Submission to Student Conference

Simulation Framework for a Collision Avoidance Inverse Kinematics model of a 7 degrees of freedom Ultrasound Imaging Robot

Luis Manuel González Villa [a] · Tolga-Can Çallar [b]

[a] Student of Robotics and Autonomous Systems, Universität zu Lübeck, Lübeck, Germany
[b] Institute of Robotics and Cognitive Systems, Universität zu Lübeck, Germany
*Corresponding author, email: luis.gonzalezvilla@student.uni-luebeck.de; t.callar@uni-luebeck.de

Abstract

In this project, a simulation framework for a collision-avoidance inverse kinematics model for a 7-DoF KUKA robot is being developed. The robot is intended to be used for ultrasound imaging of patients at the Institute of Robotics. A Proximal Policy Optimization (PPO) Reinforcement Learning (RL) algorithm is utilized to train the desired model, using a CoppeliaSim simulation as the environment and PyTorch for the implementation of the PPO method. Observations from the environment are obtained through a camera attached to the robot's end effector, with the captured image data serving as input for the algorithm. While existing inverse kinematics models focus on guiding robot movements, there is insufficient research on models for preventing unwanted collisions with patients during ultrasound imaging procedures. To address this challenge and avoid the need for ground-truth data, a RL approach was chosen. This allows the robot to learn how to avoid collisions in real time, even in the presence of patient movements.

I. Introduction

Ultrasound imaging is a clinical procedure used in medicine to visualize internal areas of interest in a patient by producing cross-sectional views of anatomical structures, enabling more precise diagnoses. This is achieved using an ultrasound sensor that operates based on the Doppler effect. The sensor emits wave signals that penetrate the body and captures the reflected signals, functioning as both a speaker and a microphone [1]. Typically, this procedure is performed by a human rather than a robot due to several challenges, including the risk of a collision between the robot and the patient caused by unintended movements of a limb or other body parts.

In recent years, significant research has focused on the development of Inverse Kinematics (IK) control for articulated robot arms using real-time images captured by a camera. This control methodology is known as Real-Time Visual Servoing (RTVS). Depending on the strategy employed, there are two primary variations: Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS). For this work, we focus on applying the IBVS approach. To analyze the robot's environment, images captured by the camera are typically processed using an algorithm to compute optical flow and derive the 3D world coordinates of detected objects [2]. These coordinates are then used in the IK process to guide the robot toward the desired goal point.

One commonly used optical flow computation method is the Scale-Invariant Feature Transformation (SIFT). This method identifies similar features between a reference (ground-truth) image and a newly captured image. Reference [3] demonstrates the application of

SIFT for RTVS in controlling a 6-degree-of-freedom (DoF) robot arm, achieving successful convergence of the proposed control to the reference pose in simulation. However, such methods rely on collecting extensive ground-truth data, requiring the acquisition of numerous images or fine-tuning the upper layers of a pretrained neural network to perform a different task [4]. Furthermore, these methods lack the capability to address real-time collision avoidance effectively.

An alternative is to use unsupervised learning techniques, which eliminate the need for ground-truth values. Instead, a Convolutional Neural Network (CNN) can be trained to learn how to move the robot based solely on the camera-provided images. Several studies have explored the use of unsupervised methods for robot inverse kinematics (IK). Reference [5] presents a successful trajectory optimization planning model that leverages a trained diffuser. This model employs a U-Net to predict all time steps of a planned trajectory simultaneously by denoising an initial trajectory sampled from a set of optimal trajectories. Research has also been conducted on multi-objective trajectory planning for 6-DoF robotic arms, with the aim of improving accuracy and smoothness while reducing energy consumption. Proximal Policy Optimization (PPO) has been employed for this purpose, demonstrating an effective and robust approach [6].

II. Deep Neural Networks Framework

To present the methodology used in this work, the first three subsections outline the theoretical concepts, while the following ones describe their implementation.

II.I. U-Net

It has been decided to utilize a U-Net Convolutional Neural Network (CNN). This type of network is commonly used for image segmentation applications, especially in biomedical image processing. Its significance lies in its ability to propagate contextual information to higher-resolution layers, thanks to its unique architecture [7].

The network is designed to learn how to navigate to a user-selected area of interest for ultrasound placement while avoiding unwanted collisions with the patient. It takes as input a 256x256-pixel RGB camera image, along with the current distance from the end-effector to the goal point as auxiliary input. The output of the network is the predicted set of seven joint positions that guide the robot toward the desired area. It has four encoding layers, $256x256 \rightarrow 128x128 \rightarrow 64x64 \rightarrow 32x32$, the bottleneck that processes 512 features and outputs 1024 features at a resolution of $16x16$. The expansive path has four decoders that start from 1024 + auxiliary features with the form $32x32 \rightarrow 64x64 \rightarrow 128x128 \rightarrow 256x256$. Finally, the final layer reduces the batch size to 7, as seen in Figure 1.

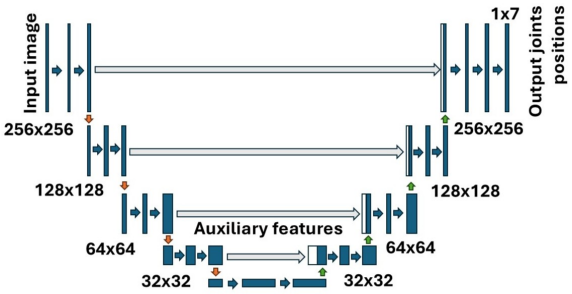

Figure 1: U-Net architecture of the robots network. It takes as an input the RGB image captured by the robot's end-effector, with a resolution of 256x256. A series of encoding layers compress the information until reaching the bottleneck. Afterwards, information is processed through the decoding layers, while retaining auxiliary features. The final output are the desired joint positions of the robot.

II.II. Proximal Policy Optimization

It is decided to use the Proximal Policy Optimization (PPO) algorithm for this project, as it offers a straightforward and effective solution for reinforcement learning tasks. Additionally, PPO is well-documented and widely adopted, with readily available implementations in PyTorch, which simplifies its integration into the system.

Deep Q-learning (DQN) is one of the most popular methods in RL due to its effectiveness in solving simple problems and its relatively straightforward implementation. However, DQN has notable disadvantages, particularly in terms of stability when dealing with complex problems or environments with many simple problems. Other approaches, such as "vanilla" policy gradient methods and trust region/natural policy gradient methods, also exhibit significant limitations. Vanilla policy gradient methods suffer from poor data efficiency and lack robustness, while trust region/natural policy gradient methods are often complex and struggle to handle problems involving noise in the architecture [8].

PPO aims to address the limitations of earlier reinforcement learning methods by achieving performance comparable to DQNs while eliminating the disadvantages discussed earlier. As a quick background, building on a policy gradient approach, policy gradients methods compute an estimate of the policy and incorporates it into a stochastic gradient ascent algorithm, as in Equation 1:

$$\hat{g} = \mathbf{E}\left[\nabla_\theta log \pi_\theta(a_t|s_t)\hat{A}_t\right] \tag{1}$$

Here, $\hat{A}_t$ is the advantage function estimator, which

measures how favorable an action a is compared to the average action at a given state s, and π_θ is an stochastic policy.

With trust region methods, the idea of a surrogate objective function is introduced, which aims to be maximized subject to a constraint. This surrogate objective can be solved using a conjugate gradient algorithm after performing a linear approximation to the objective function and a quadratic approximation to its constraint. Alternatively, the problem can be addressed by using a penalty term β instead of a strict constraint, as shown in Equation 2, where KL denotes the Kullback–Leibler divergence.

$$max_\theta \hat{\mathbf{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t - \beta KL\left[\pi_{\theta_{old}}(.|s_t), \pi_\theta(.|s_t)\right] \right] \quad (2)$$

To further improve the surrogate objective, a clipping probability ratio r_t component is introduced. This ensures that the ratio remains within the interval $[1-\epsilon, 1+t]$. This results in our PPO Equation as in 3.

$$L^{CLIP}(\theta) = \hat{\mathbf{E}}\left[min(r_t(\theta_t))\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right] \quad (3)$$

With $r_t(\theta)$ being:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad (4)$$

II.III. PPO Algorithm

Reference [8] gives a straight forward implementation of the PPO method, whose algorithm is shown in Figure 2.

for iteration=$1, 2, \ldots$ **do**
  for actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$ $\theta_{\text{old}} \leftarrow \theta$
  $\theta_{\text{old}} \leftarrow \theta$
end for

Figure 2: PPO algorithm taken from [8]. For each iteration, N parallel actors collect T timesteps of data. Then, the surrogate loss is constructed and optimized.

The reward function, as in Equation 5, is designed to benefit close distances between the end effector of the robot and the goal point, and greater distances between the links of the robot and the patient. Also, it penalizes collisions and if motion violates the robot kinematic constraints.

$$R = \left(w_1(aDEG)^2\right) + \left(w_2(bDJM)^2\right) - (w_3 col) \quad (5)$$

Where a and b are scaling factors to control the sensitivity of the measured distances, and w_1, w_2 and w_3 are weight factors. DEG is the measured distance between the end effector and the goal point, and DJM is the measured distance between the robot links and the mannequin.

II.IV. CoppeliaSim

The environment simulation is conducted in CoppeliaSim. The KUKA robot is placed on a table, mimicking its real-life setup in the laboratory. Additionally, a mannequin representing the patient is positioned on an examination table, and it performs random limb movements in the simulation. A camera is attached to the robot's last link, providing a view of the environment from the robot's perspective. The implementation is shown in Figure 3.

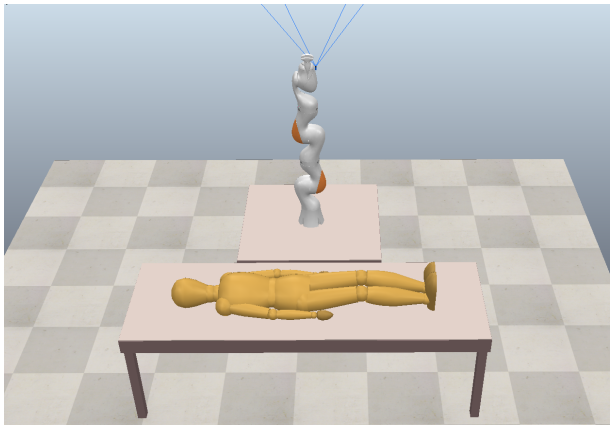

Figure 3: CoppeliaSim representation of the environment for simulation.

A script is written in Python that implements the remote API client connection and initializes or stops the simulation. It also includes all the necessary functions for moving the robot's joints, controlling the mannequin's limb movements, and capturing the camera image. To simplify the task, a KUKA class is implemented.

II.V. Gymnasium Open AI

Gymnasium is an open-source Python library used for RL simulations. It provides pre-implemented environments that can be easily configured with PyTorch due to their compatibility. Additionally, it allows for the creation of custom environments, making it easier to implement PyTorch features as shown in its official documentation. Hence, the environment script in CoppeliaSim is extended with functions for retrieving observations, resetting, and stepping through the simulation, as well as components for initializing the Gym environment. After

registering the environment, it can be used with the PPO method.

III. Results

A set of scripts containing all necessary functions and the CoppeliaSim scene for the desired simulation have been developed and can be obtained from this GitHub repository: Simulation framework.

III.I. U-Net

A U-Net architecture was correctly implemented in a class that can be called by the PPO class script. It can be initialized with a desired input dimension, output dimension, and alpha value. Additionally, for training purposes, the class includes functions that allow saving and loading checkpoints using the 'torch.save' method.

III.II. PPO pytorch

A class for PPO method was developed following the instructions given in PyTorch's official documentation. This class calls both KUKA and U-Net classes initializing them by setting the initial goal point to be chased, and U-Net dimensions. Also, it initializes PPO algorithm parameters, being 1000 frames per batch, 50000 total frames, 10 epochs, 0.2 for the clip value used on the loss, $\gamma = 0.99$ discount factor, $\lambda = 0.95$ learning rate, $\epsilon = 0.2$ clipping parameter, and $\alpha = 0.1$.

III.III. CoppeliaSim implementation

A successful API connection between CoppeliaSim and Python was established using CoppeliaSim's API, enabling seamless communication between the simulation environment and the PPO method script for training purposes.

Additionally, a function was implemented to stream video and retrieve images from the robot camera attached to the end effector. These images are subsequently processed and converted into PyTorch tensors, preparing them for integration with the PPO algorithm. Functions were implemented to simulate random movements of the mannequin, mimicking a patient performing unpredictable motions, and to enable the robot to process and execute received joint vectors as actions, thereby controlling its movement.

Finally, all necessary functions for gym-type environment registration were successfully implemented. Specifically, these include functions to obtain observations from the environment, reset the environment with random initialization, and set the next step, which returns both the reward and updated environment information after an action has been applied.

IV. Conclusion

Through this work, a simulation framework was developed to support future research focused on this topic. Various methods were studied for performing the robot's inverse kinematics without colliding with the patient. Most of these methods yielded similar results. However, PPO was considered the best option due to its flexibility with PyTorch's pre-implemented features and its lack of need for ground-truth values.

The next steps for this work involve performing a simulation using the developed framework and analyzing the results. Additionally, the implementation of the resulting simulation on the KUKA in the laboratory will be evaluated. However, the intended research has been completed, and the simulation framework is now ready to be utilized.

Acknowledgments

The work has been carried out at the Institute of Robotics and Cognitive Systems, Universität zu Lübeck.

Author's statement

Conflict of interest: Authors state no conflict of interest. ChatGPT was used for the linguistic fine-tuning of this work.

References

[1] V. Chan and A. Perlas. Basics of ultrasound imaging. *Atlas of ultrasound-guided procedures in interventional pain management*, pp. 13–19, 2011, doi:10.1007/978-1-4419-1681-5_2.

[2] P. Allen, B. Yoshimi, and A. Timcenko, Real-time visual servoing, in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 851–856 vol.1, 1991. doi:10.1109/ROBOT.1991.131694.

[3] F. Hoffmann, T. Nierobisch, T. Seyffarth, and G. Rudolph, Visual servoing with moments of sift features, in *2006 IEEE International Conference on Systems, Man and Cybernetics*, 5, 4262–4267, 2006. doi:10.1109/ICSMC.2006.384804.

[4] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, and P. Corke, Visual servoing from deep neural networks, 2017. doi:10.48550/arXiv.1705.08940.

[5] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, Planning with diffusion for flexible behavior synthesis, 2022. doi:10.48550/arXiv.2205.09991.

[6] S. Zhang, Q. Xia, M. Chen, and S. Cheng. Multi-objective optimal trajectory planning for robotic arms using deep reinforcement learning. *Sensors*, 23(13), 2023, doi:10.3390/s23135974.

[7] O. Ronneberger, P. Fischer, and T. Brox, U-net: Convolutional networks for biomedical image segmentation, in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds., 234–241, Cham: Springer International Publishing, 2015. doi:10.1007/978-3-319-24574-4_28.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, 2017. doi:10.48550/arXiv.1707.06347.