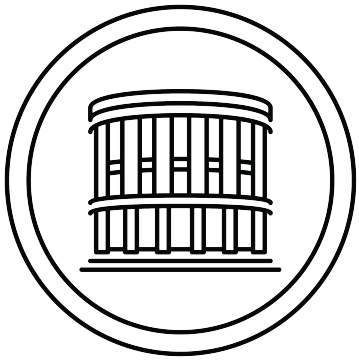
COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS PHYSICS AND INFORMATICS

OPTIMIZATION OF SIM-TO-REAL TRANSFER IN THE HUMANOID ROBOT NICO

Master thesis

2026 Bc. Juraj Gavura

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS PHYSICS AND INFORMATICS



OPTIMIZATION OF SIM-TO-REAL TRANSFER IN THE HUMANOID ROBOT NICO

Master thesis

Study program:	Applied informatics
Branch of study:	Applied informatics
Department:	Department of Applied Informatics
Supervisor:	prof. Ing. Igor Farkaš, Dr.
Consultant:	Mgr. Michal Vavrečka, PhD.

Bratislava, 2026 Bc. Juraj Gavura

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Juraj Gavura
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Optimalizácia realitného transferu u humanoidného robota NICO
Optimization of sim-to-real transfer in the humanoid robot NICO

Anotácia: Simulátory sú intenzívne využívané v robotike, vďaka rýchlemu a lacnému prototypovaniu robotických systémov bez potreby fyzického prístupu k hardvéru. Avšak kvôli existujúcim nezrovnalostiam medzi simuláciami a reálnym svetom (realitná medzera) existuje problém pre priamy prenos vytvoreného matematického modelu zo simulátora do fyzického robota.

Cieľ: 1. Využijúc robotický simulátor MyGym, ktorý obsahuje model humanoidného robota NICO, navrhnite kalibračnú metódu umožňujúcu prepojiť simulátor s fyzickým robotom.
2. Metóda bude spočívať v nameraní pozícií párových bodov (v 3D), z ktorých sa vytvorí matermatické zobrazenie (napr. pomocou neurónovej siete) medzi oboma priestormi.
3. Otestuje presnosť navrhnutej metódy pomocou vhodných motorických úkonov ramenom robota.

Literatúra:
Collins J. (2019). Quantifying the Reality Gap in Robotic Manipulation Tasks. IEEE Int. Conf. on Robotics and Automation
Kerzel M. et al. (2017). NICO — Neuro-inspired companion: A developmental humanoid robot platform for multimodal interaction. In IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)
Salvato E. et al. (2021). Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning. IEEE Access.

Vedúci: prof. Ing. Igor Farkaš, Dr.
Konzultant: Mgr. Michal Vavrečka, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 05.12.2024

Dátum schválenia: 15.12.2024 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

I hereby declare that I have written this thesis by myself, only with help of referenced literature, under the careful supervision of my thesis advisor.

.....

Bratislava, 2026 Bc. Juraj Gavura

Abstract

The growing demand for intelligent and adaptive robots has led to significant advances in simulation, control, and learning algorithms. However, transferring behaviors learned in simulation to the real world remains a major challenge due to discrepancies between simulated and physical dynamics, a phenomenon known as the reality gap. This thesis focuses on improving the sim-to-real transfer in the humanoid robot NICO, with an emphasis on reaching and grasping tasks. A two-dimensional calibration method based on haptic feedback was developed to align the simulated and real end-effector positions. The approach provides a low-cost and easily reproducible alternative to vision-based calibration systems. Subsequently, the study explores 2D grasping optimization using visual feedback and applies the calibrated model to extend grasping to 3D environments.

Keywords: reality gap, sim-to-real, humanoid robot NICO, grasping

Abstrakt

Rastúci dopyt po inteligentných a adaptívnych robotoch viedol k významnému pokroku v oblasti simulácie, riadenia a algoritmov strojového učenia. Prenos správania naučeného v simulácii do reálneho sveta však naďalej predstavuje veľkú výzvu kvôli rozdielom medzi simulovanou a fyzickou dynamikou, čo je jav známy ako „reality gap“. Táto práca sa zameriava na zlepšenie prenosu zo simulácie do reality v humanoidnom robotovi NICO, s dôrazom na úlohy dosahovania (reach) a uchopovania (grasping). Bola vyvinutá dvojdimenzionálna kalibračná metóda založená na haptickej spätnej väzbe s cieľom zarovnať simulované a reálne polohy koncových efektorov. Tento prístup poskytuje nákladovo nenáročnú a ľahko reprodukovateľnú alternatívu ku kalibračným systémom založeným na videní. Následne sa štúdia zaoberá optimalizáciou uchopovania v 2D pomocou vizuálnej spätnej väzby a aplikuje kalibrovaný model na rozšírenie uchopovania do 3D prostredí.

Kľúčové slová: reality gap, sim-to-real, humanoidný robot NICO, grasping

Contents

List of Figures

List of Tables

Terminology

Terms

Abbreviations

Introduction

With each passing year, the demand for robots of various kinds continues to grow. Robots are increasingly capable of performing complex manipulation and interaction tasks that require precision, adaptability, and autonomous decision-making. From industrial manipulators working with objects to humanoid robots that perceive and respond to their surroundings, the field of robotics has expanded rapidly due to advances in perception, control, and learning algorithms. Modern robots integrate multiple sensory modalities like vision, touch and proprioception, allowing them to interact with the environment in increasingly sophisticated ways.

Before any robot can successfully perform its assigned task, it must undergo a learning or calibration phase. This process ensures that its actions in the real world are accurate and reproducible. A commonly used method for training robots is reinforcement learning, which allows agents to develop control policies through trial and error by interacting with their environment. However, training directly in the physical world can be costly, time-consuming, and may even cause mechanical damage due to repeated or unsafe actions. Consequently, simulation has become an essential part of robotics research and development [12].

Simulators enable faster, cheaper, and safer experimentation. They make it possible to generate vast amounts of training data, perform experiments in parallel, and easily test algorithms under controlled conditions [17]. Yet, despite their benefits, simulations can never perfectly replicate the physical world. Simplifications of dynamics, imperfect sensor models, and differences in hardware specifications lead to discrepancies between simulated and real-world behavior. This mismatch, known as the reality gap, poses a significant obstacle to transferring learned models or controllers from virtual environments to physical robots [3].

Bridging this gap, often referred to as sim-to-real transfer, has become one of the most important challenges in robotics. Even small deviations in the robot's actuators or sensors can cause noticeable differences in the end-effector position, which directly affects the performance of manipulation and interaction tasks. For instance, when inverse kinematics solvers are used to control robotic arms, inaccuracies in the robot's model or physical setup can lead to errors in reaching the desired target position. The more precise the task, the more evident the discrepancy becomes.

To address these challenges, recent research has focused on developing calibration methods that minimize the difference between simulated and real-world robot behavior. Many traditional approaches rely on vision-based systems [11], external motion capture devices [8], or complex sensor fusion frameworks [14] to measure the robot's end-effector position. Although effective, these solutions can be expensive, sensitive to environmental conditions, or require specialized equipment.

The complexity of transferring robotic behavior from simulation to reality has always fascinated me, particularly in the context of grasping and object interaction. Understanding how a robot can reliably reproduce delicate, coordinated movements learned in a virtual environment is both a scientific and an engineering challenge. This curiosity motivated me to explore the topic further and gain firsthand experience with the transition between simulated and real settings.

Working directly with a humanoid robot such as NICO offers a unique opportunity to connect theoretical principles of motion control with their tangible outcomes. Observing how a robot learns to reach and grasp objects, and how small adjustments in calibration can lead to noticeably smoother and more natural movements, provides a strong sense of purpose and inspiration for this research.

Therefore, the goals of our work were to improve the accuracy of NICO robot in a 2D world using reach movement, in other words, to design a cheap and simple 2D calibration based on haptic feedback. Furthermore, to look at the challenges coming with grasping objects, also in 2D, and optimize this function using the calibration created in the first step and help of visual feedback. Finally, to try to transfer grasping from a 2D to a 3D system and quantify the reality gap in this sim-to-real transfer.

Chapter 1

Related work

1.1 Reality gap quantification

1.1.1 Identification

Before the reality gap can be quantified, it first needs to be identified. In other words, we must determine whether a discrepancy between simulation and the real world actually exists. To illustrate this, consider the typical learning process of a robot trained in simulation. At first, a controller is developed and trained using a chosen learning method, and its performance is tested in a virtual environment. If the robot fails to complete the task, the controller continues to train. If it succeeds, the same task is then repeated on the physical robot. Any observable differences between the robot's simulated performance and its behavior in the real world indicate that a problem occurred during the transfer from simulation to reality [17]. In other words, the presence of the reality gap. Once this gap has been identified, the next step is to find an effective way to measure and analyze it.

1.1.2 Quantification

The first experiment involved a simple rotation of a single joint by 100 degrees over six seconds, with all other joints fixed. This task examined basic kinematic accuracy and timing. The most consistent trajectories were produced by PyBullet and V-Rep running the Newton and Vortex engines, though these lacked the oscillatory motion seen in the real arm. Conversely, the Bullet engine simulated realistic oscillations but completed the movement about one second faster than the physical robot, while ODE was unstable and produced unreliable results.

The second test required two joints to rotate alternately, introducing gravitational effects and coordination challenges. Here, V-Rep with the Newton and Vortex engines again delivered the most realistic trajectories, closely followed by PyBullet. Still, none

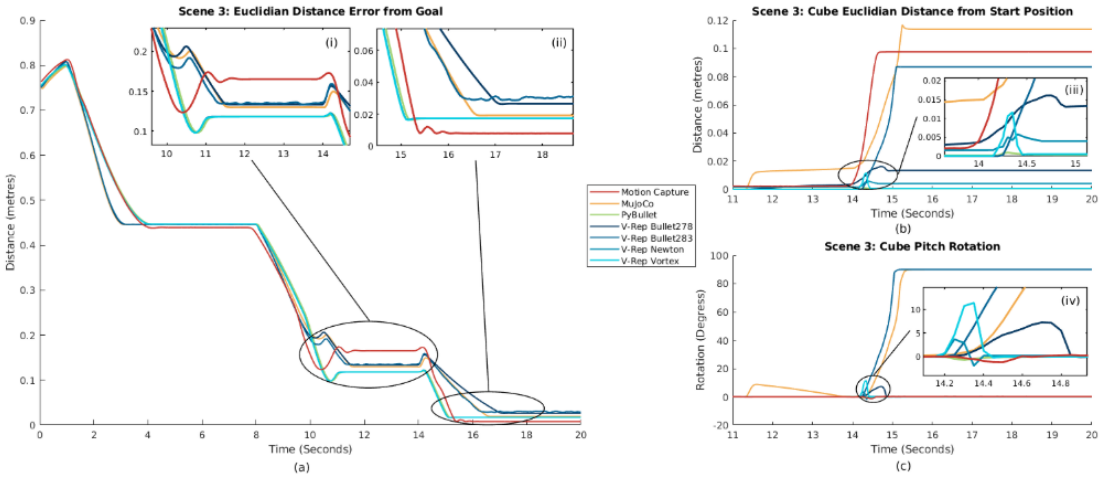


Figure 1.1: Results of the final test containing an interaction of a robotic arm with a cube in the paper from Collins et al. [3]

of the tested engines perfectly reproduced the smooth motion observed in the real arm, especially when moving against gravity.

The final and most complex scenario evaluated interaction with a cube, where the robot's arm approached the cube, made contact, and applied a gentle push. This task involved multiple joints and tested both kinematics and contact physics. As illustrated in Figure 1.1, none of the simulators were able to replicate the cube's movement with full accuracy. PyBullet showed the smallest positional error but failed to register contact, causing the arm to pass through the object. MuJoCo and V-Rep with Bullet simulated contact more realistically, yet exaggerated the cube's rotation compared to the minimal displacement recorded in the physical test.

The results clearly demonstrate that there are notable differences between the tested simulators and their physics engines. Some performed better in simple kinematic scenarios, while others handled multi-joint movements or dynamic effects more accurately. Ideally, one could imagine combining the strengths of different engines to achieve optimal performance across various tasks. However, the experiments also highlight that simulating the physical interaction between solid objects remains a major challenge and requires substantial improvement in future simulation frameworks.

1.2 Solving the reality gap

The gap between simulation and physical reality, commonly referred to as the reality gap, remains one of the central challenges in modern robotics. Numerous approaches have been developed to reduce this discrepancy, each focusing on a different source of error, such as imperfect dynamics, sensor noise, or inaccurate kinematic models. While

early work aimed primarily at identifying and quantifying the magnitude of the gap, current research is increasingly focused on developing systematic methods to minimize it.

1.2.1 Domain randomization

One of the most established and frequently used techniques is domain randomization, which exposes the controller to random variations during training to improve its robustness. By randomizing elements such as lighting, textures, mass, friction, or camera position, the robot learns to handle diverse conditions rather than overfitting to the simulated ones. This method has shown promising results, especially in robotic vision tasks, where randomized inputs lead to better generalization to real-world images. An extension of this idea is the randomised-to-canonical adaptation network, which maps both simulated and real-world observations into a shared canonical domain, improving consistency between them [17]. Related methods, such as domain adaptation and generative adversarial networks, operate on a similar principle, transforming the simulated and real data distributions to make them indistinguishable [3].

1.2.2 Adversarial reinforcement learning

Another effective method for narrowing the reality gap is adversarial reinforcement learning. In this framework, two agents are trained simultaneously: a protagonist, which learns to complete the desired task, and an antagonist, which introduces perturbations or environmental changes to challenge the protagonist. This setup encourages the learned policy to become more resilient to variations that may occur in the real world. A more advanced variant, robust adversarial reinforcement learning, further enhances reliability by letting the antagonist explicitly generate destabilizing disturbances in the environment, forcing the protagonist to adapt dynamically [17].

1.2.3 Transfer learning

A conceptually different approach is transfer learning, which aims not only to reduce the gap but to avoid its occurrence by gradually transitioning between simulation and reality. Training typically proceeds in two phases: first in simulation, then on the physical robot. The knowledge acquired during simulated training is reused in the real-world phase, significantly reducing the amount of real-world data needed. Transfer learning can be unidirectional, transferring information from simulation to reality, or bidirectional, where real-world feedback is also used to improve the simulator. In the latter case, the simulation becomes progressively more accurate, thus minimizing discrepancies between virtual and physical environments [17].

1.2.4 Simulation optimization and multi-simulator approaches

In contrast to randomization, simulation optimization focuses on refining the simulator itself to better reproduce physical reality. By adjusting the parameters of the physics model based on real-world data, such as joint velocities, friction coefficients, and time steps, the simulation can more accurately emulate actual robot dynamics. This approach, however, is often robot-specific and requires data collection from physical experiments [3].

Some researchers have also explored using multiple simulators or physics engines simultaneously. Since each simulator models physical phenomena differently, combining them can reduce the bias inherent in any single one. For instance, one simulator might handle rigid body dynamics more accurately, while another better captures frictional or compliant interactions. Using both allows for improved robustness across different motion types [3].

1.2.5 Simulation tuning

Closely related to optimization, simulation tuning involves the systematic adjustment of simulation parameters to match real-world behavior. Collins et al. [1] demonstrated that tuning key parameters, such as simulation time step, lateral friction, and joint velocity, using evolutionary optimization algorithms like differential evolution can significantly reduce the reality gap. Their experiments with PyBullet and V-Rep showed that properly tuned simulators could reproduce real-world object interactions with much higher accuracy, especially for contact-rich tasks.

1.2.6 Differentiable physics

With the advent of automatic differentiation libraries, differentiable physics has become a powerful new paradigm in robotic simulation. Differentiable physics engines allow gradients to propagate through the physics computations, enabling optimization via backpropagation. Collins et al. [2] introduced RealityGrad, a method that iteratively improves simulator accuracy through gradient-based optimization. The process includes collecting optimal trajectories in simulation, training control strategies, deploying them on the real robot, and using the collected real-world data to refine the simulator. By repeating this loop multiple times, the simulation model gradually converges toward physical reality, greatly reducing the gap between simulated and real behavior.

Chapter 2

Humanoid robot

A robot designed to resemble the human body is referred to as a humanoid robot. Its construction is typically inspired by human anatomy to enable interaction with tools created for human use and to promote a sense of comfort and familiarity during human–robot collaboration. Some humanoid robots replicate only certain parts of the body, such as the upper torso, while others include facial features like eyes or a mouth to facilitate natural communication. In many experimental and social contexts, the lower body is unnecessary, as interaction is focused primarily on gestures, gaze, and speech [15].

The development of humanoid robots has evolved over several decades. One of the earliest examples of a dynamically balanced humanoid robot dates back to the 1960s, when Ichiro Kato from Japan introduced the robot WABOT. It could communicate in Japanese, measure distances using external sensors, walk using its lower limbs, and manipulate objects through tactile feedback in its hands. This achievement was closely linked to the zero-moment point stability theory proposed by the Serbian engineer Miomir Vukobratović during the same period [7].

Today, humanoid robots are applied in a wide range of fields. In healthcare, they assist in surgeries, pain management, rehabilitation, and physical therapy. In education, they support learning processes through logical reasoning and situational analysis. Social and assistive robots are especially valuable for elderly and disabled individuals, serving as companions, entertainers, or household helpers. Interestingly, children with autism often find humanoid robots more approachable than humans, as their behavior is predictable and easier to interpret [15].

2.1 Robot NICO

To accomplish the objectives of this research, we selected the humanoid robot NICO as the experimental platform. The name NICO stands for Neuro-Inspired COmpanion,

reflecting its design as a medium-sized developmental robot intended for natural interaction with humans. NICO is capable of learning from both experience and instruction and is frequently used in studies of neurocognitive development and human–robot interaction [13]. The robot's software framework was developed by the Knowledge Technology Group at the University of Hamburg, and its complete source code and documentation are openly available to the research community [10].

NICO features a highly expressive face, with LED matrices positioned at the mouth and eyebrows that enable the display of a wide range of emotions. New facial expressions can easily be designed and added to its repertoire. The robot is equipped with two high-resolution cameras functioning as eyes, capable of capturing 4K video at 60 frames per second, and with two microphones, one in each ear, providing stereo audio input directly to the computer. Its RH4D manipulators allow for advanced grasping and object manipulation, while the body provides 10 degrees of freedom in the torso and 22 degrees of freedom in the arms [9].

The version of NICO used in this study is equipped with a touchscreen display (1920×1080 pixels) positioned in front of the robot 2.1. For the purpose of this work, the robot's index finger, serving as the end effector, was adapted to interact with the touchscreen. The fingertip was wrapped in aluminum foil connected to a conductive wire running along the forearm and covered with a piece of touchscreen-compatible glove material. This setup ensures the presence of an electric charge while maintaining the softness and conductivity necessary for reliable touchscreen recognition, effectively mimicking the touch of a human finger.

Furthermore, each of NICO's modules is connected via a dedicated USB interface, providing high modularity and flexibility. These features make NICO an ideal choice for experiments aimed at quantifying and optimizing the sim-to-real transfer in humanoid robots.

2.2 Simulator PyBullet

When selecting a suitable simulator for our experiment, we aimed for one that provided support for the NICO robot and allowed an easy transition from simulation to reality. Our initial choice was the Unity simulator, which offered access to a reasonably well-developed simulated version of NICO. However, the source code for this simulator was not publicly available, which limited the flexibility of modifications. Moreover, Unity did not provide a straightforward way to transfer control policies from the virtual environment to the physical robot, a crucial requirement for our experiments.

For the purposes of this work, we selected the PyBullet simulator as the main environment for our experiments. PyBullet is an open-source physics simulation platform

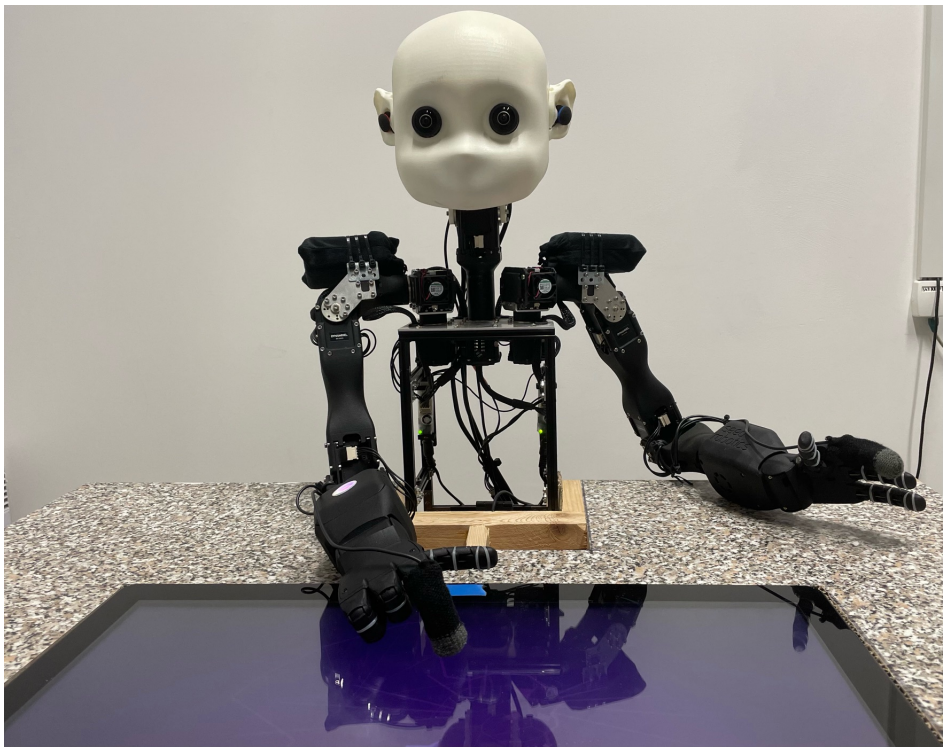


Figure 2.1: The real NICO robot at the Faculty of Mathematics, Physics and Informatics in Bratislava, equipped with a touchscreen and a fingertip (end effector) extension that enables interaction with the tablet.

built on the Bullet Physics Engine, which is widely used in both academic and industrial research. It provides accurate and efficient simulation of rigid-body dynamics, collision detection, contact modeling, and joint control, making it highly suitable for robotic applications involving manipulation and grasping [4].

Unlike some commercial simulators, PyBullet offers full access to its source code and a Python API that allows seamless integration with machine learning frameworks such as TensorFlow and PyTorch. This makes it extremely flexible for experimentation and parameter tuning. Another major advantage of PyBullet is its straightforward transition from simulation to the real world, as control policies and kinematic structures can be directly reused on physical robots.

The simulator also supports the import of custom robot models in standard formats such as URDF, which made it easy to load and configure the humanoid robot NICO for our setup 2.2. After importing the NICO model, we tested the basic functionalities, including arm movements and end-effector control relevant to our experiments. The environment proved to be very stable and responsive, and its simplicity allowed us to make modifications quickly and intuitively.

Thanks to its open architecture, realistic physics engine, and flexible control interface, PyBullet met all the requirements necessary for conducting our experiments on optimizing the sim-to-real transfer for the humanoid robot NICO.

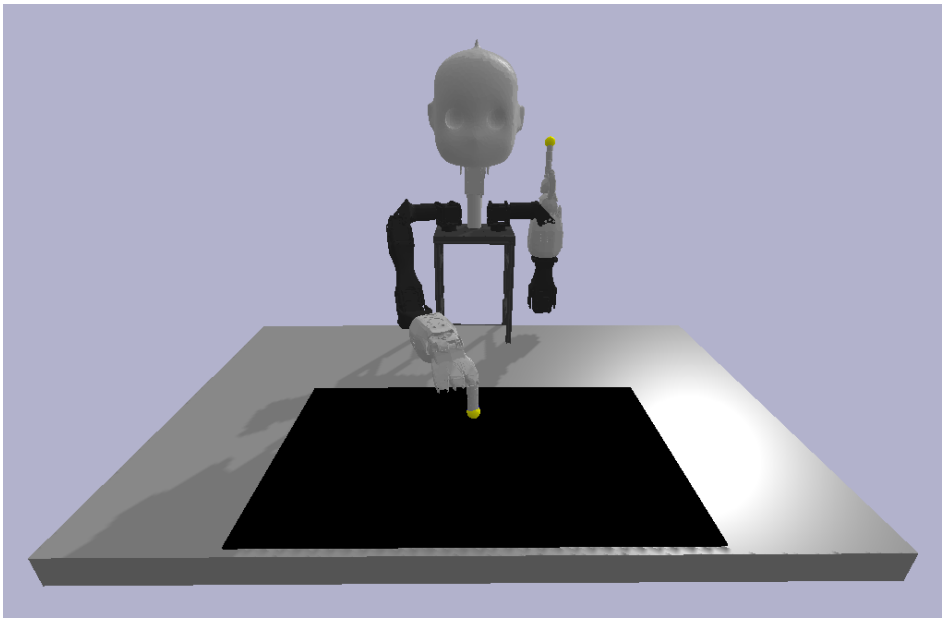


Figure 2.2: Simulated robot NICO with preconfigured setup using the PyBullet simulator.

Chapter 3

Kinematics

In classical mechanics, kinematics is the study of motion without considering the forces that cause it. It focuses on describing how the positions of bodies change over time and space within a continuous, three-dimensional Euclidean space. The main quantities of interest include position, velocity, acceleration, and trajectory.

Since movement is one of the fundamental aspects of any robotic mechanism, kinematics represents a cornerstone of robot design, analysis, control, and simulation. In robotics, kinematics deals with the motion of interconnected rigid bodies, describing how the position and orientation of each component change as the robot moves. Researchers in this field focus on finding efficient mathematical representations of spatial transformations and their time derivatives, which are essential for modeling and controlling robotic motion [20].

Two of the key problems studied in this context are forward kinematics, determining the end-effector's position and orientation from given joint angles, and inverse kinematics, which calculates the required joint configurations for a desired end-effector pose. Both are central to the movement and control of robotic arms and were directly applied in this work.

3.1 Robotic arm

A robotic arm is one of the most common types of manipulators, a mechanical device designed to handle objects without direct physical contact from a human operator. It is typically composed of several rigid links connected by joints. When arranged in series, these links and joints form a kinematic chain, which is usually attached to a fixed base or to the body of a humanoid robot.

At the opposite end of the chain is the end effector, the component responsible for interacting with objects in the environment. Each joint of the robotic arm contributes a certain degree of freedom, defining how the arm can move or rotate. In a three-

dimensional space, there are three translational and three rotational degrees of freedom, one for movement along each axis and one for rotation about it. To maintain simplicity and control stability, it is standard practice to assign only one degree of freedom per joint [5].

3.2 Forward kinematics

The forward kinematics problem describes the process of determining the position and orientation of a robot's end effector relative to its base, given the joint positions and the geometric parameters of the mechanism. The number of degrees of freedom of the manipulator corresponds to the number of joints in the kinematic chain.

Each joint is typically equipped with sensors that measure its current configuration. However, in many cases it is necessary to compute the absolute position and orientation of the joints with respect to a fixed reference frame. For this reason, forward kinematics is a critical component in robotic control and simulation.

In practice, this problem is solved by calculating a transformation matrix between the end effector and the fixed base. The end effector's pose is obtained by multiplying the homogeneous transformations of consecutive links in the kinematic chain. This effectively converts the manipulator's configuration from joint space into Cartesian space, representing the spatial position and orientation of the end effector [19].

3.3 Inverse kinematics

The inverse kinematics problem is the counterpart of forward kinematics. For a serial robotic manipulator, it involves determining the joint parameters required to achieve a desired position and orientation of the end effector relative to the base, given the geometric structure of the kinematic chain. Unlike forward kinematics, the inverse problem is non-deterministic, a single end-effector pose often corresponds to multiple joint configurations. As a result, its computation is typically more complex and time-consuming. The task represents a conversion from Cartesian space, defined by the end effector's position and orientation, to joint space, the set of joint variables that realize that pose [22].

Over several decades of research, many methods for solving the inverse kinematics problem have been developed. These approaches can generally be divided into two main categories: analytical and numerical methods. Analytical solutions use closed-form equations that take the end effector's position as input and return a vector of joint angles as output. However, such solutions are only applicable to certain manipulator structures, since the problem often has an infinite number of valid configurations. In

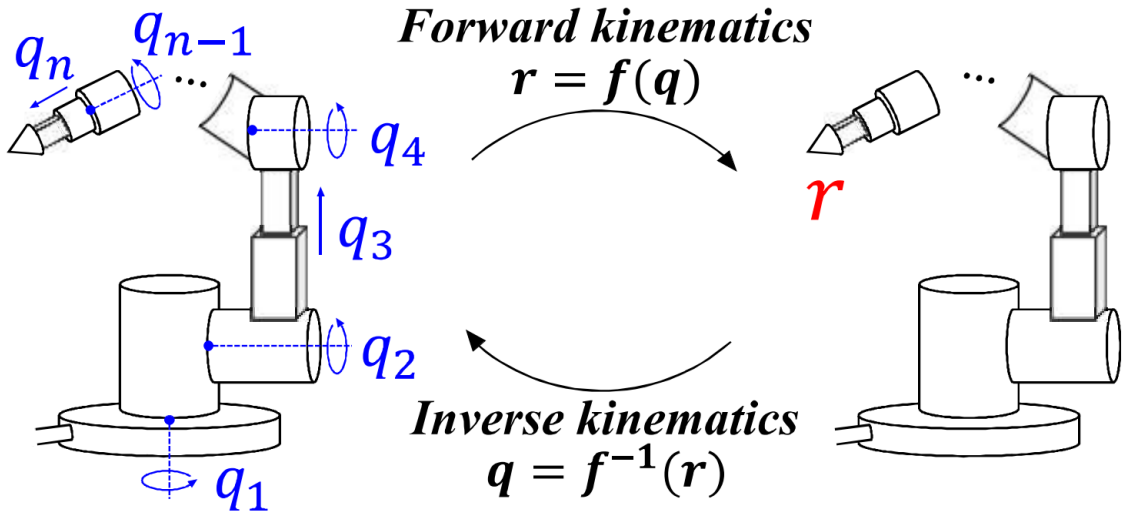


Figure 3.1: Visualisation of forward and inverse kinematics in the work of Yonezawa et al. [21].

these cases, numerical or iterative approaches are employed, which gradually optimize the result through successive approximations. Among the most widely used numerical methods are the Jacobian-based methods and the Levenberg–Marquardt algorithm, both of which iteratively minimize the error between the current and desired end-effector positions [22].

Figure 3.1 illustrates the relationship between joint space and effector (Cartesian) space: the vector q represents the joint configuration of the manipulator from the base to the end effector, while r denotes the end effector's position in Cartesian coordinates. The function $f(q)$ defines forward kinematics, mapping joint angles to the end-effector pose, whereas $f^{-1}(r)$ represents its inverse counterpart.

Chapter 4

Thesis objectives

Chapter 5

Proposed methods

Chapter 6

Implementation

Chapter 7

Results

First, we looked at the results of the first objective, two-dimensional calibration using haptic feedback. In our case, the touchscreen provides haptic feedback while robot performs the reach movement finishing with end effector touching the tablet.

The model M1 serves as the baseline for our research. We applied M1 with a 1-sec. movement to grid-based points, and plotted the contact points on the screen relative to the intended targets. The results in the upper half of Fig. 7.1 reveal that the direction of each deviation rotates depending on the target position relative to the arm base. In addition, the magnitude of the deviation increases considerably with the target distance from the base (i.e. for more stretched arm).

A similar relationship can be observed in lower half of Fig. 7.1, showing 2-sec movements. By comparison, we see that the average deviation decreased from 2.16 cm (for 1-sec) to 1.17 cm (for 2-sec), demonstrating that slower movements of NICO produce smaller errors. Further, we can see that the rotation trend is not as visible here at a 1-sec duration. Despite the slower and more precise movements, the direction of deviation from the target is less regular. 3-sec movements had very similar results to 2-sec movements.

Next, we measured 2D deviations from the target on the touchscreen at three different movement durations for each quadrant, as shown in Fig. 7.2. We can see that 1-sec movements are on average twice as inaccurate as slower movements, which may be due to overshooting the target. Movements of 2 and 3 seconds reveal similar behavior. We can also see that targets in Q1 and Q2, being closer to the robot base, yield a smaller error compared to more distant Q3 and Q4.

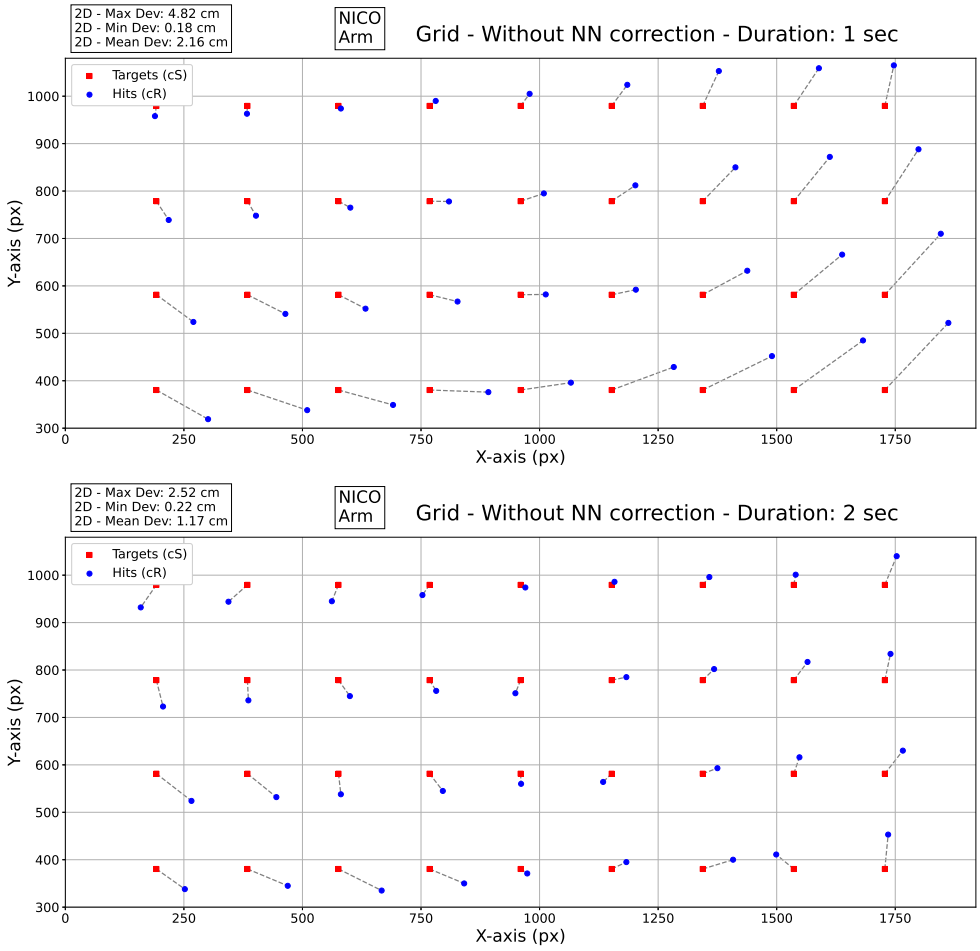


Figure 7.1: Plot of deviations of hits (blue) from grid-based targets (red) in the physical space (on the touchscreen). We used the model M1 and NICO arm movement of 1-sec (top) and 2-sec (bottom) duration. The box "NICO Arm" denotes the base position.

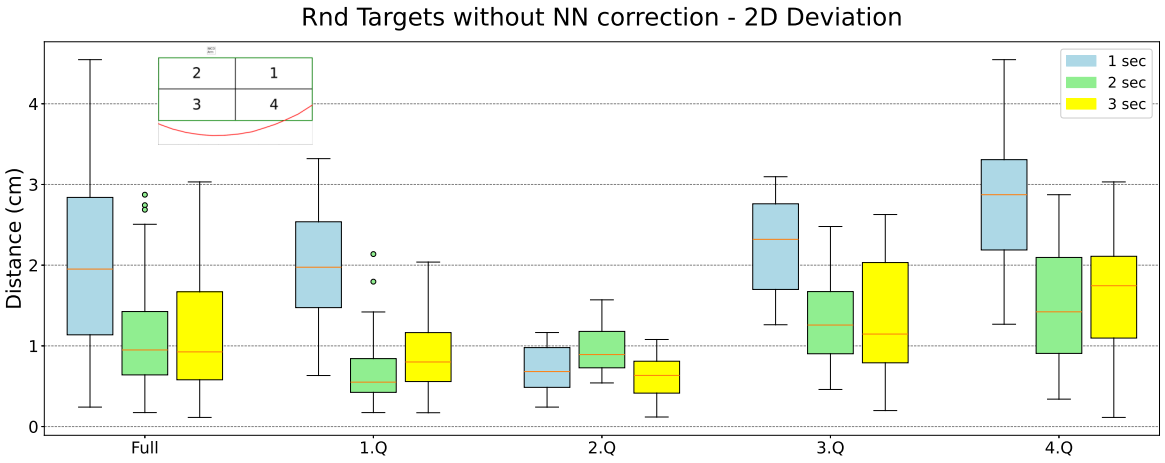


Figure 7.2: Comparison of 2D deviations of hits from random targets within four quadrants for three durations of movement. Targets were calculated using M1. Small inset image top left shows the NICO reach limit (red line) and partitioning of the testing area.

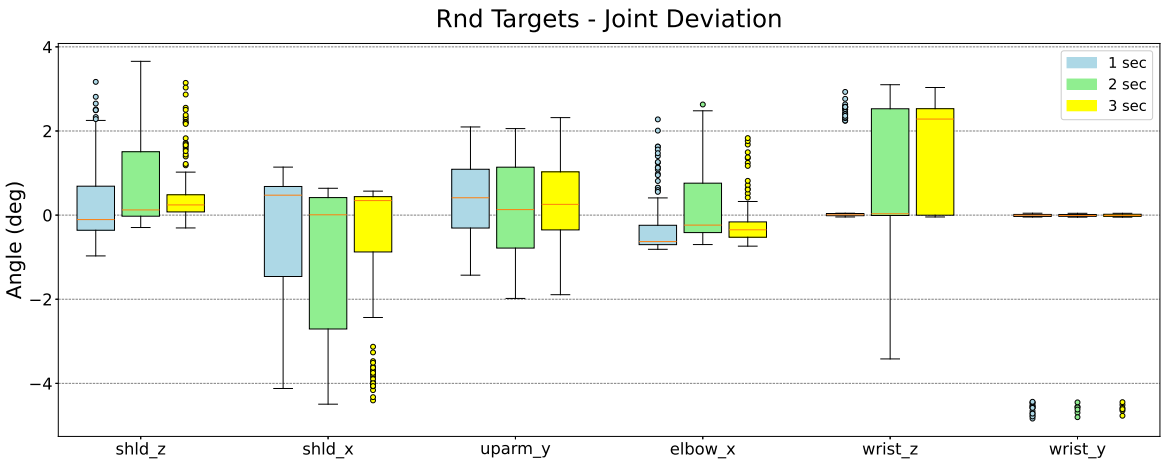


Figure 7.3: Comparison of joint angle deviations for three different durations of movement. Joint deviation is calculated as a difference of joint angles returned from inverse kinematics (angles sent to robot) and joint angles read from the robot after the end of movement (using M1).

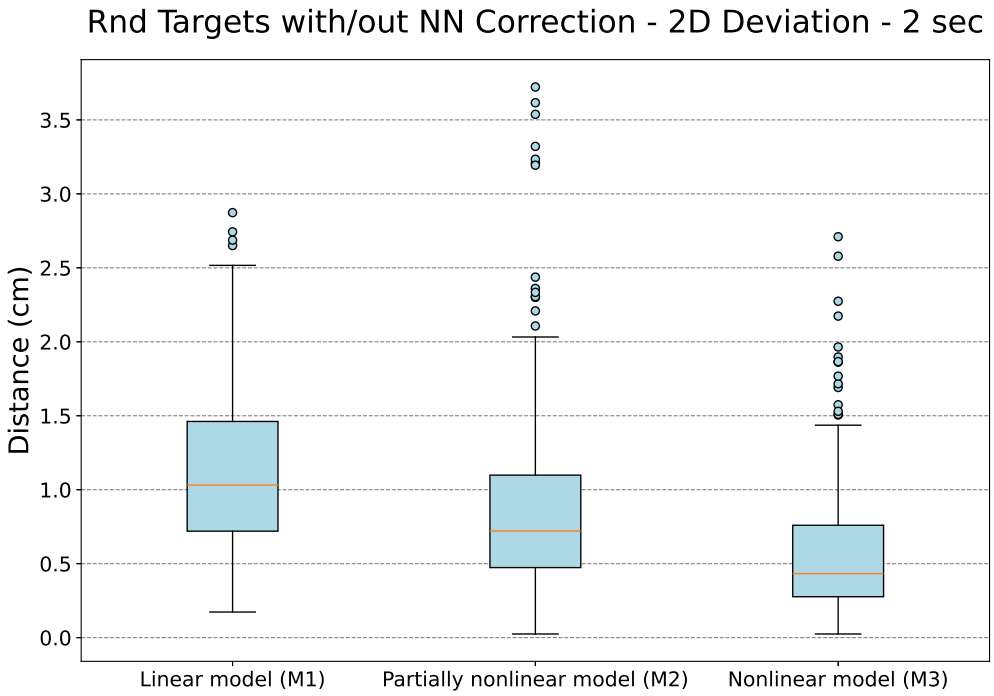


Figure 7.4: Comparison of 2D deviations (Euclidean distances) of individual model predictions for random targets.

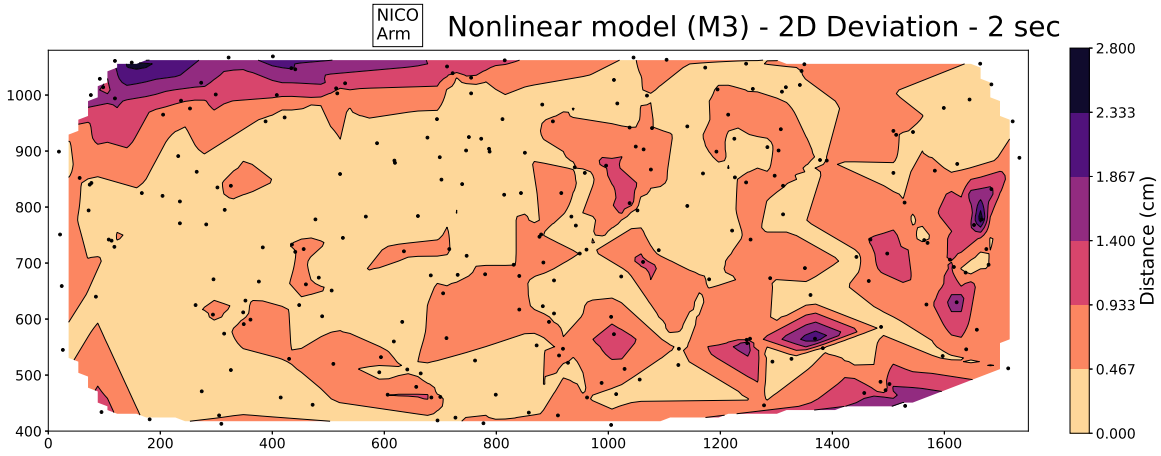


Figure 7.5: Accuracy of NICO robot after the neural network based correction, trained to predict 3D points in the simulator space from target 2D points on the touchscreen. Accuracy is calculated as a deviation in 2D space.

Conclusion

Bibliography

[1] Jack Collins, Ross Brown, Jurgen Leitner, and David Howard. Traversing the reality gap via simulator tuning, 2020.

[2] Jack Collins, Ross Brown, Jürgen Leitner, and David Howard. Follow the gradient: Crossing the reality gap using differentiable physics (realitygrad), 2021.

[3] Jack Collins, David Howard, and Jürgen Leitner. Quantifying the reality gap in robotic manipulation tasks, 2018.

[4] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[5] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson Prentice Hall, Upper Saddle River, NJ, 3 edition, 2005.

[6] Connor Gäde, Jan-Gerrit Habekost, and Stefan Wermter. Domain adaption as auxiliary task for sim-to-real transfer in vision-based neuro-robotic control. In *IJCNN*. IEEE, 2024.

[7] Waseda University Humanoid Robotics Institute. Wabot - waseda robot-. Available 10.12.2025.

[8] Ramil Khusainov, Alexandr Klimchik, and Evgeni Magid. Humanoid robot kinematic calibration using industrial manipulator. In *2017 International Conference on Mechanical, System and Control Engineering (ICMSC)*, pages 184–189, 2017.

[9] University of Hamburg Knowledge Technology Group. Nico neuro-inspired companion — seed robotics. Available 10.12.2025.

[10] Knowledgetechnologyuhh. Github - knowledgetechnologyuhh/nico-software: Software to run the nico (neuro inspired companion) robot. Available 10.12.2025.

[11] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, 2016.

[12] Quentin Le Lidec, Wilson Jallet, Louis Montaut, Ivan Laptev, Cordelia Schmid, and Justin Carpentier. Contact models in robotics: a comparative analysis, 2024.

[13] Hwei Geok Ng, Paul Anton, Marc Brügger, Nikhil Churamani, Erik Fließwasser, Thomas Hummel, Julius Mayer, Waleed Mustafa, Nguyn Linh Chi, Nguyen Quan, Marcus Soll, Sebastian Springenberg, Sascha Griffiths, Stefan Heinrich, Nicolás Navarro-Guerrero, Erik Strahl, Johannes Twiefel, Cornelius Weber, and Stefan Wermter. Hey robot, why don't you talk to me? 08 2017.

[14] Vicente Pedro. Real time graphical simulation for visual based pose estimation and self-calibrating of a humanoid robotic arm. Master's thesis, Instituto Superior Técnico, Lisbon, 2014.

[15] Prasanna Rasal. Humanoid robotics. 9:128–130, 06 2021.

[16] Alessandro Roncone, Matej Hoffmann, Ugo Pattacini, and Giorgio Metta. Automatic kinematic chain calibration using artificial skin: self-touch in the iCub humanoid robot. In *ICRA*, pages 2305–2312. IEEE, 2014.

[17] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, 9:153171–153187, 2021.

[18] Karla Stepanova, Tomas Pajdla, and Matej Hoffmann. Robot self-calibration using multiple kinematic chains—a simulation study on the icub humanoid robot. *IEEE Robotics and Automation Letters*, 4(2):1900–1907, 2019.

[19] Tony Punnoose Valayil and Rose Shaji Augustine. Methods to solve forward kinematics of parallel and serial manipulators. *AIP Conference Proceedings*, 2670(1):030003, 12 2022.

[20] Kenneth J. Waldron and James Schmiedeler. *Kinematics*, pages 11–36. Springer International Publishing, Cham, 2016.

[21] Ansei Yonezawa, Heisei Yonezawa, and Itsuro Kajiwara. Simple inverse kinematics computation considering joint motion efficiency. *IEEE Transactions on Cybernetics*, pages 1–12, 2024.

[22] Chengyi Zhao, Yimin Wei, Junfeng Xiao, Yong Sun, Dongxing Zhang, Qiuquan Guo, and Jun Yang. Inverse kinematics solution and control method of 6-degree-of-freedom manipulator based on deep reinforcement learning. *Scientific Reports*, 14(1):12467, 2024.