Initial tactile, visual and functional task verification: description, evaluation and open source software

Deliverable 4.2

Deliverable Title	D4.2 Initial tactile, visual and functional task verification
Deliverable Lead:	Commissariat à l'Énergie Atomique et aux Énergies Alternatives (CEA)
Related Work Package:	WP4: Traceability Framework
Related Task(s):	T4.1: Create the Traceability Framework based on Digital Twin, T4.2: Tactile task verification, T4.3: Visual task verification, T4.4: Functional task verification
Author(s):	Markus Vincze (TUW), Jean-Baptiste Weibel (TUW), Patrick Mania (UOB), Franklin Kenghagho Kenfack (UOB), Michael Neumann (UOB), Saifeddine Aloui (CEA)
Dissemination Level:	Public
Due Submission Date:	28/2/2023
Actual Submission:	28/2/2023
Project Number	101017089
Instrument:	Research and innovation action
Start Date of Project:	1.1.2021
Duration:	51 months
Abstract	This deliverable describes the TraceBot approach to create a verification step for realising the traceability framework for laboratory automation. It outlines how the first versions of tactile, visual and functional verification work and are integrated into the verification framework.

Horizon 2020

Versioning and Contribution History

Version	Date	Modified by	Modification reason
v.01	17.01.2023	Markus Vincze (TUW)	1st skeleton
V.02	10.02.2023	Saifeddine Aloui (CEA)	Tactile verification contribution
V.03	10.02.2023	Jean-Baptiste Weibel (TUW)	Visual verification contribution
V.04	15.02.2023	Franklin Kenghagho Kenfack (UOB)	Functional verification contribution
v.05	21.02.2023	Saifeddine Aloui (CEA)	Version ready for internal revision
v.06	21.02.2023	Carl-Helmut Coulon (INV)	1st internal revision
v.07	23.03.2023	Markus Vincze (TUW), Saifeddine Aloui (CEA), Franklin Kenghagho Kenfack (UOB)	Document revision
v.08	23.03.2023	Anthony Remazeilles (TECN)	2nd internal revision
v.09	24.03.2023	Saifeddine Aloui (CEA)	Final version ready for submission

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

Table of Contents

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

1 Executive Summary

The TraceBot project aims to establish a comprehensive traceability framework that adheres to laboratory automation regulations in the pharmaceutical industry. Our proposed verification methods employ various modalities to ensure accurate execution of processes and the creation of an audit trail. By replacing standard actions with Traceable Actions, we systematically capture and log all operations and actions conducted by the robotic system. This enables us to track the progress of the process and ensure that its ultimate goal is reached.

In addition to capturing the process structure, Traceable Actions also automatically gather and send their input and output onto a specific and unique channel. This ensures that all relevant subsymbolic and sensor data is logged and stored. To further enhance the capabilities of the traceability framework, we utilize KnowRob, a knowledge processing system for robots. KnowRob is capable of storing and processing both symbolic and subsymbolic information related to the execution of the process. By combining the data gathered by Traceable Actions with KnowRob's memory episodes, we can create a Narrative-Enabled Episodic Memory (NEEM). This NEEM can be meaningfully queried by humans during inspection, allowing them to quickly and accurately understand the success or failure of any action taken by the robot without needing to meticulously inspect all sensor data.

To ensure that the execution of the process is going as expected, we have developed a set of verification or checking actions using visual and tactile sensing, as well as functional considerations thanks to the knowledge infrastructure provided by KnowRob. These verification steps are also implemented using Traceable Actions, guaranteeing that all information is traced and logged.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

2 Introduction

This document gives an update on the methods for task verification as part of the traceability framework developed for the TraceBot project. Its purpose is to produce a meaningful audit trail adapted to the regulatory constraints of laboratory automation, and specifically, sterility testing, which is the use case that the TraceBot project focuses on.

Traceability consists in both the collection of all the data relevant to the process, and its presentation in a meaningful and usable way to spot and understand failures. The consortium therefore focused in developing tools that enable the automatic collection of all the data used during the process execution and its presentation in a meaningful format. The verification steps are essential in such a regulated domain as sterility testing. To obtain traces of what the robot is actually doing, we propose to use three modalities of verification: tactile, visual, and functional verification. The different modalities are used to verify the correct execution of the process according to the needs in different steps of the robot assembly process. Since these steps are also traced, this guarantees the inclusion of any important data into the final audit trail.

The traceability framework in itself has been presented in D4.1. No significant changes happened in the traceability framework. The concept presented in the D4.1 is still in place. Each action is wrapped automatically with custom code, that enables logging the input and output of every item happening. This code also starts a Narrative-Enabled Episodic Memory event for each action. This information is then used and presented in a compact form showing which hierarchy of actions was executed, and whether the checking actions (verifying the good execution of the process) are successful or not. Link to the detailed information contained in the NEEM is provided in the same document.

The following document outlines a comprehensive approach to task verification. Firstly, an overview of the tasks to be addressed is presented, followed by a detailed description of the tactile sensing approach used for task verification. Next, the vision-based method employed for task verification is presented, and lastly, functional task verification is addressed.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

3 Tasks for Tactile, Visual, and Functional Verification

The traceability objective cannot be completely addressed if the robot is not equipped with means to verify the successful execution of the process. Therefore, aside the implementation of the traceability framework, TraceBot aims at developing a set of operations which purpose is to confirm through sensing and acting that the environment state is as expected.

To actually verify the robot actions, three modalities are proposed following the tasks T4.2 to T4.4. This section covers these different modalities and their use toward meaningful verification: Tactile Task Verification in Section 4.1, Visual Task Verification in Section 4.2, and Functional Task Verification in Section 4.3.

An overview of the steps of the assembly process of the sterility kit is given in Table 1 below. It indicates the task, what objects are handled, a key challenge to automate the step, if one or two robot arms will be necessary to complete the task, the necessary perception and an excerpt of the verification tasks. The steps including a verification task are marked black and also refer to the Milestones as described in D1.6 v2.

Table 3.1: The process steps for assembling the sterility kit that will have verification actions (black) as part of the entire assembly procedure (other process steps in grey). The marked steps (black) correspond to the Milestone description in D1.6v2.

#	Tasks	Objects to be handled	Challenge	Arms	Necessary Perception	Verification
1	Manual Preparation	Equipment set up	Scene understanding	1-2	locate all items	Model in Digital Twin
2.1	Kit unpacking – Open Pack	Pack, Tyvek-foil	Flexible material	2	locate pull-tab top corner	foil removed
2.2	Kit unpacking – Remove Kit	SteriKit	Flexible connected parts	1-2	locate grasp point	all parts present
3.1	Kit mounting - Fit Canister To Drain	canister (2)	Click, entangle tubing, small tolerances	1	locate top of canister, seat…	Check relative pose
3.2	Kit mounting – Insert Tube Into Pump	tubes, pump	Flexible material	2	locate tube	pull at tube
4.1	Needle preparation – Remove Needle Cap	needle set (of sterility kit)	Small, force	2	locate cap element	can see needle
4.2	Needle preparation – Insert Needle Into Bottle	needles, bottle	Perforating, force	2	locate needle, sense vibrations	Check force profile
5	Wetting	Pump, canister, plugs & bottle	Operate pump	1	-	read pump data

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

6	Sample Transfer	Pump, canister, plugs and vials	Vial breaking	2	locate vials	confirm brake and emptyness
7	Filter Sample	pump, canister, plugs & bottle	Operate pump	1	-	read pump data
8	Washing	pump, canister, plugs & bottle	Operate pump	1	-	read pump data
9	Media filling	pump, canister, plugs & bottles	Operate pump	1	-	read pump data
10	Cut And Close	canister, cutter, clamp	Cutting, force	1-2	locate clips, sense vibrations	can see cut tube, detect clamp closure
11	Finish	pump, canister and tubes	No dropping of fluids	1-2	locate tubes	In storage location
12	Manual Finish	Equipment reset	Scene understanding	1-2	locate all items	Model in Digital Twin

In the following sections, we will present in detail each verification modality that we have developed. These modalities utilize various techniques such as visual and tactile sensing, functional considerations, and knowledge processing systems, mainly KnowRob. By thoroughly examining each verification modality, we aim to provide a comprehensive overview of the measures we have taken to ensure the accuracy and compliance of our traceability framework.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

4 Tactile Task Verification

Tactile verification will be implemented in a comprehensive and systematic manner for every robot grasping action and a number of assembly actions. Specifically, during grasping actions, tactile information can be utilized to ensure the proper acquisition of objects (sub section 4.3). Additionally, for specific verification actions, tactile feedback can be employed to capture and analyze the evolution of forces while executing an operation, such as when closing the clamp (subsection 4.4) or when inserting the needle into the septum (subsection 4.5). For the two latter, a generic model approach has been evaluated (subsection 4.6).

In order to ensure the accuracy and reliability of the tactile verification process, it can be mostly executed in segments. This approach entails activating the verification routine prior to the initiation of the task, and deactivating it upon completion of the task. During this time, the routine buffer the sensor signal. Once the logging is complete, a task-specific verification algorithm can be applied to produce a binary output indicating the success or failure of the task.

For continuous gripping surveillance, the TraceBot tactile sensors can also provide valuable information regarding the slip detection (subsection 4.7).

Since the TraceBot gripper is not ready by this time, we have mainly used the Franka Emika's Panda cobot manipulator with a single sensor placed on its bi-digital gripper to perform the training, then we have tested on the one finger setup of TraceBot to verify that the training can be transferred to another setup.

4.1 Notations

For the following text, we are going to be using those notations to describe the tools we are using:

- Tactipatch sensor: The hybrid piezoelectric-piezoresistive tactile sensor developed by CEA-LETI and used for each phalanx of the gripper as well as the palm (see details in deliverable D2.2).
- Tactipatch software: A metrology tool designed to connect to tactipatch sensors, log data, annotate data, as well as realtime testing. This is used for our video demonstrations.
- Piezoelectric sensor: The piezoelectric sensor part of Tactipatch, running at 10KHz per patch. There is a single sensitive element for each patch.
- Piezoresistive sensor: The piezoresistive sensor part of Tactipatch, running at 100Hz per patch. Each sensor returns a 8x8 matrix measure (64 sensels) that can be interpolated to much higher resolution using either AI-based techniques or simple interpolation algorithms.
- Sensel: A sensory pixel that returns a function of a local deformation at position x,y inside the Tactipatch piezo resistive array.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

- Panda Robot: A manipulation Cobot built by Franka Emika and used by our lab for testing our individual sensors. The cobot was modified to integrate tactipatch sensors in its end effector.
- Carousel: A linear 3 axis robot with integrated force measurement sensor and an interface that can be synchronized with our system with 10ms uncertainty.
- LSB (Least Significant Bit): A raw unit depicting the direct output of the piezoelectric sensor in binary.

4.2 Presenting Tactipatch software

Tactipatch software is a data logging tool for our hybrid tactile sensors that we have built in order to train models to do multiple tactile related tasks. In particular, the tactile-based task verification for multiple use cases. The tool enables both the recording and the real time verification process through a user-friendly interface (*Figure 4.1*).

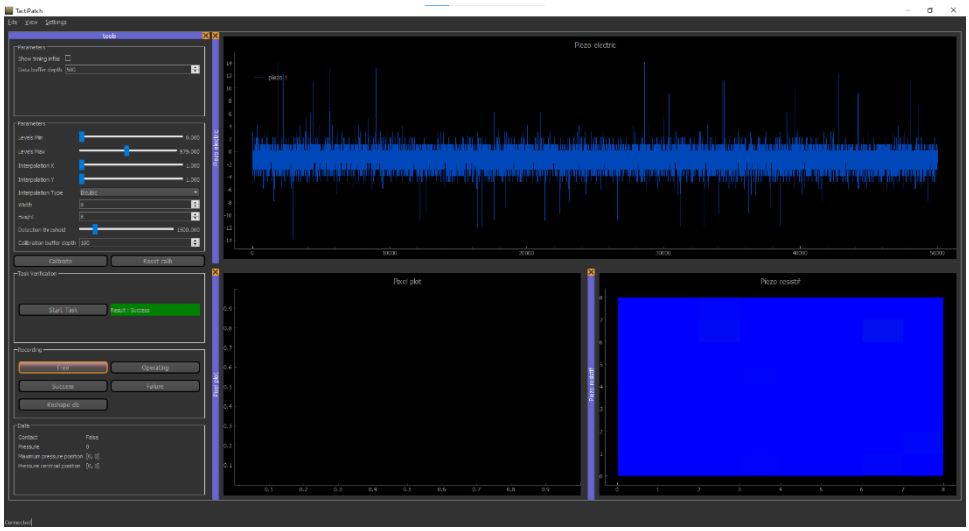

Figure 4.1 : Tactipatch User Interface for logging and Realtime data visualization as well as testing the algorithms

To record data, we place the panda robot in a ready state. We connect the tool to the Tactipatch sensor and then we start the recording process (*Figure 4.2*).

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

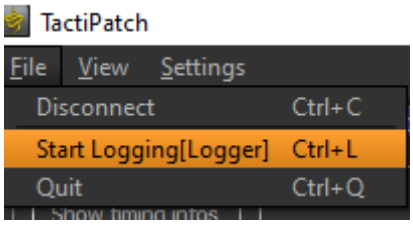

Figure 4.2 : How to start the log via the TactiPatch UI

The recording group box of buttons is used to annotate the actual status of the operation (*Figure 4.3*). By default the system is in free state which is a state. Before starting the clamp closure or the needle insertion operation (depending on what kind of verification is required), we press the operating button. This raises a flag that the system should buffer data to check if the operation is successful or not. In real application, this will be done using a ROS communication. Once the operation is done, we press free button to indicate that the operation ended, then we press either success or failure, according to the situation. Then we move to the next test. This tool enables recovering large amounts of tests with human annotations.

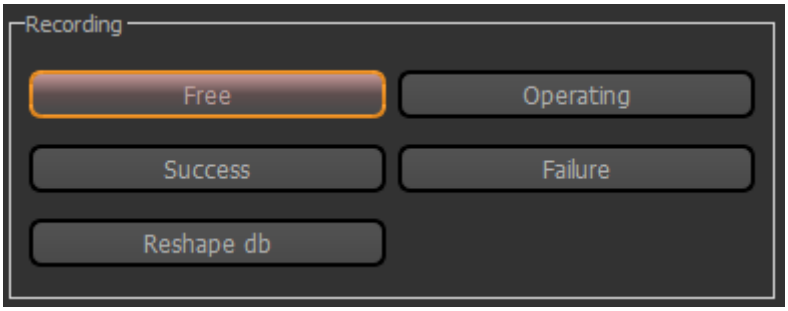

Figure 4.3: Zoom on the recording box of the TactiPatch UI

Once the experimentation session is done, we stop logging and we postprocess the file by pressing the *Reshape db* button that will transform the recording into an easier to use database for machine learning purposes.

This tool is independent from the robot as we can plug it to any of our sensors and record both piezoelectric and piezoresistive signals simultaneously while adding annotation.

4.3 Object grasping task verification

In this section, we will address the verification of the presence of an object in the hand after performing a pick and place task, such as grasping a canister, bottle, needle, etc. The verification process aims to ensure that the object to be grasped is securely held by the gripper

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

at the end of the operation. This can be achieved by utilizing tactile feedback to confirm the presence of the object.

In our case, the hand is equipped with both piezo electric and piezo resistive sensors that allow us to sense spatial and temporal tactile information at low and high frequencies. For the object grasp verification, we rely on the piezo resistive sensor that allows us to sense contact between the different parts of the hand with another surface.

The verification task is activated at the onset of the grasp operation. As the object comes into contact with the gripper, piezo resistive sensors are utilized to detect the contact by counting the number of sensels (sensing pixels) that were activated on different piezoresistive sensors as well as measuring their values. This enables the robot to understand that its phalanges are touching something, but does not confirm that the phalanges are not touching each other.

To solve this problem we rely on the robot proprioception capability, offered by the positional encoders in its articulations. Using this information combined with the tactile sensing, it is possible to determine the 3D contact positions of each activated sensel in the reference frame of the gripper. If the two phalanges are touching each other, the sensel cloud points of the two phalanges overlap. This can be used to discriminate this case (*Figure 4.4*).

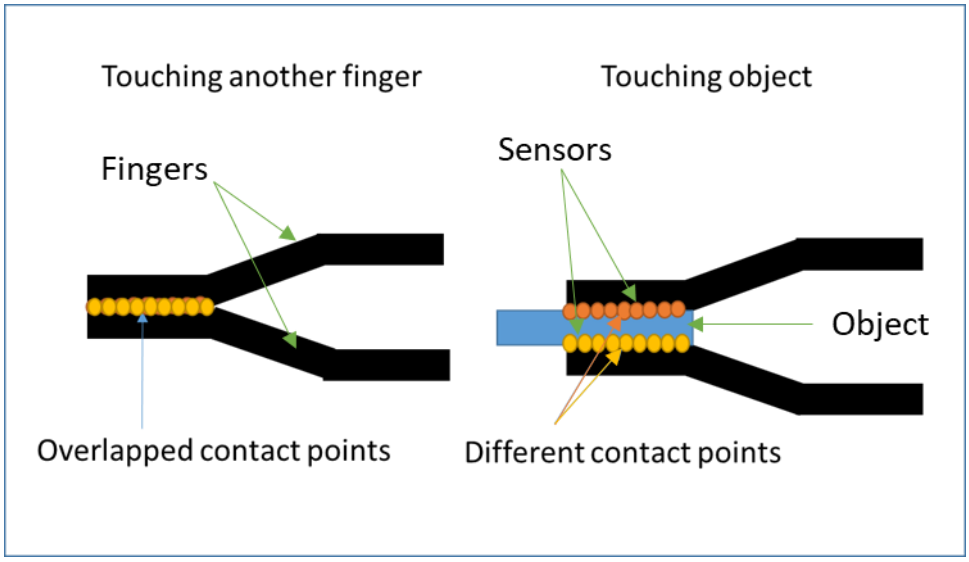

Figure 4.4: Comparison of the case of fingers (black parts) touching each other or fingers touching the object (blue part).

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

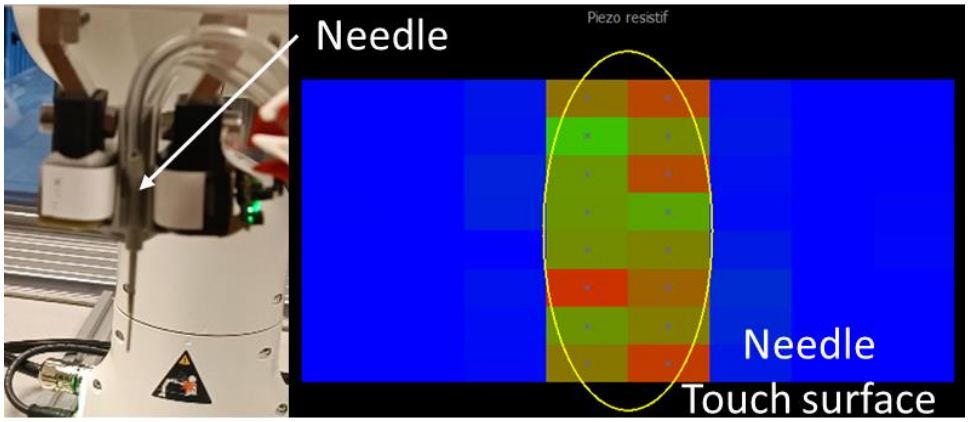

Figure 4.5: 2d touch pixels (sensels) detection while holding the needle vertically.

We have implemented the contact detection in real time using the panda robot as shown in *Figure 4.5*. The next steps (use of proprioception to validate that it is not the fingers touching each other) require the full gripper to be ready, so this will be dealt within the next deliverable.

4.4 Clamp closure task verification

The clamps must be closed by the gripper (e.g. for task #10 in Table 3.1). During this task, a click sound can be heard. To validate that the clamp was closed successfully, we can use our piezoelectric sensors. In fact, the vibration caused by the close event has a very specific signature. We use this to determine if the clamp was closed or not on our Panda Robot (*Figure 4.6* and *Figure 4.7*).

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

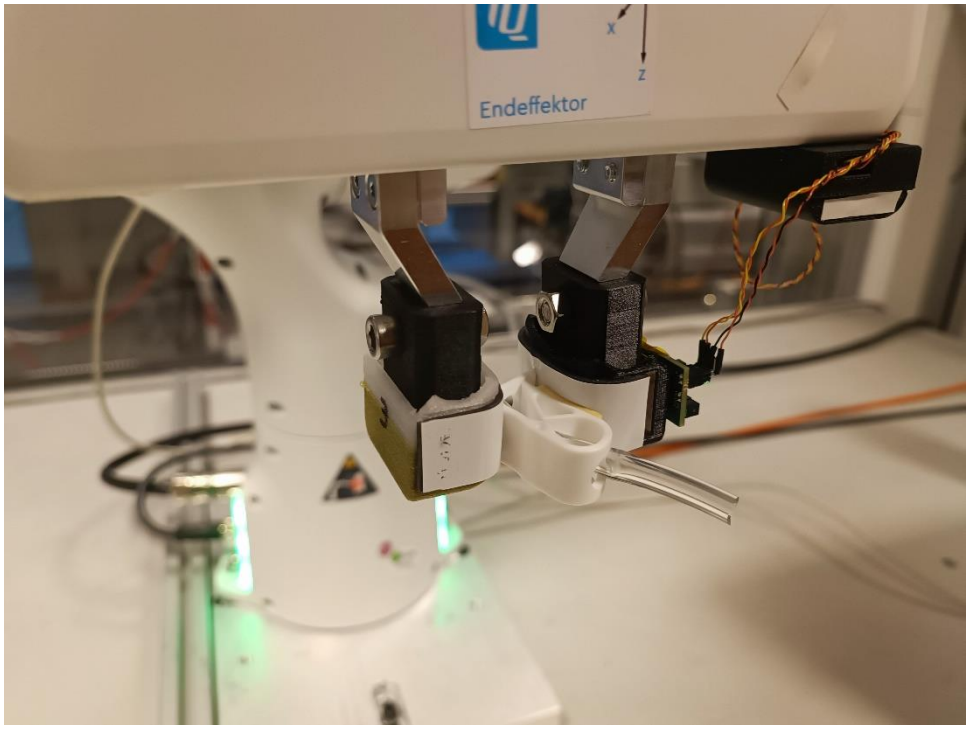

Figure 4.6: Clamp closure setup.

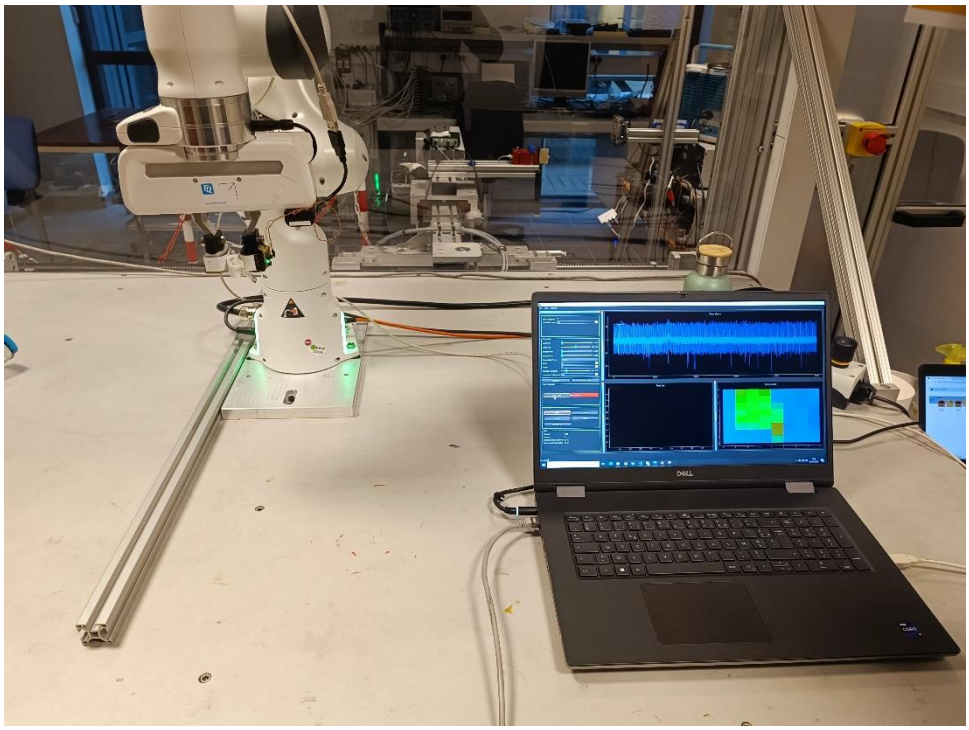

Figure 4.7: Full testing setup with the robot, Tactipatch application software using a single sensor mounted at the Pandas's gripper.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

The process is as follows:

1. Before starting the closure operation, we activate the clamp click detection module.
2. We perform the operation. The module buffers piezoelectric sensor signal.
3. When the operation is done, we query the module for success status.
4. The module stops buffering and performs the detection:
    a. Compute the mean of the signal
    b. Remove the mean from the original signal
    c. Find the maximum position by computing the absolute value of the signal.
    d. Create a 40 samples window around the maximum
    e. Remove the mean value from the new signal
    f. Feeds this signal to the classifier
    g. Return the output of the classifier (success or failure)

We have built a balanced training dataset. We have also built a balanced validation and testing datasets.

To ensure that our results are robust and applicable to real-world scenarios, we included external perturbations in all of our experiments. These perturbations simulate potential sources of unexpected events, such as vibrations from the motors, movement of pipes during operation, or accidental impacts during manipulation. We sought to incorporate realistic scenarios, as the verification period was limited and the robot was isolated, so there was no need to include highly unlikely events. This approach allows us to achieve a high degree of accuracy with minimal computational resources.

In order to prevent overfitting of the model to the robot, we established an initial training database of 120 tests, consisting of 60 successful and 60 failed trials, using a hand glove-like system (*Figure 4.8*). This approach enabled us to conduct the operation with human hands, providing a more comprehensive understanding of the task and ensuring that the model was not overly tailored to the robot's specific capabilities. This approach allowed us to establish a more generalizable model that can be applied to a broader range of robots and tasks.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

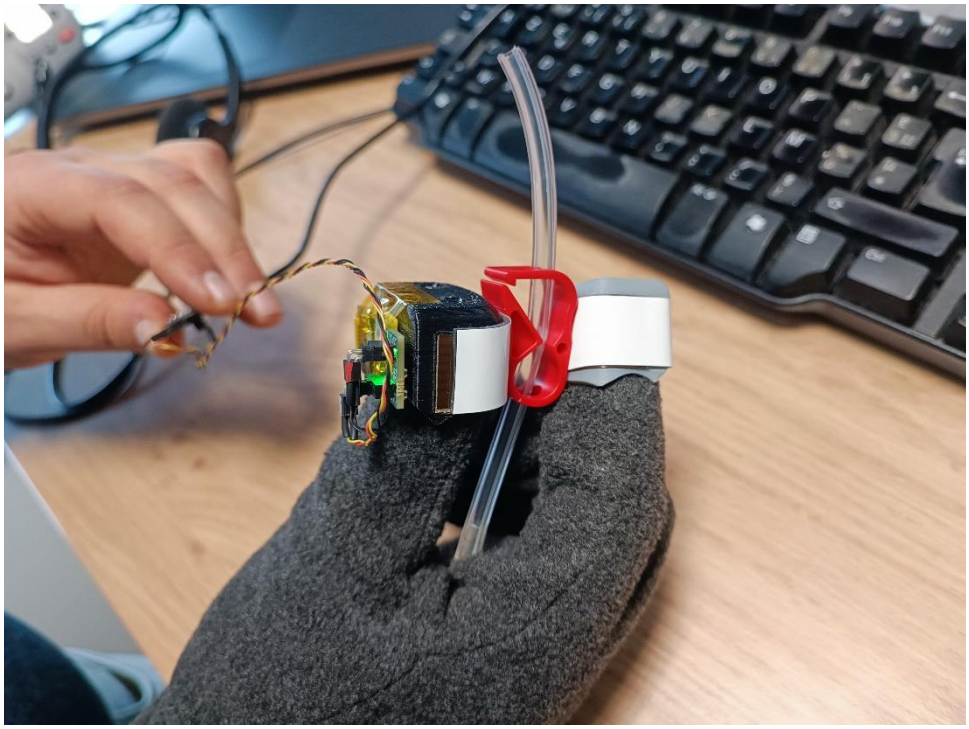

Figure 4.8: Clamp closure setup with Human hands allowing to use multiple degrees of freedom manipulation.

Subsequently, we augmented our database with recordings obtained using the Panda robotic arm gripper. Each experiment was meticulously annotated by the experimenter to ensure the accuracy and reliability of the data.

To enhance the realism of the data, we introduced artificial noise to the signals to simulate sensor noise and random spikes that were observed during the gripping operations. Those spikes are likely to be linked to the force control algorithm of the panda robot.

The noise (with a standard deviation of approximately 2.5 LSB) and spikes (with an amplitude of approximately 10 LSB) were directly measured on the robot through observation of the clamp's interaction with the phalanges while maintaining a steady grip force.

Furthermore, we observed a disparity in the representation of maximum values and we included a signal inversion to correct for this. The final dataset consisted of 37,368 samples, with some samples deliberately corrupted by random spikes to test the robustness of the model (*Table 4.1*).

Table 4.1: Number of samples for the training database before and after augmentation.

Horizon 2020

TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

Table 4.1: Number of samples for the training database before and after augmentation.

	Total number of samples	Success	Failure
Before augmentation	173	74	99
After augmentation	37368	15984	21384

A similar approach has been followed to constitute the validation (*Table 4.2*) and the test (*Table 4.3*) databases.

Table 4.2: Number of samples for the validation database before and after augmentation.

	Total number of samples	Success	Failure
Before augmentation	26	13	13
After augmentation	5616	2808	2808

Table 4.3

Table 4.3: Number of samples for the test database before and after augmentation.

	Total number of samples	Success	Failure
Before augmentation	39	20	19
After augmentation	8424	4320	4104

The classifier we opted to use is a tiny neural network that operates on the 40 samples window. This is a more robust method compared to simply using a threshold on the amplitude as the neural network learns the signature of the signal, and not only the minimum amplitude.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

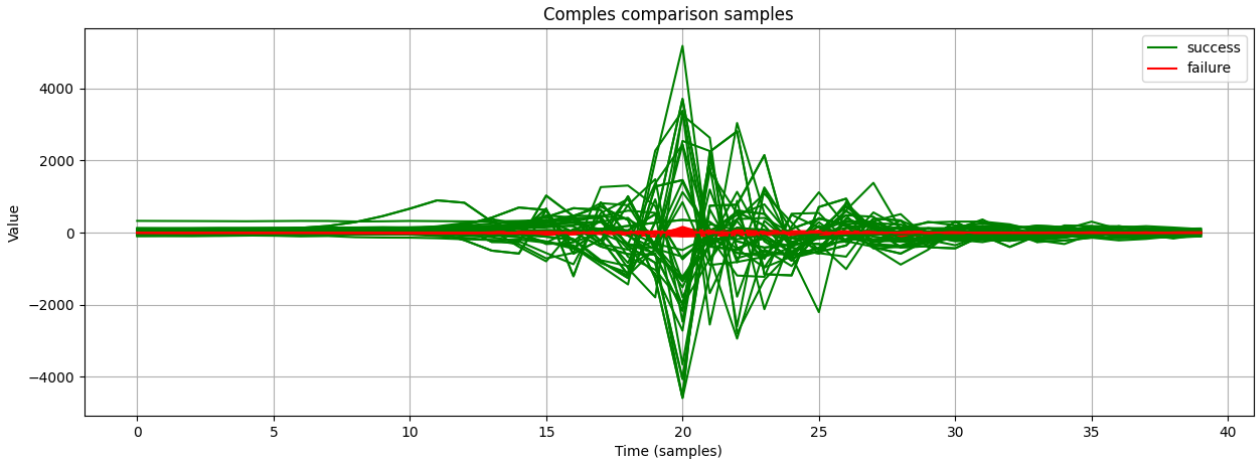

Figure 4.9: Comparison of raw piezo-electric signals (in LSB) from successful (green)and failed (red) clamp closure attempts.

It is clear that the success signals do have a typical pattern that varies a little bit in frequency according to tests (*Figure 4.9*). The neural network is an excellent tool to learn those variations with minimal tests. We used a single convolutional layer, followed by a global average pooling, a dense layer, and a final success/failure decision layer (*Figure 4.10*). This architecture allows the neural network to learn frequential information thanks to the kernels of the convolutional layer.

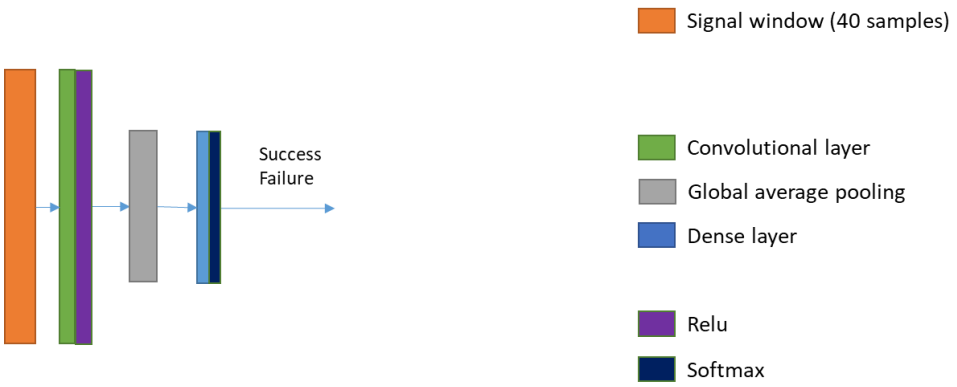

Figure 4.10: Architecture of the neural network.

The number of kernels as well as their size was optimized using a search method. The final neural network features only 128 parameters in total and since it has a shallow structure, it can run at very high speed.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

Table 4.4: Training, validation and test Results.

	Train	Validate	Test
Accuracy	100%	100%	100.0%
Total data size	37368	5616	8424

(Table 4.4) shows the obtained performances using our model on the train, validation and test databases as well as the number of augmented data size for each database. It clearly shows that the detection has been successful on all our databases.

In this work, we showed that it is possible to detect the click event using the piezo-electric sensor on the contact phalanx.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

4.5 Needle insertion detection

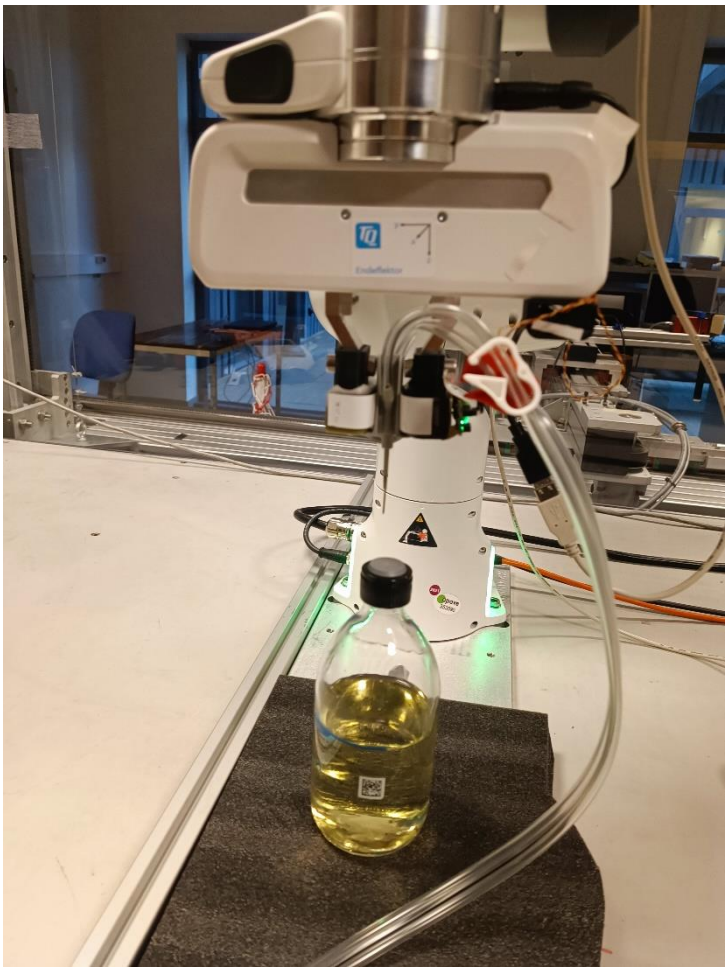

Figure 4.11: Needle insertion setup.

We have used the same setup presented in the previous chapter to train another model for detecting the success of the needle insertion in the bottle (task #4.2 in Table 4.1) using only the piezoelectric sensor. The main difference between the two events is the event signal duration as the needle insertion takes longer time than the clamp click.

To solve this difference point, we have added few steps to the operations described previously:

- We first use a large window around the max position of the signal amplitude. Here we use a window of 1 seconds (10000 points)
- We then lowpass filter the signal to only 100Hz
- We then decimate the signal by 50 to get a window of signal of only 200 samples

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

Once these operations are done, we get back to exactly the same setup than the one used to detect the click. The neural architecture had to be enhanced though because there are more variety in the success and failure signals (*Figure 4.12*) as there are many ways this can fail:

- One of the failures happens when the needle hits the border of the bottle resulting in the destruction of the needle. This failure results in a very high spike on the piezo electric signal.
- Another kind of failures happens if we completely miss the bottle, which results in no significant disturbance in the piezoelectric signal.
- The third kind of failures happens if the needle hits the rubber bottle cap without penetrating all the way.

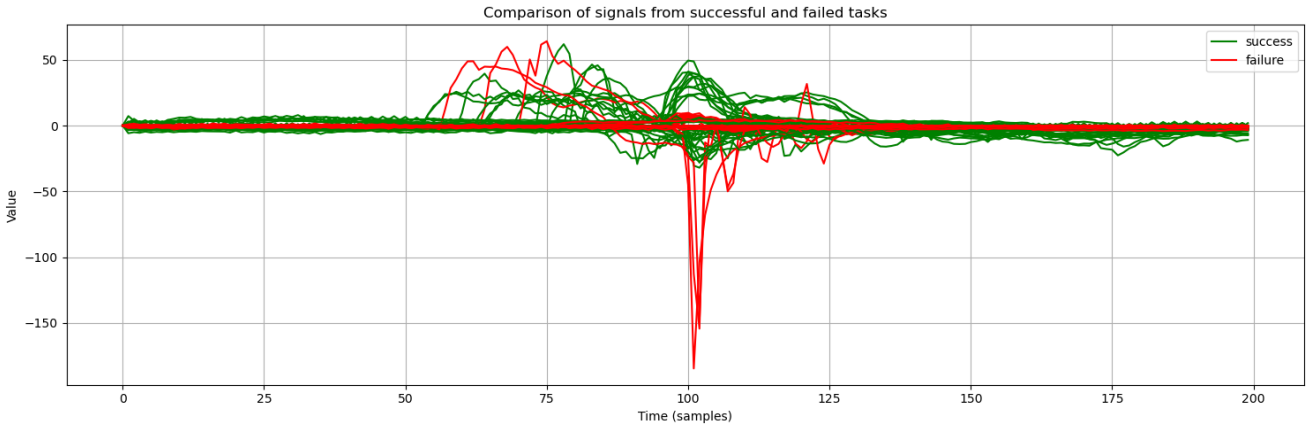

Figure 4.12: Unaugmented signals after performing all the pre-processing steps.

The signals are more diverse than in the first case and a single convolutional layer was not enough to get good results. This is why we went with a more complex architecture (*Figure 4.13*). The architecture features a succession of two layers of convolutions that discovers interesting additive features from the pre-processed signal. This part learns to build a reshaping of the original signal in order to perform the classification task. Once the signal is updated, we flatten it and apply two dense layers. The last layer has two outputs with a softmax activation which makes it capable of issuing a probability that the task succeeded or not. The residual part has many benefits that was already addressed for the case of images by the 2015 Microsoft paper about residual networks [9]. We just bring this concept to our case with timeseries using a rather shallow architecture. This helps the gradients flow smoothly during the training step. We have also added a regularization mechanism to all layers. This forces the training to obtain the simplest possible model that produces accurate solutions. This is important to avoid overcomplicated models which result in overfitting.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

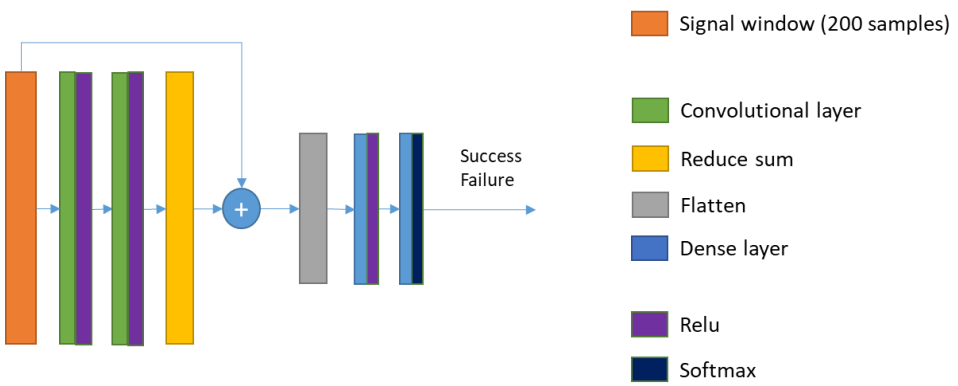

Figure 4.13: New architecture of the neural network for event detection.

There are three key elements to get a robust model:

1. The workflow parameters (optimized window size, filtering and decimation)
2. The neural network structure (the residual trick + l2 regularization)
3. The data augmentation (with realistic noise and artefacts)

The regularization mechanism, combined with the data augmentation, and the residual block architecture enhanced the ability of the network to generalize to never seen examples.

The performance on validation data increased drastically compared to using a classic deep convolutional neural network, a shallow convolutional neural network and even a deep dense neural network. Most of other architectures ended up to a high level of overfitting, leading to high performance on the training set but poor performance on the validation set.

Table 4.5: Number of samples for the train database before and after augmentation.

	Total number of samples	Success	Failure
Before augmentation	91	45	46
After augmentation	3276	1620	1656

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

Table 4.6: Number of samples for the validation database before and after augmentation.

	Total number of samples	Success	Failure
Before augmentation	18	8	10
After augmentation	648	288	360

The following table shows the train and validation results using the augmented database for train and validation data: with an accuracy of 100%, the results are quite satisfying.

Tableau 4.7: Results for needle insertion

Train	*Validation*
100%	100%

4.6 A generic model for piezo-electric based task verification

After building the needle insertion verification task model, we wanted to test our new model on the clamp closure task as the model we built for this task may be seen as a subcase of the workflow developed for the needle insertion (we use a smaller window, there is no filtering and decimation).

This resulted into a generic parametrizable version of this model that could be applied to multiple piezoelectric based tactile verification tasks. The parameters that can be learned for each task are:

- initial window size
- filtering bandwidth
- decimation level
- neural network weights

The results on the clamp closure verification task were as high as the ones obtained with the simple shallow neural network on all tests (train, validation, test).

Once the model is trained, we have tested it using the finger setup built by CEA-LIST in Paris. The tests used the intermediate phalanx to perform the clamp closure.

Since the final gripper is not yet ready, we were not able to perform this task using the entire gripper, but the results showed the capability of the model to generalize to a system it was not

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

trained on before. Here, both the coating and the architecture of the manipulator were different from the training setup described in subsection 4.4, and yet the model could distinguish successful and failed attempts with a relatively high accuracy level, as detailed hereafter.

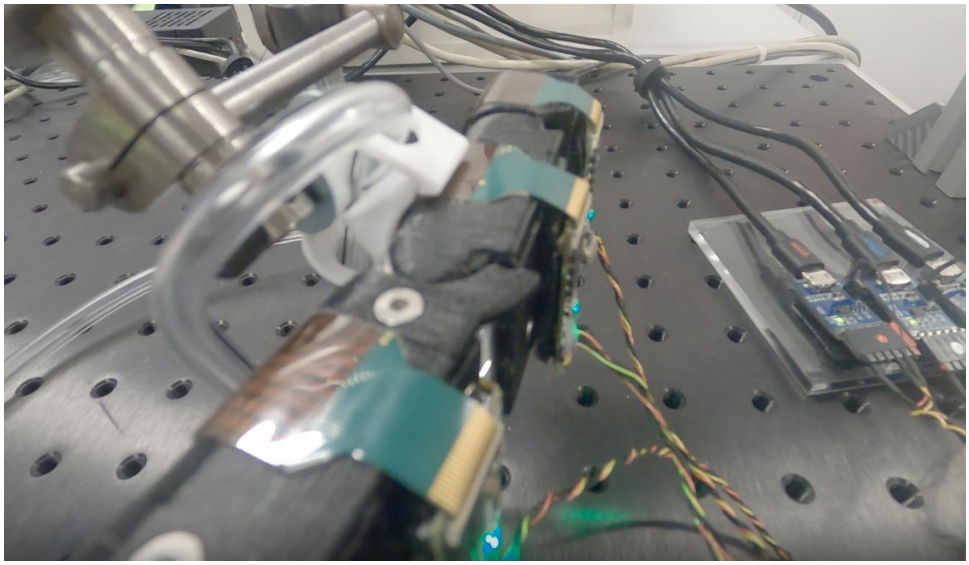

Figure 4.14: Clamp closure test setup using the first prototype of the Tracebot gripper Finger.

Twenty-nine actual closure tests have been conducted with the finger, including 13 success and 16 intended failures. As described in subsection 4.4, we have applied data augmentation to the test samples in order to have more meaningful results, increasing the samples number by 216 times (*Table 4.8*).

Table 4.8: Number of samples for the test database before and after augmentation.

	Total number of samples	Success	Failure
Before augmentation	29	13	16
After augmentation	6264	2808	3456

The tests showed a final result of 90% accuracy on the actual finger which is very encouraging, considering that the model has never seen data coming from the finger setup before.

This result will probably be greatly enhanced once the model finetuned with data from the actual gripper, which will take into consideration the new coating as well as the full parameters set (material, kinematics, the arm vibrations etc).

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

At this stage, the single finger setup cannot be used to test the needle insertion in the bottle yet. But tests will be carried out once the full gripper is ready and mounted onto the robotic arm.

4.7 Slip detection

We have also worked on slip detection and we have submitted a paper to the IEEE/ASME AIM 2023 conference[1]. Unlike the other models presented in this section, the slip detection algorithm (used for both WP4 and WP2) is being executed in real-time and does inline detection of the slippage during manipulation. The objective is to signal the system that the object is sliding as fast as possible so that the controller can take the necessary measures. That explains why we are using a completely different approach.

To detect the slippage, we have chosen to use the piezoelectric sensor and to exploit the frequential properties of the piezoelectric signal.

To perform this, we needed to build a well-controlled slippage detection database. We used our robot (Carousel) built in our lab to test our Tactipatch sensor. This robot allows us to perform various touch tests while providing synchronization of the events with the sensor signal and while providing a reference touch force measurement.

We have tested multiple configurations of slippage controlled by Carousel (*Figure 4.15*), leading to a database of 3200 tests, with:

- Still pressure (no slippage)
- Slippage with linear motion at various speeds
- No slippage, with external perturbations such as vibration
- Tests without slippage during robot motion (done with the panda robot)
- use of various contact tips to simulate different properties (shape, softness, size, material)

This allowed us to build a big dataset used to train our models, compare the performances of slip detection and measure the mean detection time compared to the reference.

[1] Spectro-temporal RNN structure for object slip detection using piezoelectric tactile sensor in robotic grasping

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

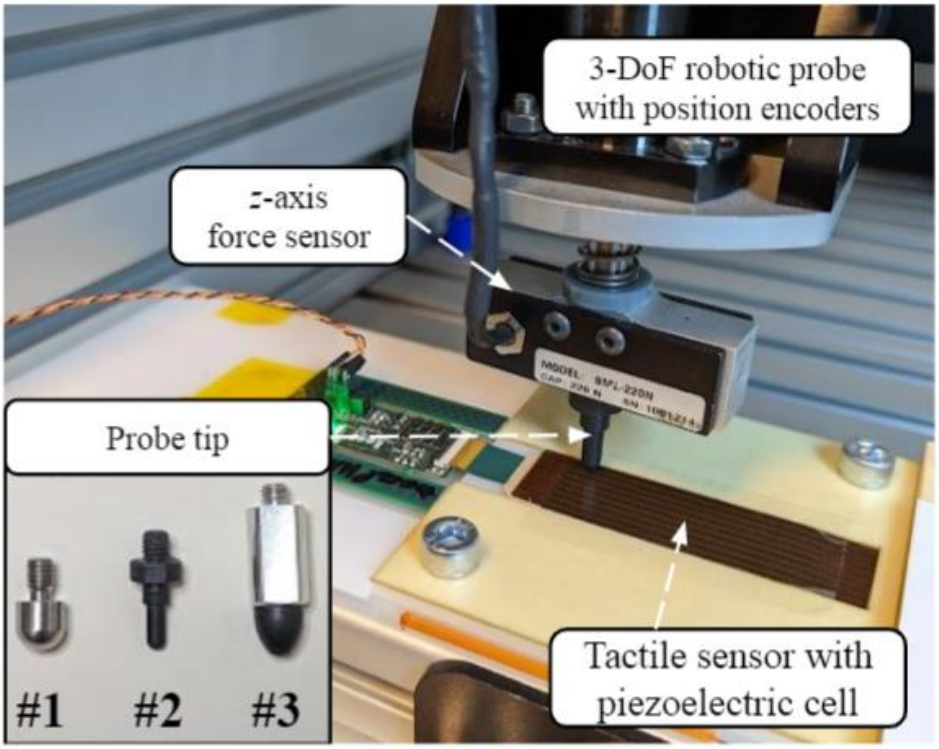

Figure 4.15: Slip detection training setup.

The algorithm used in this section can be split in two steps:

- Compute the spectrogram: To compute the Power Spectral Density (PSD) of a mean-subtracted piezoelectric signal, we use Fast Fourier Transforms (FFT) with temporal windows of 200 values (equivalent to 20ms at a sample rate of 10kHz). The spectrogram of the entire signal is created by concatenating the FFTs and forming a 2D array with frequency and time axes, providing information on the frequency content and evolution of the signal over time (as depicted in Figure 4.16). The frequency bins range from 0-5kHz with a resolution of 50Hz per bin.
- Use a GRU (Gated Recurrent Unit) based RNN: The resulting spectrogram is then fed to a Recurrent Neural Network (RNN), processing one FFT at a time, to detect temporal patterns. Classification is performed on each time-step, every 10ms. The RNN has 4 stacked Gated Recurrent Units (GRU).

The following figure presents the workflow that allows to detect the slippage by fusing information from past and present moments to predict the current slippage status. This architecture is particularly suitable for the detection of the slippage as it takes into consideration the sequence of events as well as the frequential information. We have

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

compared the use of RNN GRU vs MLP (Multi-Layer Perception) and TCN (Time Convolutional Network) and it outperformed both of them in multiple configurations.

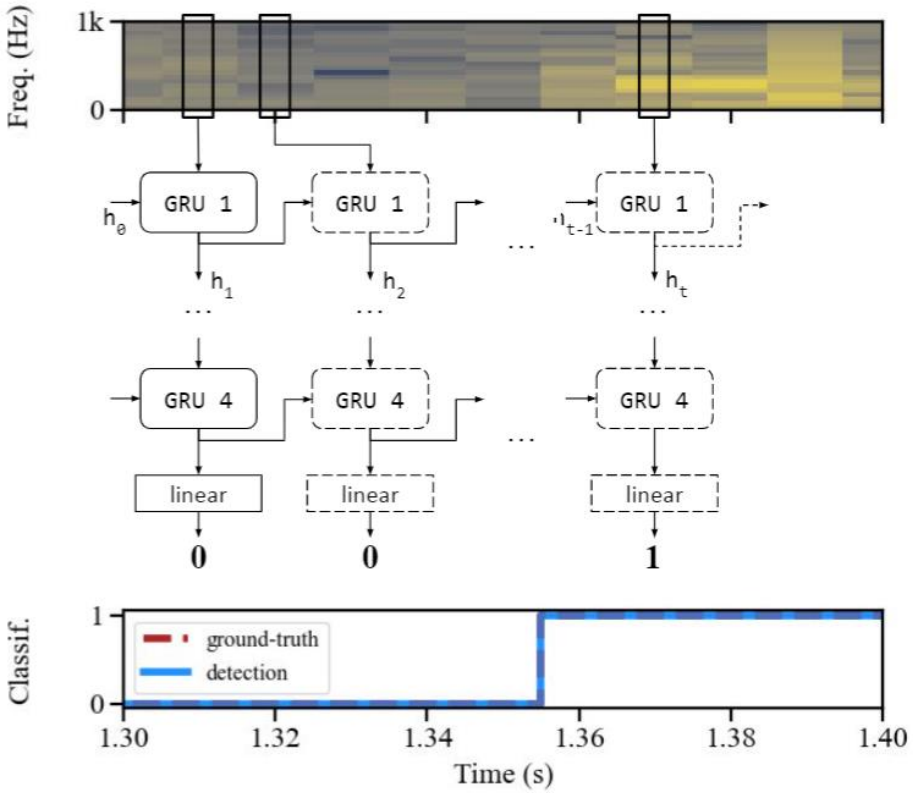

Figure 4.16: Slip detection training setup: the spectrogram (top), the recurrent neural network (middle) and the classification output (bottom).

The model resulted in an average accuracy of 98.6% over individual 10ms decision. On the 3200 tests, only 7 were incorrectly recognized in a window of 80ms around the starting of the slippage. For the rest of the samples, the average delay between the real starting of the slippage and the actual detection is of 8.5ms. We have noticed that for a delay of 20ms, in the fast motion setting, the head of the robot travelled 0.7mm (the head motion has an acceleration phase, a steady speed phase and a deceleration phase). This displacement in the use cases covered by this work can be tolerated as it can easily be recovered.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

4.8 Remarks

Although the outcomes of this research are remarkable, they have been obtained under different control conditions than those that will ultimately be implemented once the final gripper is fully operational. It is imperative to conduct additional testing to account for the variations in control strategy, mechanical components, and environmental conditions. This will ensure that the results can be accurately extrapolated to the final operational scenario.

4.9 Conclusion and perspectives

In this section we showed how it is possible to use the hybrid tactile sensor to do the following task verifications:

- Object presence in the gripper verification
- Object slip verification
- Clamp closure verification
- Needle insertion verification

We have developed multiple algorithms based on machine learning and analytic methods. The test results on the available systems are encouraging with success rates higher than 95%. A simple test for the clamp closure using the TraceBot finger resulted in more than 90% of accuracy which is encouraging given that the system used for training is completely different from the actual setup.

The future work would focus on integrating these modules in the global system via ROS interfaces and fine tuning the models on the final setup.

To ensure more robustness and to enable other verifications that may not be done using tactile sensing, visual verification will be discussed in the next section.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

5 Visual Task Verification

Visual task verification generally refers to checking whether at a certain location an object is detected and in a specific pose. This verification does not only use visual input from RGB-D sensors, but also physics simulation capability to check, whether the proposed visually observed pose of the object makes sense given the context such as supporting plane and other objects. We envision this type of visual verification to check whether objects are in the correct location and pose after a manipulation action, like inserting the canister in the tray or locating the needle before it is inserted into the septum.

5.1 Transparent Object Pose Verification

We propose to approach the problem of transparent object pose verification using differentiable rendering. This consist in the creation of a 3D representation of the scene of interest using 3D models, and the use of a differentiable renderer, that is a renderer whose every operation is implemented in a differentiable manner to obtain a rendered view of that scene that can then be compared to the captured view of the real scene. The benefit is that, given a suitable loss function definition, parameters of the 3D scene, like the pose of objects or their appearance, can be optimized directly in the image space, as gradients can flow through the differentiable renderer. The benefit of this kind of approach in the context of verification is that it can take advantage of all the existing knowledge about the scene through the scene modelling. In its simplest form, it also only requires an RGB image from the scene to perform the optimization, which makes it suitable to handle transparent objects that typically cannot be captured by depth sensors.

We therefore investigated the applicability of differentiable rendering to pose refinement and verification. Based on the initial estimate of the pose from the pose estimation module of WP3, the optimization process and its convergence inform us on the correctness of the original estimate and provide a better estimate of the pose as a side effect.

The challenge is that comparison between rendered and captured images cannot be performed directly in the RGB space, as it is very sensitive to local changes and mild differences in illumination that can be very hard to model. This creates a lot of sub-optimal local minimums, preventing a good optimization. We therefore need to select a more suitable space to compare the image and define the loss function that will drive the optimization procedure.

In a first step we evaluated the feasibility of using a detection mask to supervise the pose verification and refinement (see next Figure 5.1). Masks in this context are simple binary images that separate the object of interest from the background. Based on the currently estimated pose, it is trivial to obtain such a mask from the renderer, and we can compare it to the mask obtained in the real world from the object detector. This leads to good

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

convergence. It however relies on the accuracy of the object detector to perform the verification.

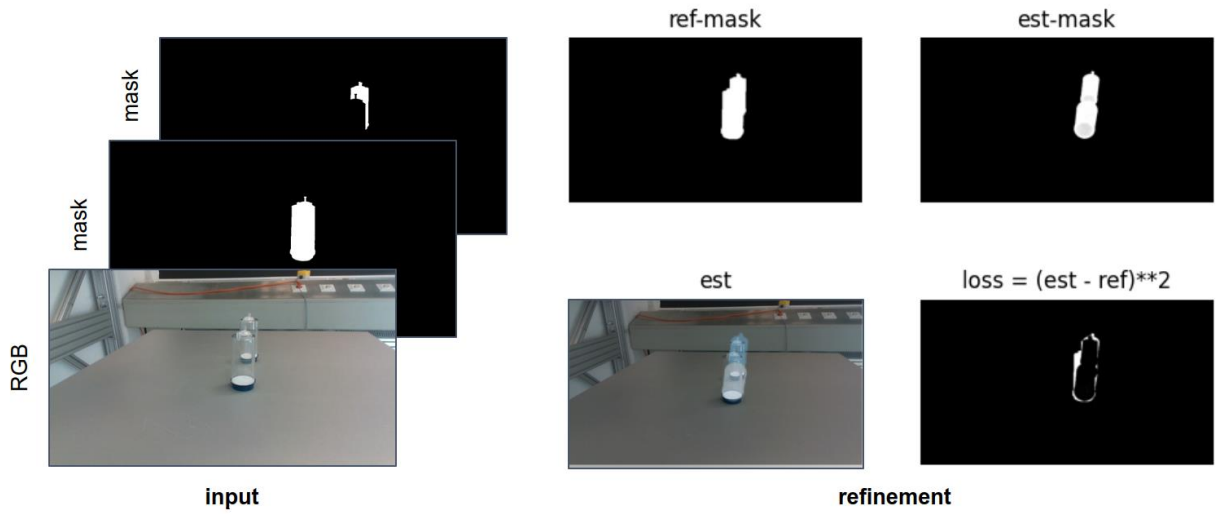

Figure 5.1:Illustration of the differentiable rendering optimisation process

This approach was then supplemented by a signed distance field-based loss to include physical constraints. Querying the signed distance field of the points of the object of interest compared to the neighboring objects, or the supporting plane lets us easily estimate whether the objects are intersecting, and express it in a way that is compatible with the representation of the scene and the optimization. This is illustrated in figure 5.2.

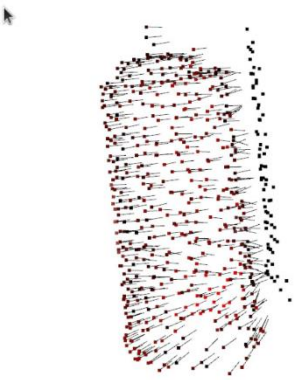

Figure 5.2: Illustration of the signed distance field-based loss. Red points are directed outwards, illustrating the repulsion between objects to avoid intersection.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

To further robustify the process, and make it less dependent on the object detector (and therefore not depend on its limitation when performing the verification step), we then investigated the use of an edge-based loss (see Figure 5.3). Edges can easily be obtained from a real image and a rendered image, are significantly less noisy than the RGB space itself, and transparent objects still provide meaningful information in that representation. They are therefore a good way to reduce the domain gap and make the optimization more robust. During the optimization, the edges of the real image are obtained once at the beginning, then the renderer computes only the contour of the object of interest, by computing the edge of the simulated depth map.

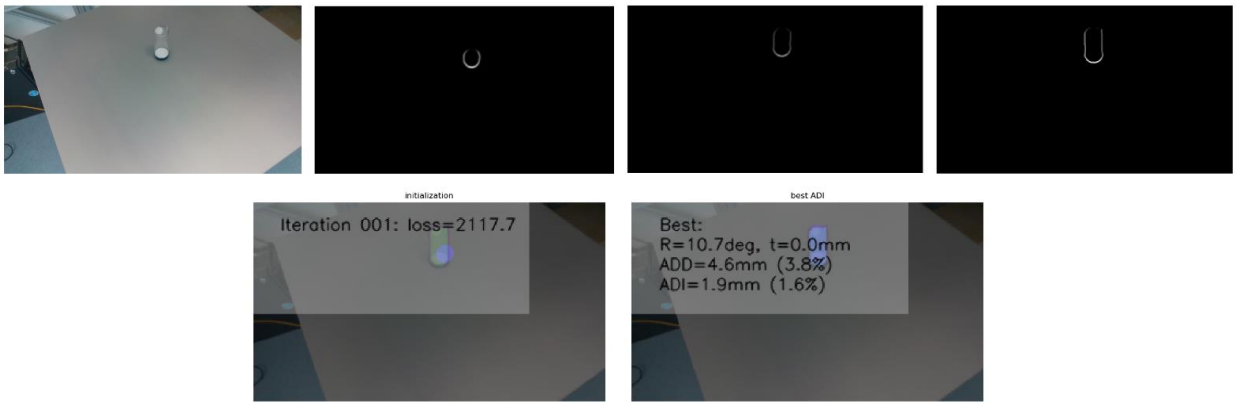

Figure 5.3: Illustration of the optimization steps based on edge maps

5.1.1 Next Steps: In-hand Pose Estimation

The two following subsections refer to verification means that are under investigation. We outline our initial results and intended plan to refine their implementation.

The verification process presented in the previous section takes advantage of the knowledge of the support plane to constrain the optimization. However, we need to expand upon this process to be able to address in-robot-hand object pose verification. The robot arm kinematics are known, giving us a good estimate of the end-effector pose. The relative pose of the object and end-effector used during the grasping procedure can also be used as an initial estimate for the object pose.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

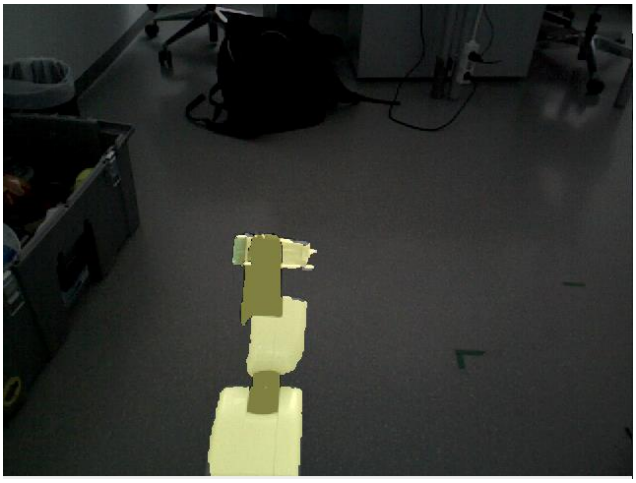

Figure 5.4: Segmentation of a robot arm and the canister from the background. Using the knowledge of the robot arm as well as the knowledge of the object enables us to optimize the in-hand object pose

The connection between the end-effector fingertips and the object is a physical constraint that we intend to model in a similar way as object intersection and the supporting plane and object interaction presented in the current system.

This scenario opens up further opportunities for verification as the ability of the robot to move enables us to perform multi-view optimization of the object pose. Under the assumption that the transformation between object and end-effector remains rigid during that procedure, moving the robot arm in front of the static camera is equivalent to moving the camera around the end-effector, which is naturally accounted for in our current setup, as cameras can be moved around the 3D scene setup that is being optimized.

5.1.2 Next Steps: Liquid Fill Level Estimation

Once the pose of a transparent container is estimated, we can refine additional parameters about the object using suitable scene representations. One such example is the fill level of that container (see Figure 5.5).

Horizon 2020

TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

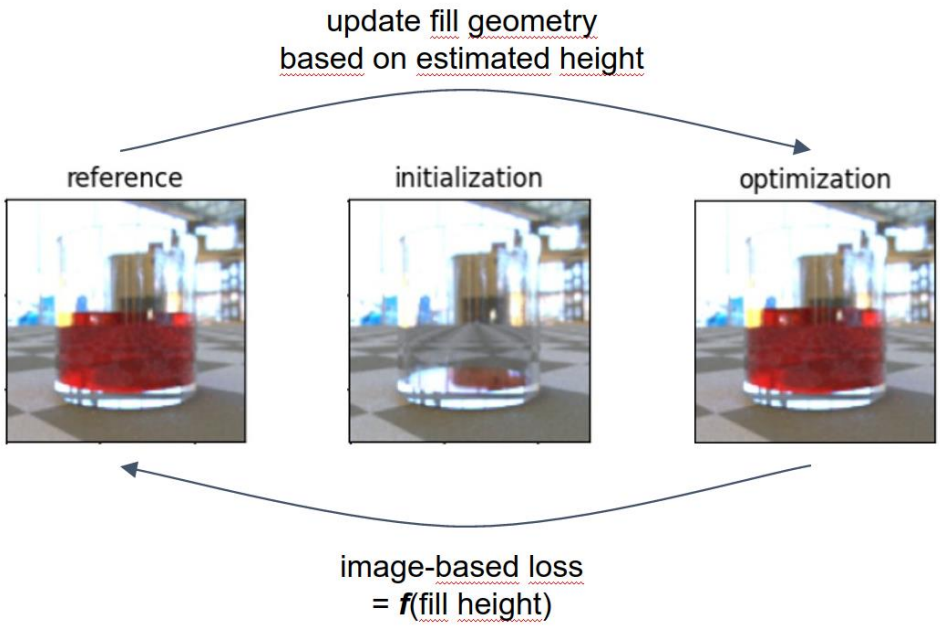

Figure 5.5: Optimization process for fill level estimation

As a preliminary step, we propose to approach this problem in a static situation by assuming a known container (including its index of refraction), a known liquid and a known gravity vector. As the light path is only modified at interfaces, this problem can then be reduced to a height estimation problem. For a given container orientation, the surface of the liquid will be orthogonal to the gravity vector, and only its height within the container needs to be estimated.

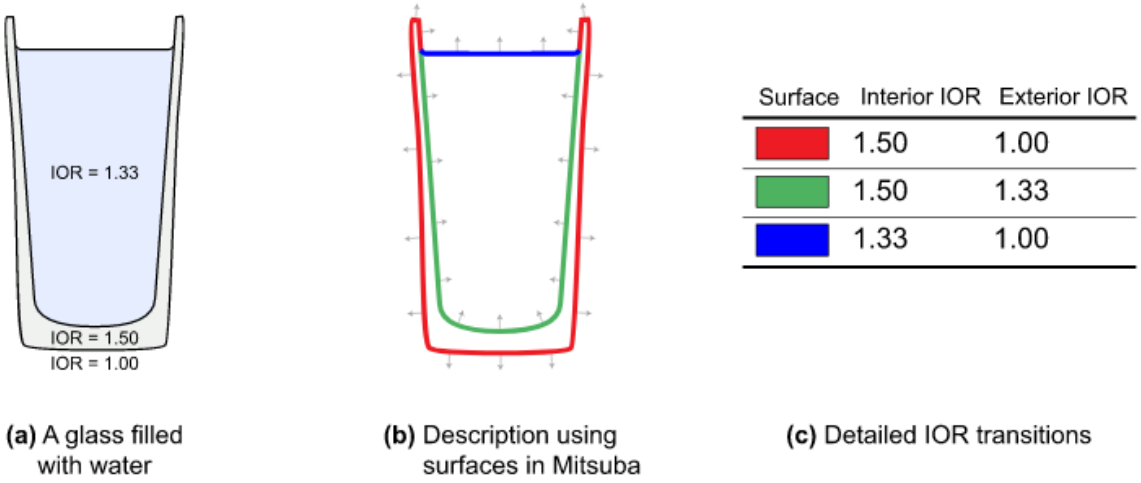

Figure 5.6: Illustration of the modeling of liquid and container

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

The corresponding model is illustrated above. As only object surfaces are modelled for rendering pipelines, and the respective orientation of the blue surface to the red and green surface is known.

While our preliminary experiments assumed a known liquid, all parameters could in principle be optimized. In the illustration below, we demonstrate the feasibility of the index of refraction and color liquid estimation.

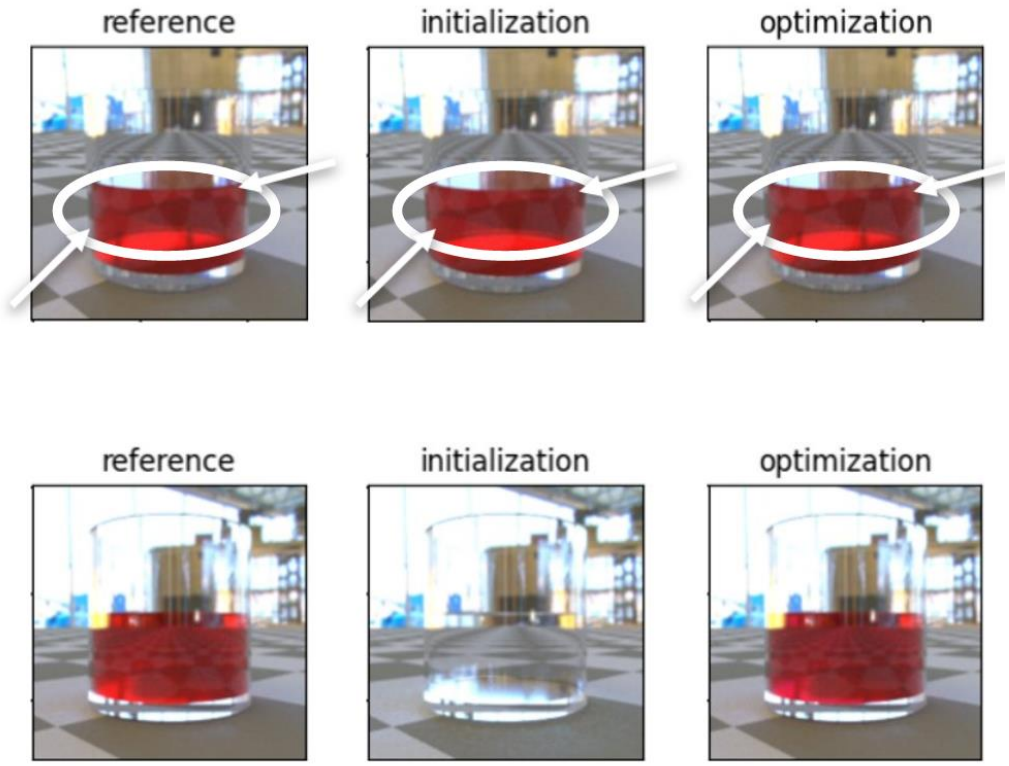

Figure 5.7: Examples of additional parameters that can be optimized with our differentiable rendering process. Top: index of refraction, bottom: color

While the proof of concept supports this research direction, further work is needed, and in particular, evaluate the reliability of such an approach with real data, as all experiments are done by using a rendered reference image instead of a reference image captured. We intend to approach this in a similar way as the pose estimation, relying on edge estimation to drive the estimation process. Relying on the ability of the robot to act on the scene will also be necessary as this enables the capture of the background scene, to be used as an environment in the rendering process. The following figure illustrates a potential pipeline to handle the liquid fill level estimation.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

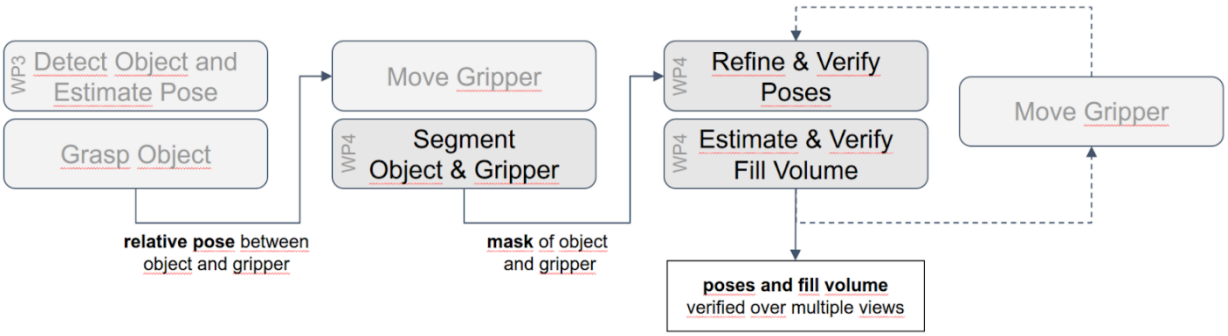

Figure 5.8: Pipeline of the Verification and fill level estimation process

5.2 Geometrical Verification through Multi-view scene collection

As an alternative approach to verification in the context of the TraceBot project, we also investigate state-of-the-art reconstruction approaches. Contrary to the work presented up until now, this approach requires capturing multiple views of the scene. This can be achieved using an in-hand camera, and obtain the camera pose using the arm kinematics. A neural radiance field (NeRF) can be trained using that approach to recover the geometry of the scene, including the shape of transparent objects, therefore overcoming the limitations of depth sensors.

Knowing the geometry of the scene lets us use depth/normals-based verification methods like the ones presented in deliverable 4.1.

Contrary to single-view depth prediction methods, NeRF is a transductive learning method, meaning it is trained on the same set of data that it is evaluated for. This makes it impossible for the network to encounter out-of-distribution samples. This makes the network behavior more predictable, which is an important property in the context of verification, but comes at a high computational cost, and the necessity to capture multiple views.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

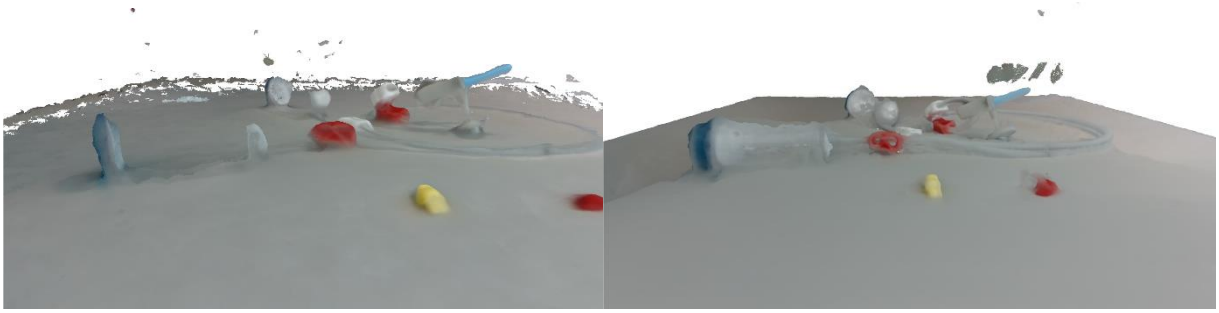

Figure 5.9: Reconstruction of the sterility kit. Left: depth sensor. Right: NeRF. Transparent objects geometry is only recovered using NeRF

In particular, we investigated the ability of NeRF methods to produce depth points for the canister in 4 scenes, illustrated below.

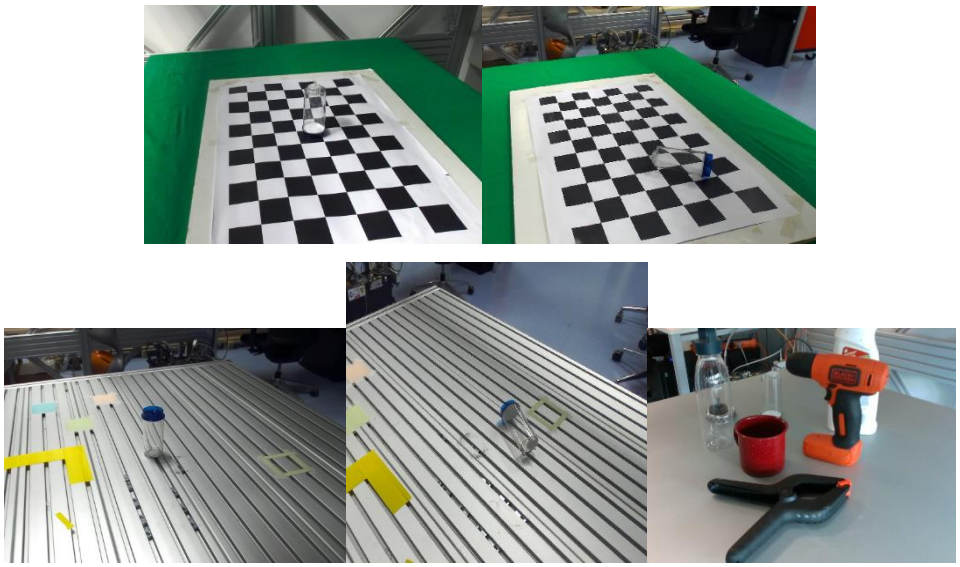

Figure 5.10: The five scenes used in the experiments (Top: Can1, Can2. Bottom: Can3, Can4, Cluttered)

For each scene, we defined an error threshold of 10 mm (max. per-pixel depth error). All pixels with an error higher than the threshold are considered to be outliers. We then evaluate the mean depth error over every inlier. NeRF models the scene occupancy rather than the geometry directly. As such, we need to threshold that occupancy to obtain a final depth prediction. The influence of such a threshold is evaluated. The results are reported in the following table.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

Table 5.1: Mean depth error (mm) / outliers (%) over all views (lowest outlier percentage in red).

Scene	Can1	Can2	Can3	Can4	Cluttered
σ = 3	2.61 / 15%	**1.67 / 8.06%**	4.65 / 38.04%	3.37 / 16.32%	2.58 / 16%
σ = 6	2.5 / 13.3%	1.54 / 8.5%	2.93 / 27.2%	2.41 / 13.04%	2.14 / 12.59%
σ = 7	2.48 / 12.87%	1.52 / 8.82%	2.73 / 25.95%	**2.24 / 12.91%**	**2.04 / 12.33%**
σ = 8	2.46 / 12.57%	1.51 / 9.2%	**2.63 / 25.58%**	2.13 / 13.2%	1.97 / 12.47%
σ = 9	2.45 / 12.4%	1.5 / 9.62%	2.58 / 26.1%	2.06 / 13.94%	1.91 / 12.91%
σ = 10	**2.45 / 12.35%**	1.5 / 10.07%	2.56 / 27.32%	2.03 / 15.1%	1.87 / 13.46%
σ = 11	2.45 / 12.47%	1.5 / 10.56%	2.56 / 29.15%	2.04 / 16.69%	1.84 / 14.1%
σ = 12	2.45 / 12.78%	1.5 / 11.09%	2.56 / 31.43%	2.06 / 18.71%	1.82 / 14.8%
σ = 13	2.45 / 13.32%	1.5 / 11.7%	2.57 / 34.01%	2.09 / 21.14%	1.81 / 15.51%
σ = 14	2.46 / 14.11%	1.5 / 12.39%	2.57 / 36.69%	2.14 / 24.02%	1.8 / 16.24%
σ = 15	2.46 / 15.14%	1.5 / 13.17%	2.57 / 39.42%	2.17 / 27.52%	1.8 / 16.93%
σ = 20	2.49 / 23.54%	1.57 / 18.78%	2.57 / 51.45%	2.19 / 43.5%	1.85 / 19.85%
σ = 25	2.49 / 33.85%	1.7 / 26.44%	2.54 / 59.74%	2.24 / 52.67%	1.93 / 22.11%
σ = 30	2.48 / 42.99%	1.82 / 34.99%	2.53 / 65.35%	2.3 / 58.9%	2.02 / 23.9%

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

We also performed the same evaluation when training the NeRF volume with an additional distortion loss, expected to limit the presence of "floaters", that is area in the volume that do not correspond to any real geometry but are created during the training of the field to model view-dependent effects.

Table 5.2: Mean depth error (mm) / outliers (%) over all views (lowest outlier percentage in red).

Scene	Can1	Can2	Can3	Can4	Cluttered
σ = 6	2.7 / 7.7%	1.9 / 12.3%	3.1 / 25.2%	2.68 / 18.7%	1.99 / 15.77%
σ = 7	2.65 / 7.4%	1.86 / 12.4%	3 / 24.95%	2.58 / 18.59%	1.94 / 15.7%
σ = 8	2.59 / 7.2%	1.8 / 12.56%	2.9 / 24.7%	2.5 / 18.62%	1.91 / 15.67%
σ = 9	2.53 / 7.2%	1.75 / 12.7%	2.8 / 24.5%	2.43 / 18.78%	1.88 / 15.67%
σ = 10	**2.48 / 7.2%**	**1.71 / 12.95%**	2.74 / 24.4%	**2.38 / 19%**	**1.85 / 15.68%**
σ = 11	2.44 / 7.25%	1.68 / 13.1%	2.67 / 24.35%	2.33 / 19.27%	1.84 / 15.7%
σ = 12	2.41 / 7.36%	1.66 / 13.4%	**2.6 / 24.34%**	2.29 / 19.58%	1.82 / 15.78%
σ = 13	2.39 / 7.54%	1.6 / 133.17.67%	2.54 / 24.38%	2.26 / 19.96%	1.8 / 15.88%
σ = 14	2.37 / 7.78%	1.62 / 13.92%	2.49 / 24.48%	2.23 / 20.36%	1.79 / 16%
σ = 15	2.36 / 8%	1.61 / 14.2%	2.45 / 24.64%	2.21 / 20.78%	1.78 / 16.16%

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

σ = 20	2.35 / 10.4%	1.59 / 15.7%	2.3 / 25.85%	2.15 / 23.47%	1.77 / 17.14%
σ = 25	2.38 / 14.23%	1.62 / 17.57%	2.25 / 27.6%	2.13 / 26.75%	1.78 / 18.32%
σ = 30	2.4 / 19%	1.68 / 20%	2.25 / 30%	2.13 / 30.19%	1.8 / 19.43%

These preliminary results are promising. Further improvements are needed, in particular because the absence of texture significantly degrades the performance of neural radiance fields, and will be investigated in the next year. The more accurate the depth prediction for objects in the scene, the better a depth-based verification will be.

5.3 Preliminary Tube Detection and Modelling

As a preliminary experiment, we evaluated the problem of tube detection and modelling. As a first step, we investigated tube detection using classical methods with limited success, as the tube used in the sterility testing kit is also transparent, limiting the applicability of such methods. We are currently investigating the use of semantic dense correspondences that can be obtained using a vision transformer trained in a self-supervised manner.

As illustrated below, when looking at the potential correspondences between a given point of the tube from a source image in a target image, the heatmap of the distance gives promising results as it highlights the point of the tube in the target image.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

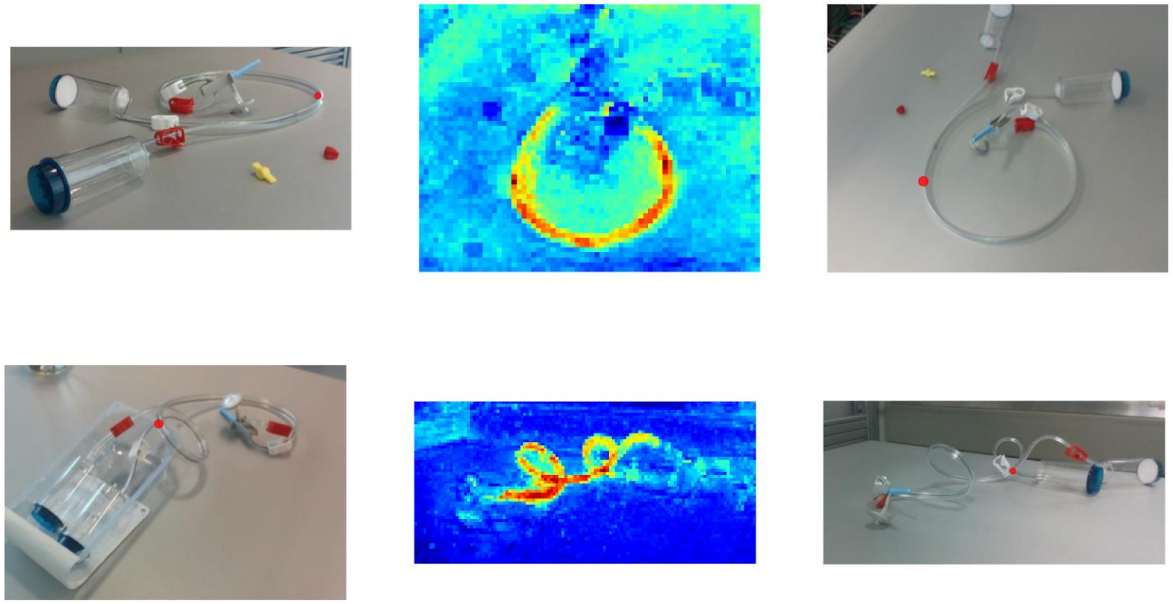

Figure 5.11: Example of semantic correspondences obtained with DINO-ViT for the tube of the sterility kit

Once a rough segmentation has been obtained, it is then possible to use a differentiable renderer to optimize the pose of the tube, growing it based on its known profile along the point of the pre-segmentation, constraining it to follow a spline. In particular, the clamps, the needle and the canister itself constitute important key points in this context, as we need to ensure that the tube prediction connects those points. Previous steps of the vision pipeline should provide us with a knowledge of the 3D position of such objects in the scene, giving us a 3D starting and end point that need to be connected with a spline based on the rough segmentation.

Preliminary experiments using markers (see Figure 5.12) to optimize such splines held promising results.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

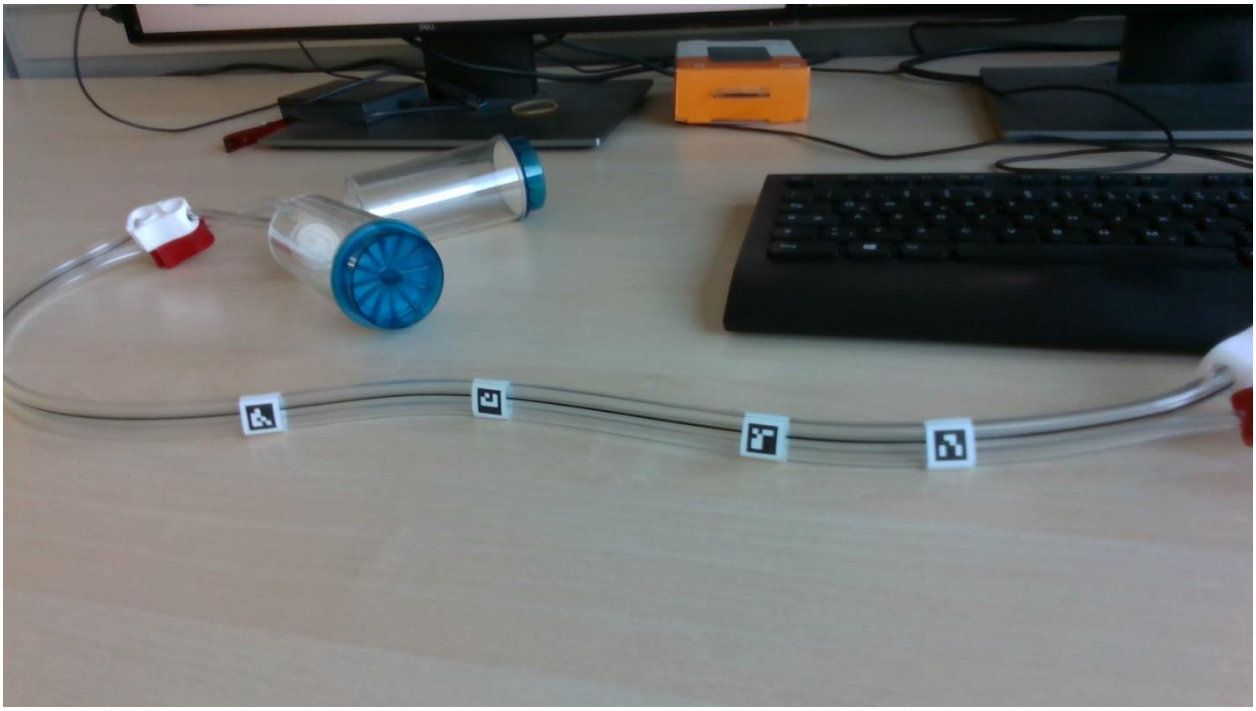

Figure 5.12: Example of the tube with markers attached to it.

5.4 Conclusion and perspectives

We have presented a set of vision methods based on the concept of differentiable rendering to verify the information inferred from the scene during the process execution. In particular, we have outlined a way to perform a verification of object poses suitable for transparent objects or deformable object depending on the exact formulation. We have also presented detailed steps to adapt this process for fill level estimation. Differentiable rendering enables the incorporation of all existing knowledge into the scene representation and compares it directly to the output of the scene's sensor, or to very simple transformations of it (like extracting the edges from the raw output image). As such it is well-suited to verification because it does not rely on the representation used to infer the object poses, but a low-level information closer to the sensors.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

6 Functional Task Verification

In the previous sections, you have certainly heard about several verification modalities (i.e., visual, tactile), still a higher-level verification modality should ultimately decide whether the actions of the robot were successful, whether it is or was even correct to perform an action within a context and whether the equipment are working properly. We term this as functional verification and logically reduce it to making the robot understand its actions and the desired effects. In this section, we present the initial work in this regard in terms of architecture, implementation and proof concepts.

6.1 Overview

Functional verification is a higher-level verification modality that should ultimately decide whether the actions of the robot were successful, whether it is or was even correct to perform an action within a context and whether the equipment are working properly. As you can see on the picture below, we digest the detailed description and specification of the TraceBot use case elaborated in the consortium into a formal ontology that establishes the fundamental truths about objects and tasks in the world (see D5.1-2 from WP5 for more details). Then, we enrich the actual state of the world also known as context with information coming from different verification sources, and combine this context with the world ontology to answer the three questions below: action feasibility, action success and equipment functioning. In this stage, the verification consists essentially in comparing the actual state of the world (context) with the expected preconditions and post-conditions of the targeted action. In the case of equipment testing (e.g., holding capacity of the drain tray), the process is sometimes augmented with the so-called verification-specific actions (e.g., pushing the canister a little bit after insertion), which are illustrated in Table 3.1. This being said, we therefore conceptually reduce the problem of functional verification to understanding the preconditions and postconditions of actions and deciding given a context if they are feasible or were successful. Another ongoing application of this module is the automated context-aware online planning of complex actions from primitive actions which is actually conducted in WP3.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

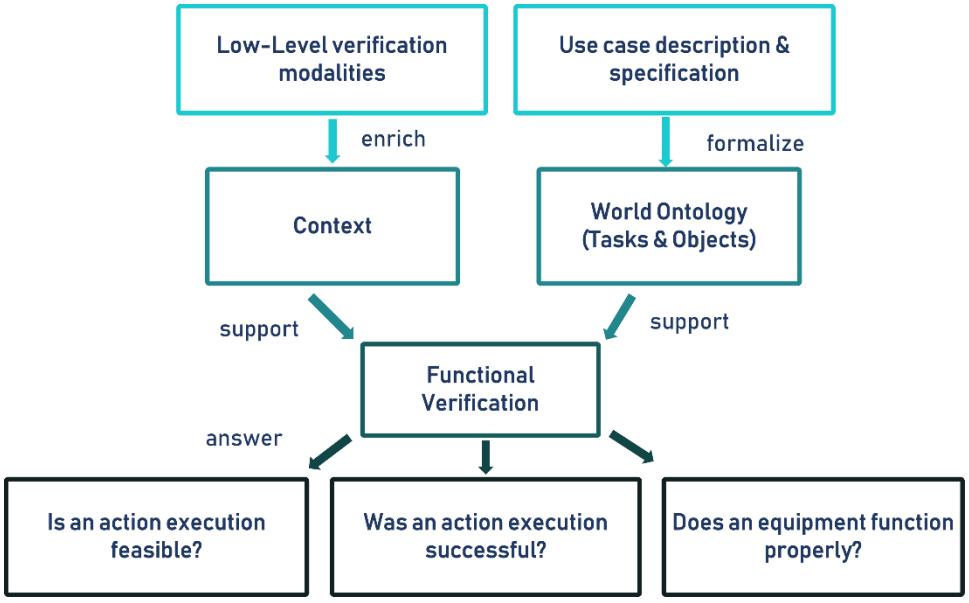

Figure 6.1: An Overview of functional verification schema

Up to this milestone, we formalized the problem of functional verification, and provided the fundamental software and data models (i.e., part of the Traceable Semantic Twin, TST, developments in WP5) for solving the problem. Furthermore, a query language (see D5.2), encapsulated into ROS actions, was also provided to the rest of the TraceBot's robotic system as an interface for submitting functional verification tasks to the reasoning system (i.e., part of the TST). Finally, the feasibility and the success verification of the grasp-canister action was illustrated.

6.2 Knowledge Representation

In this section, we present the knowledge representation that we achieved for the sake of performing functional verification, which is essentially made up of symbolic knowledge also known as world ontology about the agents, objects, actions and states as well as the grounding of the corresponding symbols into a subsymbolic representation of the environment for a better reasoning about robot actions.

6.2.1 World Ontology: Representation of Objects, Actions, States

We firstly digested the detailed description and specification of the TraceBot use case elaborated in the consortium [6] into a formal ontology (extension of SOMA [7]- Socio-physical models of activities) that establishes the fundamental truths about objects in the world. Figure 6.2 shows how objects are described in the ontology in OWL (Ontology Web

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

Language). The figure illustrates the abstract properties of a canister (i.e., color, shape, material) as well as concrete properties (i.e., size).

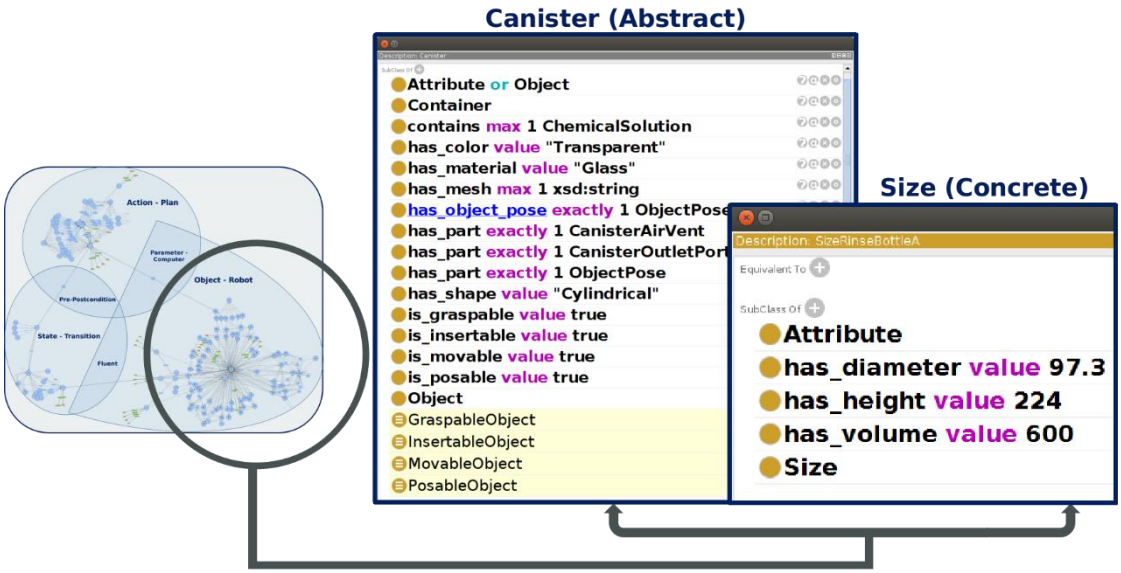

Figure 6.2: Detailed representation of objects in the ontology

Then, we represented actions in the ontology in terms of plans and parameters such as shown by Figure 6.3. Note that such description is not only fundamental to reason about the preconditions and postconditions of actions in terms of states but also help to diagnose detected failures of long-term actions (e.g., It was an intermediate action skipped that causes failure). Figure 6.3 illustrates for instance the representation of a FitInsertableToInsertee action, which is a generic action that consists in inserting a so-called insertable object (e.g., canister) into another so-called insertee object (e.g., drain tray). Such a schema can then be used for any other action with similar plan schema. The red boxes show the intermediates actions that make up the plan and their positions in the plan whereas the green boxes show how the parameters are passed from the calling action to the intermediate actions.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

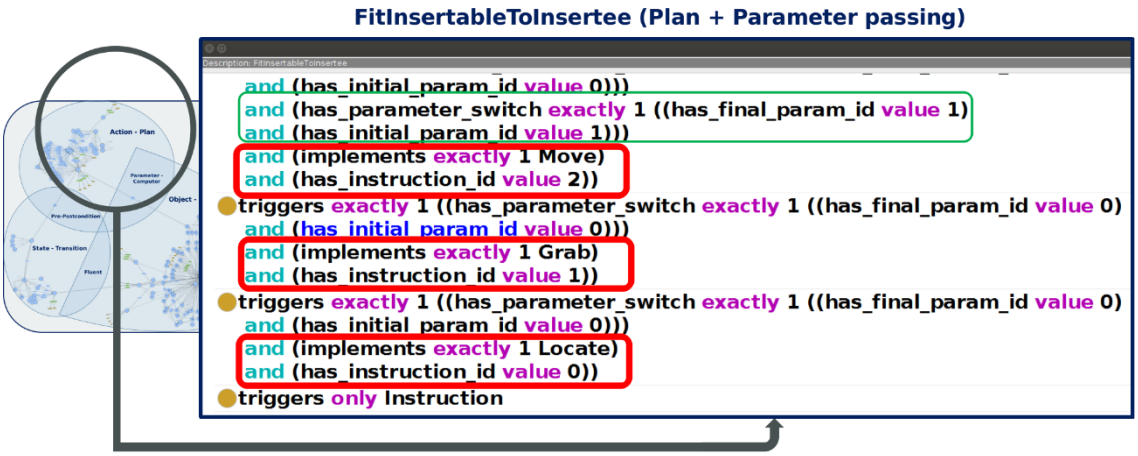

Figure 6.3: Description of actions in the ontology

Very important regarding such an ontology with respect to functional verification was the formalization of action pre- and post-conditions, which represent the necessary conditions and effects of a given action. In order to formalize the pre- and postconditions of actions, we introduced the concept of world state, which is merely a set of fluents, where a fluent is a temporal predicate about the state of scene entities. For instance, the fluent Grasped(Canister_Id, t) is true iff the canister Canister_Id was grasped by the robot at time t. Fluents differ therefore from other object properties (e.g., color, category, ...) in the sense that they are dynamic (e.g., broken, pose, appearance, ...). And whereas the precondition of an action is a state of the world in which the action is feasible, the post-condition is a state that reflects the effects of the action. This being said, Figure 6.4 illustrates the modeling of action pre- and post-conditions in the ontology using SWRL (Semantic Web Rule Language) on top of OWL for expressiveness reasons.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

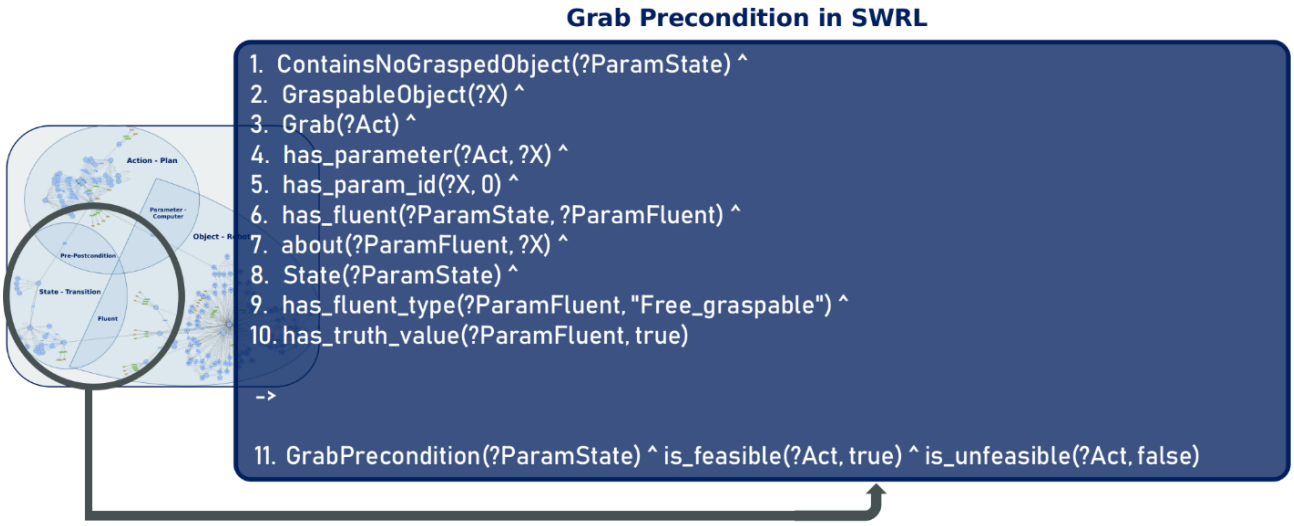

Figure 6.4: Description of action pre- and post-conditions in the ontology

This grasp/grab precondition definition states that (1) and (8) if ?ParamState is a state containing no grasped objects (expressible with OWL), (2) ?X is a graspable object, (3) ?Act is a grab action, (4) ?X is a parameter of ?Act and (5) ?X is the first parameter, (6) ?ParamFluent is a fluent (i.e., time-variable property) in ?ParamState, (7) ?ParamFluent is about ?X, (9) ?ParamFluent defines the property Free_graspable (i.e., true if the concerned object is not actually grasped), and (10) ?ParamFluent has the value true, then (11) ?ParamState is a precondition of ?Act and ?Act is therefore feasible.

6.2.2 Extending World Ontology with Subsymbolic Representations

Beyond purely symbolic reasoning for performing functional verification are physical and imagistic reasoning which are simulation-based reasoning techniques capable of escaping the complexity (e.g., combinatorial explosion of rules and reasoning paths) and limitations (e.g., lack of granularity) of rule-based reasoning while incorporating aspects of human commonsense (I.e., physics, imagination, causality) in a much more effective and efficient manner. For this reason, we also advanced the extension of the symbolic knowledge representation of the subsymbolic representation of the world, also known as DT (Digital Twin), which targets a physico-realistic replication of the real world. As you can read more about this in D5.2, the DT is able to emulate the real robotic processes for the scenario of canister insertion (MS1) and needle insertion (MS2). These advancements are illustrated by Figure 6.5.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

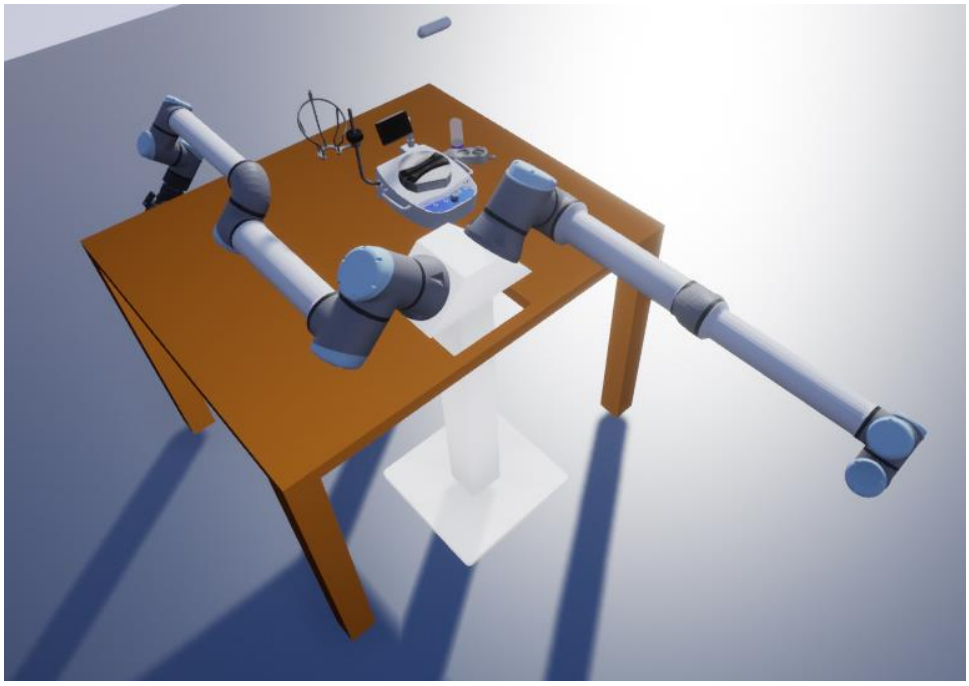

Figure 6.5: Extending world ontology with digital twin of the world

6.3 Process Context: NEEMs as grounded activity traces

Another core input for functional verification is the process context, which describes what is going on in the process (e.g., which object is where, what happens, which objects are involved, how does the scene looks like, sounds like,...). Notice that such data have been described as NEEMs (Narrative-Enabled Episodic Memories) in this project and emanate themselves from the grounding in the ontology by WP5 of robot activity traces collected by the WP4-tracer. Regarding this collection of NEEMs, we undertook two fundamental actions namely the grounding of skills from WP3 into the ontology and the implementation of a query language (see next sections but deeper in D5.2) to generate and record the NEEMs from robot activity traces. The tables below illustrate the grounding of skills into ontology. This grounding will allow not only to understand the actions (i.e. ask about/Verification) that the robot is performing but also record what the robot is doing (i.e., tell about/NEEMs/Context).

Table 6.1: Initial sequence of actions to execute from the skill engine to insert canister into drain tray

-MoveHome	Destination/Trajectory as Parameter
-Locate	Canister as Parameter
-Verify	Canister(Id) as Parameter
-Grab	Canister(Id) as Parameter
-Lookup	Grasp Canister Info (Canister (Id) as Parameter
-Compute	Grasp Trajectory (Task+Effectors + Pose as Parameter)
-Configure	Grasp (Task + Grasp Type as Parameter)

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

-Execute	Grasp Trajectory (Effectors + Trajectory as Parameter)
-Lift	Object (Trajectory as Parameter)
-Compute	lift trajectory (Task, Effectors, Canister(Id), Destination as Parameter)
-Lookup	Insertion Canister Pose (Object Type, Target action, as Parameter)
-Execute	Trajectory to move canister to tray (Canister Id, Destination as Param)
-Move towards finished	
-Insertion	
-Release	
-Retract	

After grounding the skills in the ontology, the grounded sequence of skills is shown in Table 6.2.

Table 6.2: Grounding of the action sequence in Table 6.1. for inserting a canister into the drain tray

- http://www.ease-crc.org/ont/SOMA.owl#ParkingArms
- http://www.semanticweb.org/smile/ontologies/2022/2/TraceBot#TraceablePinkingUp (Canister)
- http://www.ease-crc.org/ont/SOMA.owl#SettingGripper
- http://www.semanticweb.org/smile/ontologies/2022/2/TraceBot#TraceableLocating (Canister)
- http://www.semanticweb.org/smile/ontologies/2022/2/TraceBot#PlanningGrasp
- http://www.ease-crc.org/ont/SOMA.owl#Reaching (Approach Pose)
- http://www.ease-crc.org/ont/SOMA.owl#Reaching (Grasp Pose)
- http://www.ease-crc.org/ont/SOMA.owl#Grasping (Canister)
- http://www.ease-crc.org/ont/SOMA.owl#Lifting (Approach Pose, Canister)
- http://www.semanticweb.org/smile/ontologies/2022/2/TraceBot#TraceableInserting (Canister, DrainTray)
- http://www.semanticweb.org/smile/ontologies/2022/2/TraceBot#TraceableLocating (DrainTray)
- http://www.semanticweb.org/smile/ontologies/2022/2/TraceBot#PlanningInsertion
- http://www.semanticweb.org/smile/ontologies/2022/2/TraceBot#PlanningRetraction
- http://www.ease-crc.org/ont/SOMA.owl#Delivering (Canister, DrainTray)
- http://www.ease-crc.org/ont/SOMA.owl#Inserting (tilted) (Canister, DrainTray)
- http://www.ease-crc.org/ont/SOMA.owl#Orienting (Canister)
- http://www.ease-crc.org/ont/SOMA.owl#Releasing (Canister)
- http://www.ease-crc.org/ont/SOMA.owl#Retracting
- http://www.ease-crc.org/ont/SOMA.owl#ParkingArms

6.4 Interfaces as Formal Query Language

In order to generate and record NEEMs in the knowledge base as well as sending functional verification tasks to the reasoning system, a formal query language has been implemented and presented in D5.2. Basically, queries in this language can be reduced to two great categories of interactions namely the TELL/ASK queries. The TELL-queries allow external agents to enrich (prefixed with kb_project) the knowledge base with information or to retract/delete information from the knowledge base (prefixed with kb_unproject). To modify facts in the knowledge base, the previous fact is retracted and then the new one is inserted. The other query category is the ASK query which allows to retrieve or construct new facts from the knowledge base. These queries are prefixed by kb_call. These prefixes are then followed by a specification of a set of parameterized predicates which should be fulfilled at the end of the query execution. Imagine that the robot would like to memorize the course of

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

its activity and then reason about the success of the involved actions, this would take place as follows:

```
// Creating an action grasping
kb_project(
        [new_iri(Action, tracebot:'Grasping'), is_action(Action)]
).

// id of the created action is stored into Action
Action = tracebot:'GraspingX1GH07'

// an object of Type canister is created and set as parameter of the created action
kb_project([
new_iri(Object, tracebot:'Canister'), is_object(Object),
has_parameter(tracebot:'GraspingX1GH07', Object)
]).

// id of the created object is stored into Object
Object = tracebot:'CanisterH8YK23'

// id of the created object is stored into Object
kb_call( [ is_successful(tracebot:'GraspingX1GH07')]
```

As you can see from the above interactions with the reasoning system, an action of type Grasping is created and the unique id of that action is stored into the variable Action. Then, an object of type Canister is also created and the unique id of that canister is stored into the variable Canister. After, this, the created canister is set as the parameter of the created grasping action. And finally, the reasoning system is tasked on the success or feasibility of that grasping action. Note that before actually tasking the reasoning system, a lot more contextual information about the canister (e.g., pose, material, container, ...) and about the grasping action (e.g., effectors) could be provided to the reasoning system in order to enrich its knowledge base.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

6.5 Symbolic and Simulation-Enabled Reasoning for Verification

In this section, we demonstrate verification of an action feasibility based on symbolic reasoning and then introduce our achievements in leveraging physico-realistic simulations to enable advanced reasoning with respect to functional verification.

6.5.1 Symbolic Reasoning for Verification

At the core of the symbolic reasoning engine is KnowRob presented in D5.1 and D5.2, which itself builds on top of first-order logic reasoning engine Prolog.

Imagine that we described the precondition of the grasp action (precondition:Grab) in the ontology such as illustrated by the Figure 6.4 and 6.6.

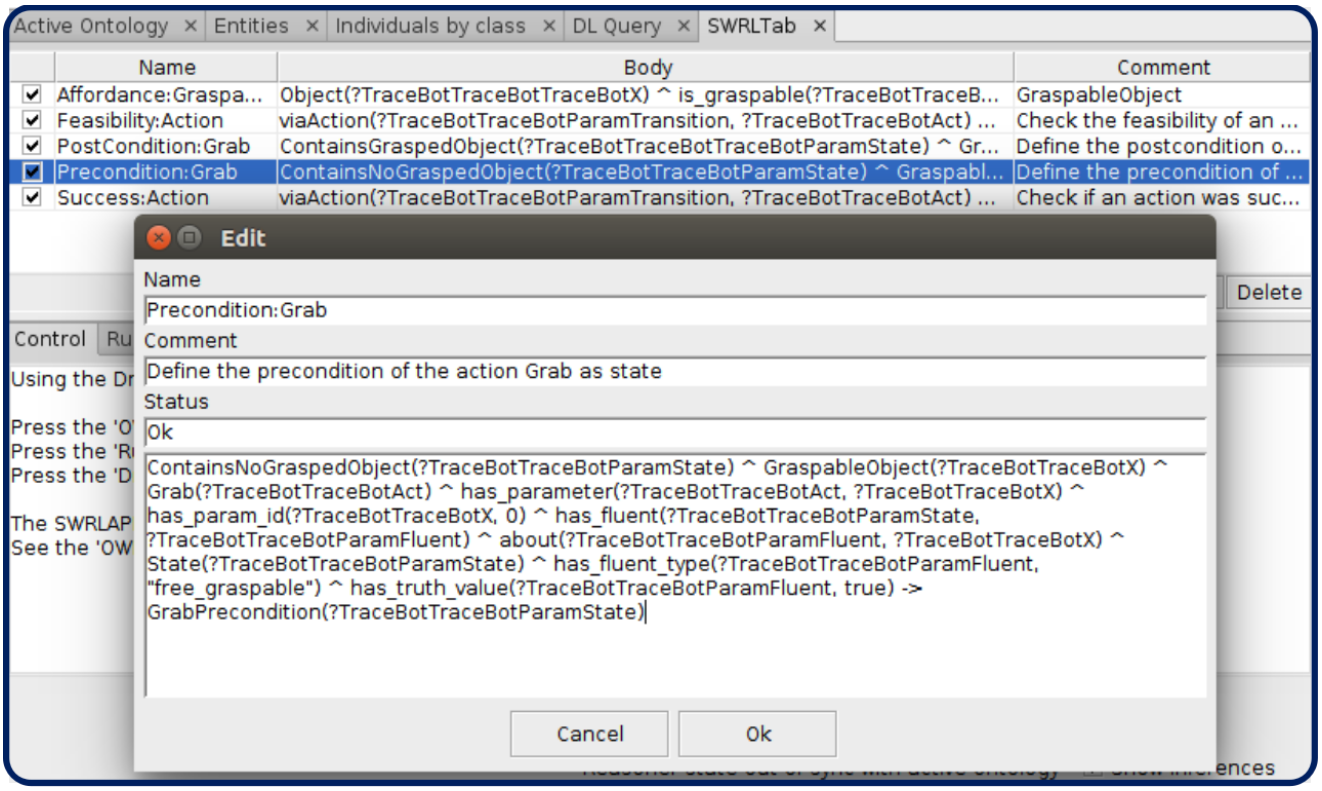

Figure 6.6: View of action post- and pre-conditions from Protégé Software

Then an object of the class Canister (scn1FrG_Canister) is detected in the scene and asserted in the knowledge as shown by Figure 6.7.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

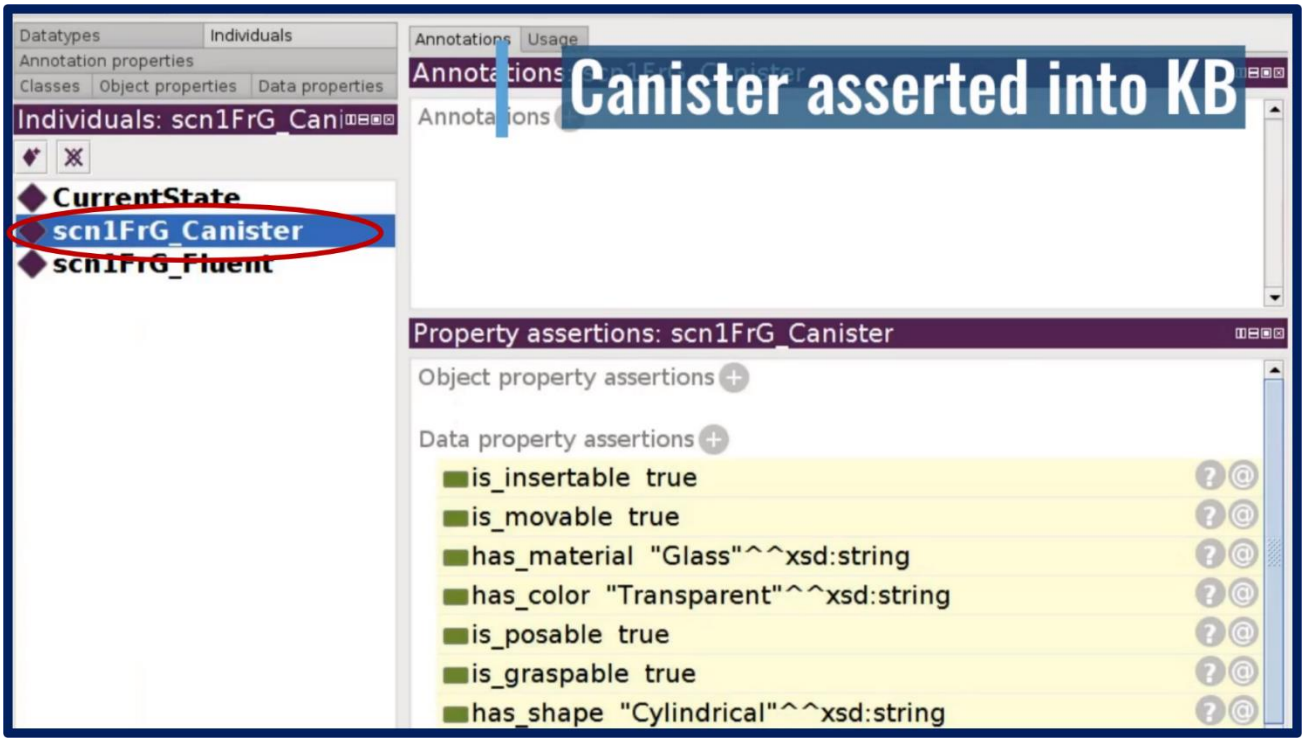

Figure 6.7: Assertion of detected canister into knowledge base

Beside the above information about the detected canister, the tactile verification tells us that there is nothing in the robot gripper which is then inserted in the knowledge base through a fluent as shown by Figure 6.8. The fluent scn1FrG_Fluent is about the above detected canister and is true if the concerned object is not in the robot gripper, hence its name Free_graspable.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

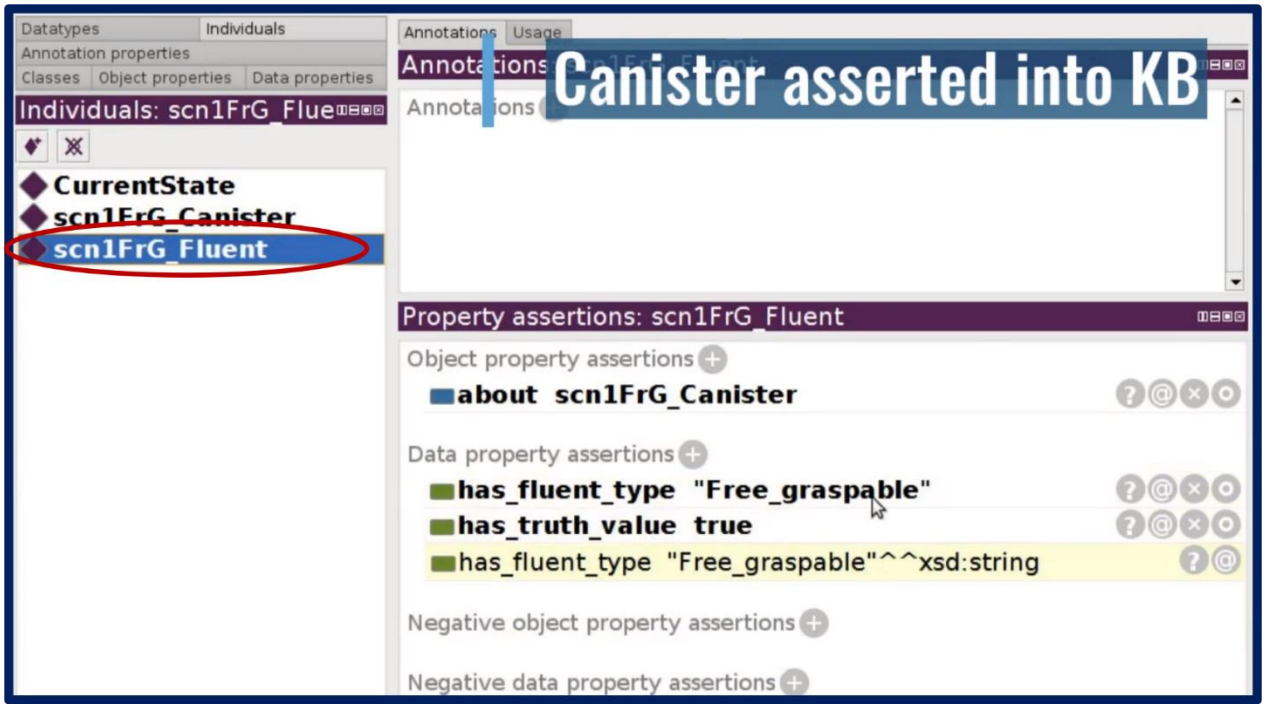

Figure 6.8: Enrichment of process context with low-level verification information

Once the fluent is created, it is attached as being part of the current state CurrentState as shown by Figure 6.9.

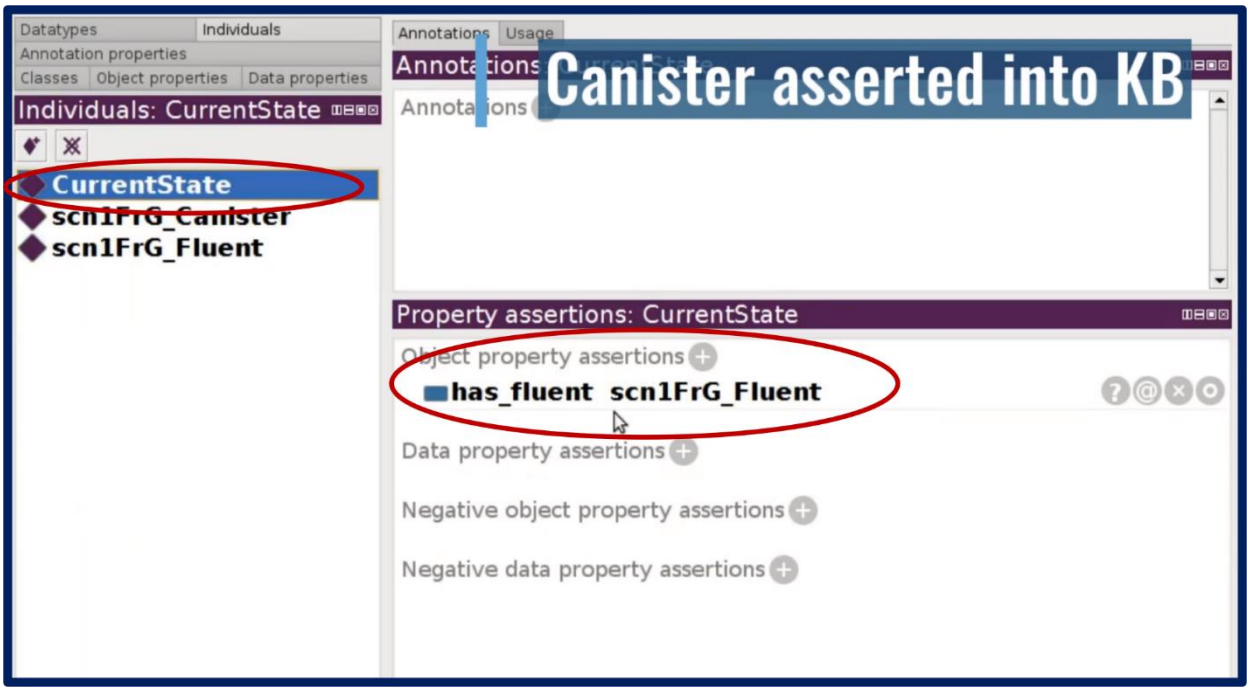

Figure 6.9: Associating a fluent to a world state

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

Then, before trying to grasp the detected canister the robot checks if the action is feasible in the given context and tasks therefore the reasoning system. For completing this task, the reasoning system creates an instance of the grasp action scn1FrG_Grab and sets the detected canister as its parameter. This is illustrated by Figure 6.10 below.

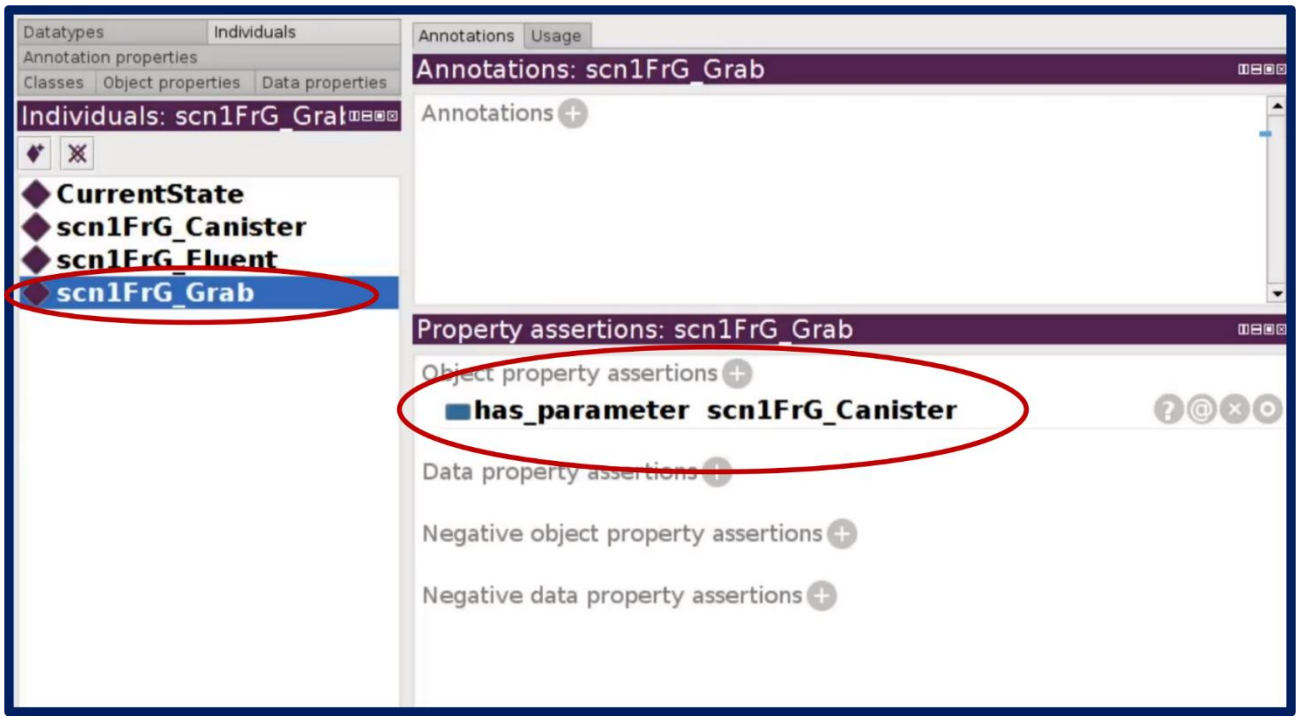

Figure 6.10: Asserting an action and its parameters in the knowledge base

After these above assertions, the reasoning system checks if the current state of the world described by CurrentState is a precondition for the grasp action given its parameters. As shown by Figure 6.11, the answer is yes since the current state fulfils all the requirements of the grasp action namely that the target object is detected and not yet in the robot gripper (Note that this can be revised in dual arm manipulation settings)

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

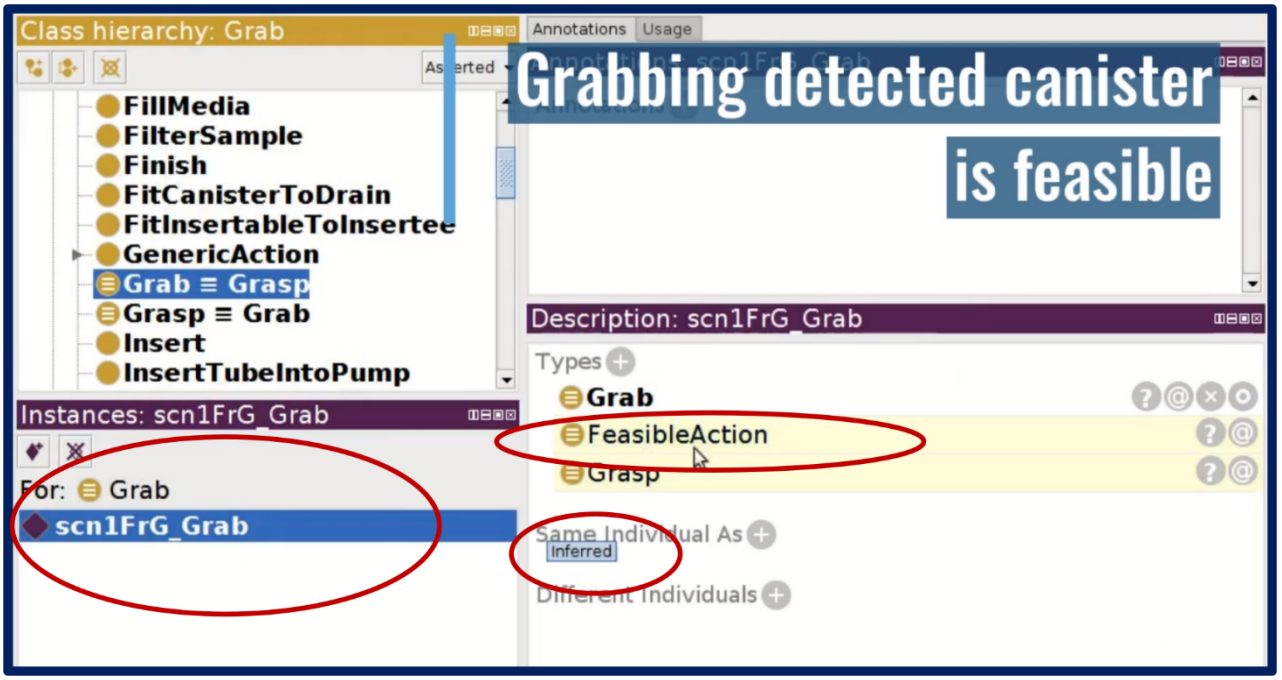

Figure 6.11: Deciding an action feasible in a given context

Finally, we suppress the information provided by the tactile verification to enrich the context, which actually consists in suppressing the fluent scn1FrG_Fluent from the knowledge base such as depicted by Figure 6.12.

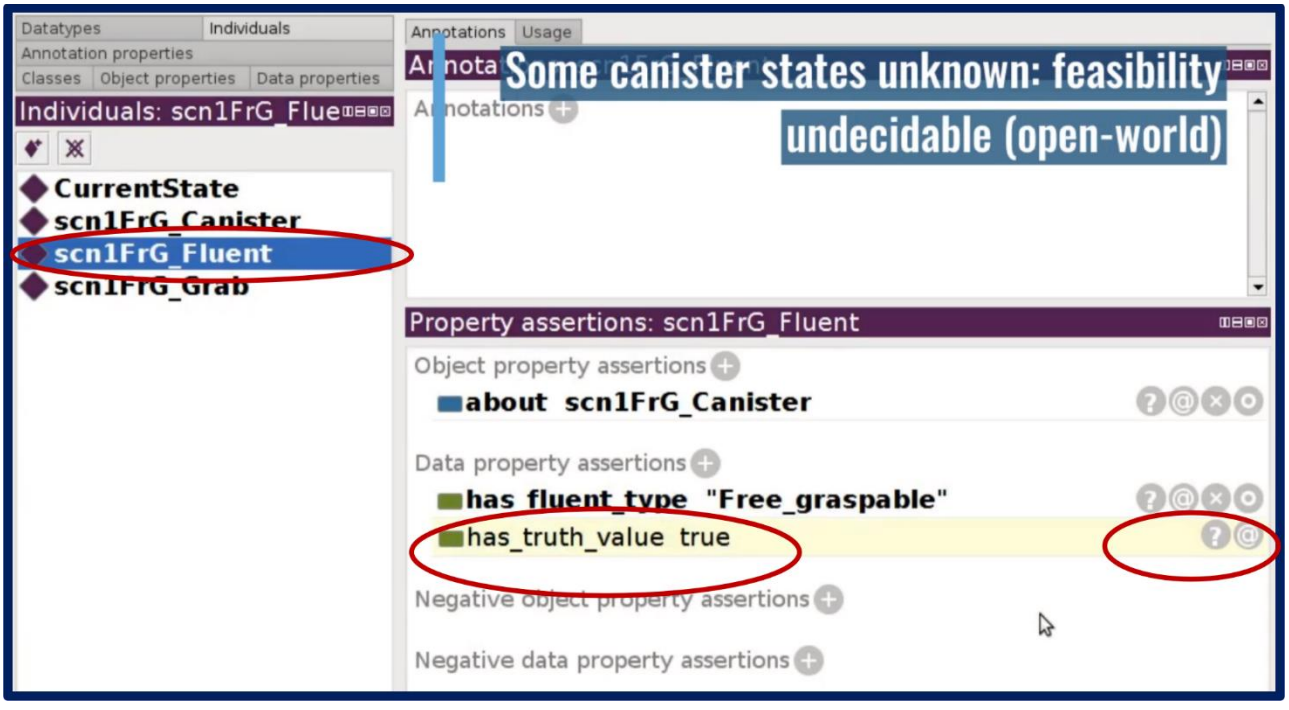

Figure 6.12: Retracting some context information from the knowledge base

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

This impoverishment of the knowledge base causes then the reasoning system to be unable to decide on the feasibility of the grasp action. Note that the reasoning system is working under the open-world assumption and this decision would be a radical "unfeasible" if it was under a closed-world assumption. This undecidability is shown by the Figure 6.13 below.

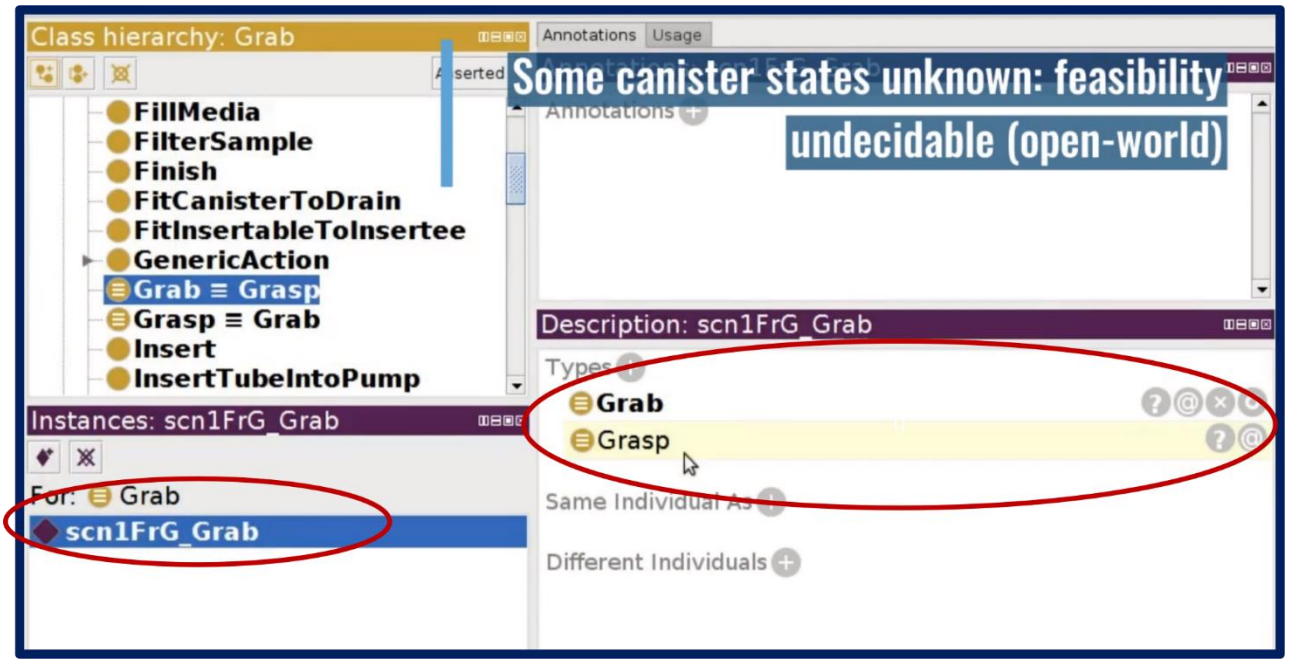

Figure 6.13: Undecidability of action feasibility under lack of context information from the knowledge base

6.5.2 Simulation-Enabled Reasoning for Verification

Notice that despite the above described capabilities of the symbolic knowledge representation and reasoning in accomplishing functional verification, this only holds at a very abstract level of task consideration (e.g., actions as sequence of symbols) and verification (e.g., binary decision: success vs failure). And this being said the consequences are fourfold. (1) First, given that the robot operates in a complex dynamic environment, the robot should not only be able to verify the success of its action given the observed effects but rather also anticipate such effects in order to avoid undesirable ones. Note that this requires the robot to maintain fine-grained models of the environment (e.g., friction of the table) as well as models of actions (i.e., mass of the canister for grasp action) in order to be able to answer questions such as what will happen if a half-full canister was to be inserted into the drain tray with a certain force. (2) Secondly, notice also that even if the canister has to be pushed a little bit after inserting it to check if it was well inserted, the force used to perform the push is crucial though it is not taken into account in the symbolic reasoning, which then leads to some test vagueness. In this case the system should also be able to estimate the world and action parameters that explain certain expected or observed effects (How/Why happen). With this capability the robot can reason about fine-grained pre-conditions of actions. (3)

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

Furthermore, with such faithful models of the world, the robot would not only be able to anticipate and explain the world state through emulation but rather also obtain a higher-level rendering of it so that a straight-forward comparison between the expected and the observed effects of an action are possible. Finally, notice that any attempt to capture such faithful models of the world symbolically for the sake of accomplishing the points (1-3) will lead to an intractable system. While the capabilities (1-3) are usually referred to as physical reasoning, the capability (3) is referred to as imagistic reasoning. Regarding this analysis, we recently designed and published a scene understanding system through process emulation based on embodied probabilistic simulations, coined as NaivPhys4RP (Naïve Physics for Robot Perception) [8], for anticipating and explaining the states and observations of dynamic worlds in a transparent and causal manner. Robot Perception because it is an established concept for scene and self awareness and naïve physics because the ultimate goal is to intuitively capture the physics of the environment which ultimately determines the world state. Figure 6.14 below illustrates the core principles of NaivPhys4RP.

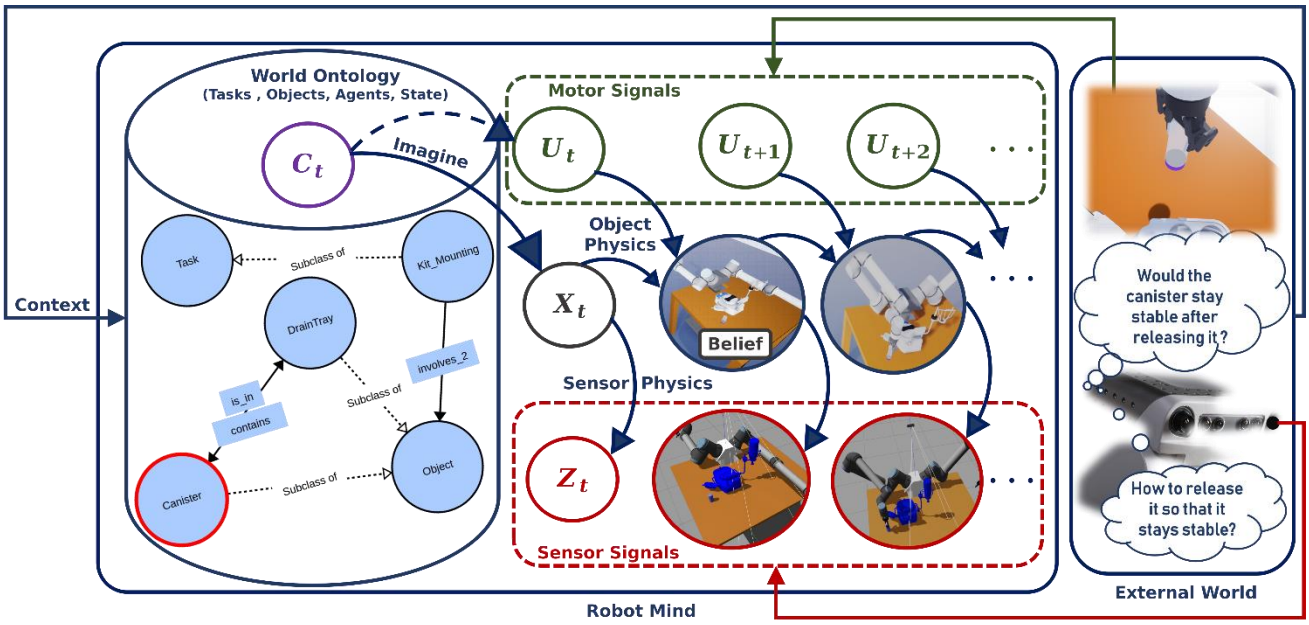

Figure 6.13: Anticipating and explaining the states and observations of the world through cognitive emulation for supporting functional verification.

We formalize the problem addressed by NaivPhys4RP in four steps. (i) We model the world state, as shown by Figure 6, as a Situated (i.e., take place in a context) Partially-Observable (i.e., only partial sensor data) Hidden (i.e., not directly accessible information) Markov Process (i.e., state dependency) (SPOHMP) that evolves through the physics that scene entities (e.g., objects, robots, sensors) undergo. The context is supposed to catch other domains of the commonsense that drive the physics such as intentions, utility and functionality. Actions are already explicitly modeled. (ii) We model the hidden state a.k.a.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

belief of the SPOHMP with the semDT (Symbolic knowledge base + digital twin), a photo-realistic and physics-faithful replication of the world grounded in the world ontology for semantics. This makes the internal world representation suitable for emulating the SPOHMP. (iii) Then, we regard perception as taskable through queries and these perceptual queries are clustered into anticipatory (i.e., consequences given causes) and explanatory queries (i.e., causes given consequences), that are abstracted as the bayesian/ markovian inference tasks. However, note that an actual accurate and rich belief of the world state is the informational source for answering these questions. Such a belief is continuously filtered over time through emulation of the SPOHMP. (iv) Finally, we efficiently implement the four main operators of the rao-blackwellized particle filter, however modified to five operators, which is a generic, practical and constructive (i.e., explainability) approach to simultaneously emulate the SPOHMP and address the bayesian inference tasks just mentioned, through embodied, physics- faithful, photo-realistic, probabilistic, partial and ontology-grounded simulations. Notice the genericity of the model as it also considers fundamental physical parameters of the world necessary for useful simulation and commits to estimating them.

This formalization is summarized by the following system of equations (S1):

$$\begin{cases} \boldsymbol{X}_t^* \sim \boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{U}_{0:t-1}, \boldsymbol{Z}_{0:t}, \boldsymbol{C}_{0:t}) & \text{, actual belief} \\ \boldsymbol{X}_{t+1}^* \sim \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{U}_t, \boldsymbol{X}_t, [\boldsymbol{C}_{t+1}]) & \text{, state anticipation} \\ \boldsymbol{X}_{t+1}^*, \boldsymbol{U}_t^* \sim \boldsymbol{P}(\boldsymbol{X}_{t+1}, \boldsymbol{U}_t|\boldsymbol{U}_{t+1}, \boldsymbol{C}_{t:t+1}, \boldsymbol{X}_t, \boldsymbol{X}_{t+2}) & \text{, state explanation} \\ \boldsymbol{Z}_{t+1}^* \sim \boldsymbol{P}(\boldsymbol{Z}_{t+1}|\boldsymbol{X}_{t+1}) & \text{, observation anticipation} \\ \boldsymbol{X}_{t+1}^* \sim \boldsymbol{P}(\boldsymbol{X}_{t+1}|\boldsymbol{U}_t, \boldsymbol{X}_t, \boldsymbol{Z}_{t+1}, \boldsymbol{C}_{t+1}) & \text{, observation explanation} \end{cases}$$

- X, is the world's hidden state (e.g., a semDT)
- Z, is the object/world observation (e.g., RGBD images)
- U, is the motion control (e.g., joint values, forces)
- C, is the process context (e.g., object, state and task knowledge)

Note that i, t, [.] and ~ respectively denote the particle index, the time index, optional priors and the argmax probabilistic sampling. Note also that we demonstrated in D5.2 how the above reasoning tasks (S1) can be formulated in the query language presented in the section 6.4.

Finally, the full implementation of NaivPhys4RP as well as their integration together with the symbolic reasoning engine in the traceability framework constitutes the object of our next developments.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

6.6 Conclusions and Perspectives

In this section, we presented in an argumentative manner a methodology for addressing functional verification in the robotic realization of TraceBot scenarios, followed by a design and an initial implementation of a framework, the basics of the framework integration in the TraceBot robotic system, and proofs of concept that realizes this functional verification. Foremost, we defined functional verification as the ultimate check of action success, feasibility and proper equipment functioning, which was then reduced to making the robot to understand at an abstract level (e.g., world state in terms of physical qualities) as well as at a sufficiently fine-grained level (e.g., world state in terms of physical quantities) the pre-conditions (i.e., world state before action execution) and post-conditions (i.e., world state after action execution) of the robot actions. For verifying the proper functioning of equipment in such a formalism of functional verification, we showed the integration of additional verification-specific actions in the TraceBot robotic process. The proposed framework for addressing this problem consists then in maintaining a hybrid model of the world, namely an ontology of the world (symbolic) grounded in a digital twin of the world (faithful physico-realistic replication of the world) which then gets instantiated during the real or imaginary robotic execution of TraceBot scenarios and enriched by the outputs of low-level verification modalities (e.g., visual, tactile) to decide on an action success, feasibility, or proper functioning of equipment. For integrating the functional verification framework in the TraceBot robotic system, a formal language has been proposed as interface.

Given that the execution of the TraceBot project is organized around the implementation of TraceBot use cases, so is the implementation of the functional verification framework. The ontology as abstract knowledge about objects and foremost actions (e.g., plans, parameters, pre-conditions, post-conditions) has been accomplished for the first use case (inserting canister in drain tray) and this continues for the actual use case (inserting needle in the bottle cap). The digital twin with the necessary physico-realism (e.g., objects and robot actions) has been accomplished as well. The interfaces for basic functional verifications (e.g., was insertion successful, is insertion successful, what happen after placing the canister on the workbench) in these use cases (canister and needle insertion) based on hybrid reasoning (ontology- and digital twin-based) have been written as well. In this regard, the object of our next workloads will be the accomplishment of the framework for the second use case (i.e., needle insertion) and the continuous development and integration of the framework with respect to the target TraceBot use cases. Note that the focus is also actually laid on verification during the real robotic process execution. However, in the long term as explained in section 6.5.2, this verification would have to also take place during imaginary robotic process executions for anticipation reasons in the complex dynamic world.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

7 Deviations from the workplan

No notable deviation from the workplan was detected so far.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

8 Conclusion and perspectives

In this report, we have explored three different modalities for robot task verification: tactile verification, visual verification, and functional verification. Our findings show that each of these modalities has unique benefits and limitations, and that a combination of all three may be necessary for comprehensive task verification.

In the tactile verification section, we demonstrated the effectiveness of using hybrid tactile sensors to verify the success or failure of various tasks, including object presence in the gripper, object slip verification, clamp closure verification, and needle insertion verification. We have developed multiple algorithms based on machine learning and analytic methods, and achieved success rates higher than 95%. Our future work will focus on integrating these modules into the global system via ROS interfaces and fine-tuning the models on the final setup to ensure even more robustness and versatility.

Moving on to the visual verification section, we presented a set of vision methods based on differentiable rendering to verify the information inferred from the scene during the process execution. Our approach enables the incorporation of all existing knowledge into the scene representation and compares it directly to the output of the scene's sensor or to simple transformations of it, making it well-suited to verification. We have outlined a way to perform verification of object poses suitable for transparent or deformable objects and provided detailed steps to adapt this process for fill level estimation.

The functional verification section of our report outlines a higher-level verification modality that decides whether the actions of the robot were successful, whether it was even correct to perform an action within a context, and whether the equipment is working properly. We have formalized the TraceBot use case into a formal ontology that establishes the fundamental truths about objects and tasks in the world, and then enriched the actual state of the world with information coming from different verification sources to answer the questions of action feasibility, action success, and equipment functioning.

To conclude, our work highlights the advantages and limitations of different verification modalities. By combining multiple modalities, we can achieve more robust and comprehensive task verification, paving the way for safer and more efficient robotic systems.

Moving forward, there are several important steps that need to be taken in order to advance the research presented in this report. Firstly, the various verification modalities that have been developed and tested, namely tactile, visual, and functional verification, need to be integrated into a single cohesive system. This will allow for the combination of all the different senses and logic, in order to improve the overall accuracy and success rate of the system. Additionally, the models developed for each verification modality will need to be fine-tuned and optimized for the final setup, and the system will need to be further tested and validated on a larger scale. Furthermore, the functional verification framework will need to be fully implemented and integrated into the TraceBot robotic system, and the ontology and digital twin will need to be continuously updated and refined for each TraceBot use case. Finally, as

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

noted in the report, there is a need for the system to be able to perform verification during both real and imaginary robotic process executions, which will require further research and development.

Horizon 2020
TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

9 References

[1] Bauer, D., Patten, T., & Vincze, M. (2020). Scene Explanation through Verification of Stable Object Poses. ICRA 2020 Workshop on Perception, Action, Learning: From Metric-Semantic Scene Understanding to High-level Task Execution.

[2] Bauer, D., Patten, T., & Vincze, M. (2020). VeREFINE: Integrating Object Pose Verification with Iterative Physics-guided Refinement. IEEE Robotics and Automation Letters (RA-L), 5(3), 4289-4296.

[3] Bauer, D., Patten, T., & Vincze, M. (2021). ReAgent: Point Cloud Registration using Imitation and Reinforcement Learning. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 14586-14594.

[4] Bauer, D., Patten, T., & Vincze, M. (2022). SporeAgent: Reinforced Scene-level Plausibility for Object Pose Refinement. IEEE Winter Conference on Applications of Computer Vision (WACV), 654-662.

[5] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian. Deep Residual Learning for Image Recognition. Computer Vision and Pattern Recognition (cs.CV), FOS: Computer and information sciences, FOS: Computer and information sciences

[6] Remazeilles, A. et al. (2023). Robotizing the Sterility Testing Process: Scientific Challenges for Bringing Agile Robots into the Laboratory. In: Tardioli, D., Matellán, V., Heredia, G., Silva, M.F., Marques, L. (eds) ROBOT2022: Fifth Iberian Robotics Conference. ROBOT 2022. Lecture Notes in Networks and Systems, vol 589. Springer, Cham. https://doi.org/10.1007/978-3-031-21065-5_19

[7] Beßler, D., Porzel, R., Pomarlan, M., Vyas, A., Höffner, S., Beetz, M., Malaka, R., & Bateman, J.A. (2020). Foundations of the Socio-physical Model of Activities (SOMA) for Autonomous Robotic Agents. Formal Ontology in Information Systems.

[8] Franklin K. Kenghagho et al., "NaivPhys4RP - Towards Human-like Robot Perception “Physical Reasoning based on Embodied Probabilistic Simulation”," 2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids), Ginowan, Japan, 2022, pp. 815-822, doi: 10.1109/Humanoids53995.2022.10000153.

[9] He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition* (arXiv:1512.03385). arXiv. https://doi.org/10.48550/arXiv.1512.03385

Horizon 2020

TraceBot receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)