Traceability framework for laboratory automation

Deliverable 4.1

Deliverable Title	D4.1 Traceability framework for laboratory automation
Deliverable Lead:	Technische Universität Wien (TUW)
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), Frank Vial (CEA), Anthony Remazeilles (TECN)
Dissemination Level:	Public
Due Submission Date:	28/2/2022
Actual Submission:	28/2/2022
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 the traceability framework for laboratory automation. It outlines the fully functional first version of the framework and architecture and gives a first verification example in one of the modalities using visual verification of the canister pose.

Horizon 2020

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

Versioning and Contribution History

Version	Date	Modified by	Modification reason
v.01	09.02.2022	Jean-Baptiste Weibel (TUW)	First version
v.02	11.2.2022	Franck Vial (CEA)	CEA input
v.03	11.2.2022	Patrick Mania (UoB)	UoB input
v.04	15.2.2022	Markus Vincze (TUW)	Table, summary, first proofread
v.05	17.2.2022	Patrick Mania (UoB)	First version check
v.06	18.2.2022	Anthony Remazeilles (TECN)	Second proofread
v.07	22.2.2022	Markus Vincze (TUW)	Final updates
v.08	23.2.2022	Patrick Mania, Franklin Kenghagho Kenfack, Michael Neumann (UoB)	Final updates
v.09	25.2.2022	Anthony Remazeilles (TECN)	Final proofread

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)

1 Executive Summary

To comply with regulations in the field of laboratory automation, the goal of the TraceBot project is to put in place a traceability framework that can log all relevant data for creating an audit trail. To achieve this, we propose a set of verification methods across modalities to guarantee that the process is executed properly and that the data supporting the execution by the robot and documenting what has happened is also logged.

The traceability framework serves the primary purpose of implementing a systematic way to capture any operation or action conducted by the robotic system. To do so, we replaced standard actions by Traceable Actions in the action execution process. These traceable components automatically gather and send their input and output (i.e. goal, feedback and result) onto a specific and unique canal. This allows to automatically keep track of the process structure but also the relevant subsymbolic data and generally the sensor data. In addition, the Traceable Actions can trigger a memory episode in the KnowRob system. The combination of both stores all the symbolic and subsymbolic information of the process executed to enable inspection. That data is the base of the audit trail, which is generalized as a Narrative-Enabled Episodic Memory (NEEM). NEEMs can be meaningfully queried by humans during inspection answering questions directly such that humans do not need to meticulously inspect all sensor data to understand the success or failure of any action taken by the robot.

The development of a set of verification or checking actions using visual and tactile sensing, but also functional considerations thanks to the knowledge infrastructure also guarantee not only that the execution is going as expected but also that all that information will be traced, as such verification steps are also implemented using Traceable Actions.

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 aims to describe the structure 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 document proceeds as follows. The traceability framework is presented in Section 3, the different verification modalities in Section 4 and finally Section 5 presents a first visual verification applied to a simple use case, the detection of the canister.

Horizon 2020

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

3 Traceability Framework

To achieve traceability in the context of the TraceBot project, we designed a framework that enables the automation of the specified testing use cases while also considering the required information flow and interpretation mechanisms necessary to generate a regulatory-proof audit trail. This architecture respects the typical distributed nature of modern robotic systems, integrates the novel modalities for verification and allows to generate structured trails for task executions. In the following sections, we highlight the core communication schema and central components required for the Traceability aspect.

3.1 Overview

The concept of traceability in the TraceBot project is realized by creating the system ability of a level of "self-awareness" during its task execution. At any time, the robot is able to check whether the task was completed according to the specifications it was given, and keeps all the information necessary for a human to later analyse the process execution.

Every task is subdivided in a sequence of actions. We distinguish two types of actions:

1. Process actions: the perception or manipulation action to carry out one step in the assembly process of the sterility kit. To ensure that these steps are executed correctly, there will be means of verification. We use the verification to detect failures and enables to recover and correct the operations to realise objective 4 (safe and failure-resistant operation).
2. Checking actions: is an additional action that needs to be introduced into the normal assembly process due to the regulatory requirements of the medical domain. The checking actions contribute to realise Objective 1 for creating traceability to allow understanding of manipulation actions such that the robot is able to describe semantically what it is handling and to determine success or failure of the action.

Note that in the context of this Deliverable, we use the term "action" to describe the robot behaviour. As commented in D6.1 and D1.1 [10,11], any behaviour of the robot, or action, is *technically* encapsulated as skill, for modularity and distributivity purposes. On a practical perspective, these skills are implemented as ROS actions. We refer here to action for these behaviour components, as the concept of traceability could be generalized to any actions, outside of a skill framework.

In terms of sequencing these actions, there is a clearly defined sequence of process sections, where a checking action may additionally follow the process action to verify that an assembly step has been correctly performed. The checking action is included whenever it is necessary to comply to the regulatory requirement for creating the audit trail.

To realise that both, process and checking action, are traced, which is necessary to achieve the complete audit trail, we created **Traceable Actions**. Since both types of actions are carried out by the robot and its supporting sensors and equipment, the distinction is only relevant when creating the sequence of actions. For execution, we treat all actions as Traceable Actions, see also Figure 1 below.

Thanks to the modularity of the action definition, a given action and its corresponding verifications action can be combined into a new action. The final objective is to only use verified actions in the

Horizon 2020

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

process definition. The verification provides a level of guarantee that the action was successful, and the data gathered both from the action and the verification step allow a better understanding of the process post-execution.

When scrutinising the different actions that are required in the project, we found two ways or channels of verifying that the (traceable) action has been executed correctly (see also D1.3[12]).

1. The **Digital Twin channel**, where all robot (perception and manipulation) actions are checked by comparing them to the Digital Twins' expectation. This enables a first kind of the "self-awareness" of the robot about its actions and if they are executed as expected.
2. The **interactive verification channel**, where the robot actions are executed to create "functional", "visual" or "tactile" data feedback with the aim to verify if the task has been executed correctly with the initial robot actions. How this verification is done is defined when creating the action and verified directly in the process. Examples are closing the clamp or visually verifying the locating of an object, see Figure 1 below.

Both these verification channels eventually feed into the Tracer and this enables the creation of the audit trail.

To generate the audit trails, the architecture shown in Figure 1 is based around two key components, the Tracer and NEEMs. One central component is the so-called Tracer, which is interconnected with all actions that are executed in the Execution Manager. The execution of the individual actions results in a sequence of executed Traceable Actions. This information provides a structure of the executed (sub-)actions including their goals, feedbacks and results. The structure is important for dividing the gathered information during the testing procedure into reasonable and referable sub steps in the audit trail. This will for example allow the user to look into the data that has been generated during the execution of a certain action including the execution context.

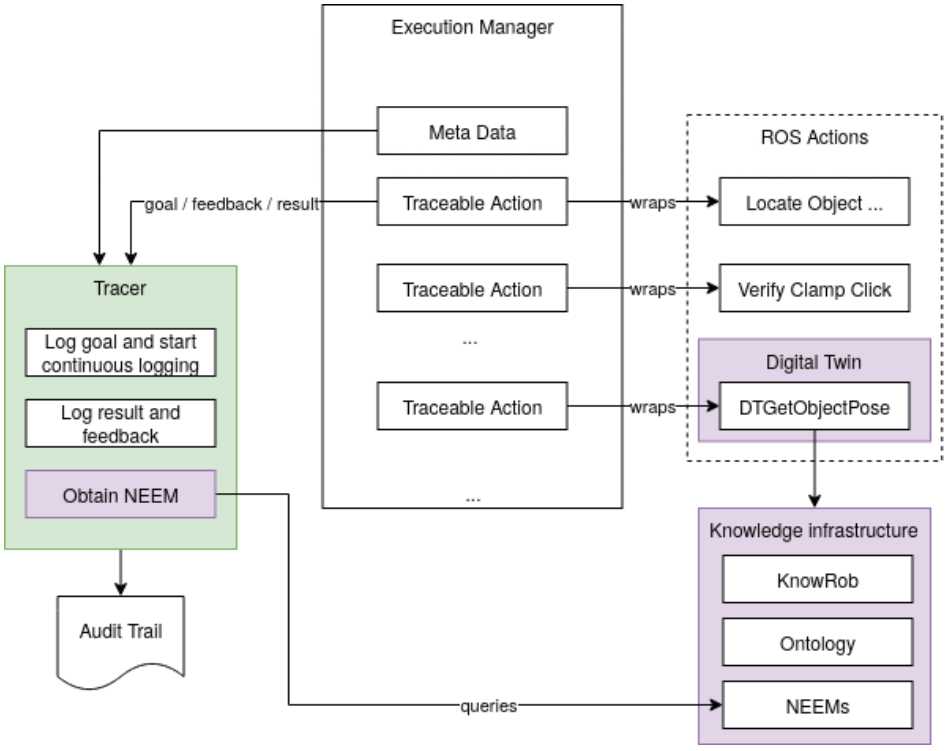

Fig. 1: Architecture of the Traceability Framework.

Horizon 2020

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

The Tracer is retrieving this information and the metadata associated with the Traceable Actions and combines it with our second central component, the so-called Narrative-Enabled Episodic Memories (NEEMs). NEEMs are a comprehensive model to represent robot memories of task executions. It allows to introspect the robot belief state of task executions at arbitrary points in time while also providing semantic information for entities like the manipulated objects or facts about the environment. This effectively enriches the logged trails by methods to represent and reason about knowledge that the robot acquired during the execution of the testing procedure. NEEMs are part of our TraceBot knowledge processing infrastructure and are integrated with state-of-the-art ontologies [7] and the KnowRob [8] knowledge representation and processing framework. In the following sections, we will provide more details about this knowledge-based module and also kindly refer to our previous Deliverable 5.1 [9] which is about the conceptual and reasoning apparatus of the overall Traceable Semantic Twin for more details about NEEMs, the Digital Twin and our knowledge processing infrastructure.

3.2 Data flow in the system

The Tracer component developed as part of the TraceBot project is responsible for creating the final audit trail based on the process execution. It therefore needs to keep track of every action the robot is taking and the data justifying this action, but also the data proving the accurate execution of any given action. Because common modern robotic software architectures for complex automation tasks are distributing their responsibilities on a plethora of submodules, a system that enables Traceability needs to address this multi-channel nature of the communication flowing in the system and should allow to assert the acquired data and the execution context into audit trails. To keep track of all the relevant information, the Tracer is directly connected to both the Execution Manager and the Digital Twin, which can represent the robot knowledge at a given time through the concept of NEEMs.

To be able to combine the information from all subcomponents in the architecture into a single comprehensive audit trail, every action will be wrapped into a Traceable Action. This includes control commands like moving an arm but also verification feedback from components like the Digital Twin or estimators for event detection based on sensor data, like for example a clamp click detection. Wrapping the execution and feedback of subcomponents into loggable Traceable Actions allows the system to combine a diverse set of (sub-)process feedback and verification modalities and generate an audit trail based on the acquired data.

3.2.1 Flow between the process and the Tracer

The data flowing from the Execution Manager is logged by the Tracer using Traceable Actions (every input, output and feedback of every Action is kept track of automatically). The notion of Traceable Action is implemented as a wrapper around an Action, enabling the automatic transmission of information from the Execution Manager to the Tracer component. In ROS terms, for any action executed, the goal, feedback and result are automatically collected and sent to the Tracer through a single topic. An ID is attached to every specific message. Each message also keeps track of the ID of its parent Action to allow the reconstruction of the entire tree of action executions.

Horizon 2020

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

An interesting point of the implementation choice we made, using a wrapper, is that this logging functionality does not require any additional work from the component developer, as it is automatically handled through the wrapper, independently of the interface of the action defined, as long as it is defined as a Traceable Action.

This structure allows for a complete reconstruction of the process execution structure. It also helps distinguish between messages even if they arrive out of order because of network bottleneck, and enables the execution of parallel Traceable Actions while guaranteeing a correct structure in the final trail. The Tracer collects all information and is then able to create a unique trail of information. The Tracer also saves the information regularly during execution guaranteeing that as little data as possible is lost in case of a software issue, a situation in which a valid audit trail is of particular importance. The entire information gathered by the Tracer for the entire process is currently saved as structured data in the JSON format. This is done to make the information easy to parse and format according to the final needs.

3.2.2 Flow between the process and the Knowledge Infrastructure & Digital Twin

At the beginning of the plan execution, an episode is started by the Tracer. Afterwards, similar to the way the actions' input and output are kept track of, whenever an action goal or result is prepared to be traced, a corresponding query is generated by the Tracer for the KnowRob system, triggering a memory episode and asserting the acquired information. Queries have a fixed structure based on their type, such as asserting skills, object detection or events. The information which is required by the query is grounded in the ontology. After the episode is finished it is possible to extract the knowledge with queries. This allows us to enhance the audit trail with semantic information. Example questions used for the audit trail, could be the timeline of executed actions, highlighting events such as click detection, the robot configuration at a specific substep in the process or images of skill results and verification tasks.

The Tracer is also connected to the Digital Twin to record the system knowledge about itself during the execution of different tasks. The DT is a provider for the sub-symbolic robot belief state, through physically simulating the state of the world based on the robot perception. If the robot releases a grasped object, the robot has to perceive the object to assert its pose. By continuously simulating the world state, it is possible to get an expected pose of the object. This expected pose can then be compared to the actual pose of the object in order to verify the result of the action execution. These methods are also wrapped into individual Traceable Actions and can therefore get logged automatically to the audit trail.

3.3 From traceable actions towards audit trails

The very reason of tracing actions is the production of audit trails. NEEMs stands for Narrative-enabled Episodic Memories. They actually encode chronological traces of activities at a symbolic (i.e., semantic) as well as at a sub-symbolic level. At the symbolic level also known as NEEM narrative, the story of the ongoing activity is collected including for instance annotations of scene objects (i.e., spatial relations, classes, colours, shapes) and robot actions (action types, transitions, events), whereas the sub-symbolic description also known as NEEM experience encompasses sensor data such

Horizon 2020

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

as entities' trajectories, poses, images, etc. In this project, NEEMs constitute the foundation of audit trails which themselves are chronological sets of records providing evidences that the processes were compliant with medical sterility test regulations. Once the process traces from different components of the system have been collected, they will be integrated into NEEMs. Figure 2 illustrates the NEEM of a kit mounting process.

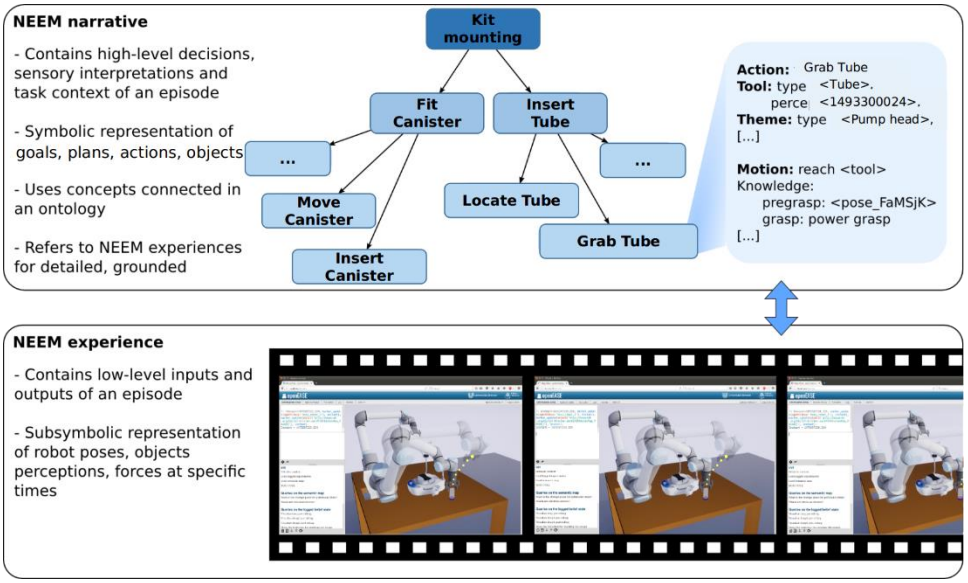

Fig. 2: NEEM of a kit mounting process.

Beyond substituting the concept of audit trail by definition and systematizing it, NEEMs are grounded in a sufficiently large ontology of the robot world, which in turn embeds more semantics in NEEMs and enables therefore a better understanding of what happened during the process. The Socio-physical Model of Activities (SOMA) is an ontological modelling approach for autonomous robotic agents performing everyday manipulation activities. It tries to catch the social as well as the physical context of activities. Note that SOMA targets generic manipulation activities though recent works have been focusing on specializing it for kitchen-related activities. In this project, the SOMA ontology is extended with medical sterility testing-specific concepts.

Concretely, the ontology will encode common knowledge about TraceBot objects such as canisters, tubes, needles, pumps, the properties or attributes of such objects such as the colour, material, shape, volume, location, 3D models but also the relations (e.g., spatial, compositional) among these objects. This information would then allow for instance to reason about plausible and feasible states of objects while interacting with them, which will then enable decision making and failure detection. Given the common spatial relations among the components of the pump, one might detect a failure if the description from the NEEMs does not match the common sense. Note that the ontology is also intended to encode knowledge about actions such as their temporal dependency (i.e., pouring precedes opening), but also their pre- and post-conditions. Reasoning on such knowledge and NEEMs would allow for instance to detect action failures. That is, the effects of an action, such as described in the NEEMs, do not match the description in the ontology. Figure 3 highlights some preliminary works in the work package 5 on the extension of SOMA for TraceBot. This is a screenshot of the TraceBot

Horizon 2020

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

ontology from the ontology visualizer “Protégé". As you can see on the top left corner of the image, the ontology extends SOMA and captures TraceBot-specific concepts such as sterility kits, canister, pump, which actually extend super-concepts in SOMA (e.g., Canister extends container, medical canister extends canister). Moreover, the relationships among these concepts such as aggregation are shown (e.g., Bottle has a lid).

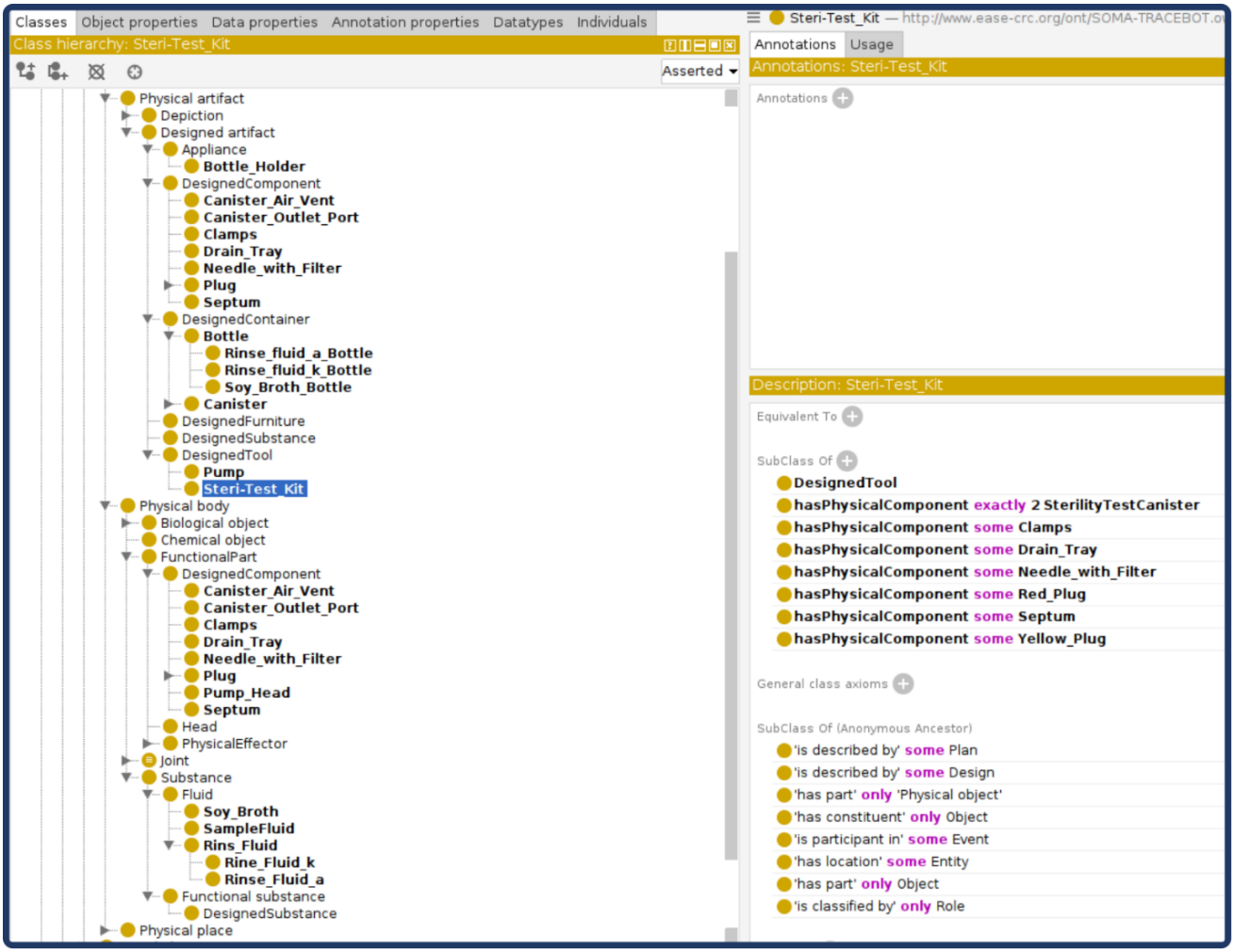

Fig. 3: Extension of the broad SOMA ontology to the TraceBot domain.

Once the NEEMs have been recorded, one should be able to access it in order to know how the process went and if there were failures. One could just focus on consulting the recorded data. However, it will be very tedious not only in time but also semantics, which is only difficultly and directly accessible by humans. For this reason, we are also developing a sufficiently rich query language that allows either robots or humans to quickly access the content of NEEMs. For instance, queries such as “what action happened after action X?”, “what is the state of object X after action Y?”, “How does the scene look like after action X?” can be issued to the reasoning system being developed in work package 5. The concepts of NEEMs, query language, reasoning and ontology have been sufficiently described in D5.1 [9], which is the first deliverable of the work package 5 and can be accessed for more information. The Figure 4 below illustrates the visual access of a NEEM of the execution of TraceBot’s simplified use

Horizon 2020

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

case, which is about grasping the canister and inserting it in the drain tray. As you can see from the bottom right corner of the image, a query is issued to the reasoning engine to extract all the trajectories for the grasp-moving actions of a canister, which are then subsequently shown or replayed. This can be used by a human auditor to attest that a specific action where successfully performed.

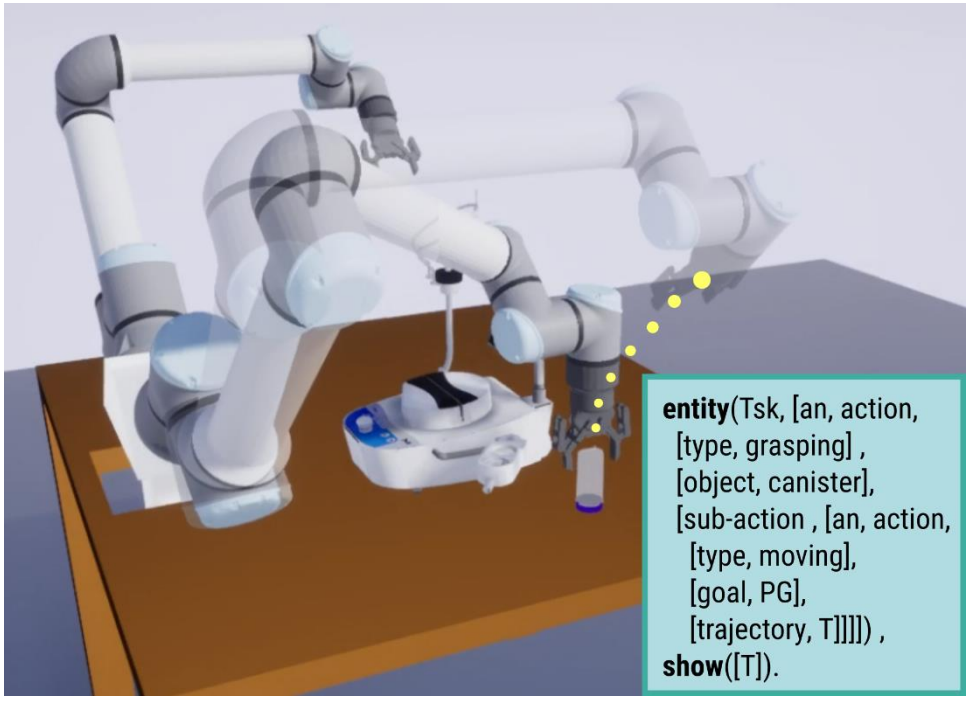

Fig. 4: Visual access of NEEMs through reasoning query.

Horizon 2020

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

4 Tactile, Visual and Functional Verification

The traceability objective cannot be completely addressed if the robot is not equipped with means to verify the good execution of it 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. Please also refer to D1.3 [12] and the verification process using acceptance tests (see Sections 3 and 4 in D1.3).

Table 1: Steps of the process steps for the sterility kit.

#	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	Check force profile
5	Wetting	Pump, canister, plugs & bottle	Operate pump	1	-	read pump data
6	Sample Transfer	Pump, canister, plugs and vials	Vial breaking	2	locate vials	confirm brake and emptyness

Horizon 2020

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

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	Cutting, force	1-2	locate clips	can see cut tube
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

4.1 Tactile Task Verification

Tactile verification is planned for every robot grasping action and several of the assembly actions. During grasping actions, tactile information will be utilised to confirm the object has been properly taken. For the specific verification actions, we will use tactile feedback to capture and exploit forces evolution while executing an operation, e.g., when inserting the needle in the septum, or when closing the clamp.

The multi-fingered gripper developed in WP2 will include several tactile modules in order to get tactile feedback on the object being held. Those modules will be attached to the phalanges of the fingers and into the palm of the gripper, where the physical contacts with the object take place during grasps and manipulations. Tactile modules are under development and will be made up of two thin sensitive layers stacked within a same assembly. The first layer will be based on a piezo-resistive material arranged in a matrix way, in order to capture the 2D pressure distribution over its surface. With a bandwidth of [0-40Hz], it will measure continuous pressure as low frequency pressure variations. It will deliver spatial and temporal information, enabling for instance to track the evolution and/or the displacement of the contact point(s) over its surface. Figure 5 illustrates an example of the pressure distribution captured by a preliminary piezo-resistive setup while grasping a flexible catheter made up of two parallel tubes.

Horizon 2020

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

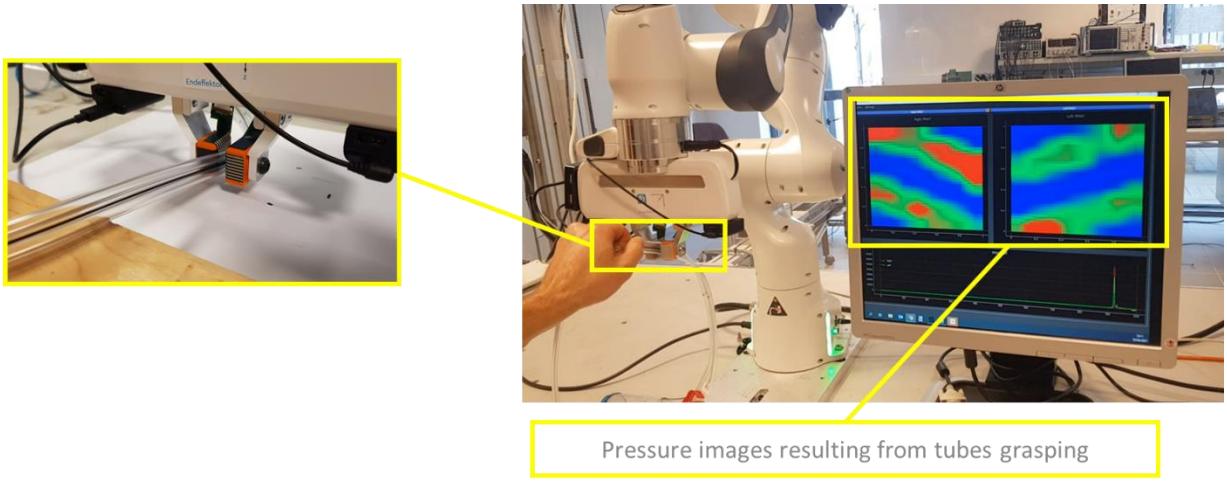

Fig. 5: Pressure images captured by a preliminary tactile sensors setup while grasping two parallel flexible tubes. The sensors setup consists in two pressure sensitive matrices, each matrix being related to one of the two fingers of the gripper. When grasped, the tubes are squeezed between the two matrices. The color scale enables to visualize the pressure distribution captured by each of the matrices, with red / green / blue colors respectively for high / intermediate / low pressure values.

The second sensitive layer will consist in a piezoelectric patch, enabling to capture mechanical vibrations up to 4 kHz. The objective here is to sense high frequency variations of the contact to be complementary to the piezo-resistive measurements. As vibrations spread within materials, a single piezo-electric cell seems sufficient and leads advantageously to a simpler and to more compact electronics than a multi-cell arrangement would.

Tactile sensing can be used to check the smooth running of operations for different use-cases. Since the tactile sensing can detect the contact with the different phalanges and the palm of the gripper, it enables to check that the gripper parts that need to be touching the object are actually in contact with it. It can be used to check whether the Digital Twin belief about the object status is correct or not, while grabbing or manipulating the object. A full or partial loss of contact with the object during the manipulation may also occur for different reasons. For example:

- if the gripper is not applying the right pressure
- if the gripping pose is not correct
- if obstacles in the environment bump into the object

This loss of contact can be reported through the tactile sensors. In practice, the motion controller could generate a reference contact array for each manipulation requiring a grasp. This array would specify for each part of the gripper whether a contact is expected or not. The tactile outputs would then be continuously compared to those expectations, and if wrong, a warning would be raised.

This verification can be refined by exploiting the spatial information provided by the tactile sensor(s). The centroid of the pressure will be computed in order to locate the equivalent contact point on the sensor surface, and by extension, on the phalanx or on the palm. This information may be particularly helpful when grabbing small or thin objects for further precise manipulation. As a use case example, we can mention the needle handling. If our plan is to grab the needle between the distal phalanges before removing its guard, a poor positioning of the needle within the phalanges would affect the success of the removal operation. More than the “yes/no” contacts expectation, the reference array

Horizon 2020

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

could then indicate the valid area of the sensor(s) in which the contact(s) should take place, whenever relevant according to the use case. The verification step here would consist in comparing the contacts location estimated from tactile measurements to the reference array content. A warning would be raised when different, enabling at the same time to identify on which phalanx the deviation has occurred.

Real-time slippage detection may also be applied in order to prevent the loss of the object. Each phalanx may locally detect a change in contact point, or a vibration caused by friction with the object. Sometimes these are intended effects, for example when reorienting the object in hand. But sometimes, it may report the unattended slippage of the object in hand during a firm grasp. If detected quickly enough, the system can compensate to prevent the loss of the object. The slippage detection may also be used for example to verify that the tube is well inserted inside the pump. After insertion, the robot could apply a moderated grip on the tube between two of its fingers and slightly pull the tube longitudinally. If the tube is well attached to the pump, the applied movement should encounter a resistance and the tactile sensor attached to the phalanges should detect a slippage, crossing a threshold to be defined. If not, then the tube moves with respect to the pump and is not well inserted.

Furthermore, the piezoelectric sensor may help to detect specific vibration patterns such as the click during the closing of the clip. To illustrate this, first closing clip experiments have been conducted with a Panda Robot's gripper equipped with a piezoelectric sensor (see Figure 6).

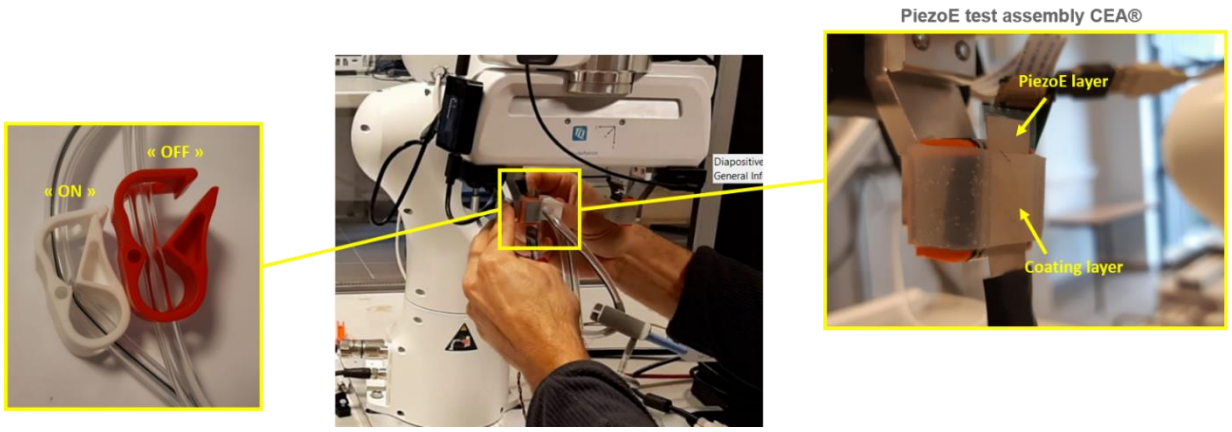

Fig. 6: Preliminary clip closing test on a piezoelectric sensor.

The spectrum of the vibrations measured by the sensor is specific and repeatable over those preliminary tests. We have verified that the motion of the hand during clip transportation did not yield a similar pattern. This means that we can use the spectrum of the vibrations in order to detect this specific event without confusing this with normal motion. This monitoring could be activated in specific phases of the manipulation to reduce any potential false positives.

Finally, using the force sensing measurements from the gripper and/or the robotic arm actuators, it is possible to detect some events such as the insertion of the needle in the bottle. When inserting the needle, first there is a resistance due to the contact with the bottle stopper. Then as the stopper is punctured, the resistance drops, and it may be possible to use force sensors to detect this particular pattern. If the needle hits a rigid part, the resistance to the motion will never drop. On the contrary, the robot should measure no significant force while the needle remains in the air.

Horizon 2020

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

4.2 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 uses the Digital Twin 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 for checks 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.

Visual verification uses the following approach: any time the pose of an object is estimated by the vision component of the system, it is compared to the belief of the system about where that object is supposed to be. This provides a level of self-awareness during the task execution. The Digital Twin based on its previous knowledge, the manipulation that was performed, and the physics simulator it contains compare the perceived pose and the estimated pose. The physical plausibility (e.g., collision, static stability, scene dynamics) of a given pose is also considered using the physics simulator.

Beside this straightforward verification, visual verification can go further and evaluate how well its prediction using an abstracted representation fits with the low-level sensor data.

Figure 7 illustrates a visual verification score: it compares the observed depth and normal images to their rendered counterparts under the estimated pose. We first detail the state-of-the-art methods for visual verification, and then details how they will be adapted to suit the project needs.

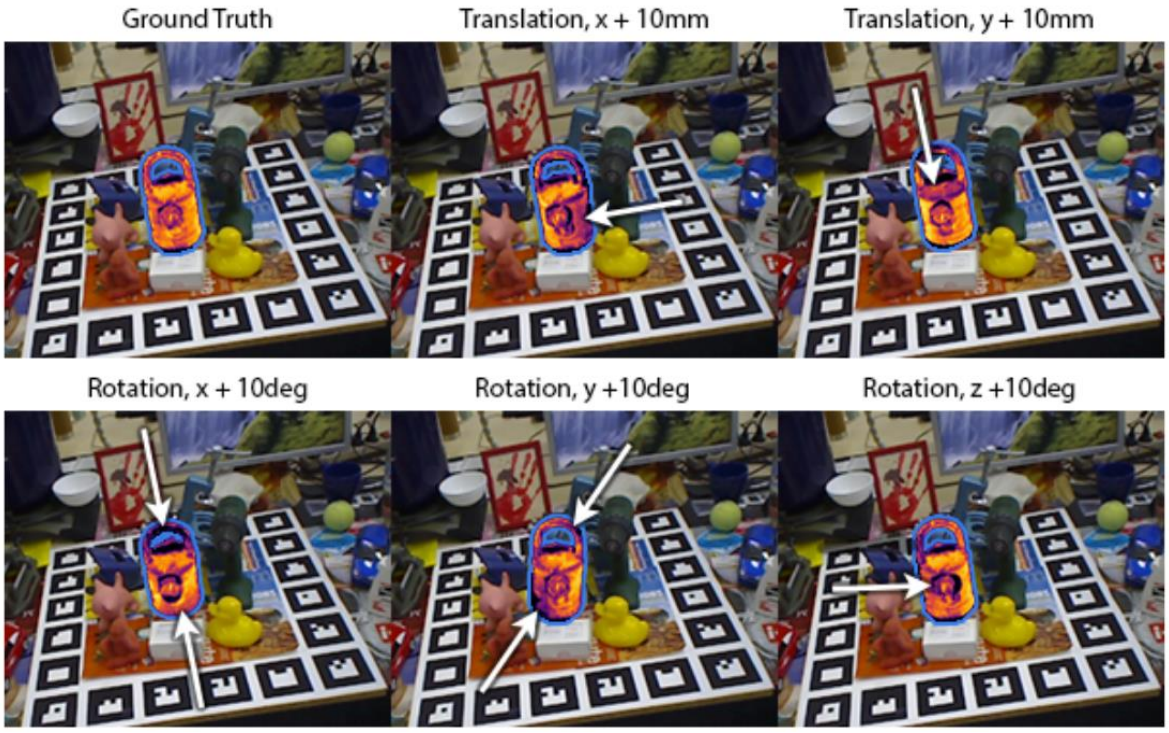

Fig. 7: Visualization of the verification score. The darker the color, the worse the alignment [1,2,4].

Horizon 2020

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

4.2.1 State-of-the-Art Visual Verification

In [1], we apply rendering-based verification to determine the visually best pose hypothesis among a set of physically plausible rest poses of the observed object. As illustrated in Figure 8, an initial rest pose is adapted by aligning the segmented object in the observation to the object rendered under this initial guess. The overall highest scoring guess is returned as pose estimate. Note that this approach assumes a single rigid object to statically rest on a planar surface.

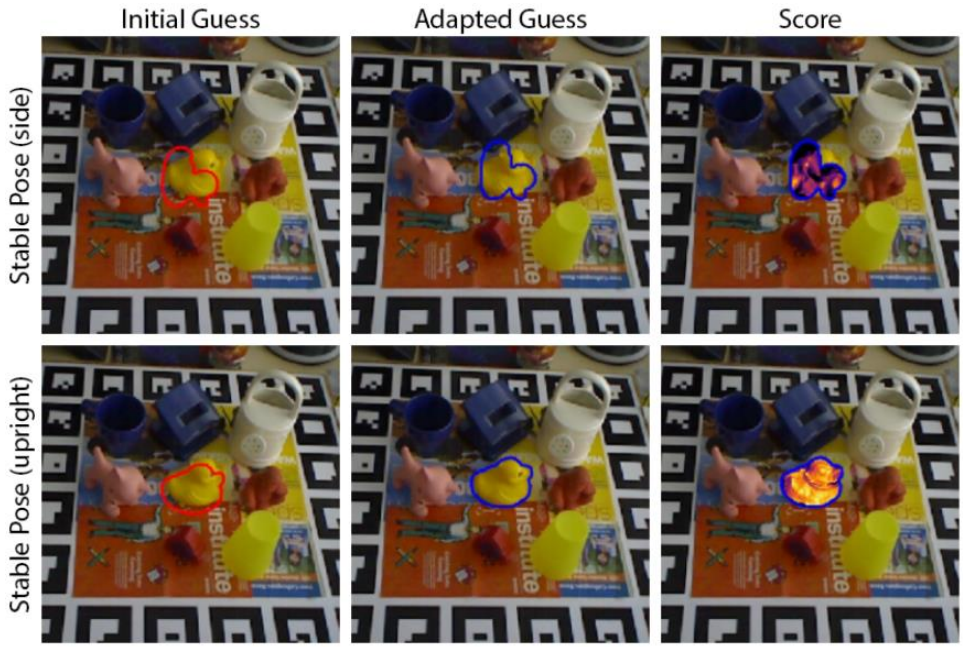

Fig. 8: Verification of physically plausible rest poses. The hypothesis with the object lying on its side (top) is visually less plausible (i.e., lower verification score) than the upright hypothesis (bottom row) [1].

This rendering-based verification is used in [2] to 1) supervise iterative refinement and 2) guide a regret minimizing hypothesis selection. Additionally, we alternate refinement and physics simulation steps to improve initialization and to avoid divergence in either substep. Considering multiple objects, we iteratively refine multiple object hypotheses and add the best hypothesis to the simulated scene used for the following object.

Finally, we proposed to consider point cloud-based refinement as a reinforcement learning task in [3]. By additionally rewarding physically plausible (non-intersecting, statically stable) scene configurations [4], implausible object poses may be resolved as shown in Figure 9. Again, rendering-based verification is used to supervise the iterative refinement process and to determine the visually most plausible (intermediary) pose, visualized in Figure 10.

Horizon 2020

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

Fig. 9: Initial pose (left) of the target object (gray) with intersecting points (red). After refinement (right), intersections are resolved, and all objects statically rest upon a support [4].

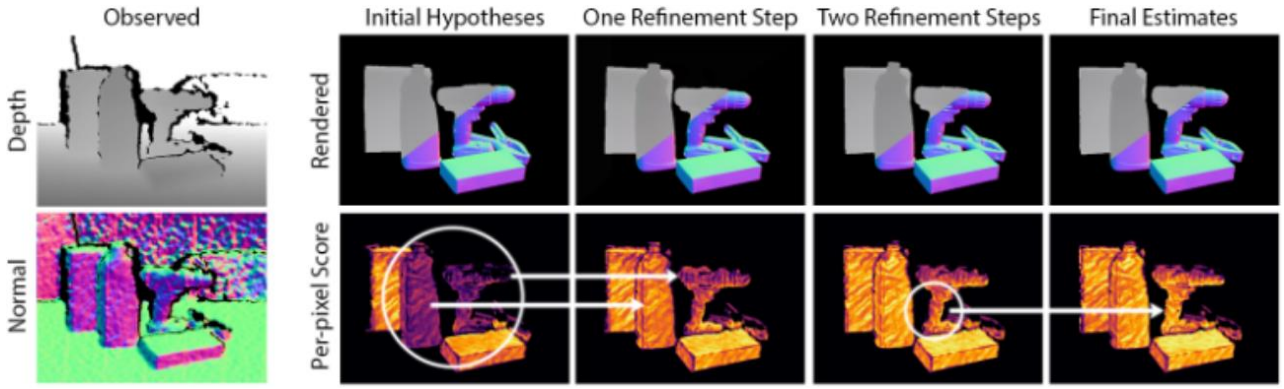

Fig. 10: Visualization of the verification score over multiple refinement iterations. The observation (left) is compared to the rendered objects with their estimated pose (right). The resulting score is visualised in the lower line (more yellow indicates better alignment).

4.2.2 Visual Verification in the context of the TraceBot project

As described in the previous section, state-of-the-art visual verification methods have been developed for non-transparent rigid objects. Transparent objects are very challenging for depth sensors, which generally deliver little valid data for those objects. This impacts the verification score described above as it relies on the observed surface normals. It further impacts the refinement steps described that also need depth observation.

The physical plausibility described in the previous section is also challenged by the objects considered in the TraceBot as they are not all rigid objects and include fluids.

These two aspects prevent a direct application of state-of-the-art methods in the context of the project. Building upon our previous work and addressing the gap in the visual domain, we will include more powerful rendering approaches, such as physics-based renderers [5] or ones exploiting ray tracing features (e.g., provided by the Unity game engine), and consider alternative representation such as [6] to reduce the domain gap and enable visual verification of refractive/reflective objects.

Both the color and depth modality degrade given refractive/reflective objects. Still, contours/edges may be extracted from color images. We will explore existing contour-based refinement methods to

Horizon 2020

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

substitute for depth-based ones and additionally propose to leverage an inverse rendering pipeline to solve for pose (instead of material). By translating both the real reference and the rendered image to the contour/edge domain, we expect to bridge the domain gap.

While CAD models are available for many objects in TraceBot, we do not know their material (e.g., defined by the reflectance properties using standard models such as BRDF - bidirectional reflectance distribution function) to render them realistically. However, this is necessary for accurate synthetic dataset generation and for rendering-based verification. We recorded views of the real object using a robotic arm on a known textured background and use those as target views in an inverse rendering pipeline [5] that optimizes for properties such as the index of refraction (IOR) or the color tint of the material, see also Figure 11. This enables an automatic estimation of the material parameters based on views of the object, making the approach easier to adapt for new objects and less dependent on expert knowledge.

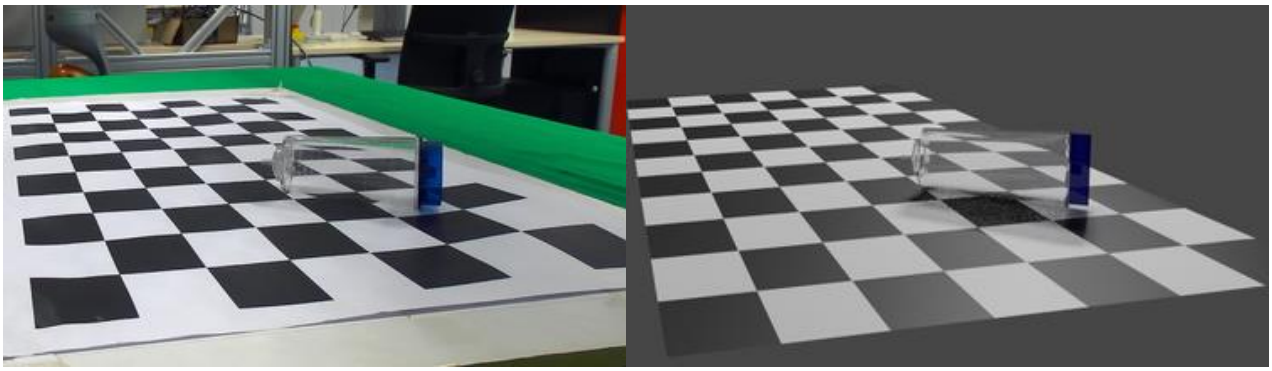

Fig. 11: Views of the canister are recorded using a robotic arm, simplifying manual annotation for ground-truth pose and segmentation (left). A material for the CAD model is manually created in Blender to recreate the refraction of the reference image in simulation (right). This step will be automated.

4.3 Functional Task Verification

An application of the knowledge infrastructure (i.e., NEEMs, ontology, query language, reasoning engine) presented in the section 3.3 is the functional verification of TraceBot processes. Functional verification consists foremost in computing the discrepancies between the expected and the actual effects of an action in order to deduce whether or not the equipment in the sterility testing cell (i.e., lab) operates properly. But it might also refer to checking whether or not an action was feasible in a certain context (pre-conditions) and that such action was performed successfully (post-conditions). For instance, how can one functionally verify that the canister is properly inserted into the drain tray, or that the needle is properly inserted into the bottle cap. For the canister insertion verification, a specific verification action is performed such as shaking the canister. If it does not fall then, one can infer that the canister was properly inserted. If starting the pump causes the circulation of liquid solutions through the tube, then this might be a hint to deduce that the needle insertion was successful and that the needle operates properly. A Bottle has no defectuous hole if the volume of the solution inside it does not unexpectedly change over time or if there is no leakage. The tube operates properly

Horizon 2020

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

if the solution circulates through it without problem. That is, the solution is pulled from the source bottle at an expected rate and filled into the destination bottle at the same expected rate. Otherwise, this might be considered as a signal that the tube is somewhere obstructed.

Recent preliminary works have been focusing on verifying the feasibility of actions depending on the context of the ongoing processes. The question is whether or not it is possible to perform a certain action within a given context. In order to achieve this, an initial ontology of action (i.e., preconditions and postconditions) was provided and the object ontology was extended with affordances (e.g., graspable, posable, posee, etc). In order to model the concepts of pre- and postconditions, we introduce the concepts of states and transitions. A state describes the actual world and can be represented by a set of predicates, known as fluents, and whose values change over time. For instance, Graspable(canister1) is true if the canister is not grasped yet and can be grasped, but becomes false either on one of the above conditions. Then, a state can be represented as set of fluents such as On(canister1, table1), Graspable(canister1) which means that the canister1 is on the table1 and can be grasped. Action preconditions are then represented as specific states in which the execution of the action is feasible. The concept of transition allows to model the effects of an action within a specific world state. The expected resulting state can be regarded as the postconditions of the action. Formally, a transition is a function that takes as parameters a given state and an action, then returns the resulting state after performing the actions. For instance, after opening a bottle, it should be visible, graspable and close, but after performing the action, the bottle should be at least open. Then, given a context such as the canister is detected and not grasped, the drain tray is empty, one could infer that the grasp of the canister is feasible as well as its insertion in the drain tray. After declaring the canister as non-detectable in the process context, our reasoning engine could infer that these actions are no more feasible. The Figure 12 below illustrates this feasibility verification.

Horizon 2020

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

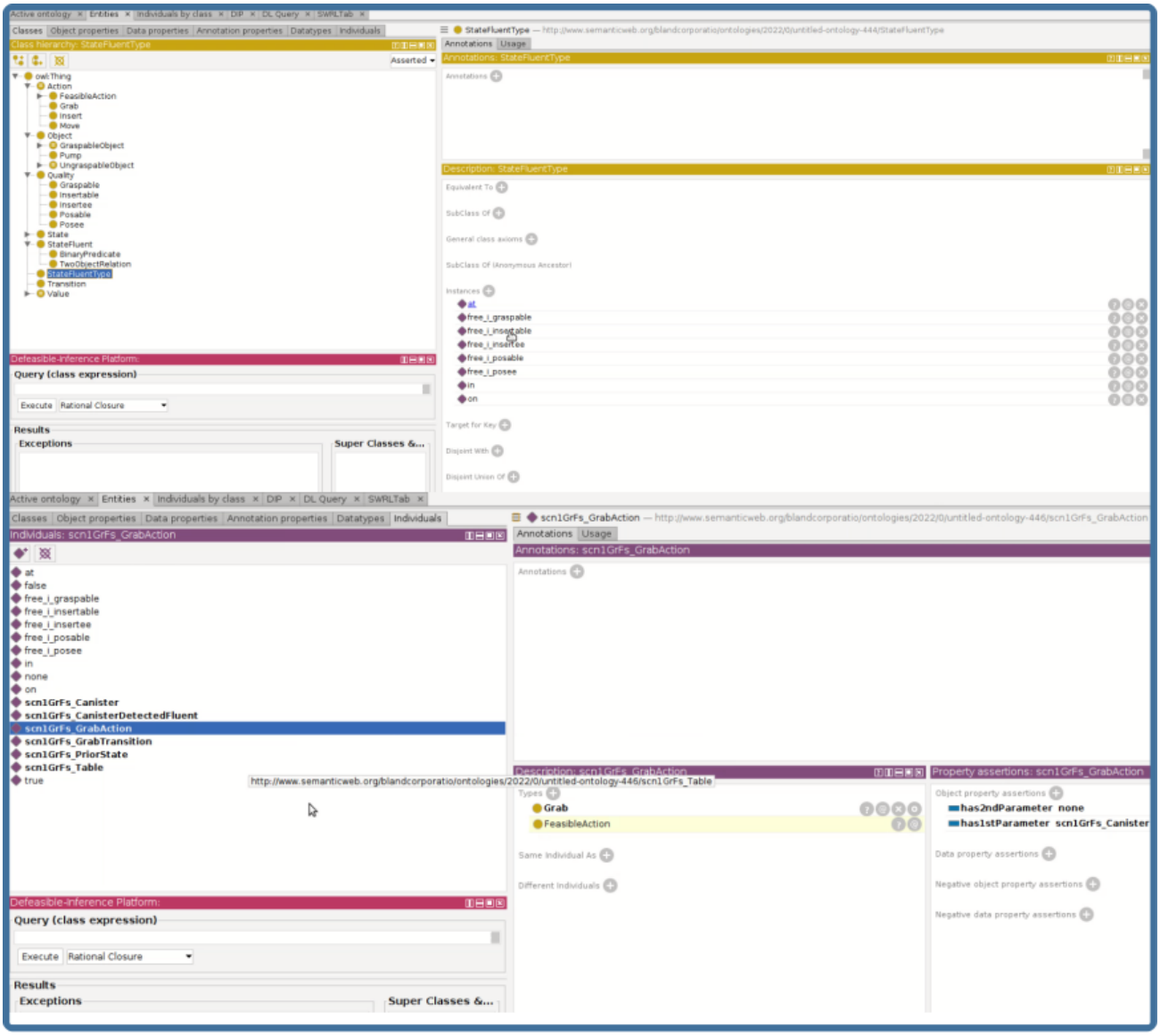

Fig. 12: Enriching ontology for action feasibility verification.

Horizon 2020

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

5 Example Task Verification

In this Section, we describe our implementation of a first visual verification for the canister pose estimation case.

5.1 Canister Pose Estimation

As commented in Section 4.2, the pose estimation of transparent objects is a challenging target that we are still working on and expect to report first results soon. As a contingency plan, we prepared a pipeline adapted to an opaque canister created by spraying out the original canister. This allows us to directly evaluate our previous work [1, 2, 3, 4] in the TraceBot pipeline and provide partners with working vision methods to investigate other downstream tasks without requiring extensive computational power.

We use plane pop-out as a simple object segmentation approach that requires no training. Point Pair Features (PPF) computed from the segmented point cloud and the known object model can be used to estimate the pose while only requiring minor preprocessing of the 3D model but no intensive training. We combine this with our rendering-based verification score to determine the best pose hypothesis, as illustrated in Figure 13.

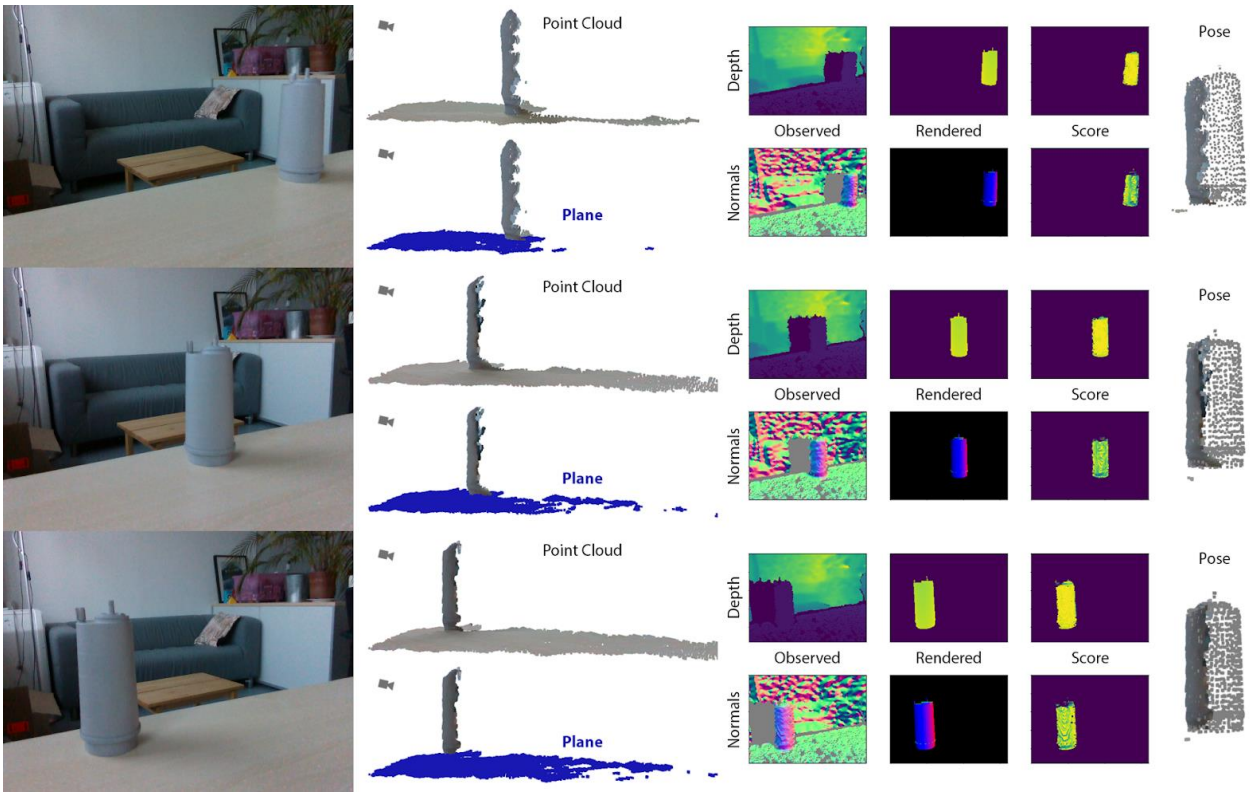

Fig. 13: A spray-painted canister is observed by a depth camera. The supporting plane is segmented, and the object pose is estimated using the remaining points. For each pose hypothesis, we compute a verification score based on the object rendered under the estimated pose. The highest scoring pose closely aligns the observation and the corresponding object model.

Horizon 2020

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

5.2 Visual Task Verification

In an experimentation illustrated on Figure 14, we compared the visual verification process for the spray-painted and the original transparent canister in isolation and in the tray (one condition per row). For each condition, three different object locations are considered, and we show for each the top-3 PPF hypothesis. Note that these results are without pose refinement.

For the isolated transparent canister, the sparse depth information may still be sufficient to yield graspable object poses. The condition of the spray-painted canister in the tray indicates the need for considering object interactions. Finally, the transparent canister in the tray is confused with the tray itself. Aligning the model to the dense depth information corresponding to the tray yields a higher score (i.e., is better supported by observational evidence) than aligning it to the sparse depth of the transparent canister. In addition to considering object interactions, this case motivates the use of further observational modalities for hypothesis generation and verification, such as color.

Fig. 14: The top-3 PPF hypotheses for three different views (left to right) and the corresponding per-pixel verification score (brighter color indicates better alignment). We compare results for (top to bottom) the spray-painted canister and the original transparent canister in isolation as well as both versions in the tray.

Horizon 2020

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

A first trial on the robot was executed on the Toyota HSR at TU Wien. The idea was to check, given the reasonable accuracy of object pose estimation, if the robot would be able to take and place the canister in the tray. The task was to grasp the canister from a table and transfer it to the tray on another table, see Figure 15. The visual methods are simplified und use a marker next to the tray location. The robot is not an industrial arm but rather a service robot with less accuracy than an industrial arm. However, it was possible to take the canister and place it in the tray with an accuracy of a few millimetres. This test is very useful, because we can now integrate the methods for object pose estimation and will be able to make tests in Vienna before making the methods available to the project partners. Certainly, TU Wien will also assist with integration at the other partner's sites, in particular with the gripper at partner CEA and the integration with partner Astech.

Fig. 15: Example manipulation using visual detection of the canister and the tray. Performed with the Toyota HSR robot at TU Wien.

Horizon 2020

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

6 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)

7 Conclusion

This document details to inner workings of the traceability framework developed for the TraceBot. On one side, Traceable Actions ensure the correct logging of the data generated by the process, both symbolic and subsymbolic, and NEEMs enable efficient querying of that data after the process ended. In combination, TraceBot processes can generate a useful audit trail that can provide information relevant to successes and failures and a structured way to search through them.

On the other side, a set of verification actions have been designed and will be implemented throughout the project relying on different modalities. These verifications actions serve as checks for the correct execution of the process and, because they are logged in the same way as any other action, guarantee that the final audit will contain all information relevant for verification.

Those two aspects help us achieve a more meaningful level of traceability and accountability for automated workflow even in highly regulated environments such a sterility testing, the main use case considered in TraceBot.

In Table 1 we presented a first overview of the process sequence and the involved actions, challenges and verification actions. Future work is to go into more detail for every step in this process and propose detailed approaches on how to tackle the goal of verifying every robot action.

Horizon 2020

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

8 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] Nimier-David, M., Vicini, D., Zeltner, T., & Jakob, W. (2019). Mitsuba 2: A retargetable forward and inverse renderer. ACM Transactions on Graphics (TOG), 38(6), 1-17.

[6] Chen, G., Han, K., & Wong, K. Y. K. (2018). Tom-net: Learning transparent object matting from a single image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 9233-9241).

[7] Beßler, D., Porzel R., Pomarlan, M. , Vyas, A., Höffner, S., Beetz, M., Malaka, R., & Bateman, J. (2021) Foundations of the Socio-physical Model of Activities (SOMA) for Autonomous Robotic Agents. In Formal Ontology in Information Systems - Proceedings of the 12th International Conference, FOIS 2021, Bozen-Bolzano, Italy, September 13-16, 2021, IOS Press.

[8] Tenorth, M., & Beetz, M. (2013). KnowRob: A knowledge processing infrastructure for cognition-enabled robots. The International Journal of Robotics Research, 32(5), 566-590.

[9] Beetz, M. (2022). Definition of the conceptual and reasoning framework and semantic models. Deliverable 5.1 of the EU-funded TraceBot Project (grant agreement No 101017089).

[10] Rosidi, S., & Gordon, B. (2021). ROS Middleware mock-up. Deliverable 6.1 of the EU-funded TraceBot Project (grant agreement No 101017089).

[11] Gordon, B., & Rosidi S. (2022). System architecture. Deliverable 1.1 of the EU-funded TraceBot Project (grant agreement No 101017089).

[12] Cichon, T., & Coulon, C.-H. (2022). Verification plan. Deliverable 1.3 of the EU-funded TraceBot Project (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)