Neural Networks for Inverse Kinematics Problems in Robotics

Franziska Schwaiger
Matriculation number: 03658670

Thomas Barthel Brunner
Matriculation number: 03675118

Introduction

In our project, we are evaluating the feasibility of using neural networks for inverse kinematics problems in robotics. In this report, we outline the progress made and discuss the results we have obtained.

Methods

Robot Simulations

To train and test our models, we developed simulations of planar robotic arms with revolute joints. As we are interested in testing the limits of our models, we created simulations of arbitraty degrees of freedom (DOFs). In general, the forward kinematics equations of a planar robot arm with link lengths l_i and N revolute joints with joint angles θ_i can be described as:

$$x_{TCP} = \sum_{i=1}^{N} l_i \cos\left(\sum_{j=1}^{i} \theta_j\right) \tag{1}$$

$$y_{TCP} = \sum_{i=1}^{N} l_i \sin\left(\sum_{j=1}^{i} \theta_j\right) \tag{2}$$

$$\theta_{TCP} = \sum_{i=1}^{N} \theta_i \tag{3}$$

These equations were modified for fast vectorized numerical computation by the element-wise computation of the sine and cosine of the matrix multiplication of the joint angles vector $\boldsymbol{\theta} = [\theta_1, \theta_2, ...]^{\mathrm{T}}$ with an upper triangular matrix with ones as elements $\boldsymbol{U}$. The results are then multiplied with the link lengths vector $\boldsymbol{l} = [l_1, l_2, ...]^{\mathrm{T}}$, as shown in Equations 4 and 5.

$$x_{TCP} = \cos\left(\boldsymbol{\theta}\ \boldsymbol{U}\right)\boldsymbol{l} \tag{4}$$

$$y_{TCP} = \sin\left(\boldsymbol{\theta}\ \boldsymbol{U}\right)\boldsymbol{l} \tag{5}$$

In our current setup, we do not take the angle of the tool center point (TCP) into consideration. Thus, the location of the TCP is defined by its x, y coordinates. As a result of this, all simulations with more than 2 DOF can up to infinite solutions of the inverse kinematics for a given TCP position. The 2 DOF robot arm can have up to two solutions.

Dataset

The dataset used for training is composed of one million samples of robot configurations for each DOF. Figure **??** shows an illustration of the datasets. Each configuration was sampled from a normal distribution $\theta_i \sim \mathcal{N}(\mu = 0, \sigma = 0.2)$. Thus, the dataset is composed mostly of configurations in which the robot arm is extended. This reflects real-world use cases of robot arms, which have limited workspaces and whose tasks are focused on one section of the workspace. Moreover, limiting the range of the joints improved the performance of the networks, as this avoids the discontinuity in the angles at $\theta = \pi$.

Considering Singularities

As already mentioned in the previous section, we avoid discontinuities by restricting the configuratoin space of the robot. Yet we implemented two approaches which consider this discontinuity in the angles at $\theta = \pi$ to generalize the implemented networks to inverse kinematics problems.

First approach: The first approach uses a non-minimal parameterization of ...

Second approach:

Conditional Variational Autoencoder

Invertible Neural Networks

Hyperparameter Optimization

For our tests, we trained a network for each robot simulation (DOF). Thus, the sizes and selection of hyperparameters of each network had to match the compleity of the simulation. To compare the performance of the networks across different DOFs in a fair setting, we used Tune [**?**] to perform a random search of the hyperparameters. To reduce computation time, we trained the models using a subset of the main dataset, which was composed of $1e4$ samples.

Implementation

The previous two model architectures (cVAE and INN) have been implemented in PyTorch and are inspired by existing implementations in [**?**], [**?**]. We utilized the Google Compute Engine to train our models.

Experimental Evaluation

Evaluation protocol

Results

Conclusion