Rutkowski Szymon 263499	Date: 10 VI 2025r
Tomczak Marcin 264322	mgr inż. Krzysztof Adamczyk
Intelligent virtualization of systems and process automation project report	

1. INTRODUCTION

As our topic, we have chosen robotic manipulator's inverse kinematics solved by machine learning. We drew inspiration from the *Neural Inverse Kinematics* [1] paper.

We have prepared two fully trained models of 3-DoF spherical manipulators: one for simulation and evaluation, and the other for controlling the real robot hardware. The simulation model is simpler – it has the first link omitted such that the two first joints are merged. Also, the second joint is reversed. The model for a real robot has forward kinematics derived using Denavit-Hartenberg for the exact match with the real world. Figure 1.1 graphically shows the simulation robot.

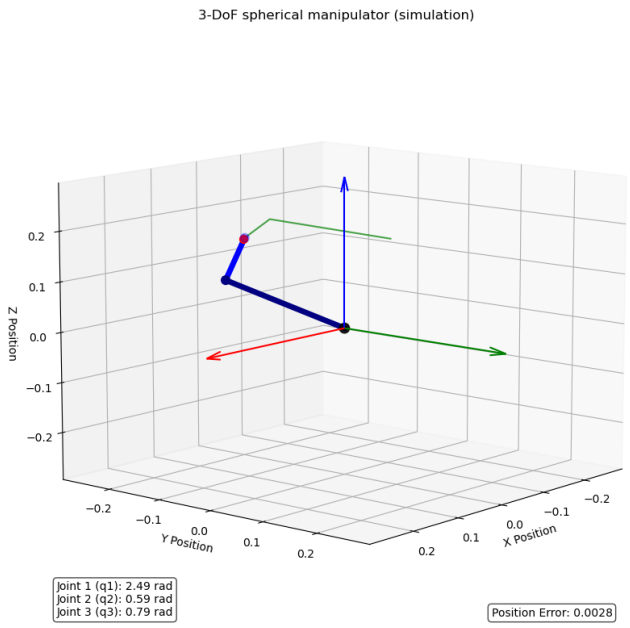

Figure 1.1: Simulation arm

Differences between simulation and real robot come from the fact that the real hardware was being developed during the course of the project.

2. DATASET GENERATION

The dataset is separately generated for every robot configuration from its forward kinematics equations. The dataset consists of 3 attributes – input position, input joint, and output joint. In the sense of inverse kinematics, input position is the desired point the effector should move to, and input joint is the current joint of the robot.

As a means of simplifying the dataset generation process, the input joint parameter is generated by adding a small random perturbation to the joint angle currently used to generate a new effector position with the help of forward kinematics. It also teaches the neural net about the local solution space – the relationship between small changes in joint angles and resulting end-effector positions. It prevents the trained robot from doing sudden movements and rapid reconfigurations. Algorithm 1 shows the process of dataset generation.

Algorithm 1 Dataset Generation for Inverse Kinematics training

Require: Joint angle ranges: Θ_1, Θ_2, Θ_3
Require: Number of random samples per configuration: $N_{samples}$
Require: Perturbation magnitude: $\varepsilon = 0.2$
Ensure: Dataset $\mathscr{D} = \{(\mathbf{p}_i, \theta_{input,i}, \theta_{target,i})\}$

Initialize empty dataset $\mathscr{D} \leftarrow \emptyset$
for $\theta_1 \in \Theta_1$ **do**
  for $\theta_2 \in \Theta_2$ **do**
    for $\theta_3 \in \Theta_3$ **do**
      $\theta_{target} \leftarrow (\theta_1, \theta_2, \theta_3)$
      $\mathbf{p} \leftarrow$ calculate_end_effector_position(θ_{target})
      for $s = 1$ to $N_{samples}$ **do**
        $\delta_1 \leftarrow \varepsilon \cdot (\text{random}() - 0.5)$
        $\delta_2 \leftarrow \varepsilon \cdot (\text{random}() - 0.5)$
        $\delta_3 \leftarrow \varepsilon \cdot (\text{random}() - 0.5)$
        $\theta_{input} \leftarrow (\theta_1 + \delta_1, \theta_2 + \delta_2, \theta_3 + \delta_3)$
        Add $(\mathbf{p}, \theta_{input}, \theta_{target})$ to $\mathscr{D}$
      end for
    end for
  end for
end for
return Dataset $\mathscr{D}$

In the case of our simulated 3-DoF spherical manipulator, forward kinematics look as follows (first and second link):

$$x_1 = l_1 \cos(q_2 + q_{2,\text{offset}}) \tag{2.1}$$
$$y_1 = l_1 \sin(q_2 + q_{2,\text{offset}}) \cos(q_1) \tag{2.2}$$
$$z_1 = l_1 \sin(q_2 + q_{2,\text{offset}}) \sin(q_1) \tag{2.3}$$

$$x_2 = l_1 \cos(q_2 + q_{2,\text{offset}}) + l_2 \cos(q_2 + q_{2,\text{offset}} + \pi - q_3 - q_{3,\text{offset}}) \tag{2.4}$$
$$y_2 = l_1 \sin(q_2 + q_{2,\text{offset}}) \cos(q_1) + l_2 \sin(q_2 + q_{2,\text{offset}} + \pi - q_3 - q_{3,\text{offset}}) \cos(q_1) \tag{2.5}$$
$$z_2 = l_1 \sin(q_2 + q_{2,\text{offset}}) \sin(q_1) + l_2 \sin(q_2 + q_{2,\text{offset}} + \pi - q_3 - q_{3,\text{offset}}) \sin(q_1) \tag{2.6}$$

All the constraints are included in the generated dataset – for example, joint position constraints are obtained by simply limiting the range of motion of the joint used to calculate end effector position. Picture 2.1 shows the generated workspace of the simulation robot for the 30 joint values and 5 nearby random positions generated for each position acquired with forward kinematics. Disturbance magnitude $\varepsilon = 0.2$.

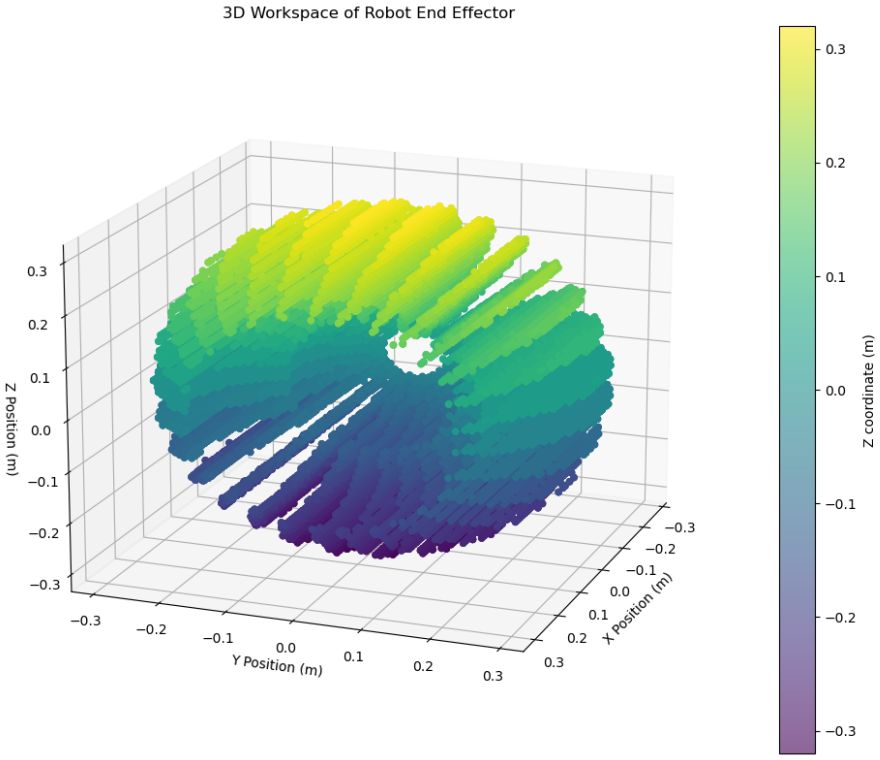

Figure 2.1: Simulation robot's workspace

3. MODEL DESIGN AND TOOLS

Artificial neural networks have been chosen as a machine learning algorithm for the task of learning inverse kinematics. Multilayered perceptron (MLP) has been chosen as the main architecture. For the 3-Dof spherical manipulator, it consists of:

- 6 inputs (three-dimensional cartesian coordinates, and the current joint positions);
- 128 neurons dense layer;
- 128 neurons dense layer;
- 64 neurons dense layer;
- 3 neurons dense layer – output (3 joint positions).

Various model hyperparameters have been chosen through tests and evaluation. This process is described in the section 4.

All the parts of the project were written in Python using Tensorflow library with Keras API. Dataset was handled using Pandas library.

4. TESTS AND MODEL EVALUATION

To evaluate the model, 3 shapes representing the path of the end effector were created. Error was measured as an integral of the square of the error (ISE). Shapes chosen for the test were: 2D circle, 3D skewed ellipse, and 2D star. These shapes are shown in figures 4.1–4.3.

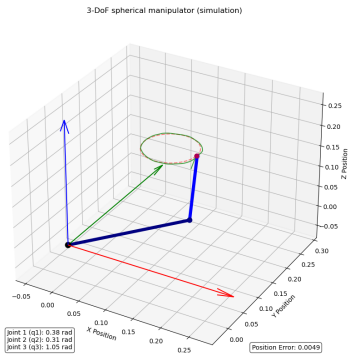

Figure 4.1: Circle

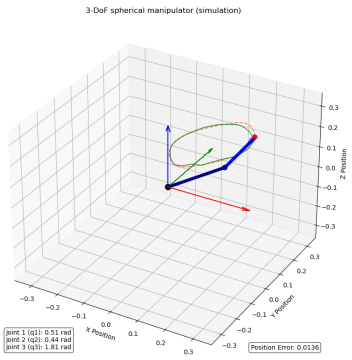

Figure 4.2: Skewed elipse

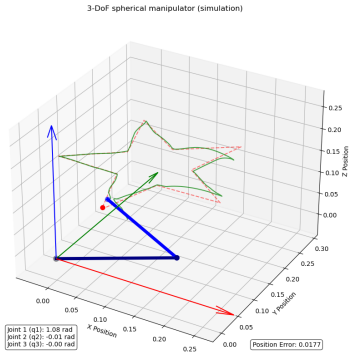

Figure 4.3: Star

In order to find best performing parameters, grid search was performed evaluating 64 configurations for each shape. Each combination was trained 10 times to for more credible scores. The search considered 4 optimizers, 4 loss functions, and 4 activation functions, with all combinations tested using identical training parameters: 170 epochs, 450 samples per batch, min delta of 0.0005 radians, and early stopping. Average training time ranged from 32 to 53 seconds per configuration. The chosen parameters are briefly described below:

1. Optimizers:
    - SGD (Stochastic Gradient Descent) – utilizes stochastic approximation of gradient descent.
    - RMSprop – adjusts the learning rate dynamically for each parameter and maintains a moving average of squared gradients to normalize the updates, preventing drastic learning rate fluctuations.
    - Adam – combines momentum and RMSprop techniques to adjust learning rates during training.
    - Nadam – combines Adam's adaptive learning with Nesterov momentum's look ahead gradient updates.
2. Activation functions:
    - relu – is defined as the non-negative part of its argument, i.e., the ramp function. Relu transforms its input into a value between 0 and 1.
    - tanh (hyperbolic tangent) – is a function which transforms its input into a value between -1 and 1.
    - swish – is defined as $f(x) \cdot sigmoid(x)$, and allows negative values in comparison to relu function.
    - sigmoid – is similar to tanh function but transforms input into values from 0 to 1.

3. Loss functions:

   - MSE – measures the average squared difference of error.
   - MAE – measures the average absolute difference of error.
   - huber – combines sensitivity of MSE with robustness of MAE. The hybrid nature of Huber loss makes it less sensitive to outliers, similar to MAE, but also penalizes small errors within the data points, similar to MSE.
   - log_cosh – takes the logarithm of the hyperbolic cosine of the error. It shares similarities with Huber loss but is infinitely differentiable.

The grid search results were visualized as heat maps for better readability. An example heat map for the **circle** shape is presented in Figure 4.4, showing performance across different optimizers. The best parameter combinations for each shape were selected with help of created heat maps and are summarized in Table 4.1.

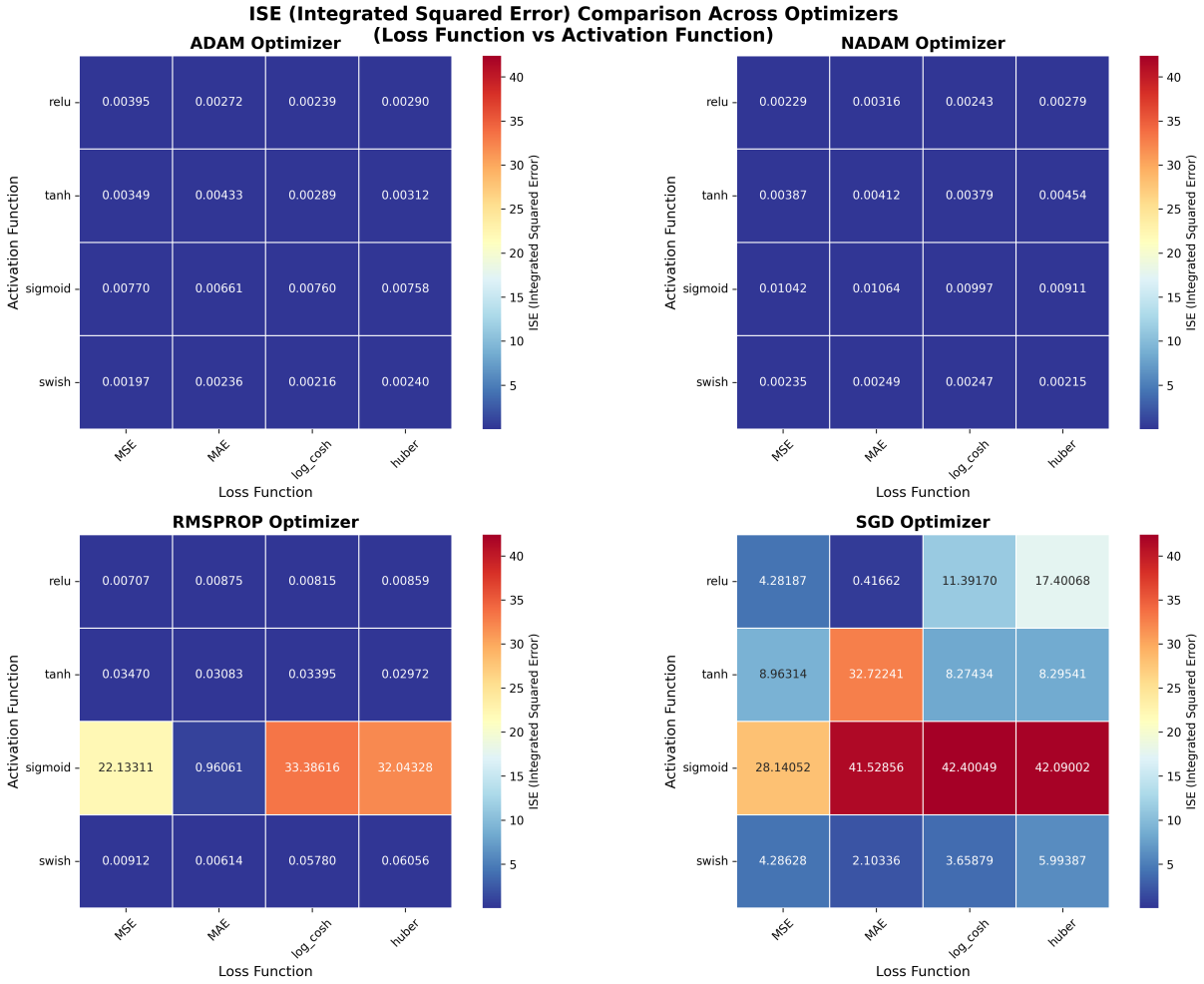

Figure 4.4: Heat maps presenting results of grid search for **circle**

Table 4.1: Best performing parameters for each shape

Shape	Optimizer	Activation	Loss
3D ellipse	Nadam	swish	MAE
2D star	Adam	swish	log_cosh
2D circle	Adam	swish	MSE

5. RUNNING THE MODEL ON REAL HARDWARE

By using Denavit-Hartenberg notation to acquire forward kinematics, positions of the end effector look as follows:

$$x_1 = 0 \tag{5.1}$$

$$y_1 = L_1 \cos(q_1 + \frac{\pi}{2}) \tag{5.2}$$

$$z_1 = L_1 \sin(q_1 + \frac{\pi}{2}) \tag{5.3}$$

$$x_2 = L_2 \sin(q_2) \tag{5.4}$$

$$y_2 = L_1 \cos(q_1 + \frac{\pi}{2}) - L_2 \sin(q_1 + \frac{\pi}{2}) \cos(q_2) \tag{5.5}$$

$$z_2 = L_1 \sin(q_1 + \frac{\pi}{2}) + L_2 \cos(q_1 + \frac{\pi}{2}) \cos(q_2) \tag{5.6}$$

$$x_3 = L_2 \sin(q_2) + L_3 \cos(q_2) \sin(q_3) + L_3 \cos(q_3) \sin(q_2) \tag{5.7}$$

$$\begin{aligned} y_3 = {} & L_1 \cos(q_1 + \frac{\pi}{2}) - L_2 \sin(q_1 + \frac{\pi}{2}) \cos(q_2) \\ & - L_3 \sin(q_1 + \frac{\pi}{2}) \cos(q_2) \cos(q_3) + L_3 \sin(q_1 + \frac{\pi}{2}) \sin(q_2) \sin(q_3) \end{aligned} \tag{5.8}$$

$$\begin{aligned} z_3 = {} & L_1 \sin(q_1 + \frac{\pi}{2}) + L_2 \cos(q_1 + \frac{\pi}{2}) \cos(q_2) \\ & + L_3 \cos(q_1 + \frac{\pi}{2}) \cos(q_2) \cos(q_3) - L_3 \cos(q_1 + \frac{\pi}{2}) \sin(q_2) \sin(q_3) \end{aligned} \tag{5.9}$$

Figures 5.1 and 5.2 show the graphical representation of the robot and the real hardware.

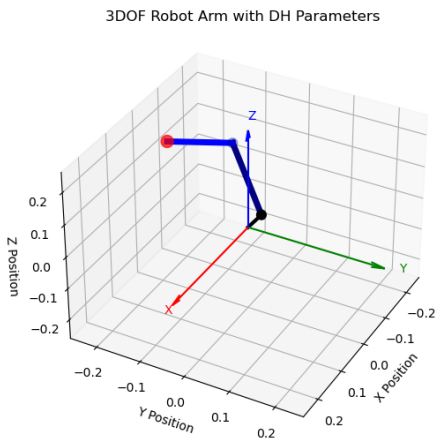

Figure 5.1: Graphical representation

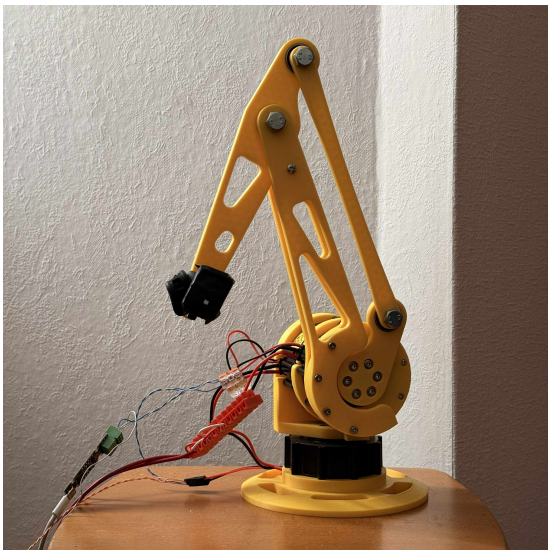

Figure 5.2: Real hardware

To run the model on the real robot, ROS2 node was created to convert joints predicted by the model to ROS2 joint trajectory messages and send them to the hardware interface controller. Tests on real hardware were supposed to just visually show that the model is working; no evaluation of the error of the end effector position was carried out.

Shape chosen to be drawn by the robot was a star as it is the most visually interesting. To capture the image, an app for light painting was used (it allowed for video with frames with long exposure). The robot had a red bike light strapped to the end effector. Picture 5.3 shows the star drawn in the air.

Figure 5.3: Star drawn by the robot

The star appears skewed because the robot was rotated slightly on its x axis.

REFERENCES

[1] Raphael Bensadoun et al. *Neural Inverse Kinematics*. 2022. arXiv: 2205.10837 [cs.LG]. URL: https://arxiv.org/abs/2205.10837.