Building a Framework to Solve Insertion Tasks with Residual Reinforcement Learning in the Real World

Adriaan Mulder [1]   Nick Striebel [1]

Abstract

Solving robot manipulation tasks requiring high accuracy in contact rich environments grows gradually complex with traditional control methods. This is why today's control increasingly relies on machine learning methods. Basis for using an machine learning approach in such an environment is a solid understanding of a robots workings and intricacies due to safety reasons. Even then implementing a learnable control policy is not a trivial task. This is why we build a basis to implement reinforcement learning (RL) algorithms and policies to control a robot, focusing on creating a residual RL framework. We show the importance of RL methods on the example of the insertion task.

1. Introduction

Fine manipulation tasks require precise, closed-loop feedback and involve high degrees of coordination to be solved. One example of such a fine task is the insertion of building blocks into fitting slots. While this task is already challenging to solve in simulated environments, real-world settings bear the additional challenge of unknown, unmodeled or poorly modeled outer influences like friction or contact forces. Conventional robots counter these problems with a high control stiffness, which makes them more precise, but also faster, stronger and generally more dangerous outside controlled environments (Hyun et al., 2010).
In today's world though, where robotic and human workspace move closer together, robotic control is needed, which delivers similarly good task accuracy while being considerate for its immediate environment and thus minimizing danger and damage to humans, tools and building parts.

[*]Equal contribution [1]Department of Computer Science, Technical University of Darmstadt, Darmstadt, Germany. Correspondence to: Adriaan Mulder <adriaanmariojan.mulder@stud.tu-darmstadt.de>, Nick Striebel <nick.striebel@stud.tu-darmstadt.de>.

Proceedings of the IAS Conference on Robot Learning, TU Darmstadt, Germany. Winter term 23/24. Copyright 2024 by the authors.

While there are many approaches using machine learning, especially imitation learning, to solve this challenge, our task revolves around letting the robot find out for itself how to act safely in high precision environments with the help of reinforcement learning (RL). While we will not show the actual process of RL with the insertion task, we will discuss why RL strategies make sense in this context and how it can be used to learn the task.
We will build an underlying framework that allows the application of residual RL algorithms. The latter will be examined in the next semester, building on the work of J. Hellwig (2023). For the implementation and testing of the framework, we use a Franka Emika panda robot and implement a custom low level impedance controller in ROS, which gives the possibility to change controller parameters in high level programming. We then use 3D-printed parts (a peg and a base, see Figure 1) to evaluate the performance for the insertion task.

2. Related Work

In recent years, many methods have emerged trying to increase robotic safety and performance in contact rich environments (Chen et al., 2023; Stepputtis et al., 2022). Some imitation learning approaches based on dynamic movement primitives (DMPs) use a mechanism to exert a force onto the manipulator to lead it across a planned trajectory (Ijspeert et al., 2013; Paraschos et al., 2018). While these methods are very sample efficient, they do not generalize well in contact rich environments and they bear the problem of constant stiffness, which makes them unsuitable for human or object interaction.
Other Methods based on behavioral cloning (Zhao et al., 2023) perform outstandingly well in contact rich environments, even when using two manipulators working together, but require visual feedback to perform a task. Our focus will be on reinforcement learning (RL), as recent approaches based on RL algorithms also succeeded in many tasks involving robotic control (Ghadirzadeh et al., 2017; Shi et al., 2017; Agrawal et al., 2017).
We build a basis for solving a high precision insertion task with RL, that is based on ideas of Residual RL with Stable Priors (Hellwig, 2023) which revolves around using a nomi-

nal policy to lead the manipulator to the goal location in a stable manner and a residual policy to perform minor corrections to the nominal policy. Our work delivers a framework to parameterize the traditional dynamic PID controller and an interface to learn and execute such a policy with mushroomRL using only the feedback of the internal sensors and actuators of the robot.

3. Theoretical Background

In this section the theoretical basics needed in this work are presented. Furthermore, the selection of the methods is motivated.

3.1. Kinematics

The forward kinematics of a robot

$$\boldsymbol{x} = f(\boldsymbol{q})$$

maps the joint configuration $\boldsymbol{q}$ to the operational space coordinate vector $\boldsymbol{x}$ (position and orientation). With the inverse kinematics

$$\boldsymbol{q} = f^{-1}(\boldsymbol{x})$$

One can derive the needed joint vector $\boldsymbol{q}_{\text{des}}$ to reach a desired target position $\boldsymbol{x}_{\text{des}}$. The robot can then be controlled by using a controller in joint space which compares $\boldsymbol{q}$ and $\boldsymbol{q}_{\text{des}}$. Regarding the insertion task this has multiple downsides. For instance, when linearly interpolating the joint variables between $\boldsymbol{q}$ and $\boldsymbol{q}_{\text{des}}$ without via-points, the path in the operational space is unclear. Inserting something into a hole can fail because the end-effector might not hit the hole from above. Planning the joint trajectory accordingly is complex. In addition, the control using the inverse kinematics only works for open kinematic chains (von Stryk, 2022). During the insertion the peg can get in contact with the base plate resulting in a closed chain.

3.2. Jacobian of a Robot

The Jacobian $\boldsymbol{J}(\boldsymbol{q})$ connects the joint velocities and accelerations with the corresponding components in the operational space (Peters & Schaal, 2006):

$$\dot{\boldsymbol{x}} = \frac{\partial f(\boldsymbol{q})}{\partial \boldsymbol{q}} \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}\,, \quad \ddot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}}\,. \tag{1}$$

3.3. Operational Space Control

The Jacobian $\boldsymbol{J}$ can be used to connect the joint torques $\boldsymbol{\tau}$ with the forces and torques in the operational space $\boldsymbol{F}$ at the end-effector (Peters & Schaal, 2006):

$$\boldsymbol{\tau} = \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{F}\,. \tag{2}$$

Specifying the forces and torques in the operational space is intuitive and easy to implement and has the advantage that the path of the end-effector is directly present, as it follows the force field. Controlling the robot is done by specifying a control signal $\boldsymbol{u} = \boldsymbol{F}$. Using equation (2), $\boldsymbol{u}$ can be converted to the corresponding joint torques. How $\boldsymbol{u}$ is derived is explained in the sections 3.4 and 3.5.
Only applying this would not lead to the desired behavior: Gravitational and coriolis forces must be compensated ($\boldsymbol{\tau}_{\text{g}}$ and $\boldsymbol{\tau}_{\text{c}}$) and it is usual to use a so-called null space control ($\boldsymbol{\tau}_{\text{null}}$). This is done to guide the robot around problematic singularities and to keep him in a configuration of high manipulability (Karnam et al., 2020).
Another element that is needed for high accuracy is the compensation of friction. As in our case no friction model is available, this component is not taken into account and the total joint forces can be computed via

$$\boldsymbol{\tau} = \underbrace{\boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{u}}_{\boldsymbol{\tau}_{\text{task}}} + \boldsymbol{\tau}_{\text{null}} + \boldsymbol{\tau}_{\text{c}} + \boldsymbol{\tau}_{\text{g}}\,.$$

3.4. Impedance Control

The advantage of a PD controller is that it implements a compliant behavior if moderate gains are used (Dong et al., 2020). This is particularly desirable in dynamic and contact rich environments. For targeting a static goal position ($\dot{\boldsymbol{x}}_{\text{des}} = 0$) the PD control law is the following:

$$\boldsymbol{u} = \boldsymbol{K}_{\text{P}}(\boldsymbol{x} - \boldsymbol{x}_{\text{des}}) - \boldsymbol{K}_{\text{D}}\dot{\boldsymbol{x}}\,. \tag{3}$$

The damping $\boldsymbol{K}_{\text{D}}$ is needed to prevent the end-effector from overshooting. Critical damping is achieved for $\boldsymbol{K}_{\text{D}} = 2\sqrt{\boldsymbol{M}\boldsymbol{K}_{\text{P}}}$ (von Stryk, 2022) and incorporates the mass matrix $\boldsymbol{M}$.
Using this control in the operational space has the advantage that the trajectory is known: If no cross-correlation is used, the PD controller (in combination with the operational space control of equation (1)) leads to a movement on a straight lines per axis.
The computation of the error $(\boldsymbol{x} - \boldsymbol{x}_{\text{des}})$ in equation (3) is straight forward for the translational components and can be done element wise. For the rotational error, the axis angle error is chosen as the measure.
As no friction model for compensation is given, high accuracy can be achieved with high gains only. Increasing the gains reduces the compliance and might cause fatal damage in some situations. Another option is the use of a PID controller.

3.5. PID Control

A PID controller expands the control law by a integral term:

$$\boldsymbol{u} = \boldsymbol{K}_{\text{P}}(\boldsymbol{x} - \boldsymbol{x}_{\text{des}}) - \boldsymbol{K}_{\text{D}}\dot{\boldsymbol{x}} + \boldsymbol{K}_I \underbrace{\int \boldsymbol{x} - \boldsymbol{x}_{\text{des}}\,\mathrm{d}t}_{\boldsymbol{e}_{\text{I}}}\,. \tag{4}$$

By integrating the error over time the friction error can be eliminated, but an increasing integral term (e.g. if the robot gets stuck during insertion) can lead to force buildup that can be fatal again. Furthermore, guaranteeing stability is a lot more complex.
Despite these downsides and inspired by the results of Luo et al., 2024 the PID controller is tested as well. In the implementation of Luo et al., the problem of a rising integral value is avoided by clipping the value at every time step:

$$\boldsymbol{e}_\mathrm{I} \leftarrow \mathrm{clip}\left(\boldsymbol{e}_\mathrm{I}, \boldsymbol{e}_\mathrm{I}^\mathrm{min}, \boldsymbol{e}_\mathrm{I}^\mathrm{max}\right) \ . \tag{5}$$

Even though it can become unstable. the integral term can be set to zero every time there is a new $\boldsymbol{x}_\mathrm{des}$.

4. Implementation

4.1. Controller

The controllers from the equations (3, 4) are set to run at a frequency of 1 kHz.
During the experiments the orientation is not varying a lot, but larger gaps between target position and actual position can occur. Because of that, the maximum translational difference between desired and current position is clipped to 5cm along every axis. This prevents too high torques which would result in very high end-effector speeds. While the robot could handle this, the 3D printed parts could experience damage.
The gains $\boldsymbol{K}_\mathrm{P}$,$\boldsymbol{K}_\mathrm{D}$ and $\boldsymbol{K}_\mathrm{I}$ are implemented as diagonal matrices. If this later fails to provide sufficient performance, cross-correlation between the single error components can be added.
Another measure to prevent too high control signals is to restrict the single gain values. The translational and rotational stiffness values of the P component have an upper limit of $1000\,\frac{\mathrm{N}}{\mathrm{m}}$ and $50\,\frac{\mathrm{Nm}}{\mathrm{rad}}$. The damping gains are chosen to implement critical damping and the integral gain is limited by a value of $10\,\frac{\mathrm{N\cdot s}}{\mathrm{m}}$ and $5\,\frac{\mathrm{Nm\cdot s}}{\mathrm{rad}}$. The integral part is restricted even more by clipping the integral value $\boldsymbol{e}_\mathrm{I}$, see equations (4, 5).

4.2. Interface

The interface for communication between the low level controller and the high level RL framework allows the access to the manipulator's sensor outputs and the PID controller's coefficients. In this way, we can update the robots translational and rotational stiffness $\boldsymbol{K}_\mathrm{P}$ dynamically and enable learning by observing sensorical values in accordance to chosen actions. This is achieved by using a `dynamic_reconfigure::server` from ROS.

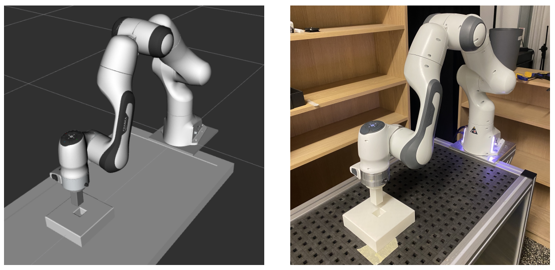


Figure 1. Environment for the insertion task. *Left:* Visualization in rviz. *Right:* Photo from the lab.

4.3. Environment

The environment is chosen to be simple on the hardware side. The peg is a simple cuboid stretched in z-direction. Accordingly, the base provides a square hole. The single elements can be seen in Figure 1.
For testing two configurations of peg and hole are chosen: Both use a squared hole size of $d_\mathrm{h} = 30\,\mathrm{mm}$, but different peg dimensions of $d_\mathrm{p}^{(1)} = 25\,\mathrm{mm}$ and $d_\mathrm{p}^{(2)} = 29\,\mathrm{mm}$, resulting in a larger toleranze $\Delta d^{(1)} = 5\,\mathrm{mm}$ and $\Delta d^{(2)} = 1\,\mathrm{mm}$.

4.3.1. Reward function

The reward function is the same as the reward that is used by Hellwig, 2023:

$$r(\boldsymbol{s}, \boldsymbol{a}) = -\frac{||\boldsymbol{e}||}{||\boldsymbol{e}_0||} \,,$$

and takes the norm of error vector $\boldsymbol{e}$ consisting of the translational error in the first three components and the axis angle error in the last three components. Additionally, this norm is normalized by the initial error at the start of an episode.

4.3.2. Experiment procedure

The procedure of the experiment is displayed in Algorithm 1. During the insertion the policy runs with 10Hz.

The state $\boldsymbol{s}$ is not limited to capture any information. In this case it contains the current configuration $\boldsymbol{x}$, the goal position $\boldsymbol{x}_\mathrm{g}$, the measured forces at the end-effector $\boldsymbol{f}$ and the currently used stiffnesses. Depending on future performances the state definition can be adjusted, foe example by using an external signal such as the RGBD image of the scene.
The policy π returns an intermediate target position $\tilde{\boldsymbol{x}}_\mathrm{g}$ and adapted stiffnesses as the action $\boldsymbol{a}$.
Sending the selected action to the robot takes $(5.7 \pm 0.8)\,\mathrm{ms}$. This takes roughly 5% of the 10Hz interval. This has to be considered later if the RL algorithm and the complexity of

Algorithm 1 Insertion Experiment

```
 1: Input: home $\boldsymbol{x}_\mathrm{h}$, goal $\boldsymbol{x}_\mathrm{g}$, goal radius $\boldsymbol{e}_\mathrm{max}$
 2: for $n = 1, \ldots, N$ do
 3:   Reset($\boldsymbol{x}_\mathrm{h}$, $\boldsymbol{x}_\mathrm{g}$)
 4:   Randomly shift start position
 5:   $\boldsymbol{s} \leftarrow$ current state
 6:   for $t = 1, \ldots, T$ do
 7:     $\boldsymbol{a} = \pi(\boldsymbol{s})$
 8:     Execute action $\boldsymbol{a}$ and observe $\boldsymbol{s}'$ and $r$
 9:     [Potential RL step using $\boldsymbol{s}$, $\boldsymbol{a}$, $r$]
10:     if $\boldsymbol{e} < \boldsymbol{e}_\mathrm{max}$ then
11:       break
12:     end if
13:     $\boldsymbol{s} \leftarrow \boldsymbol{s}'$
14:   end for
15: end for
```

Algorithm 2 Reset Function

```
1: Input: home pos. $\boldsymbol{x}_\mathrm{h}$
2: $\boldsymbol{x} \leftarrow$ current end-effector configuration
3: if $\boldsymbol{x}_z < z_\mathrm{min}$ then
4:   $\boldsymbol{x}_z \leftarrow z_\mathrm{min}$
5:   Move upwards to $\boldsymbol{x}$
6: end if
7: Move to $\boldsymbol{x}_\mathrm{h}$ in a straight line
```

the agent gets specified - although the time delay caused by executing the command is small.
The reset function implements a save way to go to the home position and is illustrated in Algorithm 2. Before moving straight to the home position it makes sure that a minimum z-height is reached. This prevents the robot from getting stuck inside the bases hole or at an edge of the basis. Having a good running rollout function with a save reset function is important to later run longer RL trainings without having to restart the experiment by hand every time something went wrong.

4.4. Policy

Before a manual policy is designed, it makes sense to consider the possible scenarios. Figure 2 displays the three possibilities. In the first, the robot can move the peg on a straight line into the hole. This case might be the rarest. In the second scenario the robots end-effector starts high enough above the base surface that he can reach the hole without having to move upwards. In the last case, the robot has to move upwards on its way to the hole, otherwise the peg will get stuck at a wall of the base.

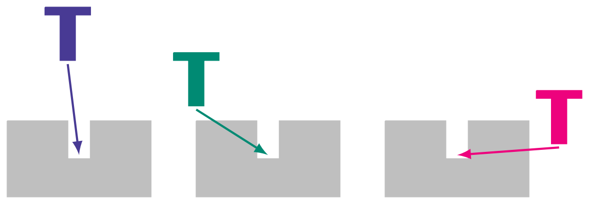


Figure 2. Possible scenarios for the insertion task. Blue: The peg can be inserted directly. Green: The peg starts above the base surface but cannot reach the target in a straight line. Red: The peg starts below to top surface of the base.

4.4.1. Plain PD/PID Controller

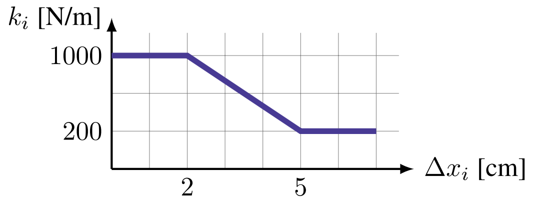
Using a simple controller with fixed gains is only capable of inserting the peg if the hole can be reached in a straight line - blue scenario in Figure 2. As soon as the peg gets in contact with the surface of the base, the friction gets too high for the manipulator to move further. This also fails in the PID case, as only a small integrational gain can be used due to higher gains leading to oscillating behavior.

4.4.2. PD/PID Controller with Variable Gains

Extensive testing in the environment with important components like the reset function is only possible if a policy is used that has the chance to solve the task. Therefore, a simple hand-made policy $\tilde{\pi}$ is constructed which adjusts the gains of the controller in a simple way. At this point, it should be mentioned, that a policy should be developed that can solve the insertion task without requiring major engineering effort, as further improvements are to be achieved later with RL.
As a first consequence the designed policy does not have any via-points and directly uses the final position as the goal position for the controller. $\tilde{\pi}$ adjusts the translational stiffness, the damping gain is chosen accordingly for critical damping and if the integral term is used the integral gain is kept constant.
The basic adjustment for the translational stiffness is based on the distance to the target position. Far away from the goal position, the controller gains are kept low, as the variance in the movement is high. Getting closer to the goal, the gain is increased linearly up to a maximum value, see Algorithm 3, line 4. With this behavior the robot gets stiff only in situations where it matters: Close to the hole.
On top of this behavior, the translational gains are adjusted referring to the force measurement: If the force in one dimension i surpasses a threshold f_i^lim this is classified as contact and the dimensional gains $K_{\mathrm{P},i}$ and $K_{\mathrm{D},i}$ are set to zero. This drastic reduction of gains reduces the force in the affected dimension. The affected gains will slowly be increased over time again as the directional movement in the other dimensions will close the gap to the goal position

Algorithm 3 Hand-Coded Policy $\tilde{\pi}$

```
1:  Input: $\boldsymbol{x}$, $\boldsymbol{x}_\text{g}$, $\boldsymbol{k}^\text{old}$, $\Delta\boldsymbol{k}$, $\boldsymbol{f}^\text{lim}$
2:  for $i \in \{x, y, z\}$ do
3:      $\Delta x_i \leftarrow |x_i - x_{\text{g},i}|$
4:      Select $k_i$ based on distance:
```



```
5:      if $|f_i| > f_i^\text{lim}$ then
6:          $k_i \leftarrow 0$
7:      end if
8:      $k_i \leftarrow \text{clip}(k_i, 0, k_i^\text{old} + \Delta k_i)$
9:  end for
10: $\boldsymbol{K} \leftarrow \text{diag}(k_x, k_y, k_z)$
```

and we need the gain in the last dimension to complete the insertion. In this case the change of the stiffness is limited by Δk (per axis) compared to the previous time step. Algorithm 3 illustrates the total behavior. Figure 5 highlights the gain selection with respect to the force measurement.
With more engineering effort designing a superior policy would be possible. Yet, the goal of this work is to derive a framework, that enables successful insertion tasks without this additional effort. In saying that, RL is justified if a rather simple policy is not capable of solving the task.

4.4.3. Starting below the base surface

If the end-effector has to move up before targeting the entry of the hole, the hand-coded policy fails as well, see Figure 6. It would be possible to implement a more complex behavior to achieve a successful insertion by adapting the gains and/or the intermediate target position. Also using Dynamic Movement Primitives or other hand-coded policies with intermediate target positions would be an option for this insertion. Both options again include engineering effort. Achieving a well performing policy is left for next semester. It will then be the goal to use a residual policy that learns to adapt a simple policy (e.g. a PD controller).

5. Experimental Results

Some general experiments with the real Franka robot are presented before the actual insertion task is evaluated.

5.1. General Accuracy

Before performing an actual insertion the general accuracy of the robot is evaluated. Therefore, the robot is controlled to target 100 random positions in box of size $10 \times 10 \times 10$ cm, starting in the center of the virtual box.

Table 1. Accuracy of different controllers depending on the translational gains. The translational error $e_{\boldsymbol{x}}$ displays the euclidean error. The orientation error $e_{\boldsymbol{\alpha}}$ is measured by using the axis angle error. The stiffness's refer to the translational ones, the rotational ones are kept constant with 10 $\frac{\text{N}}{\text{m}}$, 6 $\frac{\text{N}\cdot\text{s}}{\text{m}}$ and 5 $\frac{\text{N}}{\text{m}\cdot\text{s}}$.

$\boldsymbol{K}_\text{P}$ $\left[\frac{\text{N}}{\text{M}}\right]$	$\boldsymbol{K}_\text{D}$ $\left[\frac{\text{N}\cdot\text{S}}{\text{M}}\right]$	$\boldsymbol{K}_\text{I}$ $\left[\frac{\text{N}}{\text{M}\cdot\text{S}}\right]$	$e_{\boldsymbol{x}}$ [MM]	$e_{\boldsymbol{\alpha}}$ [°]
200	28	-	22 ± 9	2.7 ± 1.1
1000	63	-	4.9 ± 2.0	2.4 ± 1.1
200	28	10	20 ± 8	1.1 ± 0.9
1000	63	10	3.9 ± 1.9	1.2 ± 1.0

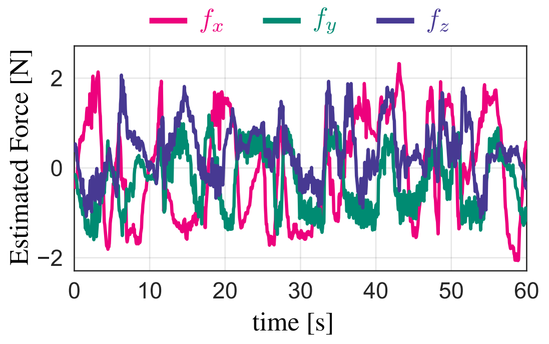


Figure 3. Internal force estimation of Franka. The robot is randomly moved in a $10 \times 10 \times 10$ cm large box. The force estimation is usable, as it stays in a range of approx. ± 2 N for external forces $\boldsymbol{F}_\text{ext} = 0$.

The accuracy is evaluated for controllers of the cross-combinations of $\{\text{PD}, \text{PID}\} \times \{\text{low gains}, \text{high gains}\}$. The damping is always chosen to be $\boldsymbol{K}_\text{D} = 2\sqrt{\boldsymbol{M}\boldsymbol{K}_\text{P}}$, implementing a critical damped controller in the PD case. The integral gain is chosen to be constant - but small - to avoid the problem of too strong error accumulation. To further reduce the risk of a rising integral value, we use the technique of Luo et al. to clip the value as well as resetting the integral term to zero every time we set a new $\boldsymbol{x}_\text{des}$ which happens at a frequency of 10Hz. The translational accuracy is measured by the euclidean norm, while the orientation error is evaluated using the axis angle error. The results can be seen in Table 1.
It can be seen, that the integrational term is important for scenarios requiring a high accuracy. Using high gains in a PD controller might show good results too, but as already explained the missing compliance of the robot is a factor, that weakens the results of this option.

5.2. Force Measurement

An important component to perceive contact during the insertion is the internal force measurement provided by

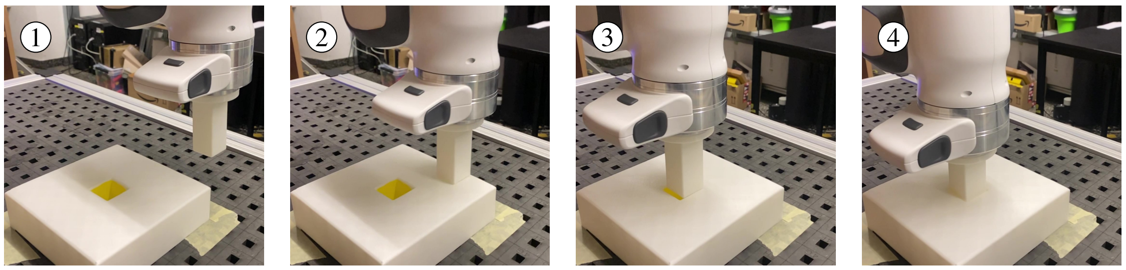


Figure 4. Successful insertion with the PID-policy and the small tolerance of 1 mm. The frames correspond to the markers in Figure 5. Between frame 2 and 3 the peg stays in contact with the surface all the time.

Table 2. Evaluation of the hand-coded policy for 100 episodes each. Each episode has a random start position above the surface of the base. Scenarios like in Figure 6 are not included.

Δd[MM]	$\tilde{\pi}$	SUCCESS RATE	COMPLETION TIME [S]
5	PD	100%	1.76 ± 0.31
	PID	100%	1.64 ± 0.07
1	PD	FAILS	FAILS
	PID	100%	3.72 ± 2.60

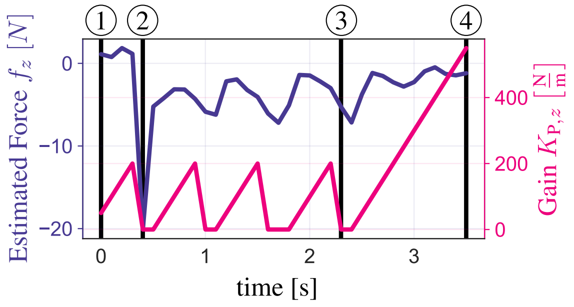


Figure 5. Internal force estimation of Franka during insertion in z-direction. Additionally, the stiffness in z-direction, derived by the hand-coded policy is shown. Please refer to Figure 4 for the marked positions.

Franka. To analyze the quality of these values a force measurement is carried out. During this measurement the robots end-effector moves randomly in a $10 \times 10 \times 10$ cm box. The results are displayed in Figure 3.

If the force estimation would be ideal, constant forces of 0 would be measured during the movement. This is clearly not the case. However, the force values stay in a limited area. This allows to identify the force limits of Algorithm 3 to $f^{\mathrm{lim}}_{x,y,z} = 5\,\mathrm{N}$. This value is chosen to be close to the maximum observed forces while leaving some room for small outliers.

5.3. Solving the Insertion Task

In this section, the performance of the hand-coded policy $\tilde{\pi}$ is evaluated. Therefore the two standard measures *success rate* and *completion time* are evaluated. Please note that testing was done with start positions above the base surface only.

Testing with small ($\Delta d = 1$ mm) and large ($\Delta d = 5$ mm) insertion tolerance the two policies $\tilde{\pi}^{\mathrm{PD}}$ and $\tilde{\pi}^{\mathrm{PID}}$ are analyzed. The two policies only differ in the use of an integral term $\boldsymbol{K}_{\mathrm{I}} = 0$ and $\boldsymbol{K}_{\mathrm{I}} \neq 0$.

An example for an successful insertion can be seen in Figure 4. The corresponding forces and gains are plotted in Figure 5. The quantities for evaluation can be found in Table 2.

In the first phase, the manipulator moves towards the goal position in a linear fashion until it comes in contact with the base. In the second phase, the manipulator still tries to move towards the goal position linearly, which leads to a measured positional error and thus a force buildup, as it cannot continue along the worlds z-axis. Breaching the force threshold will reset the z-dimensional gains as mentioned in the policy. This reduces friction and gives the manipulator time to move further towards the hole while the z-dimensional gains increase again. This process is repeated several times as seen in Figure 5 between markers 2 and 3, which results in a sawtooth wave for the gain in z direction. These jumps are ultimately an undesired behavior and should be eliminated through a learned policy. After reaching the hole, the peg will slide into it with increasing z-dimensional gains.

An entire episode's success and duration depends on the choices of controller and required accuracy. When starting the episode over the base block and with a desired accuracy of 5 mm, the peg is inserted successfully and far enough into the hole every time. With an desired accuracy of 1 mm the PD controller is unable to complete the task, what is expected, regarding the accuracy in Table 1 and the fact, that no compensation for friction is implemented. For the

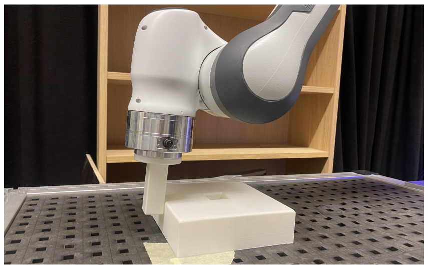


Figure 6. Failure case. Starting below the surface of the base, the peg gets stuck at the wall of the base.

1 mm tolerance the PID controller takes up to four times as long but still manages to complete the insertion. This high accuracy seems surprising in comparison to the results of Table 1. This is because of the choice of the robots null space task. It is a relevant factor to reach precise results, as the success rate also falls drastically when choosing a null space configuration that is not directly above the hole. A possible explanation can be given by considering numerical errors and estimation errors that occur calculating τ_{null}. With these errors the calculated torque also has a small influence outside of the null space task. Cases in which the randomized starting position is below the bases surface are not taken into the statistic, as the hard coded policy is not suitable for those cases, as shown in Figure 6.

6. Conclusion & Outlook

Crafting a real-world framework to implement residual RL algorithms for a robot manipulator proves to be quite challenging. This is why a deep understanding of the intricacies of the robot, its control and its environmental influences is necessary to build a basis which provides safety to its users and environment. Using the gained knowledge about the working of the system allowed for building a mechanism that enables learning and executing control policies. In this case this was used to program a policy executing a relatively simple insertion task. Even though this hand-crafted policy works in a small spacial range, it does not provide a smooth behavior and thus a learned policy is needed to substitute or improve the simple policy used in this project. In future, more complex insertion tasks could considered. Due to more complex arrangements or tasks that require a high level of sensitivity, policies with a very high level of engineering effort would be needed. The goal is to create and test an residual RL framework that enables the learning of efficient policies without much prior knowledge or domain knowledge. It should be mentioned that developed framework is specifically suited for residual learning. General RL would require more safety measures.
Using this work as basis, the learning and implementation of a residual policy using stable priors as defined by (Hellwig, 2023) is possible - in the real world. This should result in smoother movement with higher accuracy and fewer undesired contacts between the manipulator and its environment as well as the elimination of the integrational controller component. The latter is generally not desired in trajectorial control due to its error buildup.

References

Agrawal, P., Nair, A., Abbeel, P., Malik, J., and Levine, S. Learning to poke by poking: Experiential learning of intuitive physics. 2017.

Chen, Y., Geng, Y., Zhong, F., Ji, J., Jiang, J., Lu, Z., Dong, H., and Yang, Y. Bi-dexhands: Towards human-level bimanual dexterous manipulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–15, 2023. doi:10.1109/TPAMI.2023.3339515.

Dong, Y., Ren, T., Wu, D., and Chen, K. Compliance control for robot manipulation in contact with a varied environment based on a new joint torque controller. *Journal of Intelligent & Robotic Systems*, 99(1):79–90, Jul 2020. ISSN 1573-0409. doi:10.1007/s10846-019-01109-8.

Ghadirzadeh, A., Maki, A., Kragic, D., and Björkman, M. Deep predictive policy training using reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2351–2358, 2017. doi:10.1109/IROS.2017.8206046.

Hellwig. Residual reinforcement learning with stable priors, 2023.

Hyun, D., Yang, H. S., Park, J., and Shim, Y. Variable stiffness mechanism for human-friendly robots. *Mechanism and Machine Theory*, 45(6):880–897, 2010. ISSN 0094-114X. doi:10.1016/j.mechmachtheory.2010.01.001.

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., and Schaal, S. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013. doi:10.1162/NECO_a_00393.

Karnam, Parini, Eugster, Cattin, Rauter, and Gerig. An intuitive interface for null space visualization and control of redundant surgical robots. *Proceedings on Automation in Medical Engineering*, 2020.

Luo, J., Hu, Z., Xu, C., Tan, Y. L., Berg, J., Sharma, A., Schaal, S., Finn, C., Gupta, A., and Levine, S. Serl: A software suite for sample-efficient robotic reinforcement learning, 2024.

Paraschos, A., Daniel, C., Peters, J., and Neumann, G. Using probabilistic movement primitives in robotics. *Au-*

tonomous Robots, 42, 03 2018. doi:10.1007/s10514-017-9648-7.

Peters, J. and Schaal, S. Learning operational space control. *Proceedings of Robotics: Science and Systems (RSS), Philadelphia, PA*, 01 2006.

Shi, J., Woodruff, J. Z., Umbanhowar, P. B., and Lynch, K. M. Dynamic in-hand sliding manipulation. *IEEE Transactions on Robotics*, 33(4):778–795, 2017. doi:10.1109/TRO.2017.2693391.

Stepputtis, S., Bandari, M., Schaal, S., and Ben Amor, H. A system for imitation learning of contact-rich bimanual manipulation policies. 07 2022. doi:10.48550/arXiv.2208.00596.

von Stryk, O. *Grundlagen der Robotik*. 2022.

Zhao, T., Kumar, V., Levine, S., and Finn, C. Learning fine-grained bimanual manipulation with low-cost hardware. 04 2023.