid link robots:

Closed Form Solutions: These are explicit solutions for solving the inverse kinematics. These solutions evaluate very quickly and are very accurate but can suffer the inability to be computed for suitably complex robotic systems. Moreover, additional logic may be required because there may be more than one solution, whence choosing the context of the solution is important. The Jacobian is also closed form in these solutions, so there is an explicit knowledge of the manipulability and associated joint angular velocity situations that are important for practical robots. Bottom line is the types of robots that closed form solutions are relevant to are limited, but the precision and controllability is high. These solutions are seen in precision industrial and manufacturing robots.

Iterative Solutions: In this case the forward kinematics are well known, but the inverse kinematics are more challenging to solve for explicitly. The Jacobian is often still known, so the manipulability and practical concerns for joint angular velocity are often known. The problem is that it is often unknown how long the solution will take to calculate, as it depends on the specific equations being solved, where on the manifold the solution lies, what the initial guess is, etc. While this method offers precisions, it isn't always appropriate since there is no guarantee of the timeframe for a solution, or if the solver will find a solution (for example problem conditioning). This type of solution is more commonly seen in unusually complex rigid body robots where there is no constrain on the time required to solve for the joint angles.

Neural Network Solutions: In this case a neural network is used to learn the inverse kinematics based on forward kinematic training. The Jacobian can be known but may not be integral to the concept of a solution. The idea is that this form of solution offers more flexibility (ability to resolve multiple possible solutions by using training from forward kinematics) with a consistent evaluation time at the expense of precision. Since the Jacobian itself is not part of the training, it can still be used for determining manipulability and angular joint velocity, though this is not part of the main idea of the technique. Since this type of solution does not offer high accuracy, it is not a solution that would be found in industrial and manufacturing robots. However, for classes of robots that do not require precision, and for which manipulability may not be of paramount importance, this type of solution is quite flexible in that only the forward kinematics needs to be known. Since this is not of relevance to most industrial problems, it might not be considered a common solution.

The robot kinematics studied here is for a 3R robot due to simplicity, but it is easily extendable to the most common 6R articulated arm.



In this work Denavit-Hartenberg parametrization convention is used for computing the composite homogeneous transformation. For training, I am strictly using the forward kinematics (the generator.py) to produce a set of features (Cartesian position of the end effector) and labels (joint angles). Forward kinematics mapping was used here rather than the inverse kinematics because the ultimate goal is to be able to actuate a robot without knowledge of how to find a closed form inverse mapping and understand how to create and train a model that provides acceptable accuracy. After establishing a link chain based on the DH parameters, a composite homogeneous transformation is produced as standard. Summary information about the transformation and the Jacobian are provided. The forward transformation is evaluated to train the network. The plotting part is still work in progress, so unfortunately unless we are exporting the angles, positions and configuration of the DH chain to something external we can’t see a general representation of the segments positions.

The results aren't bad. However, in general, the greater the slope, the worse the performance.

Since we are especially interested in areas near singularities for inverse kinematics, the accuracy at higher gradients is a problem. One solution is to use more neurons, but the number of neurons grows quickly with the size of the work area and the gradient of the manifold we are approximating. Another option is to use a significantly different type of activation function having a geometric aspect, such as radial basis functions as was seen in a previous attempt from a paper (This was investigated but not yet finished).

Text

Description automatically generated

Validation against the original function