BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

SPECIALIZATION COMPUTER SCIENCE

BACHELOR THESIS

Robot Arm Inverse Kinematics Solver using Reinforcement Learning

Supervisor

Assoc.Prof. Rares Florin Boian, Ph.D.

Author

Andrei-Viorel GRIGORESCU

2023

UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA

FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENŢĂ

Rezolvarea unui sistem cinematic invers pentru un braț robotic folosind învățare prin întărire

Conducător ştiinţific

Conf.Dr. Rares Florin Boian

Absolvent

Andrei-Viorel GRIGORESCU

2023

Abstract

This work reviews some of the methods that have been proposed in the literature as solutions to the problem of inverse kinematics, which is introduced in Chapter 1 as being under-constrained and having multiple solutions which need to be selected according to properties desirable for the application of interest. In Chapter 2, after reviewing the classic methods for solving the inverse kinematics problem, which have been successfully employed in a number of domains such as computer graphics and robotics, attention will shift towards more recent trends in the proposed methods, such as data-based and hybrid ones. The accent will be put on solutions that make use of the reinforcement learning paradigm, and in Chapter 4 a study will be carried to review and compare the results of Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) learning algorithms with respect to the task of solving for a set of joint angles so that the motions produced result in the end effector of a robot arm reaching the desired destination, following experimental setup descriptions offered in Chapters 3. The possibility of implementing additional functionalities and constraints, such as desired end effector orientation and motion smoothness, will be verified. The results obtained, having insufficient precision but capability of recovering from more difficult configurations, have suggested implementation of a hybrid method in Chapter 5, which is then quantitatively and qualitatively compared to the traditional method for success rate of reaching the target position. Finally, conclusions are emitted in Chapter 6 regarding the proposed endeavor.

Originality statement

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

Table of Contents

1. Introduction

Inverse kinematics problem

Many situations present a system composed of segments or links of known lengths, connected through rotational joints. For this system, a kinematic chain called “arm”, we can find the position of the end effector, or tip of the arm, by using forward kinematics, which leverages the knowledge about the system and the relative angles between the segments (or the relative angles in the joints). The problem of inverse kinematics is that of finding the joint angles for a known system that lead towards the desired end effector position. Compared to the problem of forward kinematics, inverse kinematics has multiple possible solutions, being under-constrained, especially in a tridimensional coordinate frame.

At a basic level, the role of inverse kinematics is to offer a higher-level method of controlling such a system, abstracting the need to specify relative angles for each individual joint. In highly redundant systems, which have numerous such joints, this approach might even prove to be necessary. Moreover, the redundancy of such systems allows for the surplus degrees of freedom to be used in order to satisfy other constraints in operation, such as respecting an indicated orientation, maintaining a constant speed, following a given path or avoiding obstacles in the environment.

The two main industries where inverse kinematics solvers are used are those of computer graphics and robotics. The first employs the use of inverse kinematics either in editors, to simplify the task of animating characters, including but not limited to humanoid ones, or in procedural animation, where the configuration of a target end effector position can lead to more complex interactions with the virtual environment. The other domain, that of robotics, uses the inverse kinematics solver to verify that the desired end effector position is reachable, and then to derive the necessary joint actuations so that the end effector reaches the desired destination. This task might present several constraints, such as limited joint rotation angles, velocity or avoidance of obstacles while pursuing the target configuration.

Of course, the problem of inverse kinematics appears in other domains as well, one example being that of pose estimation [1]. This concept describes a method that allows the pose of a human, their hands, or other subjects to be reconstructed after analyzing images or video footage.

Reinforcement learning applications

Reinforcement learning represents a paradigm in artificial intelligence, separated from that of supervised or unsupervised learning. Considering an agent, an environment, an observation (complete or incomplete) of the environment, available actions and a function that describes the reward or punishment based on the performance of an action from a state to another, reinforcement learning methods attempt to learn an approximation of the reward function through trial and error, building experience. This is done so that the agent can learn a policy that maximizes the reward function or minimizes punishment, as to obtain an optimal behavior in the environment. Elements of reinforcement learning that were previously described can be seen as a Markov decision process, which the methods attempt to learn and to exploit in order to produce best results.

This paradigm has seen numerous developments in the recent years, attracting even more interest in application in domains ranging from playing games such as chess or videogames [2], to discovery of new proteins by solving the complex problem of protein folding [3], representing significant advances for biomolecular medicine. Reinforcement learning has also seen numerous applications in robotics. In combination with imitation learning and adversarial motion priors, a control policy was learnt for the control of a quadrupedal robot [4]. Training took place in a physically accurate simulation of the robot, and iterations (or episodes) were accelerated by modern parallel computing methods. The policy was then adapted to the physical robot, demonstrating robust behavior even when subjected to real-life conditions and challenges. This demonstrates learning of a flexible and efficient policy which would have been otherwise extremely complicated to implement through regular means.

Figure 1 Stand up-sequence in simulation and on the real robot. The policy is able to stand up, navigate large distances on two legs, and finally sit down. [4]

Proposed approach

The goal of this paper is to investigate whether an approach based on reinforcement learning is justified to solve the problem of inverse kinematics. There are a number of related works, such as one investigating several methods from artificial intelligence [5], or a similar study [6] that solves the problem of robot arm trajectory planning not by solving the joint angles, but by indicating a displacement relative to the end effector, that is then resolved with existing inverse kinematics solutions. In the aforementioned work, this was done as to reduce the dimensionality of the inputs in the problem, but the current work intends to verify whether the original, higher-dimensional problem of inverse kinematics can be solved directly. A qualitative comparison of reinforcement learning methods will be carried, and one method will be chosen for training of a control policy.

Summary of chapters

In this paper, a literature review will follow that will investigate results obtained so far, observing that this problem has seen intense activity and numerous trends in the types of solutions proposed. The types of methods covered are classic methods, based on analytical, numerical, or heuristic methods, recent evolutions such as an algorithm that relies on dynamic programming [7], but also mentioned are data-driven methods. Attention is then given to artificial intelligence-based or otherwise learnt methods (which are referred in [5] as soft computing methods), highlighting the results obtained, but also suggested improvements. Finally, the literature review concludes with the review of a study [6] that uses a similar setup to the one intended to be used in this paper, and differences between approaches and experiments are mentioned.

The next chamber proceeds to describe the proposed approach with a finer level of detail, covering reinforcement learning methods, environments, and the defined tasks. Implementation details are also enumerated, exposing the choice for the environment-algorithm interface (OpenAI Gym), reinforcement learning algorithms implementation and training environment.

Two case studies are presented, both containing the experiments ran, evaluation metrics used, results obtained and a discussion around these results. The first case study tests and compares two reinforcement learning algorithms, one of which is on policy and the other off policy. Additionally, the second case study, which is derived from the results of the first, contains further implementation details of a refined, hybrid method and the changes made to the environment to support evaluation of the desired metrics. The second experiment compares

the success rate of the hybrid method compared to a traditional method offered by the environment implementation. Results are further discussed, and a mention is done in regards to a setup that would allow client code to control the target position.

Finally, conclusions of the experiments and the justification of reinforcement learning as a method of solving the problem of inverse kinematics are offered. Weak points in the study are presented and further directions of research and development are pointed out.

2. Literature Review

Inverse kinematics solutions

The problem of inverse kinematics has been long researched in the literature, a survey [8] classifying no less than 40 works in this area under the following categories: analytic, numerical, heuristic, data-driven and hybrid.

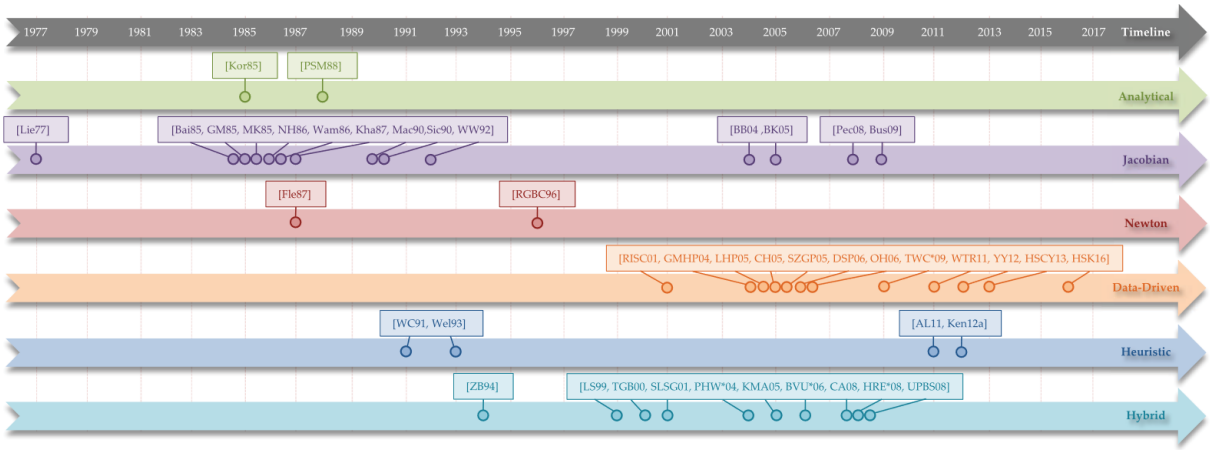

Figure 2 A timeline showing how the IK approaches have progressed over the years by examining publication years of key methods [8]

When considering any of these categories of methods, the desired application must be considered, be it robotics, computer graphics or others, as each method presents different properties when compared to others. Such factors include:

- treatment of singularities and their avoidance (especially in physical systems),
- stability, smoothness, and life-like motions (for example: natural, human-looking motions in computer graphics),
- implementation complexity and necessary computation time (thus dividing between algorithms that can solve the problem in real-time, as compared to those which need to solve ahead-of-time under known scenarios),
- scalability (use in a system with redundant degrees of freedom, such as continuum robots),
- and additional constraint satisfaction.

Classic methods

Analytic methods

Analytic methods attempt to find all the possible solutions for a given configuration, usually being constrained by the stability of the joint angles θ so that a unique motion planning

solution can be given. The aforementioned survey mentions that they offer a global solution, are not affected by singularities, can be efficiently computed in closed loops, and have seen extended use in robotics applications. However, given the non-linear nature of the problem, this family of methods does not scale well for larger numbers of degrees of freedom, implementations being made for as many as 6 rotational joints, which is a common occurrence in robotic arms.

Numerical methods

The next family of methods mentioned is that of numerical solutions. These methods find the solution by iteratively converging towards a result that minimizes an error function, simultaneously offering a trajectory to follow. Jacobian inverse methods solve this using partial derivatives of the forward kinematics, offering solutions that get closer and closer to the desired configuration, but they also present problems in crossing singularities, this definition being ill-conditioned in the vicinity of these points. That, in turn, results in very large changes near the singularities, oscillations for small changes and unstable behavior, which is most often solved by avoiding the singularity points altogether. The methods also pose some problems in implementing constraints, but present scalability, making them often employed in continuum robots.

One such example of a numerical algorithm would be that of Jacobian transpose [9], which avoids using the inverse of the Jacobian matrix of the system with respect to the joint angles. This, in turn, results in a solution which is not ill-conditioned near singularities. However, some disadvantages are that it requires a higher number of iterations, and the produced motions are considered unnatural.

Another presented algorithm is that of Damped Least Squares (DLS). By introducing a damping constant λ, the derivative of the joint angles is stabilized, removing the problem of crossing singularities that appear in many of the pseudo-inverse methods. However, large λ damping values can slow convergence rate, reduce accuracy, and generate oscillation or shaking.

Meanwhile, Newton methods, which use a second-order Taylor approximation, avoid the problem of singularities, and allow for easier implementation of constraints, while being considerably more complex and expensive to compute.

Heuristic methods

The heuristic methods manage to offer a solution to the inverse kinematics problem by solving the problem locally, per joint. They do so in a similar manner to numerical methods,

arriving at the solution after a number of iterations. Such a heuristic that has been widely adopted for use in robotics is represented by Cyclic Coordinate Descent (CCD) [10]. This method is simple to implement and compute, involving the alignment of the end effector on a line between the target position and each other joint, sequentially. Notable is the possibility of using this method along with prismatic joints, which produce translational motions instead of rotational ones. This approach, however, leads in a motion characterized by curling and uncurling of the arm as it approaches the target position, producing high angle changes and poor distribution of rotation among joints. This may result in unstable behavior and unnatural motions.

Another notable heuristic is Forward and Backward Reaching Inverse Kinematics (FABRIK) [11]. This method solves the problem of inverse kinematics in position space rather than joint orientation space in two phases: first it estimates the position of the end effector starting from the desired position and then working towards the base of the chain to reconnect the links, then another one in which the base is brought back to its original position and the chain is once again linked together starting from the base. These iterations pull the end effector closer to the target position and distribute the rotations along all joints in the arm. It demonstrates quick convergence, natural motions, and stability, but alike other heuristic methods, it requires additional complexity to integrate constraints, especially related to orientations of the joints, due to its position-oriented approach.

Recent evolutions

One of the more recent approaches [7] uses dynamic programming as opposed to the previous families of solutions that were identified. The solutions obtained are globally optimal, and the algorithm is capable of introducing collision avoidance constraints, as well as other complex limitations suited for a highly redundant robot arm. An implementation is offered in the Robot Operating System (ROS), which is a popular ecosystem in robotics applications.

Some of the limitations of this approach are the required computation time, thus having the necessity of knowing the constraints prior to computing a trajectory plan, making it less suited for dynamic, real-time and unexpected situation.

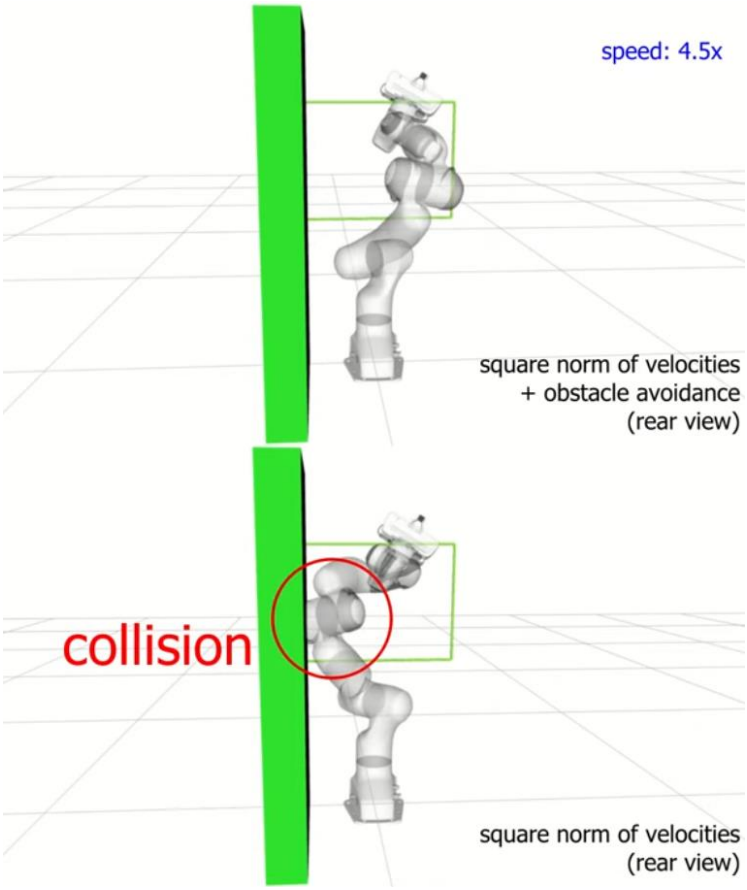

Figure 3 Demonstration of collision avoidance in dynamic programming algorithm proposed by [7], adapted from the presentation video

Data-driven methods

The survey on inverse kinematics [8] also takes note of a more recent trend that makes use of the available motion-captured data to solve the problem, calling this family of methods data-driven inverse kinematics. These methods learn from a large dataset of motions representing examples for the algorithm. The algorithm then builds a probability distribution function around these examples to find the next most likely pose to be in. This approach results in motions that are natural, and in fact allow for multiple styles when performing the task. However, motions are constrained in the area normally available to a human, heavily relying on the training dataset.

Artificial intelligence-based or learnt methods

Also mentioned are deep learning methods that either learn to replicate reference motions in a physically simulated environment, or those that learn to perform motions without any reference at all. These methods learn policies that map high-level target parameters to low-level actuator commands, used to control the system by solving the problem of inverse kinematics through optimization. To train such a policy, however, a considerable

computational effort is required, as many iterations may be needed before learning an acceptable model.

A study [5] reviews the efficiency of different artificial intelligence techniques, described as "soft-computing methods", for derivation of an inverse kinematics solutions applied on a robot arm with five links. It is claimed that the solutions can be computed in shorter time but maintain a desirable level of accuracy compared to traditional methods. The techniques that were tested are:

- artificial neural networks (ANN),
- adaptive neuro-fuzzy inference systems (ANFIS),
- and genetic algorithms (GA)

Upon performance of several experiments, it was observed that genetic algorithms obtain the smallest error out of the 3 proposed methods but require 2 orders of magnitude more computation time to solve, while having a high rotational angle error for the end effector. The authors proceed to propose a minimized error function (MEF) to improve upon the results of ANN and ANFIS by using the forward kinematics equation to verify the found solution, then bring it within a desirable threshold by iteratively proposing new target positions based on the error. Once MEF was applied, the accuracy of the results has improved considerably, resulting in a precision with several digits better. The computation time incurred remains acceptable compared to the result achieved by the genetic algorithm method.

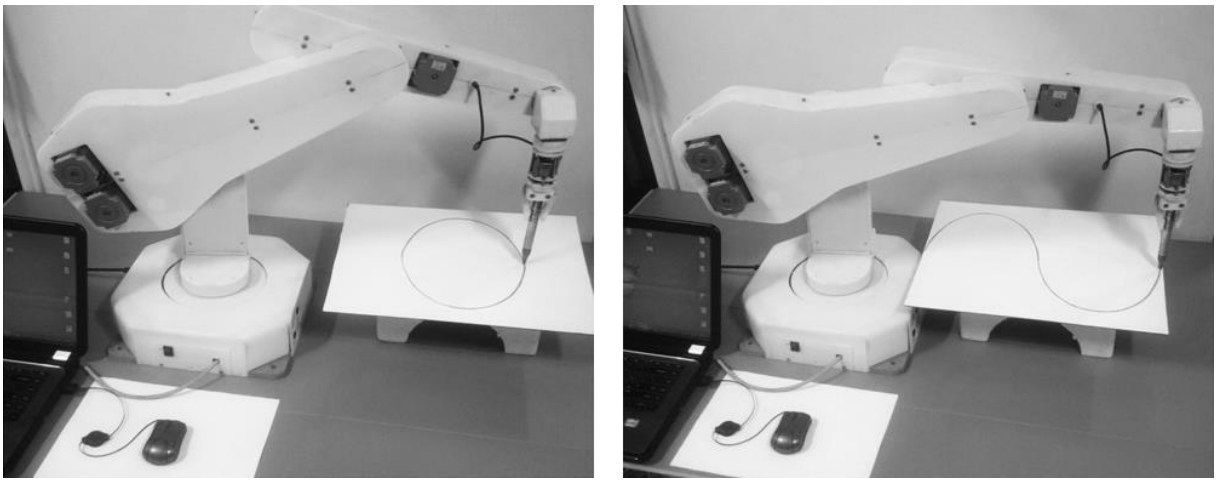

Figure 4 Robot arm real drawing circle and sin wave [5]

Reinforcement learning methods

An important work that employs the use of deep reinforcement learning methods is given by [6], where two methods that operate in continuous action and observation spaces, without knowledge of the model, and namely Soft Actor-Critic (SAC) and Deep Deterministic Policy

Gradient (DDPG), are used to plan the trajectory of a robot arm with 7 degrees of freedom while avoiding randomly positioned obstacles in the environment. It is argued in the favor of these off-policy algorithms because of their sample efficiency, which is indeed a relevant aspect when considering to adapt policies learnt in a virtual environment to a physical model of the robot arm, as each episode rollout becomes more expensive to train, and with more considerations such as safety and stability.

However, in order to reduce the number of dimensions in the problem, the mentioned work offers actions on the end effector in a Cartesian coordinate frame, using existing inverse kinematics solutions to solve for the necessary joint rotations.

Considering the distance to each obstacle known from sensors, the paper presents the training of a policy that solves the trajectory planning problem while avoiding the obstacles in the environment. The observations available to the agent consist of the end effector's position and velocity, as well as the relative position and velocity between the end effector and the closest obstacle, and with the target respectively.

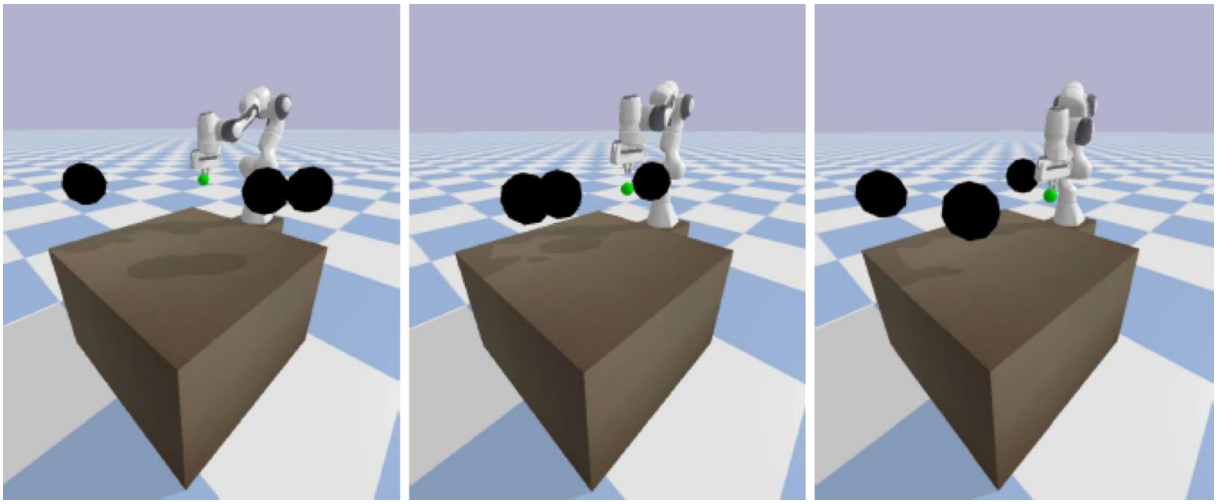

Figure 5 From left to right, first, the robot learns to reach the goal. Second, avoid collisions with dynamic obstacles. Third, keep reaching the goal. [6]

The reasoning for using relative measurements was given as a reduction in the time necessary for the policy training to converge towards an acceptable solution.

The reward function was modeled as a weighted sum between the distance to the goal and an expression containing the inverse of the distance (incremented by a constant as to avoid division by zero) towards the closest obstacle, distances relative to the end effector. Finally, in order to maximize the goal function, this formula is expressed with a negative sign, punishing the agent for keeping a distance from the goal and for approaching obstacles. By maximizing this function, the agent is learning a policy that fulfills the desired objectives.

For the experiments, two simulation environments were prepared, containing either static or dynamic target and obstacles, respectively. The initial position of the end effector, obstacles and target were randomized in order to ensure that the learnt policies can adapt to variable conditions.

Experimental results demonstrated in [6] have shown that while both deep reinforcement learning methods are capable of learning a policy to fulfill the initial objectives, DDPG does so after more episodes than SAC, while proving more inconsistencies between episodes. The authors explain this phenomenon as deriving from the deterministic characteristic of the DDPG method, which impacts exploration of the solution space. This, in turn, makes DDPG more sensitive to hyperparameters and more unstable while performing high-level tasks.

As accuracy and safety rate were defined as the proportion of timesteps in the episode in which the end effector approached the target position satisfactorily close, respectively, avoided collision with all the obstacles, the results in the paper do not come close to 100%. This is because the timesteps since the initial, rest position in the beginning of the episode are being taken into consideration in this metric. Even under these circumstances, both learnt policies have reached a safety rate of 100%, with SAC reaching an accuracy as big as 82% in a static environment and 79% in a dynamic one.

3. Proposed approach

Model

In the following, a formal definition of the problem will be given for a simplified bidimensional case.

An "arm" is a system composed of n segments (or links) of lengths l_i, connected in serial using rotational joints at angles θ_i, with an end effector at position $p = (x, y)$, for $i = \overline{1, n}$, as seen in Figure 6.

We can, moreover, consider constraints upon the minimum and maximum rotation of each

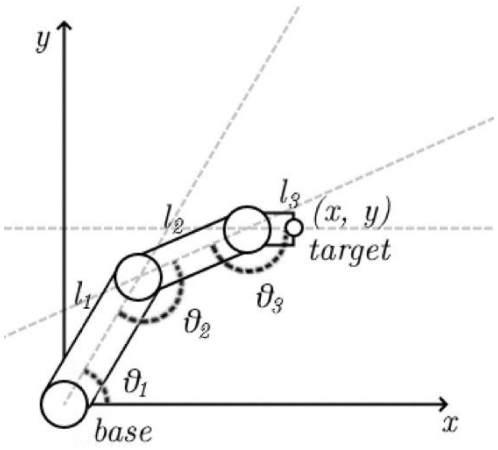

Figure 6 Formal definition of the inverse kinematics problem

joint. These can be defined by θ_{min} and θ_{max} such that the following condition is met: $\theta_{min_i} \leq \theta_i \leq \theta_{max_i}$.

Solving the forward kinematics problem is trivial, as all variables that are needed to find the position of the end effector are known. The equation can be formulated as such:

$$p = \sum_{i=0}^{n} l_i \left(cos\left(\sum_{j=0}^{i} \theta_j \right); sin\left(\sum_{j=0}^{i} \theta_j \right) \right)$$

However, solving the joint angles θ for a given end effector position p is not similarly trivial, having either no solution or multiple, depending on the arm configuration and target position. Considering the situation of $n = 2$ segments in a bidimensional space, the desired position of the end effector p and also constraining the orientation of the end effector, the problem becomes sufficiently constrained, allowing for a solution. However, for situations where $n > 2$, and especially those which unfold in a tridimensional coordinate frame, the problem of

inverse kinematics becomes once again under-constrained, having multiple potential solutions.

Learning environments

The learning environment will consist of a physically simulated workspace containing a highly redundant robot arm (such as one with 7 degrees of freedom). The robot arm must support control given through target angles and perception of current state, such as position, orientation, and velocity of the end effector. The robot arm shall have its base fixed on a table in this workspace.

Tasks

The task will be that of reaching a desired target position, optionally including a neutral orientation so that the joint control method imitates the properties of that in end effector Cartesian coordinate space. For observations, the end effect position, orientation, and velocity, as well as the target position, will be given, albeit the velocity being currently unused for the purpose of this study. Results found in [6] show that a description of the displacement relative to the end effector led to more simplified learning, but these results have not been applied in this work.

For the trained policy to be more general, the initial joint angles of the robot arm will be randomized by applying a perturbation to each of the neutral joint angles equal to a value in the range $[-\pi, \pi]$, multiplied by a factor that will be further named "perturbation term".

In a dense formulation of the reward function, distance between the end effector and the target position can be subtracted so that it represents a punishment, encouraging the intelligent agent to minimize this distance. Similarly, a term is defined for the norm of the difference vector between the current orientation and the one that is desired (neutral, in this case), orientations expressed in axis angles format.

However, a potential problem arises in this dense formulation of the reward function, as it may conflict other goals such as a limited end effector velocity. This is because the distance punishment minimization implies that the end effector moves from the initial position to the target one as quickly as possible. This, in turn, suggests that the results of a policy trained on a sparse definition, one where positive reinforcement is given when the criteria is met, are worth investigating.

Reinforcement learning algorithms chosen

The previously mentioned study [6] used the Deep Deterministic Policy Gradient (DDPG) and Soft Actor-Critic (SAC), which represent off-policy algorithms. These algorithms learn

the value function independently from the policy that is being used but based on different possible actions. It was argued that a previous work identified this type of reinforcement learning algorithms to be more sample efficient, which is indeed a desirable property when training policies that are then meant to be adapted on real-life systems, where training become much more expensive to perform, and with concerns related to safety and stability. This study will compare the results of SAC, which was pointed out as having better results, with those of the Proximal Policy Optimization (PPO) algorithm, which is an on-policy algorithm, in the attempt to solve the task.

Implementation

Implementation consists of 3 elements:

- interface between reinforcement learning algorithms and environments,
- reinforcement learning algorithms,
- and environments.

Environment-algorithm interface – OpenAI Gym

For the interface between the reinforcement learning algorithms and the environments, Gym [12], an open-source Python library from OpenAI was chosen. The library allows independent development of learning algorithms and environments by assuring a common interface is respected. It is widely used in research projects in the domain of reinforcement learning and some of these works are also seen in further implementation decisions.

Reinforcement learning algorithms – Stable Baselines 3

For the implementations of reinforcement learning algorithms, Stable Baselines 3 [13] is used. This is a set of reinforcement learning algorithms, including algorithms such as: DDPG, DQN, PPO, SAC or TD3. It is compatible with the Gym interface and is used to help replicate the results, baseline models being offered for a various number of tasks. This also simplifies the process of comparing different reinforcement learning algorithms in the solving of a task.

Training environment – panda-gym

For the environment, panda-gym [14] was chosen. This environment, which is compatible with the Gym interface and with Stable Baselines 3 (specifically, panda-gym version 2), offers a number of example task implementations for a Franka Emika Panda robot arm with 7 joints and degrees of freedom, with specifications offered in Figure 7 and Table 1.

The tasks include reaching a target position, pushing cubes in a desired position, picking, and placing cubes or flipping them. The environment uses the PyBullet physics engine for the simulation of the environment.

While there is a standard PandaReach task defined, it only verifies the position of the end effector, and the starting position is not randomized. Changes were made and integrated into a PandaCustomReach task so that it matches the properties described in the Tasks subchapter, such as additional observations, constraints, and task reward definition, but also the capacity to offer a random number generator seed in order for the episodes to be replicable or to programmatically set the target during execution of the episode from the client code. In Figure 8, a visualization of the volume within which the goal can be sampled is shown, and in Figure 9, a diagram summarizing the code changes made to panda-gym 2.0.0 is offered.

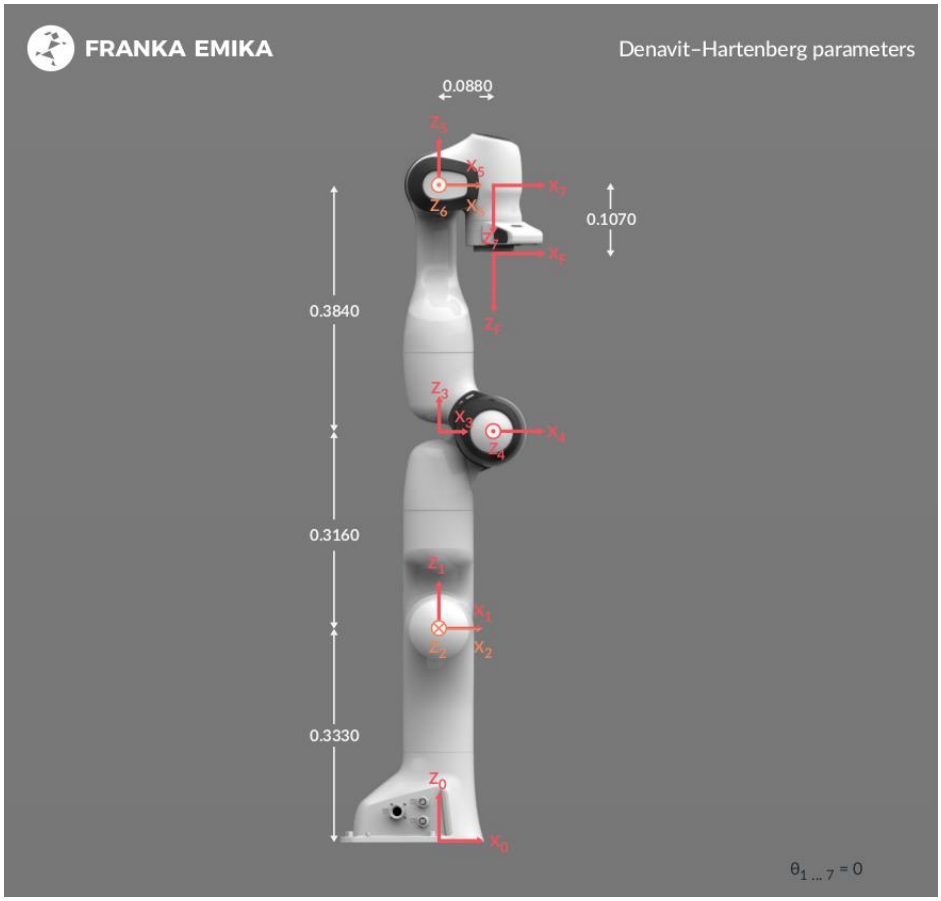

Figure 7 Panda's kinematic chain [15]

Name	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7	Unit
q_{max}	2.8973	1.7628	2.8973	-0.0698	2.8973	3.7525	2.8973	rad
q_{min}	-2.8973	-1.7628	-2.8973	-3.0718	-2.8973	-0.0175	-2.8973	rad
$\dot{q}_{max}$	2.175	2.175	2.175	2.175	2.61	2.61	2.61	$\frac{rad}{s}$
$\ddot{q}_{max}$	15	7.5	10	12.5	15	20	20	$\frac{rad}{s^2}$

$\ddot{q}_{max}$	7500	3750	5000	6250	7500	10000	10000	$\frac{rad}{s^2}$
$\tau_{j_{max}}$	87	87	87	87	12	12	12	Nm
$\dot{\tau}_{j_{max}}$	1000	1000	1000	1000	1000	1000	1000	$\frac{Nm}{s}$

Table 1 Joint space limits for Panda [15]

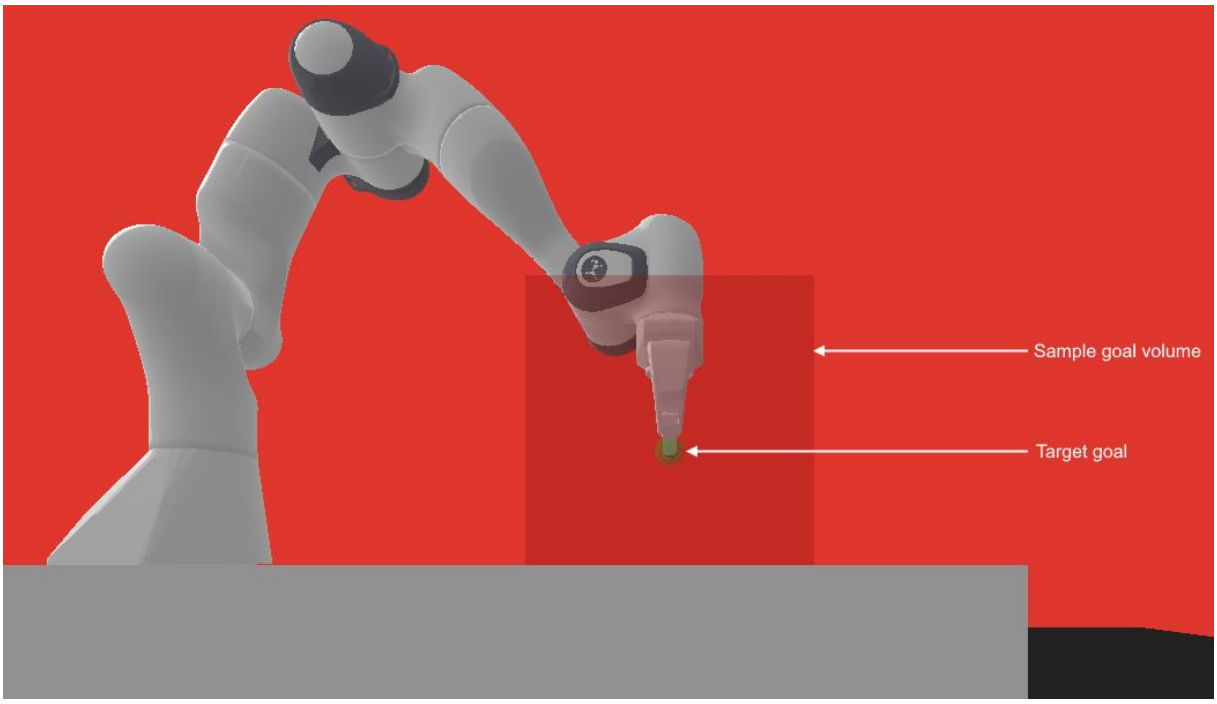

Figure 8 Depiction of the simulation environment, including the volume within which the goal is sampled

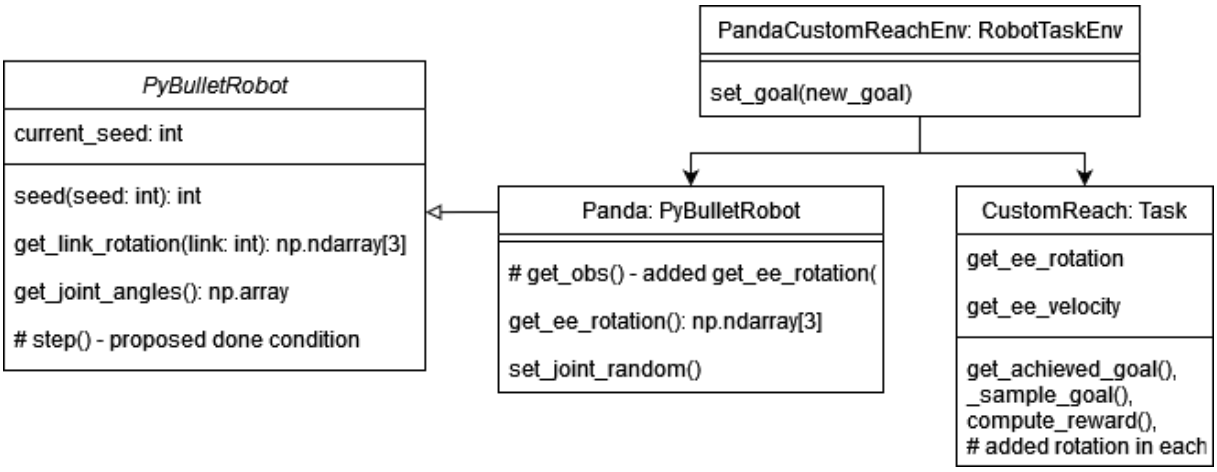

Figure 9 Summary of properties and methods added or changed within panda-gym 2.0.0

4. Case study 1: Proximal Policy Optimization versus Soft Actor-Critic

Experiments

In order for the training to occur more reliably, it will take place in several sequences consisting of 200000 frames each (150 frames per episode). Each sequence will train the policy obtained in the previous one while variating the initial robot arm configuration by larger increments. The initial sequence will have no perturbation and instead will focus on simply reaching and maintaining the target position, sampled within a box volume of 0.3m x 0.3m x 0.3m, with a standard orientation shared by the Cartesian coordinates space control method. As the increment gets larger, the control policy will have to learn how to perform actions that lead to the recovery of a viable joint angles configuration, as some configurations heavily restrain the movements that can be produced.

Evaluation metrics

The same study [6], as previously mentioned to be similar, has described two evaluation metrics: accuracy and safety rate. Both describe the number of frames in the length of an episode in which the target position was reached, respectively in which no collision took place (number of corresponding frames / 100, the length of an episode in their experiment). While the safety rate could reach 100%, the metric for accuracy, which differs from the classical definition of distance between expected and resulting positions, could not, as it included the frames since the initial position and through its movement towards the target position.

The accuracy metric could use a better definition, or at least the results should be compared to those obtained by a traditional method in order to confirm the motivation behind using reinforcement learning-based techniques in solving the problem.

Given the dense definition of the reward function, the mean reward (or punishment) itself could be used as a metric of comparing the results and of confirming that relevant learning takes place. The results here will focus on that and the loss values in the reinforcement learning process. Moreover, considering the variables and constraints imposed on the experiment, results will be also described qualitatively.

Results

For higher perturbations, it was noticed that a greater number of episodes are required for the policy trained by PPO to adapt. As such, first sequence contains no perturbation (perturbation term equal to 0), second sequence contains a perturbation term of 0.2, and the next 2 sequences containing a perturbation term of 0.25 The training results are shown in the

diagrams from Figure 10 and Figure 11, with the note that the last two sequences were conjoined in this situation:

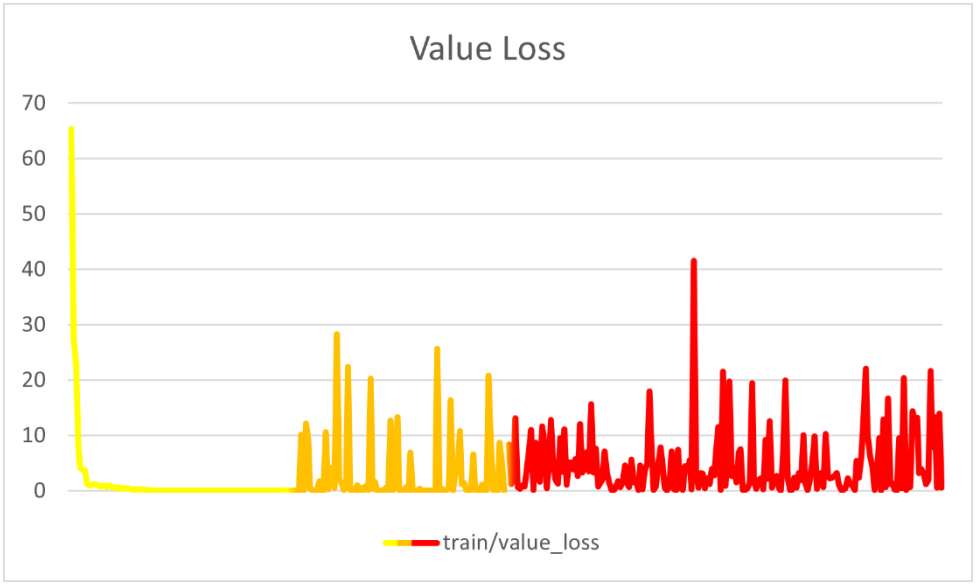

Figure 10 Value loss of PPO algorithm during the training sequences

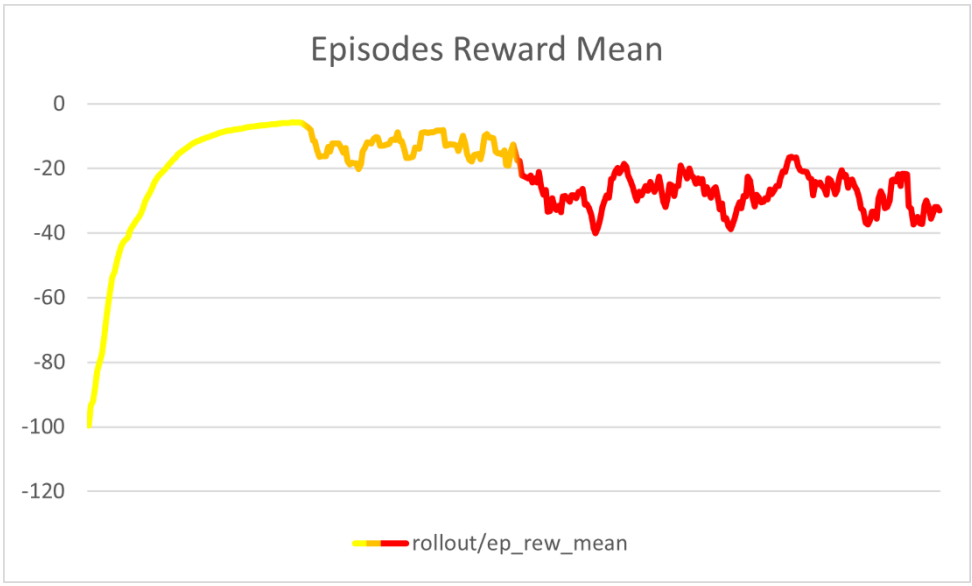

Figure 11 Episodes' reward mean of PPO algorithm during the training sequences

Meanwhile, for SAC, the policy learnt could better adapt against higher perturbations, reason for which the last 2 sequences contain a perturbation term of 0.4 but are separated. This led to the results portrayed in Figure 12 and Figure 13:

Figure 12 Actor loss of SAC algorithm during the training sequences

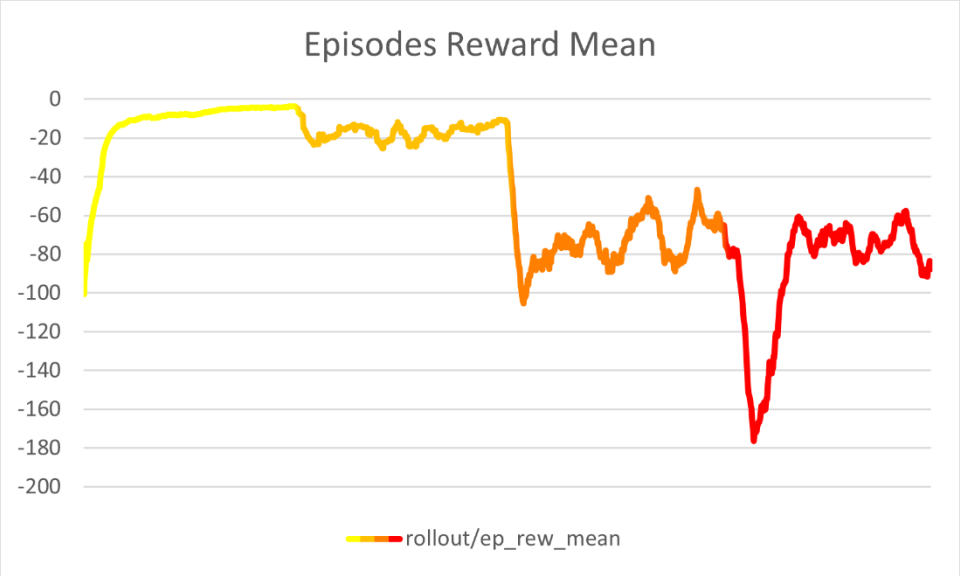

Figure 13 Episodes' Reward Mean of SAC algorithm during the training

Discussion

Discussing the results from a qualitative perspective, results obtained by PPO in first training sequence have proven to be unstable upon reaching the target position, with the end effector oscillating. This behavior has been visibly improved upon the addition of orientation constraints, initial joint angle perturbation and through prolonging of the episode duration (from 50 frames to 150). The correlation with the episode duration comes from the observation that the joints do not stop moving upon reaching the target destination. While increasing the length of the episode, this motion becomes slower, but once the joint limits are

reached, they make keeping the target position complicated, problem which should be solved by introduction of a punishment term proportional to the movement caused by the action, as to minimize the energy necessary, or to stop when done.

The learnt policies do not seem to offer the accuracy of traditional methods, offering instead a result within a certain margin of error. The dense formulation of the reward function (as punishment proportional to the distance between the end effector and the target position) leads to quick movements that one on hand might be desirable but could conflict with constraints related to orientation or velocity. This, in turn, motivates the need to investigate the results based on a sparse formulation of the reward function (as reward given for each frame in which the desired target position is achieved).

It might be worthwhile to implement a similar strategy as noted in [6] where the displacement observation is given relative to the end effector rather than in global coordinate space in order to improve upon the convergence towards an optimal solution. Another way to improve the results obtained by the learnt policy is to adopt a similar strategy as the minimized error function described in [5], which corrects the target position based on the observed error.

A useful result, however, is that the robot arm has managed to recover from configurations that would have otherwise posed obstacles for direct methods, such as moving in Cartesian space proportional to the displacement. However, not all motions observed can be described as being safe, stable, or successful, the policy sometimes getting stuck in locally optimal configurations.

These results are taken further by the policy trained with SAC, which, while not presenting satisfying results in terms of classic accuracy, it has learnt viable strategies of controlling the joints so that the robot arm configuration can reach close to the target position, unimpeded by local minima configurations or those that would have impacted even the direct solution that does not use reinforcement learning. This justifies use of a reinforcement learning-based policy for recovery from difficult configurations of the robot arm, while using traditional methods to achieve best accuracy.

One of the aspects that can be improved is that the randomization of the starting joints configuration, as implemented, sometimes causes the grippers of the end effector get disconnected, while still retaining influence over the robot arm. This impacts the results of the training and can lead either to inaccurate training data or even perturbations in the trained policy. These illegal states represent a weak point in both training and evaluation of the methods.

5. Case study 2: Traditional method versus Hybrid method

Hybrid method

The previously obtained results indicate the fact that the trained model could be combined with a traditional method, so that the advantages of both: recovery and accuracy, respectively, to be merged. The proposed approach for this is to discretely switch the method used based on a distance threshold (such as either 0.1 or 0.2, in the simulation environment), so that the trained model is used at large distances, while the traditional method (PyBullet's DLS implementation) is used in the close vicinity of the target position. Getting ahead of the results, the static, sudden threshold leads to oscillations in the end effector when crossing the threshold, caused by contradictory commands issued by the two methods. This observation led to a refinement, changing between the methods continuously instead of discretely, leading to a smoothed hybrid approach. The action of the robot arm is represented as a weighted sum between the solutions given by the trained model and DLS. The weights are limited between 0 and 1, and they sum up to 1, the equations for them being:

1) $2.5d - 0.5$ (for the trained model),
2) $1.5 - 2.5d$ (for DLS),

where d represents the distance between the end effector and the target position.

In Figure 14, a graphical representation of the weight functions can be seen:

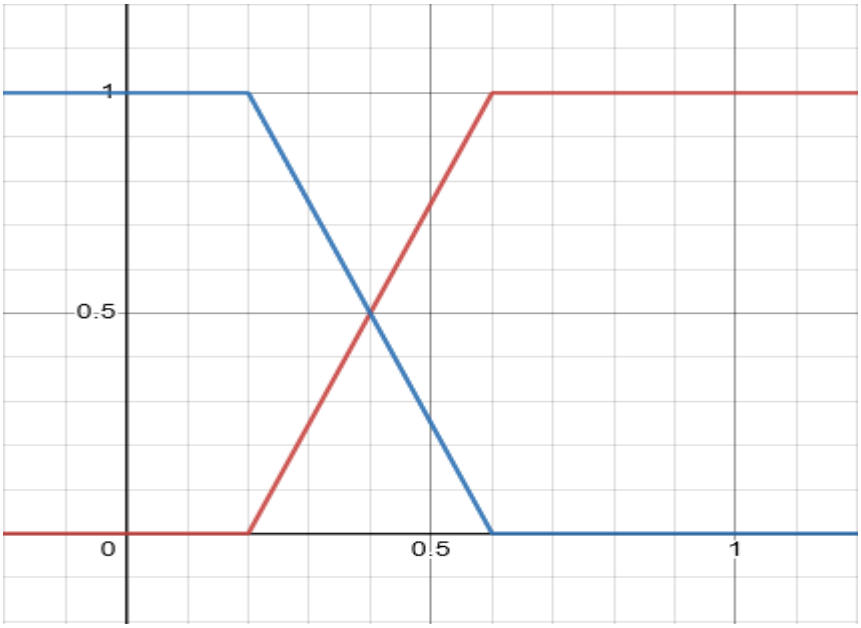

Figure 14 Weight functions for the smoothed hybrid method (based on distance, DLS with red, trained model with blue)

Implementation details

While panda-gym offers the possibility to control the end effector either in Cartesian coordinates (via DLS) or through direct manipulation of the joint angles, this is done through supplying of two different environments. The code used to implement the former control mode was reused inside the hybrid method, although it has been found to converge slowly, so a proportional multiplication factor (40) was introduced. The value improves convergence speed while avoiding oscillations, but the implementation of the traditional method leaves large room for improvements.

In order to allow for execution of test scenarios under same circumstances, the environment was modified to allow seeding of the episode so that both the target position and initial joint angles configuration to be consistently replicable. This allows different methods to be tested under same circumstances, allowing for comparison between them.

Experiments

Using the following methods: traditional (DLS), hybrid with 0.1 threshold, hybrid with 0.2 threshold and smoothed hybrid; 1000 seeded episodes were processed for each of the methods, the success of each episode being recorded at the end, after 300 timesteps. The purpose of this is to quantitatively analyze the success rate of each method, to observe the different situations derived from combinations of success across methods and their frequency. This experiment is meant to check whether the hybrid method, and thus the reinforcement learning trained model can be justified in the use for an inverse kinematic solver.

Evaluation metrics

Given that all the methods use the traditional inverse kinematics methods for accurately positioning the end effector, measuring accuracy in a classic sense is not justified. While number of episodes required to reach the target could be tracked, it is not implemented in this work. The way success is defined in this experiment is whether the end effector was or not within a margin of error at the end of each episode. Across these 1000 episodes, the success rate will be measured, but also the number of each scenario, defined by the combination of success results from each method.

Results

The results from running the total of 4000 episodes are contained in Table 2 and Table 3.

Method	DLS	**Hybrid-0.1**	**Hybrid-0.2**	**Smoothed Hybrid**
Success count	775	**829**	**843**	**840**
Success rate after removing general fails	87.77%	**93.88%**	**95.47%**	**95.13%**

Table 2 Success count for each method (out of 1000 episodes each) (using SAC)

Method	**DLS**	**Hybrid-0.1**	**Hybrid-0.2**	**Smoothed Hybrid**	**Combination Count**
Success	FALSE	FALSE	FALSE	FALSE	117
	FALSE	FALSE	FALSE	TRUE	3
	FALSE	FALSE	TRUE	FALSE	2
	FALSE	FALSE	TRUE	TRUE	7
	FALSE	TRUE	FALSE	FALSE	5
	FALSE	TRUE	TRUE	FALSE	14
	FALSE	TRUE	TRUE	TRUE	77
	TRUE	FALSE	FALSE	FALSE	5
	TRUE	FALSE	FALSE	TRUE	15
	TRUE	FALSE	TRUE	FALSE	1
	TRUE	FALSE	TRUE	TRUE	21
	TRUE	TRUE	FALSE	FALSE	4
	TRUE	TRUE	FALSE	TRUE	8
	TRUE	TRUE	TRUE	FALSE	12
	TRUE	TRUE	TRUE	TRUE	709

Table 3 Count of each combination of methods' successes (using SAC)

It is worth noting that success rate has been calculated after subtracting the number of “all fail” cases, considering naively that these scenarios have been caused by the limitations identified earlier in randomizing initial starting position and the subsequent collisions. The variation in the number of such cases can be verified when later comparing with PPO.

Examples of the results obtained by the methods in the episode with the seed 14 can be observed in Figure 15, Figure 16, Figure 17, Figure 18 and Figure 19, the screenshots being sampled at an interval of 1 second. Notable is the aforementioned situation of illegal states that the random configuration of joint angles can lead to, as caused by the resulting intersecting collisions. These can offer a qualitative insight on the performance of the discussed methods.

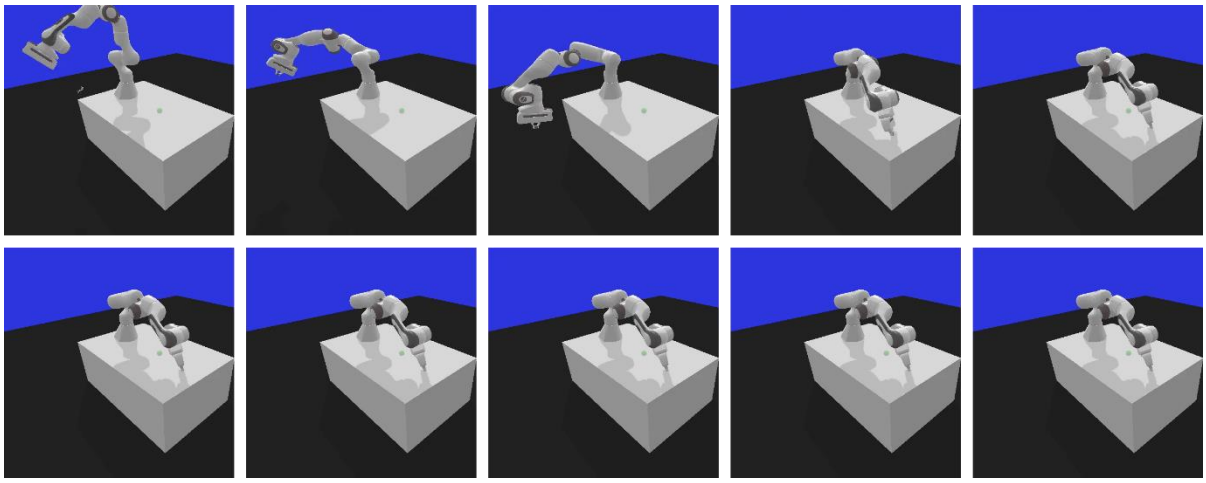

Figure 15 DLS results on episode 14

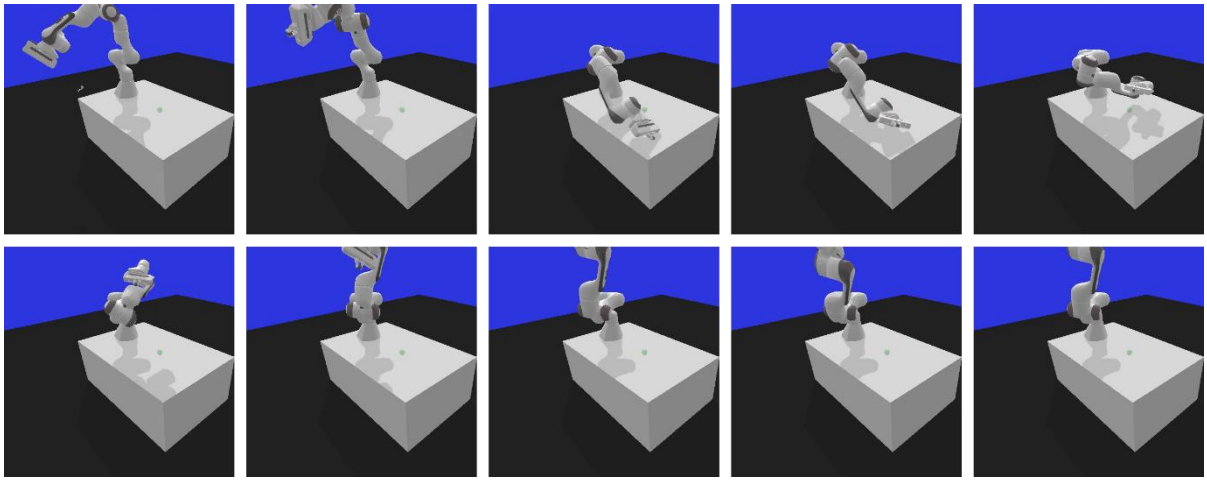

Figure 16 PPO results on episode 14

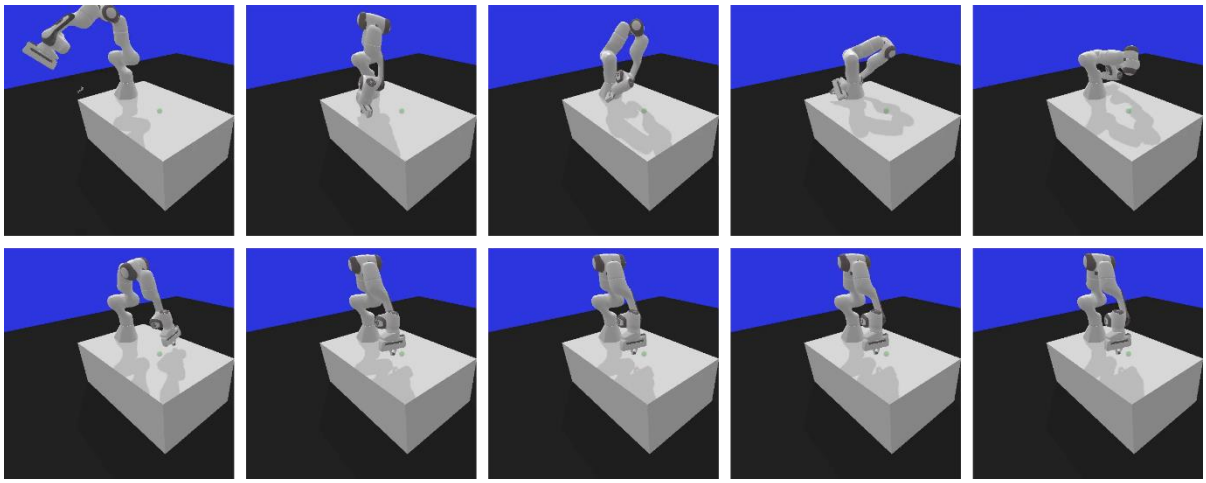

Figure 17 SAC results on episode 14

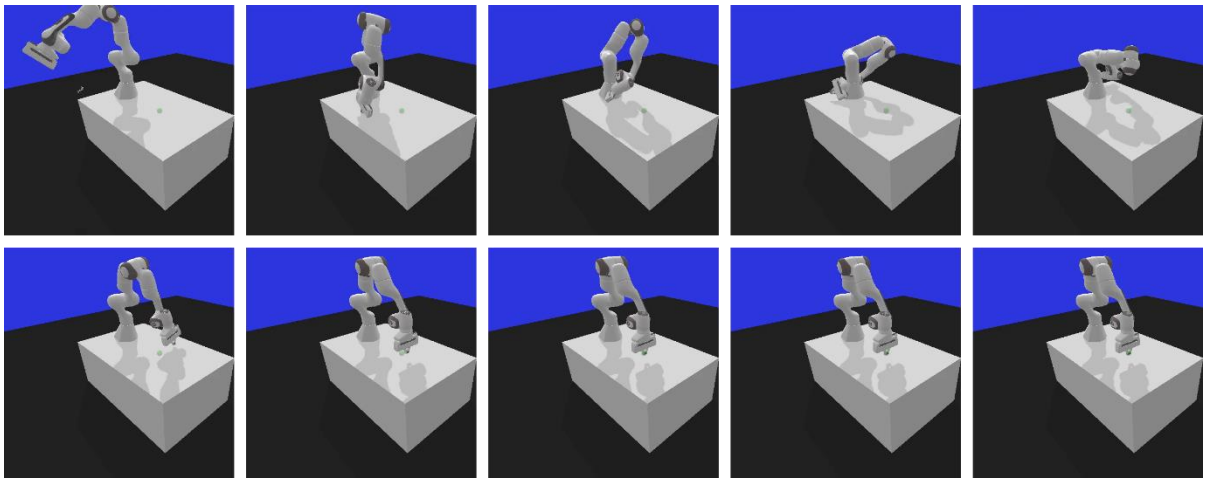

Figure 18 Hybrid-0.1 results on episode 14

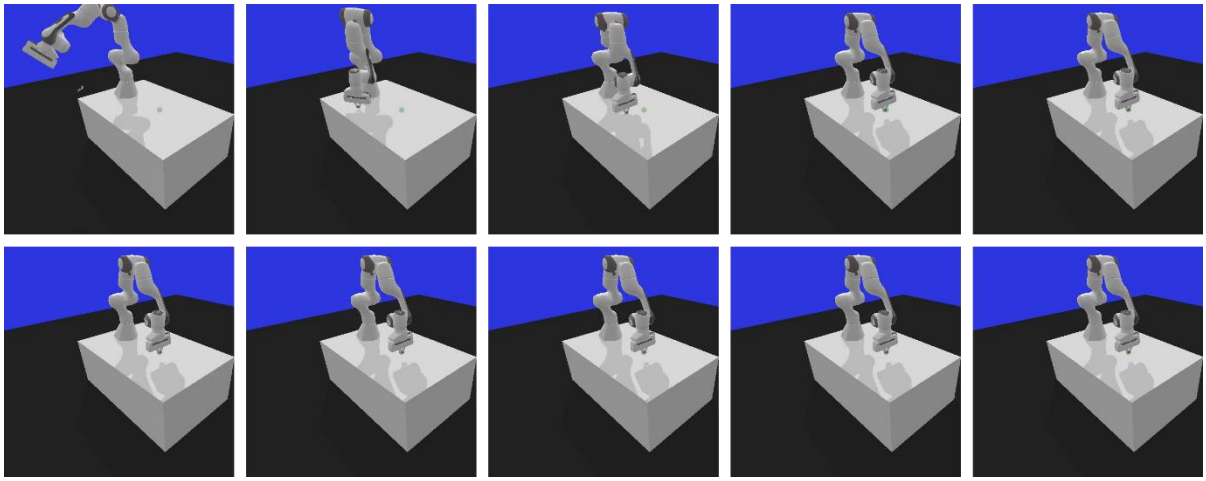

Figure 19 Smoothed Hybrid results on episode 14

The hybrid methods used the model trained through SAC. The results obtained using the model trained through PPO are similar between those that apply the hybrid approach and the one which does not, in that the end effector does not approach the target sufficiently close for the traditional approach to be chosen. An exception to this situation is represented by the smoothed hybrid approach, which results yet into a steady-state error because of the contradiction between the traditional approach (which itself has a steady-state error) and the trained model. For a complete view, results of the same experiment applied to hybrid algorithms using PPO are provided in Table 4 and Table 5.

Method	DLS	**Hybrid-0.1**	**Hybrid-0.2**	**Smoothed Hybrid**
Success count	775	**676**	**665**	**689**
Success rate after removing general fails	91.18%	**79.53%**	**78.24%**	**81.06%**

Table 4 Success count for each method (out of 1000 episodes each) (using PPO)

Method	**DLS**	**Hybrid-0.1**	**Hybrid-0.2**	**Smoothed Hybrid**	**Combination Count**
Success	FALSE	FALSE	FALSE	FALSE	150
	FALSE	FALSE	FALSE	TRUE	6
	FALSE	FALSE	TRUE	TRUE	1
	FALSE	TRUE	FALSE	FALSE	6
	FALSE	TRUE	FALSE	TRUE	3
	FALSE	TRUE	TRUE	FALSE	7
	FALSE	TRUE	TRUE	TRUE	52
	TRUE	FALSE	FALSE	FALSE	126
	TRUE	FALSE	FALSE	TRUE	31
	TRUE	FALSE	TRUE	FALSE	2
	TRUE	FALSE	TRUE	TRUE	8
	TRUE	TRUE	FALSE	FALSE	7
	TRUE	TRUE	FALSE	TRUE	6
	TRUE	TRUE	TRUE	FALSE	13
	TRUE	TRUE	TRUE	TRUE	582

Table 5 Count of each combination of methods' successes (using PPO)

Discussion

Quantitatively, the hybrid methods have managed to pass a noticeably larger number of tests compared to DLS. However, looking at the variety of situations, as defined by the combination of the results, and their non-trivial frequency, a quantitative analysis is not enough, as depending on the scenario, some methods may pass the test while others do not. A result is, however, the fact that with one exception, there weren't situations where any of the hybrid methods could not solve the problem while the traditional method could.

Episodes were split in the following classes, based on the qualitatively observed results:

- fail caused by illegal state – these situations can easily be observed when the grippers are visibly detached from the end effector
- fail under normal conditions – although a solution appears trivial to the observer, none of the methods reach the target position
- success that is trivial – solved normally by all methods
- success that is not trivial – success obtained as a result of the proposed approach
- success that is not acceptable – reaches target destination successfully, but compromises on self-collision.

For each of the described classes, a commentary is given for an example episode that is representative of each classs:

- episode 78 – fail caused by illegal state – the grippers are visibly missing from the end effector, indicating the illegal state and the impact of improper physics simulation over the results and training
- episode 37 – it is observed that DLS does not succeed in reaching the objective, being stuck in a difficult configuration (fully extended joints). The hybrid methods tend to fail, using same DLS to resolve the last part, except for Hybrid-0.1, which manages to explore sufficiently to avoid the local minimum that DLS cannot solve
- episode 1 – trivial success – no commentary required
- episode 90 – non-trivial success – in this situation, the trained model is capable of recovery from a configuration that DLS cannot resolve
- episode 14 – unacceptable success – while the hybrid models can solve this scenario, they do so through compromises that are practically unacceptable, such as self-collision. Remarkably, however, is that the smoothed hybrid method succeeds in this scenario, with qualitatively acceptable results.

Even though the refinement of hybrid methods into the smoothed hybrid method offers satisfactory results qualitatively, careful consideration of the action combination method has

to be given, as certain parametrizations and situations can lead to steady-state errors. The results obtained certainly encourage further research to adopt hybrid methods that adapt thresholds dynamically.

Weak points that should be addressed to improve the credibility of the results are the following:

- starting position randomization causing illegal physics states, which negatively impacts training and evaluation
- self-collision allows the trained model to reach solutions while making unpractical compromises
- lack of flexbility or dynamism in the hybrid methods, or of a thorough analysis in general.

Finally, in order to allow use of the developed hybrid method, a new environment was implemented that allows client code to set the target position during runtime. This environment was tested by changing the target position so that the target position moves on a circular trajectory. The results observed were that the robot arm converges towards the target position, but as described previously, precise convergence is slow due to limitations in the implementation of the traditional method. This, in turn, causes the end effector to remain behind the target position, but this represents a result that could be further improved.

For adaptation of use for a physical system, an additional environment would have to be developed that would transmit commands and sensors' readings to and from the physical robot arm, replacing the simulated robot arm and environment. Additional training would also be recommended for the trained model to adapt to differences between the simulated environment and the real one.

An example program was conceived which allows the user to control the target position of the simulated robot arm's end effector in real time, using PyBullet's addUserDebugParameter method in the ExampleBrowser's debug GUI. The user can bring up the debug GUI by pressing G, then control the x, y and z coordinates of the target position using the sliders.

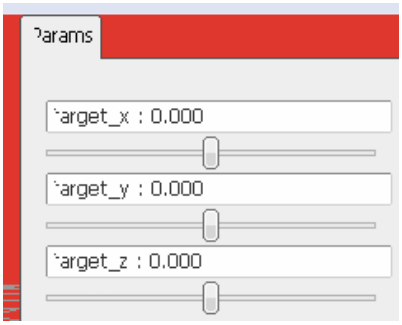

Figure 20 Control parameters sliders

6. Conclusions

In conclusion, while the problem of inverse kinematics has been actively researched for a time period that to date represents several decades, with methods ranging from analytical and numerical to heuristics and data-driven, this area of study still offers new, promising techniques to providing a solution, leveraging developments in connected computer science and mathematical domains. Deep reinforcement learning has been shown in literature to successfully solve related problems while offering additional capabilities such as avoiding collision between the end effector and dynamic, unexpected obstacles. Noteworthy is also a recent algorithm that offers a globally optimal solution for redundant robots, while implementing additional constraints, including avoidance of collision with known obstacles, albeit the computation time does not allow for runtime computation, thus limiting the application to planning.

Future work

Future work will need to explore definition of better evaluation metrics and more detailed reward functions for the solutions based on artificial intelligence, and to consider comparison of those methods with traditional ones, in terms of computation time and solution accuracy, to better justify the use of such frameworks.

While the action space used in training of the policy in one of the works was one reduced to the relative Cartesian coordinates of the end effector, solving a trajectory planning problem, the training could be extended to accommodate the joint rotations as an action space instead, giving an end-to-end trained approach for solving the problem.

Another direction that can be taken is to train a generalized policy, with variable segment lengths, joint limits, or perhaps even number of degrees of freedom, to allow the computation time trade off to be rewarded in flexibility and transferability of the learnt policy to a greater number of systems.

References

[1] B. Sapp and B. Taskar, "Modec: Multimodal decomposable models for human pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3674-3681, 2013. https://doi.org/10.1109/CVPR.2013.471

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint* arXiv:1312.5602, 2013. https://doi.org/10.48550/arXiv.1312.5602

[3] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko and A. Bridgland, "Highly accurate protein structure prediction with AlphaFold," *Nature,* vol. 596, no. 7873, pp. 583-589, 2021. https://doi.org/10.1038/s41586-021-03819-2

[4] E. Vollenweider, M. Bjelonic, V. Klemm, N. Rudin, J. Lee and M. Hutter, "Advanced skills through multiple adversarial motion priors in reinforcement learning," *arXiv preprint* arXiv:2203.14912, 2022. https://doi.org/10.48550/arXiv.2203.14912

[5] A. El-Sherbiny, M. A. Elhosseini and A. Y. Haikal, "A comparative study of soft computing methods to solve inverse kinematics problem," *Ain Shams Engineering Journal,* vol. 9, no. 4, pp. 2535-2548, 2018. https://doi.org/10.1016/j.asej.2017.08.001

[6] L. Chen, Z. Jiang, L. Cheng, A. C. Knoll and M. Zhou, "Deep reinforcement learning based trajectory planning under uncertain constraints," *Frontiers in Neurorobotics,* vol. 16, p. 883562, 2022. https://doi.org/10.3389%2Ffnbot.2022.883562

[7] E. Ferrentino, F. Salvioli and P. Chiacchio, "Globally optimal redundancy resolution with dynamic programming for robot planning: a ROS implementation," *Robotics,* vol. 10, no. 1, p. 42, 2021. https://doi.org/10.3390/robotics10010042

[8] A. Aristidou, J. Lasenby, Y. Chrysanthou and A. Shamir, "Inverse kinematics techniques in computer graphics: A survey," in *Computer graphics forum,* vol. 37, no. 6*,* pp. 35-58, 2018.
https://doi.org/10.1111/cgf.13310

[9] H. Das, J. E. Slotine and T. B. Sheridan, "Inverse kinematic algorithms for redundant systems," in *Proceedings. IEEE International Conference on Robotics and Automation*, pp. 43-48, 1988.
https://doi.org/10.1109/ROBOT.1988.12021

[10] L.-C. T. Wang and C. C. Chen, "A combined optimization method for solving the inverse kinematics problem of mechanical manipulators," *IEEE Transactions on Robotics and Automation,* vol. 7, no. 4, pp. 489-499, 1991.
https://doi.org/10.1109/70.86079

[11] A. Aristidou and J. Lasenby, "FABRIK: A fast, iterative solver for the Inverse Kinematics problem," *Graphical Models,* vol. 73*,* no. 5*,* pp. 243-260, 2011.
https://doi.org/10.1016/j.gmod.2011.05.003

[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba, "OpenAI Gym," *arXiv preprint* arXiv:1606.01540, 2016.
https://doi.org/10.48550/arXiv.1606.01540

[13] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research,* vol. 22, no. 268, pp. 1-8, 2021.
https://dl.acm.org/*doi*/abs/10.5555/3546258.3546526

[14] Q. Gallouédec, N. Cazin, E. Dellandréa and L. Chen, "panda-gym: Open-Source Goal-Conditioned Environments for Robotic Learning," in *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS,* 2021.
https://doi.org/10.48550/arXiv.2106.13687

[15] Robot and interface specifications - Franka Control Interface (FCI) documentation
https://frankaemika.github.io/docs/control_parameters.html#limits-for-panda
Accessed on date: 13/06/2023

Table of figures