Ref. Ares(2023)8898178 - 29/12/2023

OpenDR

—

Open Deep Learning Toolkit for Robotics

Project Start Date: 01.01.2020
Duration: 48 months
Lead contractor: Aristotle University of Thessaloniki

Deliverable D7.5: Integration and experimental demonstration of the OpenDR toolkit in the specific use cases

Date of delivery: 29 Dec 2023

Contributing Partners: AUTH, TAU, AU, TUD, ALU-FR, CYB, PAL, AGI

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449.

Title	D7.5: Integration and experimental demonstration of the OpenDR toolkit in the specific use cases
Project	**OpenDR** (ICT-10-2019-2020 RIA)
Nature	Demonstrator
Dissemination Level:	**PU**blic
Authors	Niclas Vödisch (ALU-FR), Abhinav Valada (ALU-FR), Akif Ekrekli (TAU), Mikael Petäjä (TAU), Roel Pieters (TAU), Anton Muravev (TAU), Moncef Gabbouj (TAU), Kakaletsis Efstratios (AUTH), Symeonidis Charalampos (AUTH), Tzelepi Maria (AUTH), Nousi Paraskevi (AUTH), Manousis Theodoros (AUTH), Nikolaidis Nikolaos (AUTH), Tefas Anastasios (AUTH), Babis Emmanouil (AUTH), Tsampazis Konstantinos (AUTH), Kirtas Emmanouil (AUTH), Avramelou Loukia (AUTH), Spanos Dimitrios (AUTH), Katsikas Dimitrios (AUTH), Moustakidis Vasileios (AUTH), Nikolaos Passalis (AUTH), Alea Scovill (AGI), Nicolai Knudsen (AGI), Halil Ibrahim Ugurlu (AU), Amir Ramezani Dooraki (AU), Erdal Kayacan (AU), Alexandros Iosifidis (AU), Thomas Peyrucain (PAL), Gizem Bozdemir (PAL), Bas van der Heijden (TUD), Jelle Luijkx (TUD), Jens Kober (TUD), Robert Babuska (TUD), Olivier Michel (CYB)
Lead Beneficiary	PAL (PAL Robotics)
WP	7
Doc ID:	OPENDR D7.5.pdf

Document History

Version	Date	Reason of change
v1.0	09/10/2023	Deliverable structure template ready
v2.0	25/11/2023	Initial Draft ready
v3.0	27/11/2023	First version for review
v4.0	29/12/2023	Final version to be submitted

Executive Summary

Deliverable D7.5 marks a significant achievement within the OpenDR project, focusing on the successful integration and experimental validation of the OpenDR toolkit across three distinct domains: Agriculture, Healthcare, and Agile Production. This executive summary provides an overview of the integrated OpenDR tools and the compelling evidence, including photos, that illustrate the tangible benefits derived from these integrations. The tools required for all use cases have been successfully integrated in the OpenDR toolkit. Each use case has integrated and evaluated the OpenDR toolkit based on the use case specifications from D2.3, with the objective of evaluating the usefulness and efficiency of the toolkit. In addition integration results and feedback are provided.

1 Introduction

OpenDR aims at developing an open, non-proprietary modular toolkit that can be easily used by robotics companies and research institutions to efficiently develop, evaluate and deploy AI and cognition technologies to robotics applications. As industries evolve, the demand for intelligent and versatile robotic solutions becomes increasingly apparent. The OpenDR toolkit serves as a testament to the project's commitment to addressing these evolving needs.

This deliverable report signifies a momentous stride in the successful integration and experimental validation of the OpenDR toolkit within three distinct domains: Agriculture, Healthcare, and Agile Production. The introduction of the OpenDR toolkit into these specific domains not only exemplifies the adaptability of robotics but also signifies a crucial step towards realizing the practical applications of advanced technologies in real-world scenarios. This introduction sets the stage for an exploration of the integrated tools, the rigorous evaluation processes undertaken, and the tangible benefits observed across the diverse landscapes of these three domains.

This deliverable (D7.5) reflects the effort of the integration and experimental evaluations of the OpenDR toolkits to robot architectures for the 3 specific use cases; Healthcare, Agriculture and Agile Production.

2 Integration and experimental evaluation for Agricultural Robotics Scenario

The tools integrated in the agriculture use case have focused on the agriculture tasks, detection of crop plants, increasing the safety, scene understanding, plant row guidance system, realistic simulations and simulation of path planning methods to avoid obstacles using ground robot and drone. The tools have been integrated on a Nvidia Xavier or a Nvidia TX2. All demonstration links have been recorded in a field in either real conditions or close to real conditions.

2.1 Detection of crop plants in rows in field

The integration of the Crop & Weed tool has been integrated in two systems - edge and cloud.

The edge-based system acquires, then analyzes the image on the robot's Nvidia Xavier GPU. The metadata is sent to the cloud where it is stored, waiting for retrieval. Some images are also uploaded so the user and AGI can verify that the tool is working properly.

The cloud-based system acquires the image on the robot, sends the images in a rosbag to the cloud where it is extracted and stored. The images are automatically put in queue to be analyzed by the Crop & Weed tool. After the images are analyzed, the metadata is stored, waiting for retrieval.

For both systems, after the metadata is stored, the user can view the heat maps through the Robotti portal, see Figure 2.1.1. The locations of where images were acquired are overlaid onto Google maps, where the user can click and view the images, allowing them to learn more about their field. The images can be enhanced by the tool. The heat maps allow the user to spatially see the count or coverage of a weed or crop. The navigation to the heat maps and UI was improved in winter 2023.

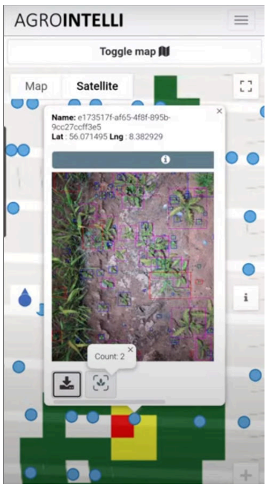


Figure 2.1.1: Example of the Crop&Weed tool being used by a farmer.

The figure is a screenshot of the Robotti portal. The top layer shows the image processed by the Crop & Weed tool. The layer under the image shows a heat map of the thistle count for the chosen field. Part of the demonstration video (first link below).

Currently, the cloud-based system is available on commercial Robottis that have the CropEye system (1-4 downward facing cameras with Xavier GPU). The cloud-based system has the advantage that AGI receives more images that can be incorporated into the tool and AGI can easily verify that the tool is performing satisfactorily. The disadvantage is the data requirements are considerable and the robots often work in areas where the bandwidth of the mobile connection is low. However, we don't explicitly trust the tool and the cloud-based system allows us to monitor and update the tool centrally.

Ultimately, the plan is for AGI to transition over to offering the edge-based system to farmers instead of the cloud-based system. The advantage of the edge-based system is the data requirements are significantly lower, reducing the running costs of the robot (both mobile data use and AWS cloud costs). The disadvantage is that there will be fewer images collected to update and verify the tool's performance. If there is a problem (e.g. cannot find maize because it is stressed and looks different than the model's dataset), then if only 5 or 10 images are recorded per ha, it may not be enough to correct the issue.

The Crop&Weed tool (custom YOLOv5) has been integrated into the OpenDR platform. Part of the Crop&Weed tool's dataset has been published and a smaller version has been integrated into the OpenDR platform, available for everyone to use. Because the completed dataset and tool is proprietary, the full model has been kept confidential. This is a demonstration of the full Crop&Weed model integrated into OpenDR:

https://www.dropbox.com/scl/fi/w4wa7btbzzd2nbfwhukc7/Opendr_custom_model_works.mp4?rlkey=kdhr58aww3v0ar33ad9w8el5h&dl=0

Demonstration of the Crop&Weed tool on AgroIntelli infrastructure:

Agrointelli Educational - Doing Virtual Field Walk via Mobile Phone while Robotti nurse berry bushes - YouTube

https://youtu.be/TnCB3aTEvCg?si=jg6YnTkNNzAxmIhW

2.2 Increase safety through object detection and tracking of humans in the field

SSD, single shot detection, object detection tool and YOLOv5 object detection tool (humans and tractors) have been integrated into Robotti using the Docker image. YOLOv5 (humans and tractors) has been integrated into Robotti's infrastructure, using Robotti's front and rear cameras and Nvidia Jetson GPU.

AGI will continue the integration and integrate the analyzed images into Robotti's remote supervision page on the Robotti portal, so they can be shown automatically to users when viewing the live streaming. In addition, the tools will trigger notifications to farmers and increase Robotti's autonomy by reducing the number of events requiring the user to go to the field to restart the robot. This is expected to be completed by winter 2024.

All integration issues with the Docker image have been resolved with the help of the OpenDR consortium.

Demonstrations can be viewed here:

SSD object detection tool:
https://www.dropbox.com/s/9cf0yd0yhppwzbf/opendr%20demo%20human%20detection1.mp4?dl=0

https://www.dropbox.com/s/7uy4pk4b9rw5w3f/Opendr_test_3.mp4?dl=0

YOLOv5:
https://www.dropbox.com/scl/fi/w1m0urhabr99sn3d785aw/Screen_Recording_20230223_142153_Chrome.mp4?rlkey=ut7ql5ej9k9fuazfoh5p8jx6p&dl=0

https://www.dropbox.com/s/rmrfzbilysary5u/Opendr_yolov5s_test_1.mp4?dl=0

(The black 'loading' circle is because the treeline beside the road reduces the bandwidth of the mobile signal. The delay is not due to tool performance.)

2.2.1 Refining object detection for agriculture-related vehicles

To enhance safety via object detection in agricultural fields, we fine tuned a pre-trained YOLOv5 model specifically for vehicles that often appear in agricultural fields. To this end, we considered several different options, as outlined below, including introducing a new class, "tractor"", and subsequently trained YOLOv5 on an appropriately constructed dataset. This additional training aims to enable the detection of tractors, which are vital in the agricultural domain. A detailed analysis of this procedure is provided below.

The used dataset was sourced from Kaggle[1] and comprises both truck and tractor images, along with their corresponding annotations. Specifically, it contains 132 images of tractors and 194 images of trucks. For evaluation, we partitioned the data into approximately 80% for training and 20% for testing.

[1] https://www.kaggle.com/datasets/dataclusterlabs/construction-vehicle-images

To enhance the performance of YOLOv5 for the aforementioned tasks, we first froze the backbone model responsible for extracting high-level features that emphasize the most prominent characteristics and patterns in the input data and trained only the head model, which interprets the feature maps generated by the backbone to make final predictions or decisions.

Table 2.2.1: Training and validation curves for a model exclusively trained on truck images.

Model	**mAP @ 50** *in trucks/tractors*	**mAP @ 50 - 95** *in trucks/tractors*	**mAP @ 50** *in COCO*	**mAP @ 50 - 95** *in COCO*
Default	0.342	0.248	0.713	0.475
Finetuned (trucks)	0.769	0.582	0.156	0.095
Finetuned (trucks+COCO)	0.740	0.559	0.398	0.288

The first option that we evaluated was to train the model exclusively on truck images that could appear in agricultural settings, while also treating tractor images as trucks, assuming they share similar feature representations. This approach provides us with 260 training samples and 66 evaluation samples. We train the YOLOv5s model over 200 epochs, using mAP50 and mAP50-95 as evaluation metrics. In Figure 2.2.1, we provide training and validation curves, while in Table 2.2.1 we report the performance of the models. The default model represents YOLOv5s before any training. The aforementioned finetuning process is abbreviated as "Finetuned (trucks)" in Table 2.2.1. Based on these results, it's evident that the model has significantly improved in detecting trucks and tractors. However, when tested on the COCO128 dataset, our model appears to struggle with recognizing other classes, a phenomenon commonly known as catastrophic forgetting.

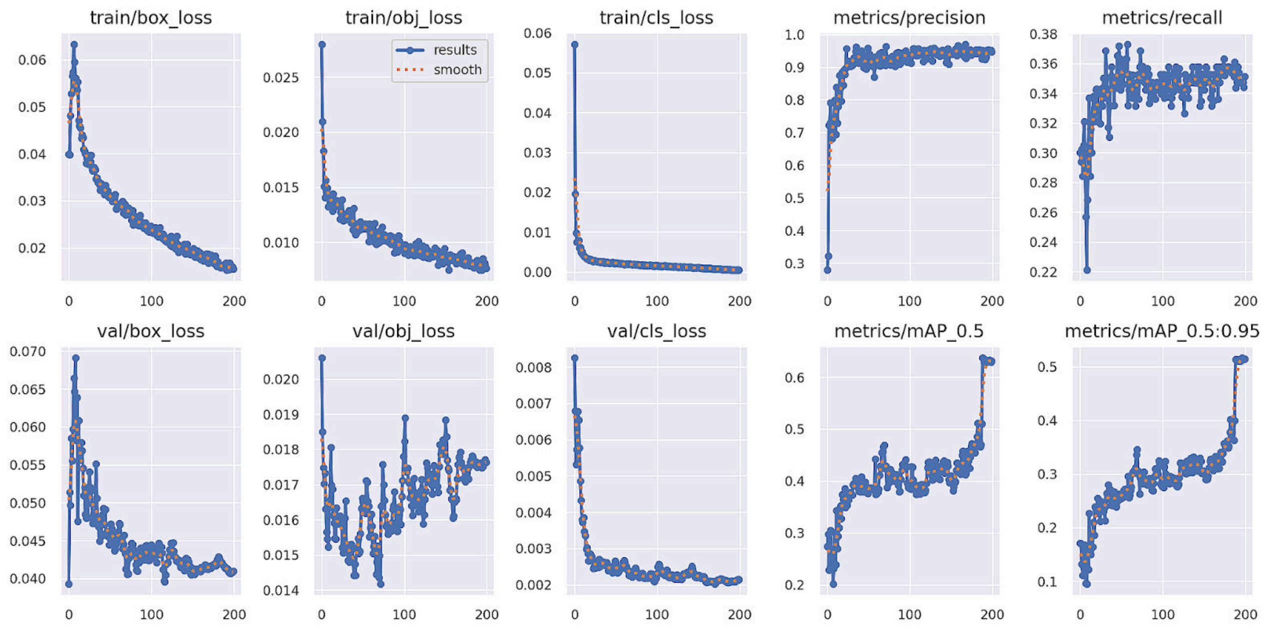


Figure 2.2.1: Training and validation curves for a model exclusively trained on truck images and tested on trucks.

To address the issue of catastrophic forgetting we choose to augment the YOLOv5s with images that include not only the new domain, but also the original one (as provided by the COCO dataset), in order to mitigate these issues and provide potentially more robust models. To this end, we also sample training images from the COCO dataset. The training/validation curves are shown in Figure 2.2.2. The mAP is also reported in Table 2.2.1, with the proposed method abbreviated as ``Finetuned (trucks+COCO)". The new model achieves almost the same performance as the previous one in the agricultural domain, while performing better in the original COCO domain. Please note that the aim of this experiment is not to retain the original COCO knowledge, since most part of this is probably irrelevant to the task at hand, but rather to examine if we can retain part of this knowledge without harming the detection performance for the task at hand.

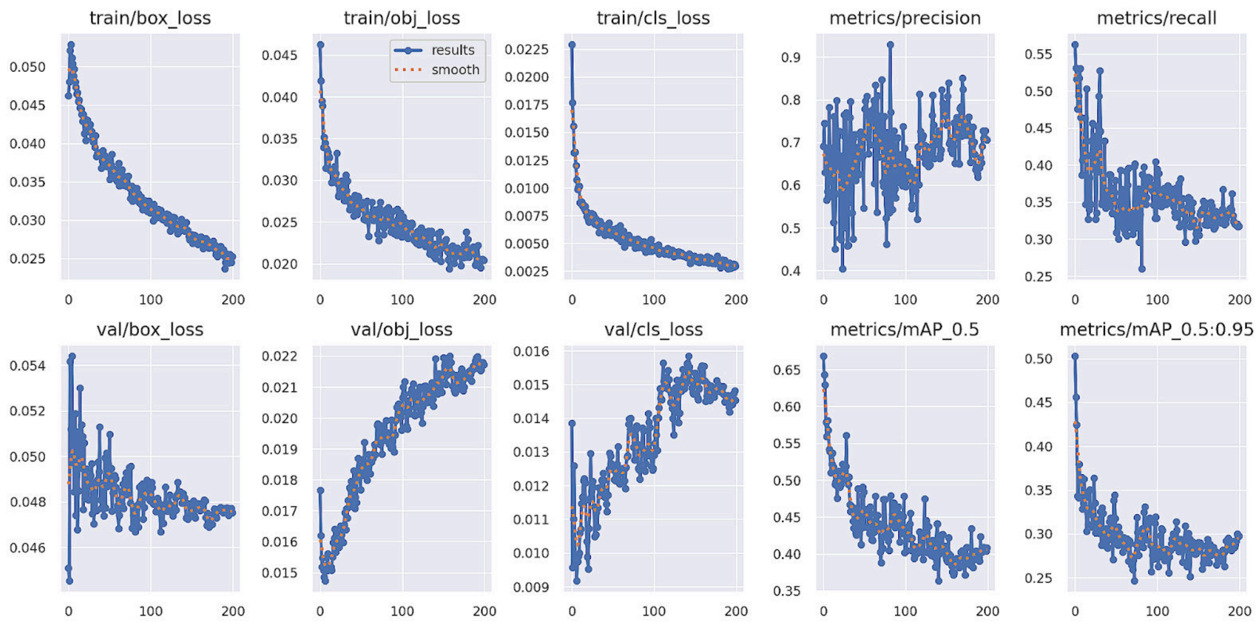


Figure 2.2.2: Training and validation curves for a model trained-tested both on truck images and COCO images.

Finally, we also finetuned a model in which tractors are added as a new class. We follow the same approach as before and we also include an equal number of COCO images in the training to mitigate forgetting issues. However, as shown in Figure 2.2.3, this approach works worse compared to the previous one. However, such an approach might be useful in cases where we need to know the exact type of detected vehicle. In Figure 2.2.4, we provide some examples of tractors along with the corresponding model predictions.

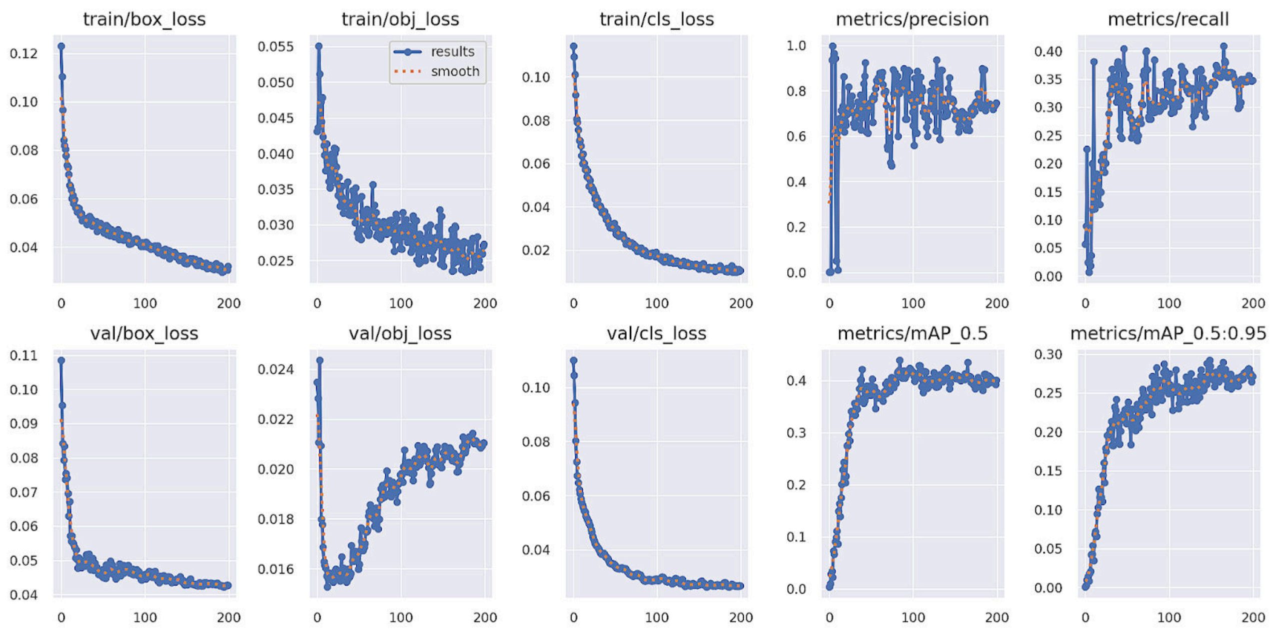


Figure 2.2.3: Training and validation curves for a model trained-tested both on truck images and COCO images (a separate tractor class is added).

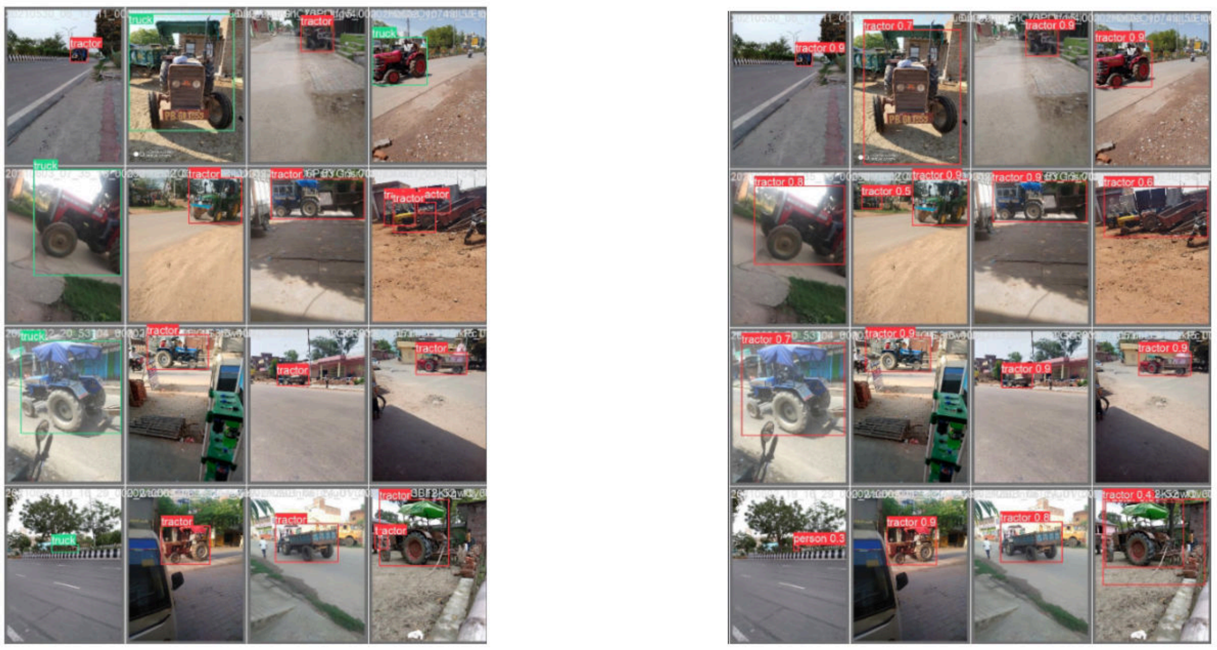


Figure 2.2.4: Examples of annotation and model predictions for indicative tractor images, as provided by the original dataset.

2.3 Scene understanding for agriculture fields

For scene understanding of agriculture fields, we perform dense semantic segmentation from the onboard cameras of the employed robot. Since a vast majority of traditional semantic segmentation approaches require a large set of precisely annotated images, we leverage few-shot training techniques to reduce the cost of initial image labelling. In particular, we utilize SPINO, recently proposed by ALU-FR. At the time of submission, this work is still under review at a major robotics conference. In the following, we provide a brief introduction to SPINO. Note that for this particular use case, we only employed the semantic segmentation capabilities without performing panoptic segmentation.

Markus Käppeler, Kürsat Petek, Niclas Vödisch, Wolfram Burgard, and Abhinav Valada. "Few-Shot Panoptic Segmentation With Foundation Models", *arXiv. Preprint: arXiv:2309.10726*, 2023.

Most contemporary panoptic segmentation techniques rely on supervised learning, necessitating the availability of extensively annotated panoptic maps, a process that is both costly and time-consuming. To address this challenge, an alternative approach relies on utilizing pseudo-annotated data for supervised training of larger models.

Our proposed SPINO aims to produce highly accurate pseudo-labeled data for state-of-the-art panoptic segmentation models, requiring only a limited quantity of actual annotated data. This is achieved through the application of transfer learning on foundational models, such as DINOv2, which enables the extraction of rich semantic information from any input image. Moreover, the acquired semantic information can be leveraged to generate task-specific outputs, such as semantic maps and boundary maps, with minimal training on top of the pre-trained foundation model. This training strategy significantly mitigates the need for extensive human annotation.

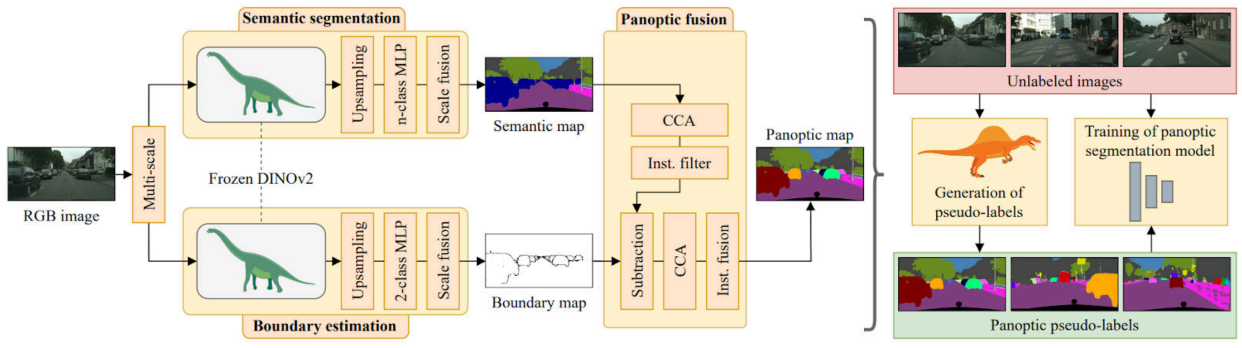


Figure 2.3.1: SPINO architecture.

SPINO applies the aforementioned strategy to derive boundary and semantic maps from an RGB image. Subsequently, through the process of panoptic fusion, it generates pseudo-label panoptic outputs. However, it is essential to note that rather than utilizing these pseudo-labeled data solely for prediction, it is advisable to employ them for training a state-of-the-art panoptic segmentation model. This approach is preferred due to concerns regarding the inefficiency in inference time associated with the direct use of pseudo-labeled data as predictions. The visual overview of the method can be seen in Figure 2.3.1..

Below (Figure 2.3.2), we show qualitative results of the predictions by SPINO for scene understanding of fields.

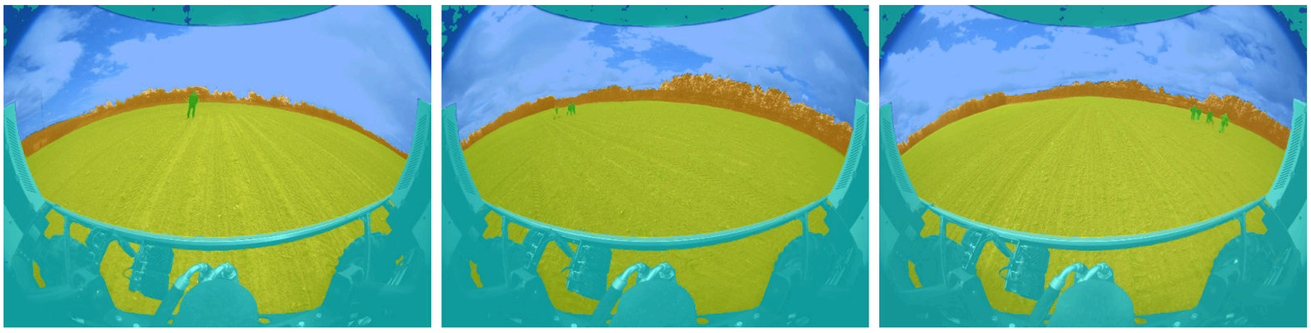


Figure 2.3.2: Results of SPINO in the agriculture use case analyzing images taken in real farming conditions.

2.4 Plant row guidance system

The plant row guidance system has been integrated into Robotti's infrastructure, see Figure 2.4.1.. The plant row guidance system acquires an image, analyzes it while finding the plant stem emergence zones and the crop. The crop row is found by counting the pixels along the x axis for all of the plant stem emergence zones that are found for the crop and the pixel column that has the highest count is the crop row. The cross track error is calculated (see D4.4 for calculation details) and published to the robot's 'nudge' topic, see figure 2.4.1. The nudge allows the robot to change position to the left or right of the planned line. The robot is then nudged based on the actual position of the crop and then the robot moves into the corrected position. The RTK GPS position data is sent and recorded to the cloud. A map is created so that the robot can follow in sequential operations. When turning, the robot uses the RTK GPS, as there are no visual clues as to where it should drive. The user can use the recorded path in future operations.

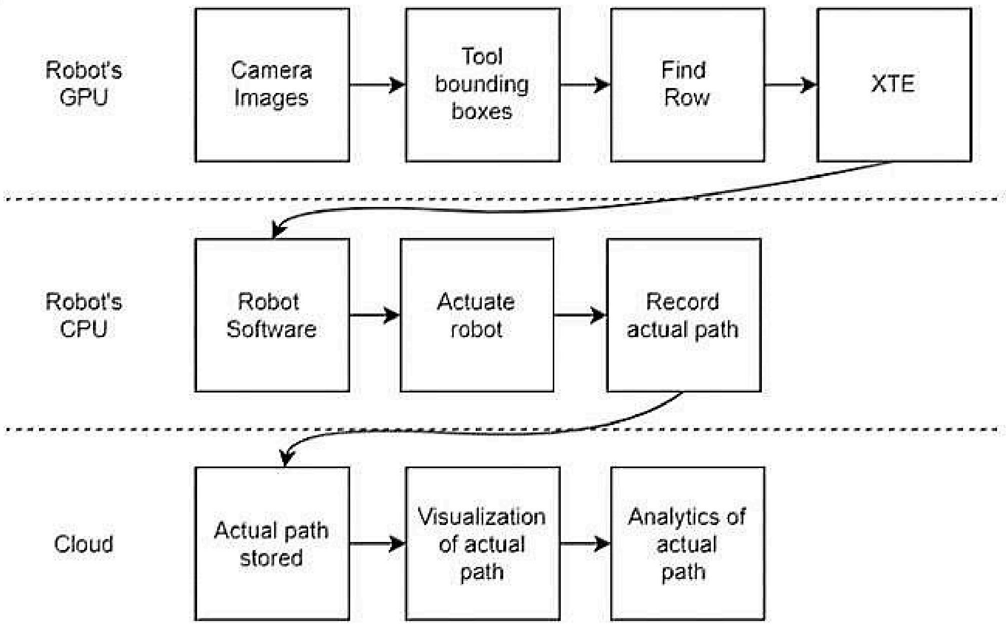


Figure 2.4.1: Infrastructure of the row guidance system

The demonstration can be viewed here:

open_dr_row_guidance_tmp_3_time_14_13.mp4

Explanation of the video: First the Robotti drives in automatic mode, then it slows (0:20 - 0:28). This is when the row guidance system is being activated, the lights are turned on at 0:20. In the top left, the crop images are shown when they are being recorded (starts at 0:28). In the top center, this is the video from the person recording the demonstration. In the top right, this is the video of the rear camera, also showing the person recording. Just below the videos, the path and location of where the robot drove is shown. In the bottom, the graph shows the cross track error (XTE) plus the vision estimation (row guidance) of the robot, see Figure 2.4.3x (the orange line in the graph below is the same as the red line in the video). The moving dotted line shows where in the time progression the video is on the graph. The red line in the graph is the nudge calculated from the row guidance system. The blue line is robot cross track error plus the row guidance estimation of the cross track error.

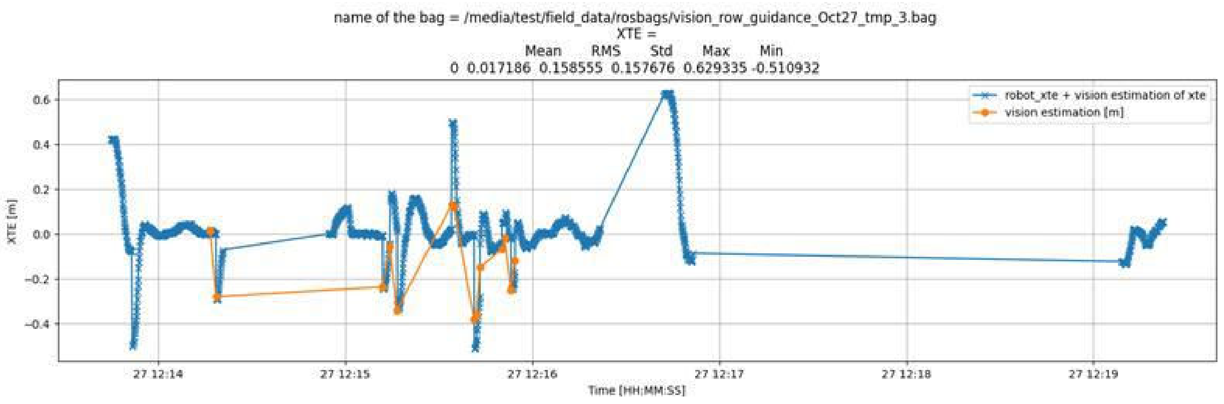

```
name of the bag = /media/test/field_data/rosbags/vision_row_guidance_Oct27_tmp_3.bag
XTE =
       Mean       RMS       Std       Max       Min
0  0.017186  0.158555  0.157676  0.629335 -0.510932
```

Figure 2.4.3: The blue line shows the robot cross track error plus the vision estimation (row guidance). The orange line shows the vision estimation (row guidance), also called the nudge.

2.5 Development efficiency by getting more realistic simulations

The simulation has been used to evaluate safety events. The simulation models were refined to provide more realistic results. In particular, the Robotti model and embedded sensors were improved, based on their corresponding datasheets. The Robotti camera, Robotti module and Robotti module wheels were implemented as separate PROTO files in Webots. In addition, an agricultural warehouse PROTO model was implemented to allow for more realistic scenarios. We also reused a number of farm environment PROTO files developed previously, such as Tractor, Silo, Ditch, Barn, various farm animals, etc. Finally, a special simulation world file was designed and implemented with a realistic scenario where a Robotti robot is detecting humans in a field environment.

The demonstration can be viewed here:

https://webots.cloud/AcNlzA6

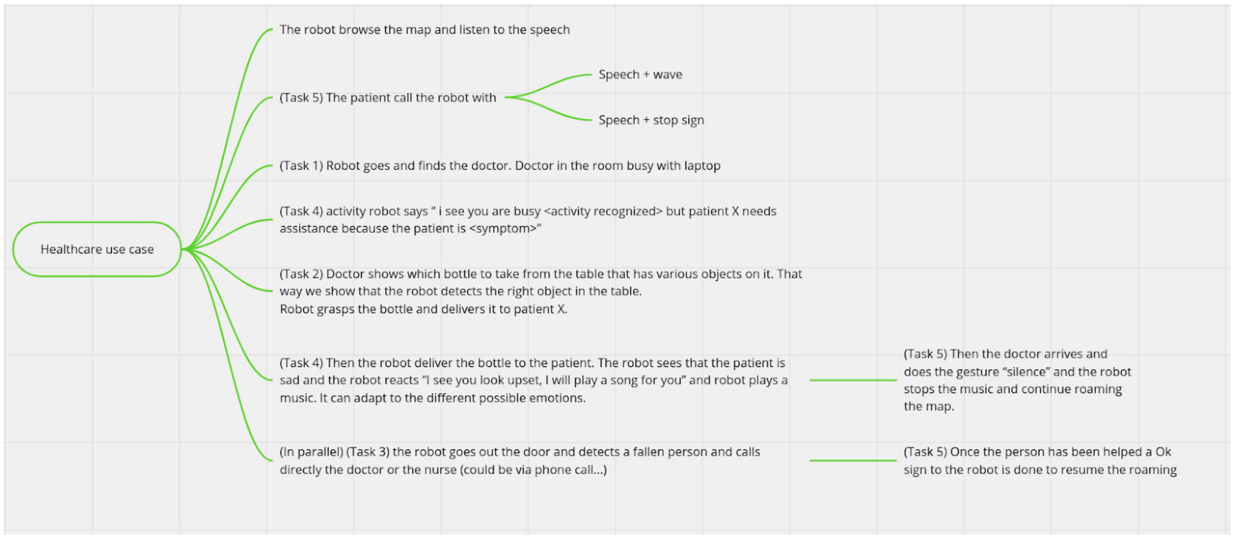
2.6 Simulation of path planning methods to avoid obstacles using ground robot and drone

In addition to the previously reported showcase of end-to-end planning UAV guiding agricultural UGV, an autonomous dataset generation approach for an agricultural environment is simulated in the Webots simulator. The simulated agricultural environment includes two active vehicles. The first is an unmanned ground vehicle (UGV) called Robotti, and the second one is an unmanned aerial vehicle (UAV) called Mavicpro. The dataset generation tool and a sample collected dataset are integrated with the toolkit.

3 Integration and experimental evaluation for Healthcare Robotics Scenario

The general setup for experiments in the healthcare scenario includes one TIAGo robot that is equipped with a RGBD for visual perception and a microphone for speech recognition. Computation is performed on different platform to run all the nodes at the same time: Jetson NX, Linux laptop (NVIDIA GeForce GTX 1650 + 2 * 8GB DDR4 RAM 3200 MHz), PC (NVIDIA GeForce RTX 3080 Ti + 4 * 8 GB DDR4 RAM 2400 MHz) and all robot (TIAGo robot) communication and control utilizes ROS and PAL software.

The updated scenario is provided below to highlight the different tools used in the healthcare use-case:


Healthcare use case
The robot browse the map and listen to the speech
(Task 5) The patient call the robot with
Speech + wave
Speech + stop sign
(Task 1) Robot goes and finds the doctor. Doctor in the room busy with laptop
(Task 4) activity robot says " i see you are busy <activity recognized> but patient X needs assistance because the patient is <symptom>"
(Task 2) Doctor shows which bottle to take from the table that has various objects on it. That way we show that the robot detects the right object in the table. Robot grasps the bottle and delivers it to patient X.
(Task 4) Then the robot deliver the bottle to the patient. The robot sees that the patient is sad and the robot reacts "I see you look upset, I will play a song for you" and robot plays a music. It can adapt to the different possible emotions.
(Task 5) Then the doctor arrives and does the gesture "silence" and the robot stops the music and continue roaming the map.
(In parallel) (Task 3) the robot goes out the door and detects a fallen person and calls directly the doctor or the nurse (could be via phone call...)
(Task 5) Once the person has been helped a Ok sign to the robot is done to resume the roaming

3.1 (Task 1) Person finding

The integration of face detection and face recognition tools is integrated at different levels of the scenario.

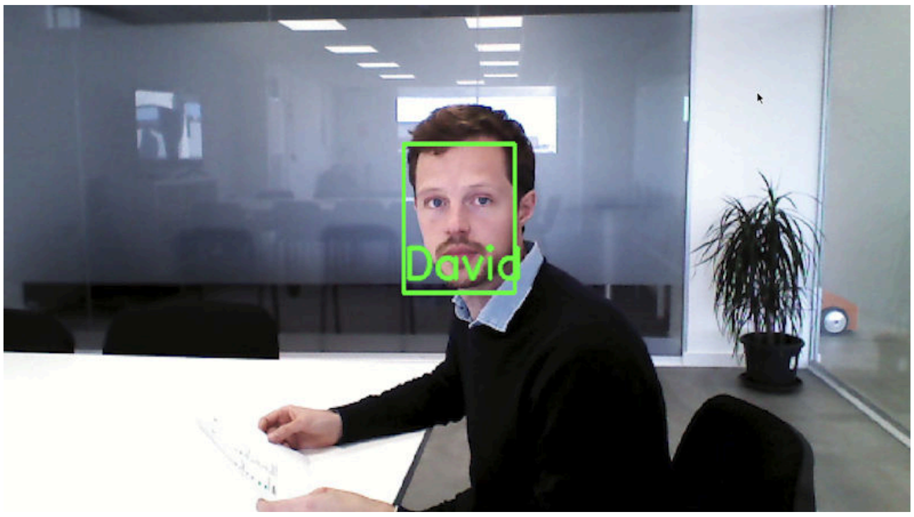


Figure 3.1.1. Detection and recognition of a person

Upon initiating interaction, the robot recognises the person as in **Figure 3.1.1** (if the person is not found in the database the robot will ask for the patient's name in order to add to the database **Figure 3.1.2**) for subsequent use within the use case. For instance, upon reaching the doctor, the robot states the patient's name to facilitate the doctor's awareness of the prescribed medicine. Additionally, upon entering the doctor's office, the robot endeavors to identify the doctor.

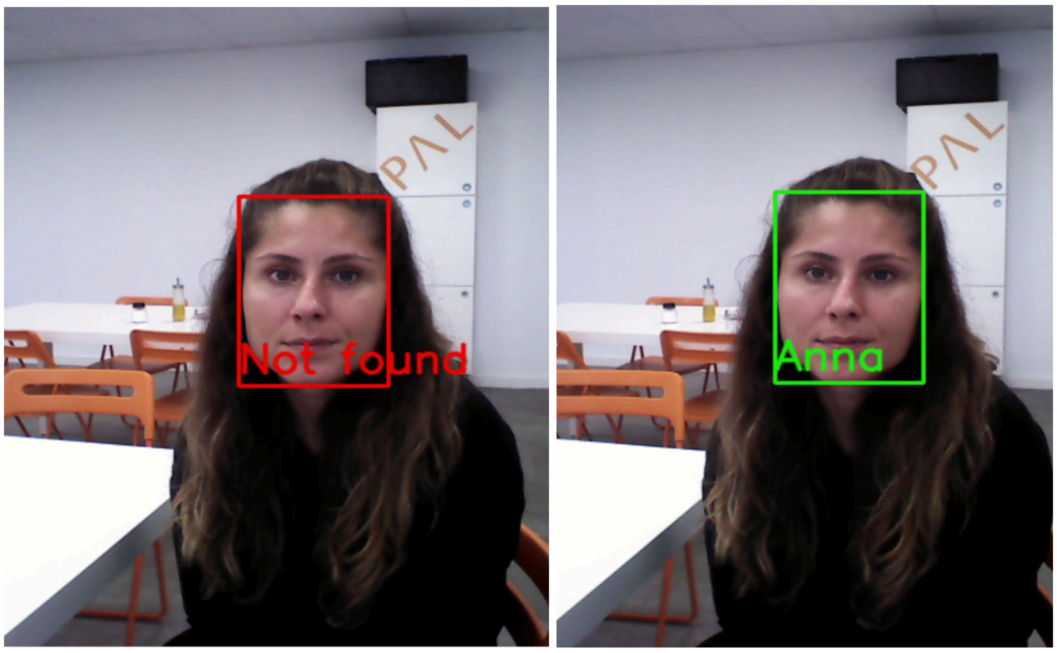


Figure 3.1.2. Adding a new person to the database using speech recognition and PAL chatbot

Utilizing PAL's sophisticated autonomous navigation system, TIAGo navigates to various points of interest (poi1, poi2, poi3) and surveys the environment (Figure 3.1.3); the different blue arrows are the points of interest that can easily be moved (poi1, poi2, poi3, patient_poi, doctor_poi) and the orange arrow is the localization of the TIAGo robot on the map. Upon receiving a patient's request, the system records the location of the patient, poi on the map for the doctor's reference.

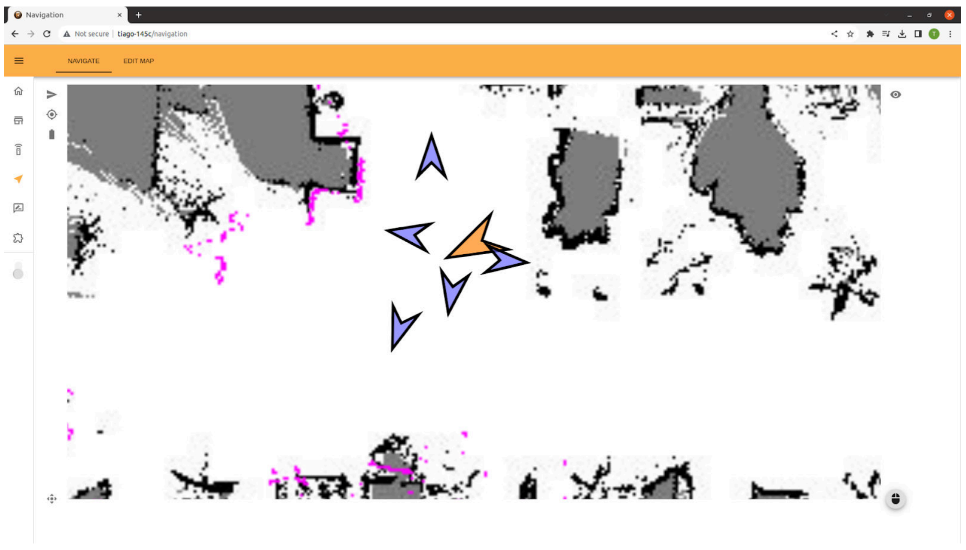


Figure 3.1.3. Updated map with the localization of the patient on the map (most bottom arrow)

Furthermore, in order to better recognize persons whose appearance may vary due to environmental conditions, e.g. lightning conditions, or changes that occur to facial features through time, an active approach on top of the existing face recognition module is implemented in OpenDR. Based on the approach, the facial features extracted by a backbone deep neural network that exists in the known-persons database are gradually and dynamically updated with new facial information. When the system correctly recognizes a person, but with low confidence, facial features are saved in a buffer, which then updates the facial information of that person in the database, making the system more confident in recognizing this person in future occurrences. This method proved to be more robust and more accurate in identifying persons in different environmental conditions and through changes that occur in facial features through time. Additionally, when the algorithm fails to recognize a person, a module to add this person to the database was developed. In the Healthcare use case, the Tiago Robot asks for the person not recognized to state its name, and the corresponding entry in the database is created.

3.2 (Task 2) Manipulating and delivering an object

For this task PAL integrated OpenDR inside the advanced grasping pipeline packages created for the TIAGo robot.

The Advanced Grasping (AG) (Figure 3.2.1) software package provides a framework to use the mobile manipulation capabilities of TIAGo and TIAGo++. It can be used with the PAL gripper, the Robotiq 2f-85 and the Robotiq 2f-140. The package combines behaviour trees and MoveIt! to allow the robot to perform complex grasping tasks in a broad range of environments. Furthermore, the behaviour tree structure provides an infrastructure to adapt to the needs of the application easily.

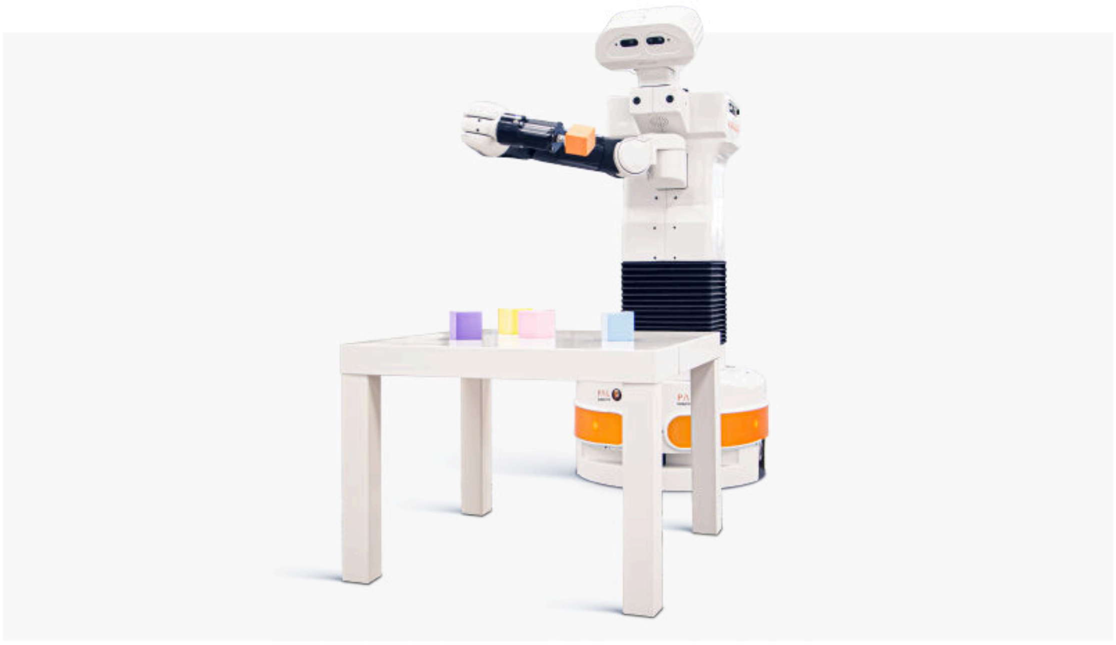


Figure 3.2.1: TIAGo grasping an object using the Advanced Grasping package

The internal architecture of the AG package is based on the server-client structure, in other words, a client sends a goal with specific features to the server and it performs the action without blocking the normal behaviour of the robot. Once the server finishes its task, it returns the result to the client. These servers are implemented using ROS Actions. The three available servers are:

- Perception server.

- Grasp server.
- Place server.

The perception server was adapted to the OpenDR Yolov5 object detector. It is now taking the bounding box message provided by the OpenDR tool and filtering the point cloud provided by the RGBD camera to output a 3D bounding box (cylindrical or rectangular) that fits the detected object, as shown in Figure 3.2.2.

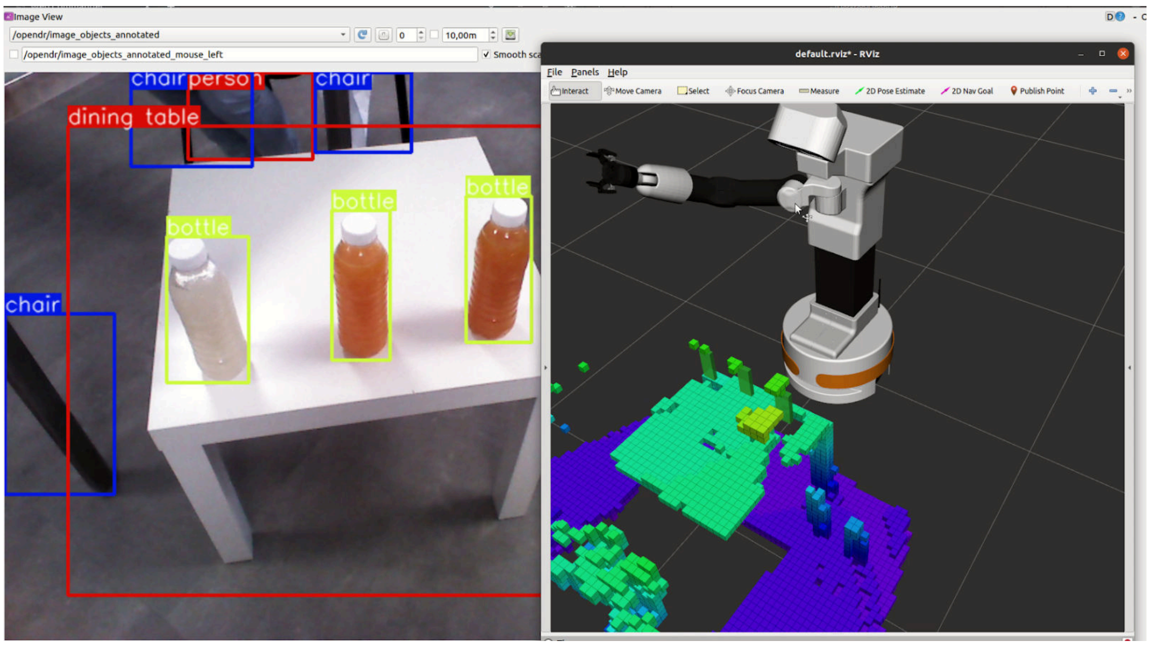


Figure 3.2.2: rviz detection of 3 objects + octomap

The Advanced Grasping detection outputs an ordered list detailing the object positions. Subsequently, the robot observes the doctor, who indicates the preferred bottle for grasping. The selection is determined based on the right wrist position as shown in Figure 3.2.3, with initialization involving pointing first to the left bottle and then to the right bottle before making the final choice of the bottle to be grasped.

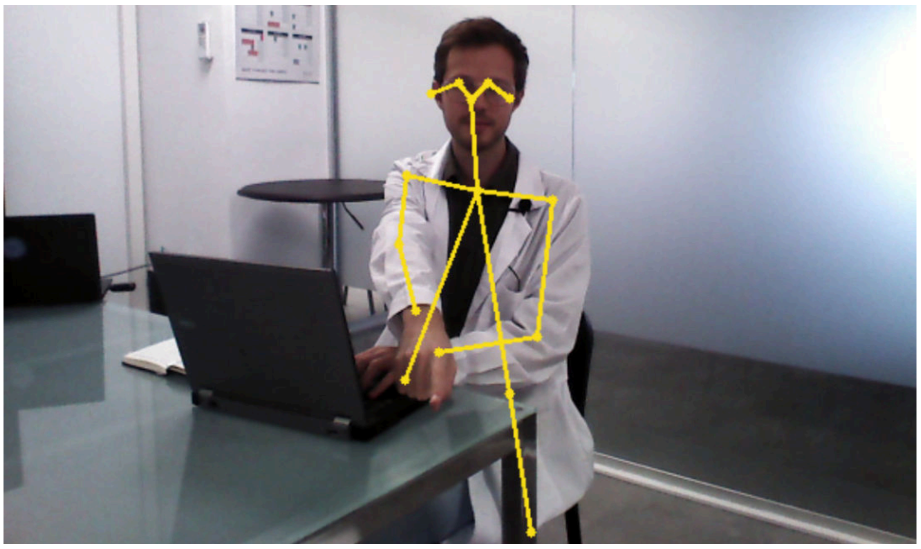


Figure 3.2.3: skeleton detection: pointing at bottle

A new dataset was created to focus on the detection of medicine cups. 25 images were taken and annotated. This dataset was extended by modifying the brightness of the image in order to adapt to different lighting conditions. It was then trained to perform the detection of the medicine cup as seen below in Figure 3.2.3.

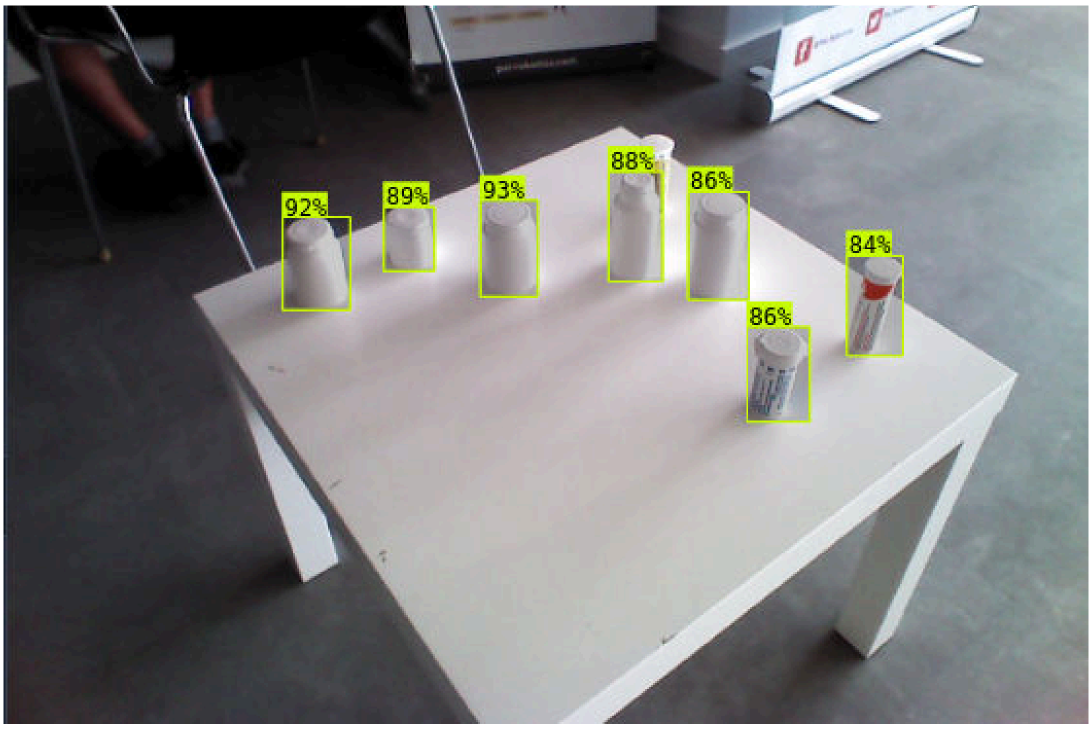
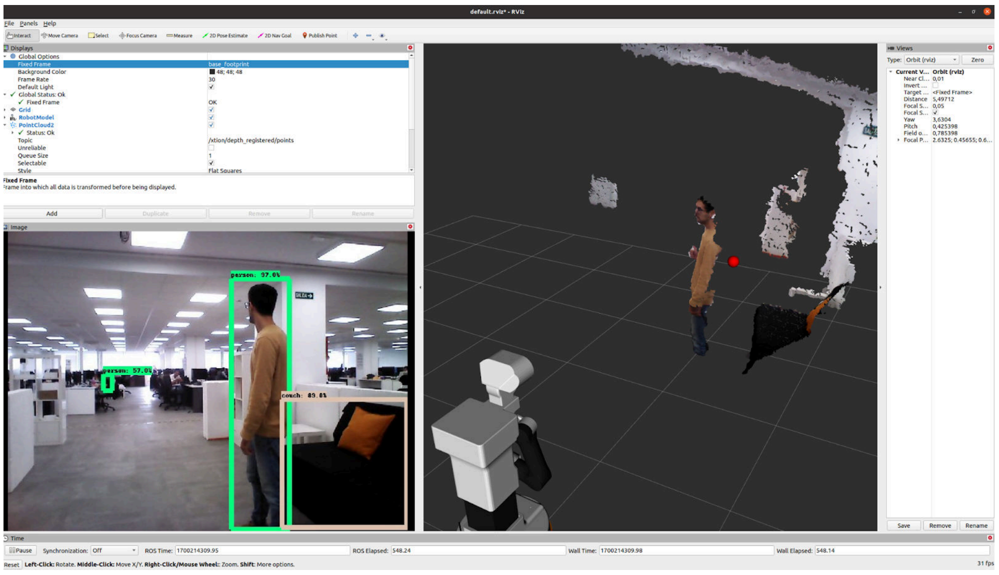


Figure 3.2.3: Yolov5 detection of medicine cups.

3.3 (Task 3) Detection of fallen persons

This task is focused on adapting the fallen person tool that is known to have false detection of sitting persons.

Until now the detection of a fallen person is done by taking the 2d detection of every person in the field of view and outputting a 3d pose of the mean of the point cloud included in this bounding box as shown in Figure 3.3.1. A threshold is then set to differentiate standing persons from fallen persons.



Global Options
Fixed Frame: base_footprint
Background Color: 48; 48; 48
Frame Rate: 30
Default Light: ✓
Global Status: Ok
Fixed Frame: OK
Grid: ✓
RobotModel: ✓
PointCloud2: ✓
Status: Ok
Topic: /xtion/depth_registered/points
Unreliable: ☐
Queue Size: 1
Selectable: ✓
Style: Flat Squares

Fixed Frame
Frame into which all data is transformed before being displayed.

Type: Orbit (rviz) Zero
Current V... Orbit (rviz)
Near Cl... 0,01
Invert ... ☐
Target ... <Fixed Frame>
Distance 5,49712
Focal S... 0,05
Focal S... ✓
Yaw 3,6304
Pitch 0,425398
Field o... 0,785398
Focal P... 2.6325; 0.45655; 0.6...

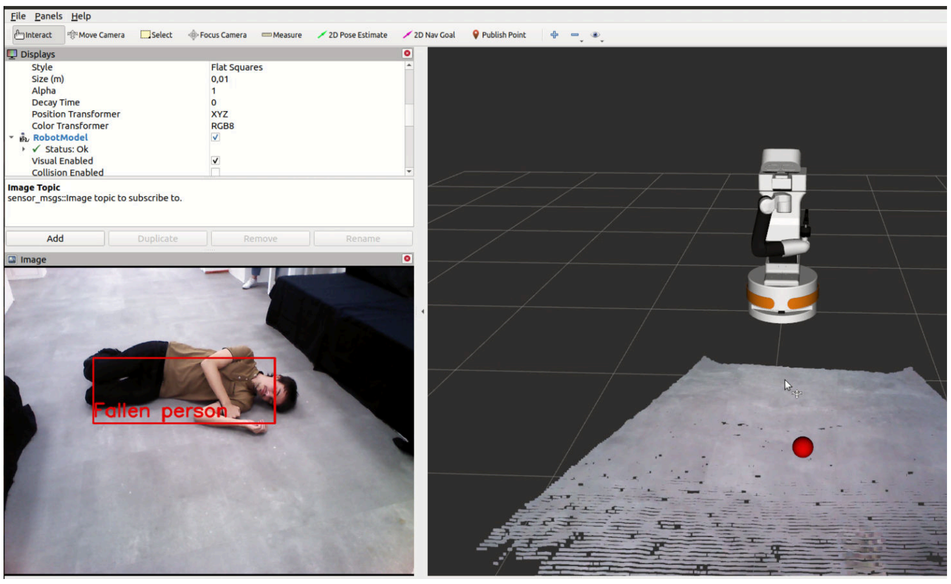
Synchronization: Off ROS Time: 1700214309.95 ROS Elapsed: 548.24 Wall Time: 1700214309.98 Wall Elapsed: 548.14

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.

Figure 3.3.1:: Person detection + pointcloud

This operation requires computing power and can be costly if several persons are detected in the frame.

The package was adapted to take the output of the fallen person OpenDR tool instead of every person detected in the frame, Figure 3.3.2 highlights this change, if a person had been detected standing next to the person it would not have been processed since it would not have been detected as fallen.


File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Image
Add Duplicate Remove Rename
Fallen person

Style: Flat Squares
Size (m): 0,01
Alpha: 1
Decay Time: 0
Position Transformer: XYZ
Color Transformer: RGB8
RobotModel: ✓
Status: Ok
Visual Enabled: ✓
Collision Enabled: ☐

Image Topic
sensor_msgs::Image topic to subscribe to.

Figure 3.3.2 Fallen person detection + pointcloud +red dot of the mean of the point cloud where the person has been detected

3.4 (Task 4) Emotional reactions

The emotional state of the person is recognized at the end of the interaction to understand the appreciation of the medicine delivery to the patient.

If the patient is happy the robot is saying a joyful phrase: “I am glad that your day is going great.”

If the person is sad as in Figure 3.4.1, the robot says: “I can play some music to cheer you up! If you want me to stop you can do a silent sign.” And plays a piece of cheerful music until the person performs a “silence” sign

If the person looks surprised: “Sorry to have frightened you.”

If the person is angry: “Please don't be angry. Everything will be fine!"

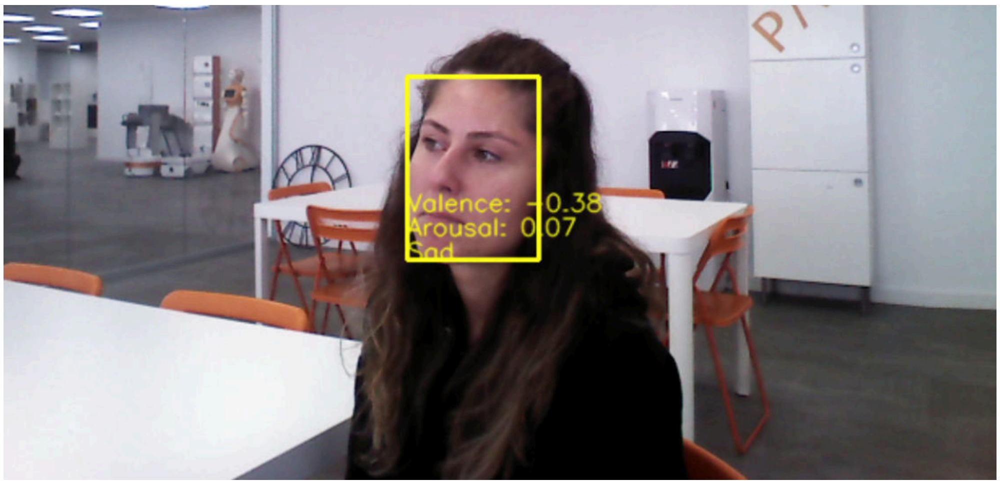


Figure 3.4.1: Emotion detection from TIAGo camera

3.5 (Task 5) Multi-modal human-robot interaction (HRI)

The main focus of this task was to integrate different tools and modalities to interact with the robot.

Before the pipeline is triggered the robot needs to understand with confidence that a patient wants to interact. For this purpose, the natural speech recognizer is combined with the wave detector (Figure 3.5.1) or the gesture recognizer (Figure 3.5.2). A specific keyword is used to trigger the process and then the robot looks at the person, if a wave or a “stop” sign is detected the chatbot will be triggered and the use case can proceed, otherwise the robot will continue roaming the environment.

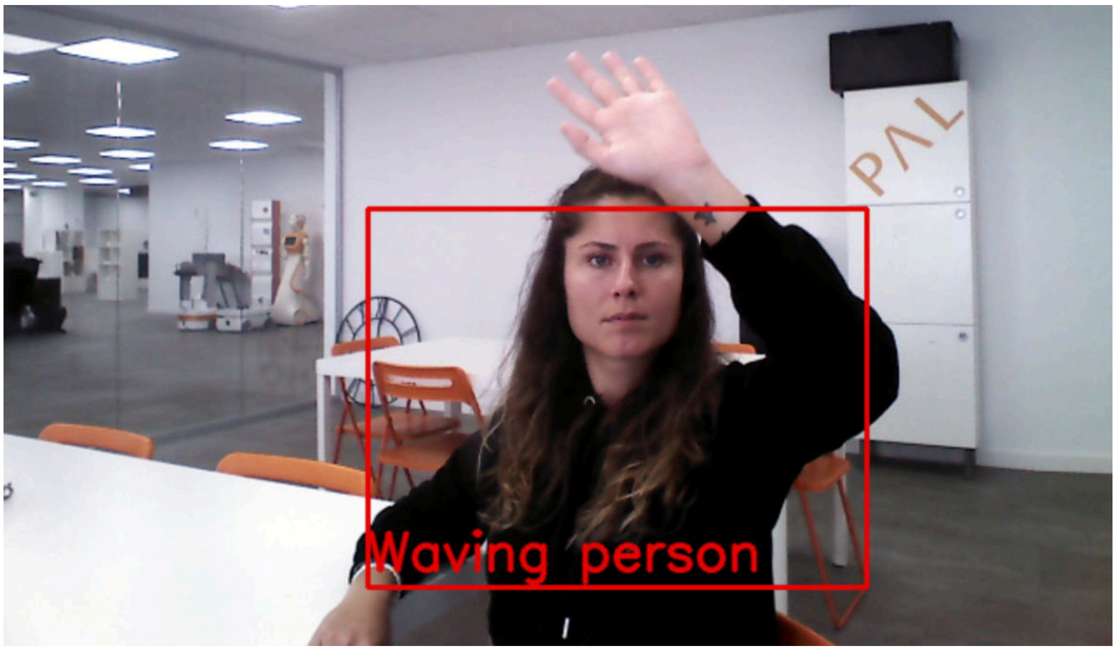


Figure 3.5.1: Person waving at robot

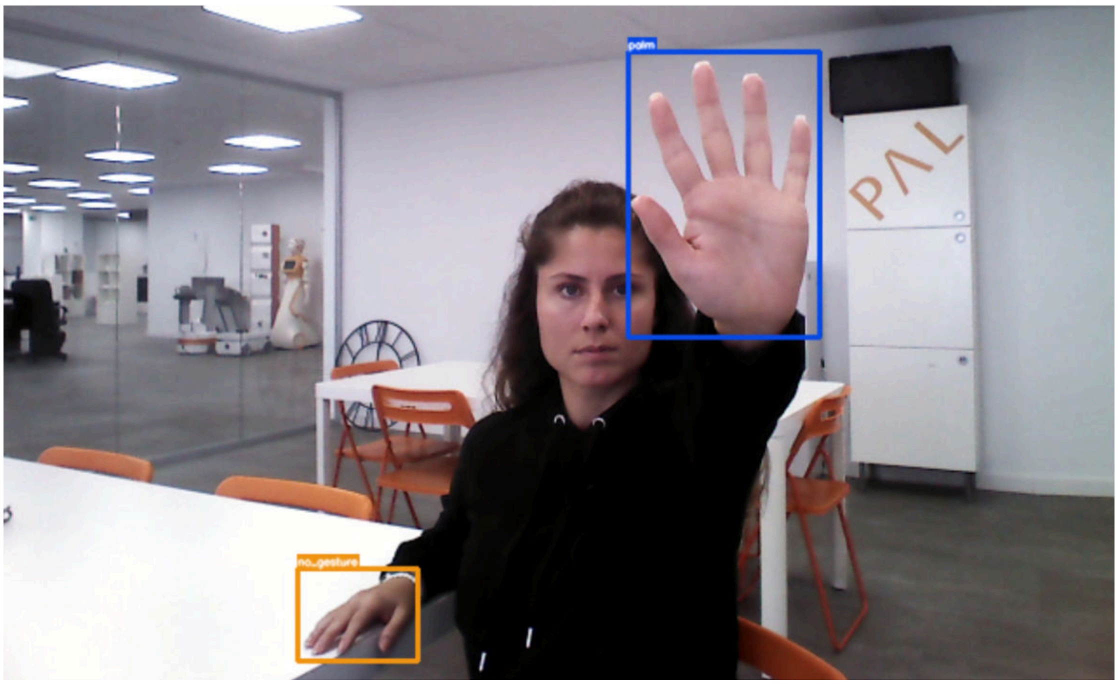


Figure 3.5.2 : Person with hand gesture recognition to stop robot

Once the interaction has been successfully detected the chatbot is activated. The natural speech tool was integrated with the PAL chatbot (the architecture can be found in Figure 3.5.3). The left part of Figure 3.5.3 shows from the top left to bottom left the process of the chatbot, first, the reSpeaker microphone adapts environment sounds to a ROS message that is used by the OpenDR natural speech recognizer. Then the OpenDR tool outputs a standard ROS4HRI message hri_msgs/LiveSpeech, this allows the tool to be easily integrated with pipelines that use the ROS4HRI architecture as a standard. Then the recognized text is provided first to the soft_wakeup_word package to trigger the chatbot. Finally, once the chatbot has been triggered it takes the phrases and processes them with RASA to output a specific intent that can contain specific information about the phrase. RASA is a framework that is open source and designed for constructing applications that operate through text and voice. It employs a modular collection of fundamentals for tasks such as natural language understanding and dialogue management. This approach enables the creation and expansion of advanced conversational AI solutions. In parallel it is sending a response to the text-to-speech of the TIAGo robot to be able to communicate with the user. This integration is able to recognize the intent of the person and reply in a seamless manner.

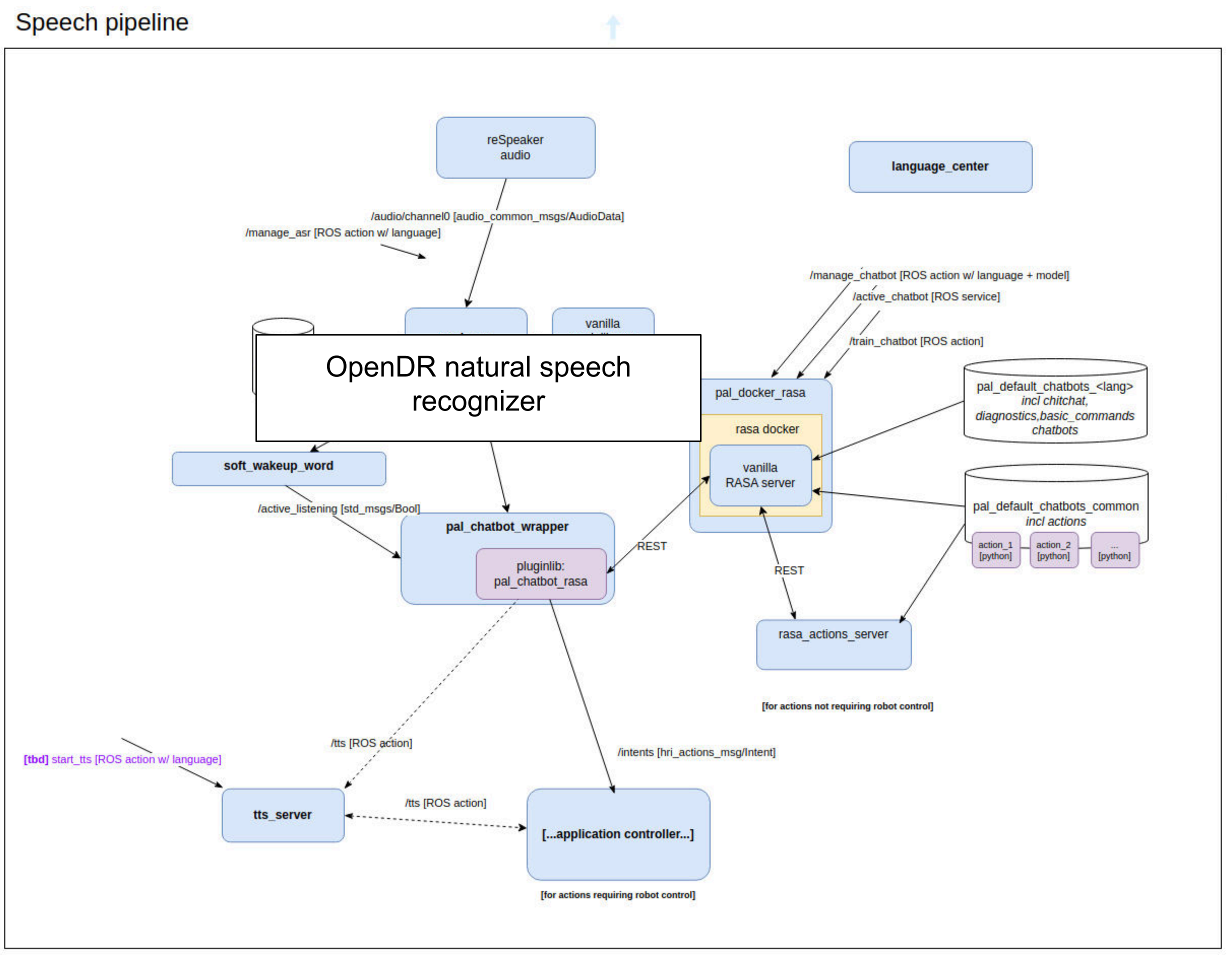

Speech pipeline
reSpeaker audio
language_center
/audio/channel0 [audio_common_msgs/AudioData]
/manage_asr [ROS action w/ language]
/manage_chatbot [ROS action w/ language + model]
/active_chatbot [ROS service]
/train_chatbot [ROS action]
vanilla
OpenDR natural speech recognizer
pal_docker_rasa
rasa docker
vanilla RASA server
pal_default_chatbots_<lang> incl chitchat, diagnostics,basic_commands chatbots
pal_default_chatbots_common incl actions
action_1 [python]
action_2 [python]
... [python]
soft_wakeup_word
/active_listening [std_msgs/Bool]
pal_chatbot_wrapper
pluginlib: pal_chatbot_rasa
REST
REST
rasa_actions_server
[for actions not requiring robot control]
[tbd] start_tts [ROS action w/ language]
/tts [ROS action]
/intents [hri_actions_msg/Intent]
tts_server
/tts [ROS action]
[...application controller...]
[for actions requiring robot control]

Figure 3.5.3: Speech pipeline with OpenDR tool

The RASA rules are set by different packages that will be explained more in depth below, the structure of those packages is shown on the right of **Figure 3.5.3** where pal_default_chatbot provides the chitchat protocol to be able to handle simple everyday phrases like what time is it? or How are you doing ? and other parts customized to fit OpenDR needs. It is then used to reply with a specific phrase or can trigger an application inside the application controller developed for the OpenDR healthcare use case. This approach is modular and can be adapted to different use cases.

The different intents that can be recognized are listed below:

```
intents:

- chitchat

- get_weather

- nlu_fallback
- greet

- goodbye

- get_time

- out_of_scope

- ask_doctor

- stop

- thanks_medicine
```

The greet intent is, for example, outputting one of those phrases that are then pronounced by the robot using text to speech Acapela application:

```
utter_greet:

- text: Hi! What can I do for you ?

- text: Hello, what can I do for you ?
```

The ask_doctor can be triggered by any phrase that is listed below and an extension of those thanks to RASA:

```
- intent: ask_doctor

 examples: |

   - Call a [doctor](symptoms) for me please.

   - I'm [not feeling well](symptoms), can I get some medicine?

   - Could I speak with a [healthcare](symptoms) professional?

   - I'm [sick](symptoms) and need medicine.

   - I'm [feeling sick](symptoms) and need medicine.

   - I'm [not feeling good](symptoms). Can I see a doctor?

   - I think I have a [cold](symptoms)

   - I'm feeling [nauseous](symptoms)

   - I've got a [cough](symptoms). Is there a medicine that could help?

   - I have a [stomachache](symptoms)

   - I have an [headache](symptoms)

   - I think I have [fever](symptoms)?
   - Is there a medicine I can take for my [insomnia](symptoms)?


- synonym: feeling sick
 examples: |
   - not feeling good
   - not feeling well
   - healthcare
   - sick
   - doctor
- synonym: having a cold
 examples: |
   - cold
- synonym: having a cough
 examples: |
   - cough
- synonym: having a stomachache
 examples: |
   - stomachache
- synonym: having an headache
 examples: |
   - headache
- synonym: having a fever
 examples: |
   - fever
- synonym: having insomnia
 examples: |
   - insomnia
```

It is also handling symptoms, for example, if the person says:

```
I have a stomachache and need medicine.
```

Synonyms can also be provided in order to understand the same symptoms with different ways of specifying them.
The robot will extract one of the possible symptoms listed above, here it will be the stomachache then the code will return “having a stomachache” for the robot to include it in the phrase to say to the doctor.

The ask_doctor triggers the task to navigate to the doctor's office and save the position on the map, the name and the symptom of the patient.

In different parts of the use case, hand gesture recognition is also used to confirm or stop actions of the robot:

- To release the object after handing it to the patient to be sure to not make it fall (Figure 3.5.4)

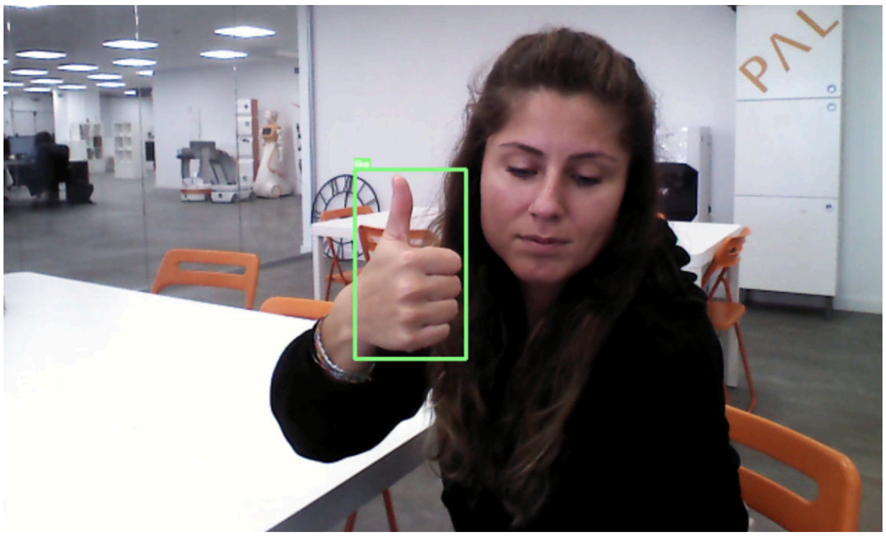


Figure 3.5.4: Thumbs up gesture to release object

- to stop the music after helping the patient and resume the roaming activity (Figure 3.5.5)

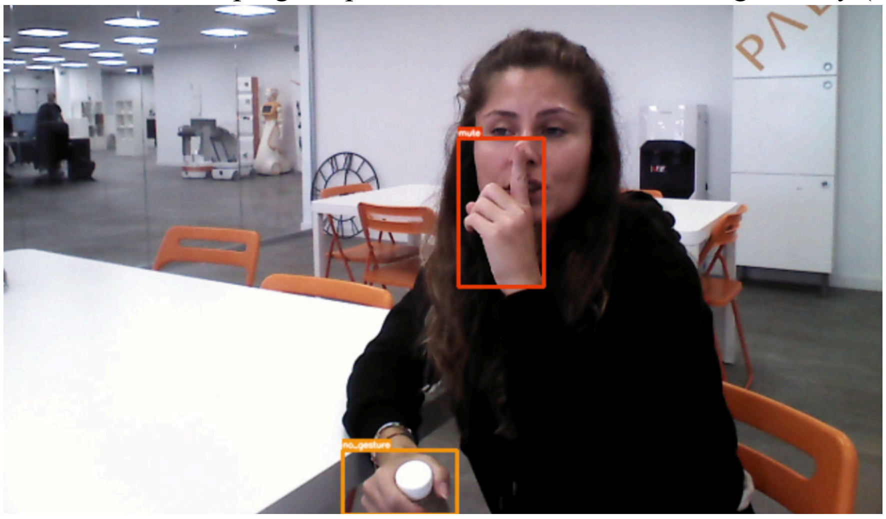


Figure 3.5.5: Mute gesture to stop music and resume roaming

4 Integration and experimental evaluation for Agile Production Robotics Scenario

The general setup for experiments in the agile production scenario is depicted in Figure 4.1.1 and includes two cameras (Intel Realsense D435) for visual perception (one front-facing and one on the robot end-effector) and a microphone for speech recognition. Computation is performed on a standard Desktop PC running Ubuntu Linux with Nvidia GTX 1080 Ti GPU, and all robot (Franka Emika) communication and control utilizes ROS.

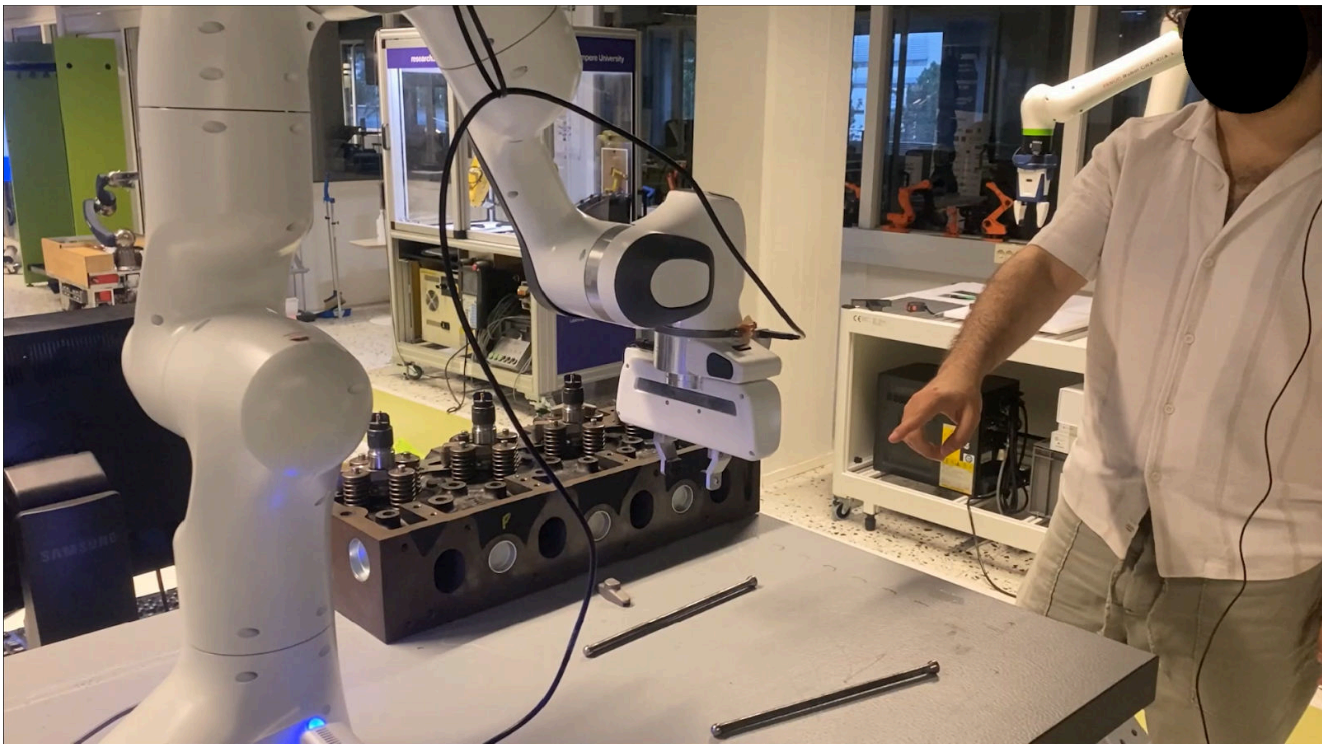


Figure 4.1.1: General setup for the agile production scenario includes a collaborative robot, Diesel engine and its different parts to be assembled and the human operator. Sensors included are one camera on the end-effector and one front-facing towards the person.

4.1 Human detection and tracking

Detection of a human in the agile production scene is done with OpenPose tool, a real-time multi-person human pose detector. For a successfully detected human pose, the method returns a list of 18 2D image key points of the human skeleton with associated key point abbreviations. The method in this work utilizes the pre-trained MobileNet model, which was trained and evaluated with the COCO 2017 dataset under default training parameters. Examples of human detections can be seen in Figure 4.2.1.

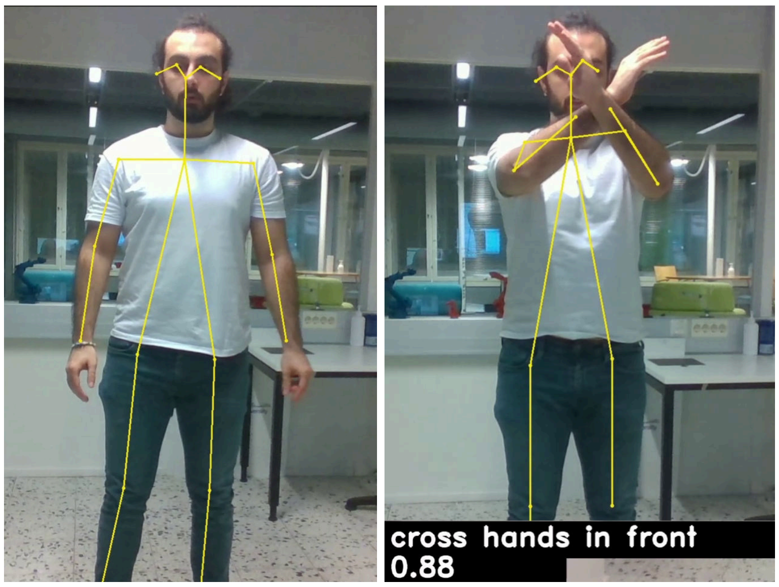


Figure 4.2.1: Human skeleton detection skeleton-based tracker Lightweight OpenPose. Recognized action is ’cross hands in front, with its corresponding confidence score.

Recognition of human actions is done with a real-time skeleton-based human action recognition tool (ST-GCN), as it utilizes the lightweight OpenPose model. The method takes the location of the human joints in every image, and generates a sequence of detected human skeleton graphs, connected both spatially and temporally. Depending on the dataset the method can detect a large number of different human actions, ranging from daily activities to complex actions with interactions. For the use case, the smallest training dataset is selected (NTU-RGB+D), as it contains the most relevant human action classes (60 classes). Two examples of human actions recognized are depicted in Figure 4.2.1, with their corresponding confidence scores.

As additional functionality for human-robot interaction, the recognition of wrist gestures was extracted from the skeleton detection tool. Wrist detection takes the wrist node of a detected skeleton and, when presented in a certain image area, can serve as trigger for robot actions (e.g., stop, continue) or refer to certain objects in the scene. Figure 4.2.2 depicts the detection of the wrist in specific image areas.

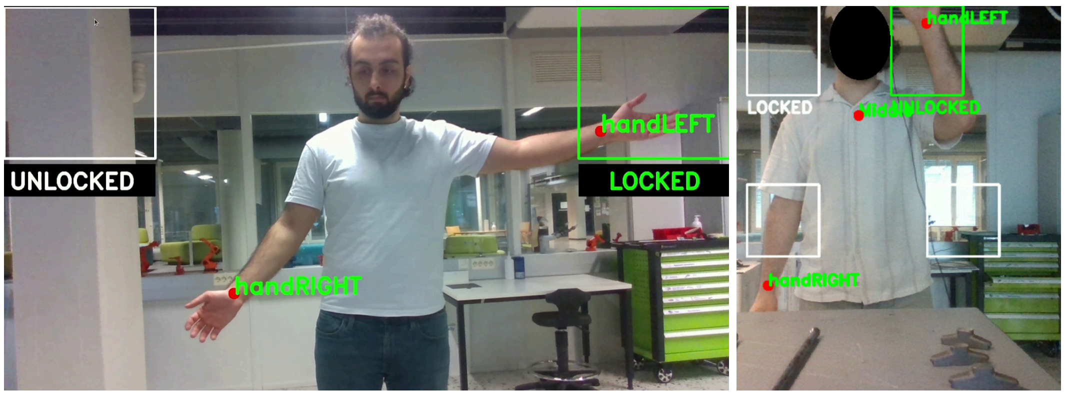


Figure 4.2.2: Wrist detections are obtained with the skeleton-based tracker Lightweight OpenPose, from the detected wrist nodes.

Integrated experiments of the human detection functionalities can be seen in Figure 4.2.3 and the following video: https://youtu.be/b_ISrhOlcC8. This describes the collaborative scenario with a human and a robot sharing the task of assembling different parts of the Diesel engine to the engine block. The specific scenario is described as well in the following sections, as other tools were required for integrated experiments.

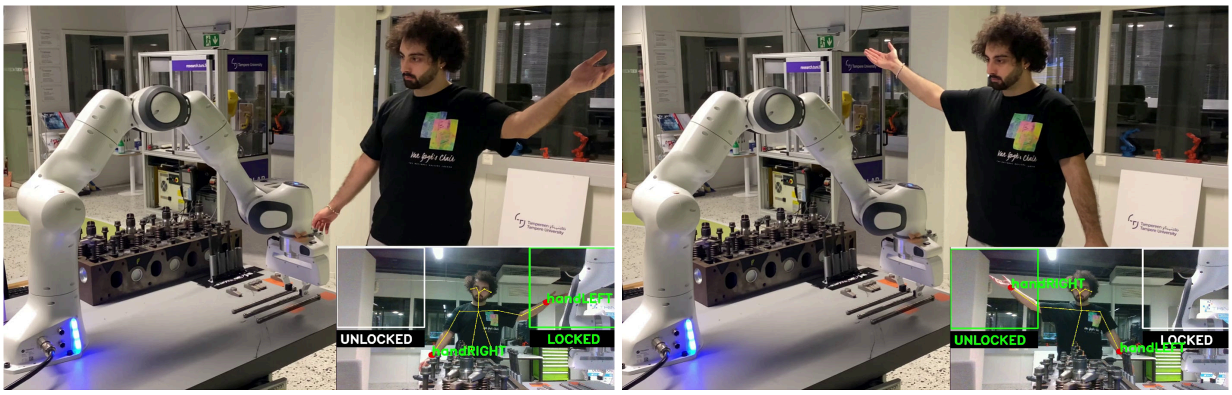


Figure 4.2.3: Results of human-robot collaboration experiments. Both images depict human task coordination by visual detection of
the left wrist (handLeft), for halting the robot and performing manual assembly actions, followed by right wrist detection
(handRight) for resuming robot actions.

4.2 Object detection and pose estimation

Object detection and pose estimation serve to detect objects in the Diesel engine assembly scenario and utilize the information for further robot actions, such as grasping, pick and placement and robot to human hand-overs. Mask R-CNN from Detectron2 was selected for object and target detection in the scene, as performance was preferred over inference time. Mask R-CNN combines a Region Proposal Network (RPN) with the CNN model, to simultaneously predict object bounds and objectness scores at each position. After detection, orientations are estimated in each bounding box by the second order moment from a segmented object or target.

As the assembly objects and targets are novel with respect to existing datasets, a custom dataset needed to be generated. For this, 200 images of eight object and target classes were annotated with segmentation polygons, as depicted in Figure 4.3.1. The object classes included rocker arms, bolts and pushrods, and the target classes included the Diesel engine, small and big pushrod holes, bolt holes and rocker arm locations. This data was augmented to include a broad variation in noise and lighting conditions, to form the custom dataset of around 280,000 images. The dataset is available at Zenodo: https://zenodo.org/records/7669593

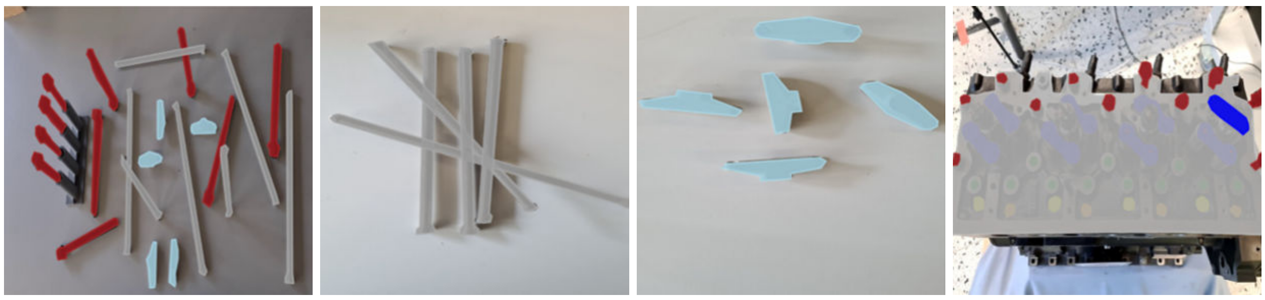


Figure 4.3.1 : Image annotations for objects, including bolts (red), pushrods (grey) and rocker arms (light blue) and targets, including Diesel engine (grey), small (yellow) and big (orange) pushrod holes, bolt holes (green) and rocker arm locations (dark blue).

Results of the detection model are depicted in Figure 4.3.2 and a video can be seen here: https://youtu.be/A3uqe8AYh0M

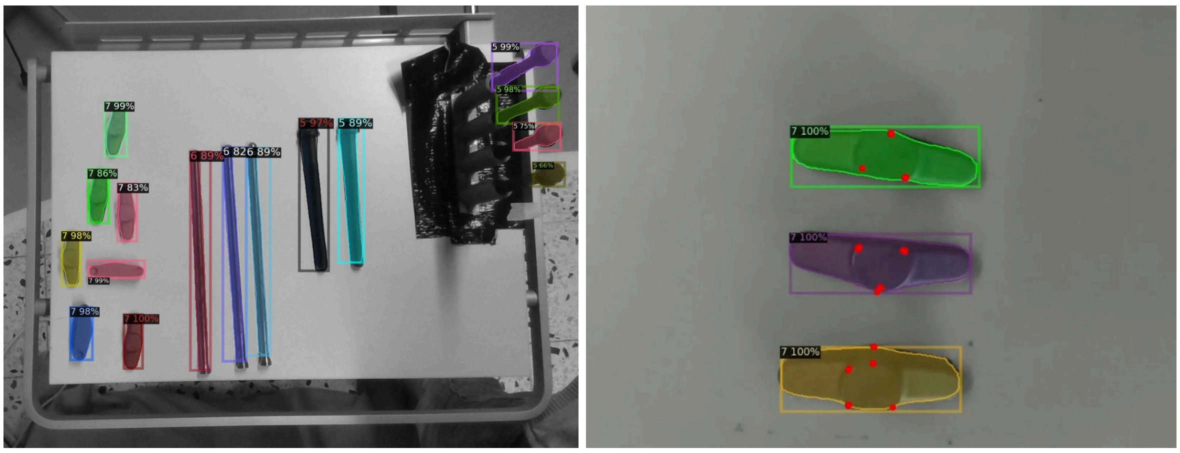


Figure 4.3.2: Results of visual perception for object and target detection utilizes Detectron2. The images depict detection of objects (three classes): bolts (class 5), pushrods (class 6) and rocker arms (class 7), labeled with the detected class and their corresponding confidence score.

4.3 Recognition of targets

Target recognition was achieved with the same dataset and trained model as in Section 4.2. Results of the model for targets can be seen in Figure 4.4.1 and a video can be seen here:

https://youtu.be/A3uqe8AYh0M.

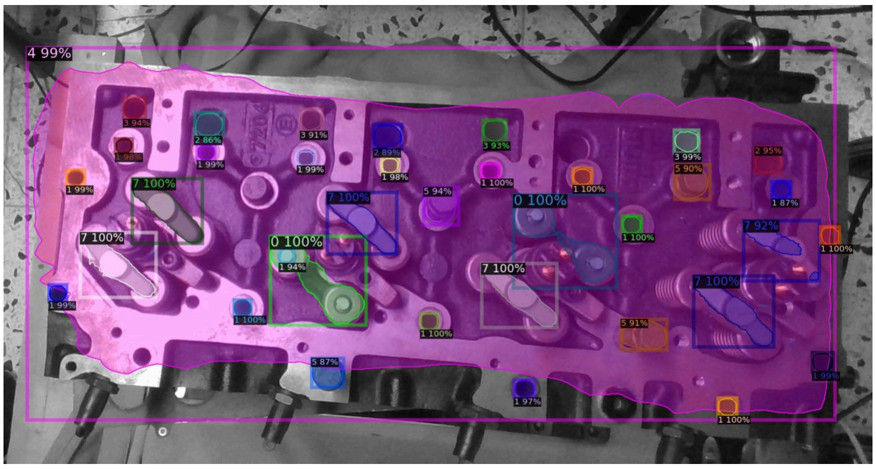


Figure 4.4.1: The image depicts detection of targets (five classes): rocker arm location (class 0), bolt holes (class 1), big and small pushrod holes (class 2 and 3) and engine (class 4). Each detection is labeled with the detected class and their corresponding confidence score.

4.4 Object grasping

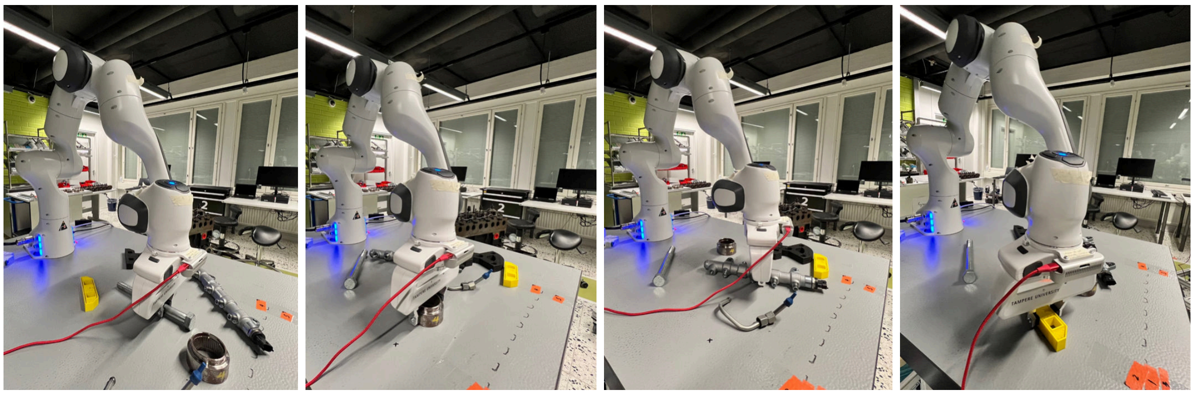


Figure 4.5.1: Results of the robot grasping tool.

The grasping of objects utilizes the object detection tool with custom generated datasets, to detect object positions and orientation from a camera mounted on the robot end-effector. Besides perception, the tool also includes different robot functionalities, such as robot motion planning and gripper actions, in the form of a Python library. Developments for this have been made in such a way that any of the perception tools can be utilized to execute robot grasping actions. Results of one of the grasp detection models can be seen in Figure 4.5.1 and in the video: https://youtu.be/-B_5nLEtiec. In addition, other results of the grasping actions can be seen in the videos from the other Sections, as the tool was integrated in different use case tasks.

4.5 Object manipulation

The manufacturing of Diesel engines involves assembly steps that are hard to automate, such as contact placement and manipulation of parts with various degrees of freedom. For example, rocker arm placement, push rod insertion and bolt fastening all have different constraints with respect to the final manipulation of the part to the engine. Rocker arms can be moved freely in 3D task space before placements, push rod insertion requires vertical motion into a pushrod hole and bolt fastening requires rotational motion and compliance orthogonal to vertical motion. In addition, parts to assemble are complex in shape, metallic and require lubricant for assembly and for operation. This means traditional robotic operations for picking and placing are not suitable for assembly and manual actions are the standard approach for manufacturing. A promising alternative, however, is to utilize the robot as an assistant and assign tasks to it that support the assembly procedure and the ergonomy of the human operator. These are easy, but repetitive tasks, such as pick and placement, and actions for operator assistance such as hand-overs of parts and tools. Where possible, other tools are utilized to provide perception input, i.e., object perception for object detection and pose estimation (Section 4.2 and 4.3), person and speech perception for task coordination and HRI (Section 4.1, 4.7 and 4.8) and grasping actions (Section 4.4).

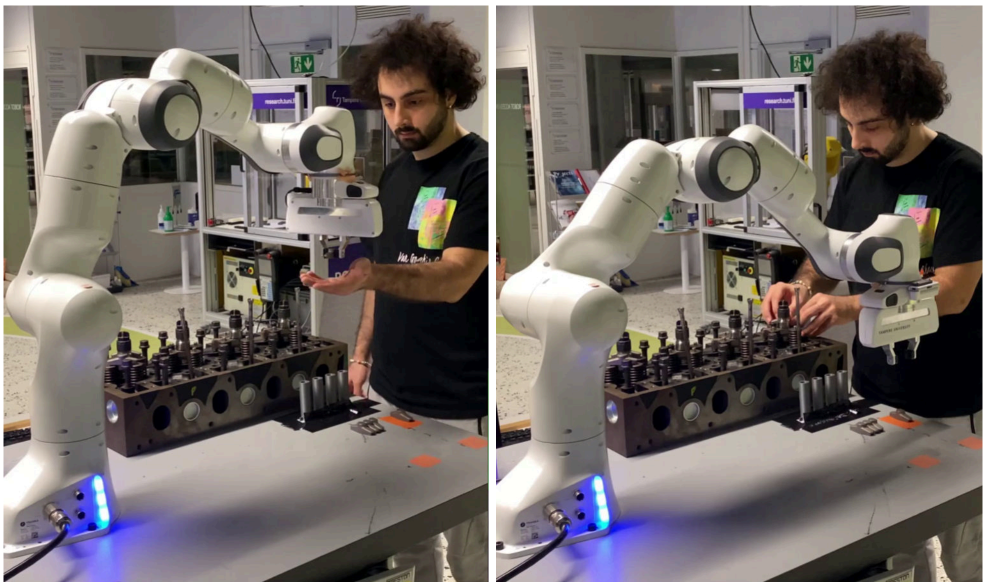


Figure 4.6.1: Results of robot-human hand-over and assembly tracking experiments depict the hand-over of a rocker arm from robot to human and the human assembly action of the rocker arm, while the robot fetches another rocker arm.

Results for object manipulation can be seen in Figure 4.6.1 and a video of the integrated experiments can be seen here: https://youtu.be/3z3yiLdznrY .

4.6 Task policy generation and sim2real

A unified software pipeline is facilitated by the EAGERx tool, enabling seamless transition from simulation to real-world deployment and supporting rapid prototyping. OpenDR tools, such as speech recognition and PARTNR, have been integrated within the engine-agnostic graph structure of EAGERx. This integration allows for effortless interchange between various simulators and real-world applications. Both the camera (Realsense d435) and manipulator (Franka Emika) are encapsulated as abstract objects in EAGERx, with distinct implementations for simulation and real-world contexts. Consequently, tools can be initially integrated and validated in a simulated environment, thereafter requiring no additional modification for real-world experiments. The simulated and real-world version of the use-case are depicted in Figure 4.7.1, while the graph that is used in the use-case is depicted in Figure 4.7.2. The task policy is interactively learned using the PARTNR tool of OpenDR. See Section 4.10 for more details.

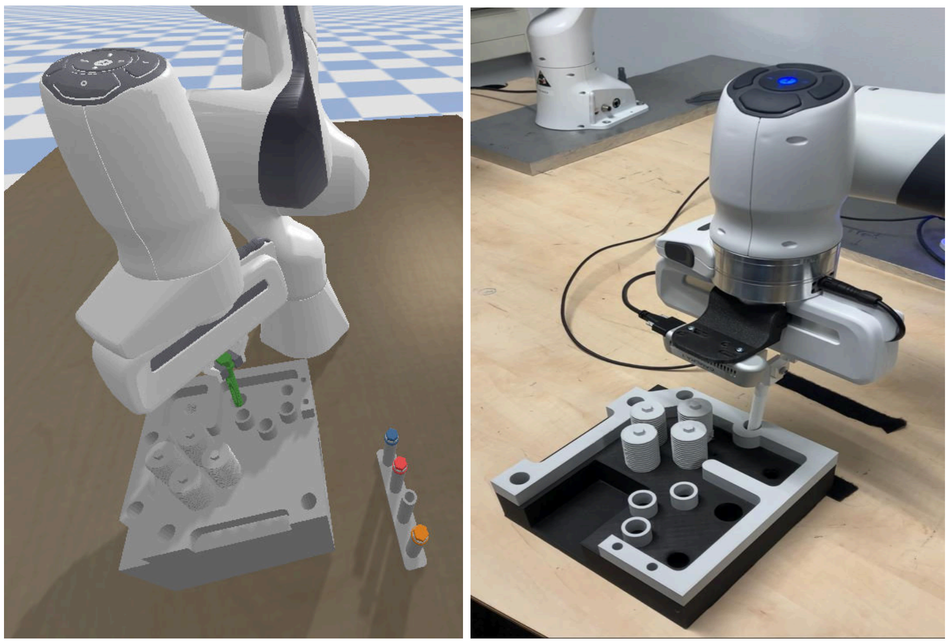


Figure 4.7.1: Transitioning from PyBullet to a physical setup that uses 3D-printed engine components and EAGERx.

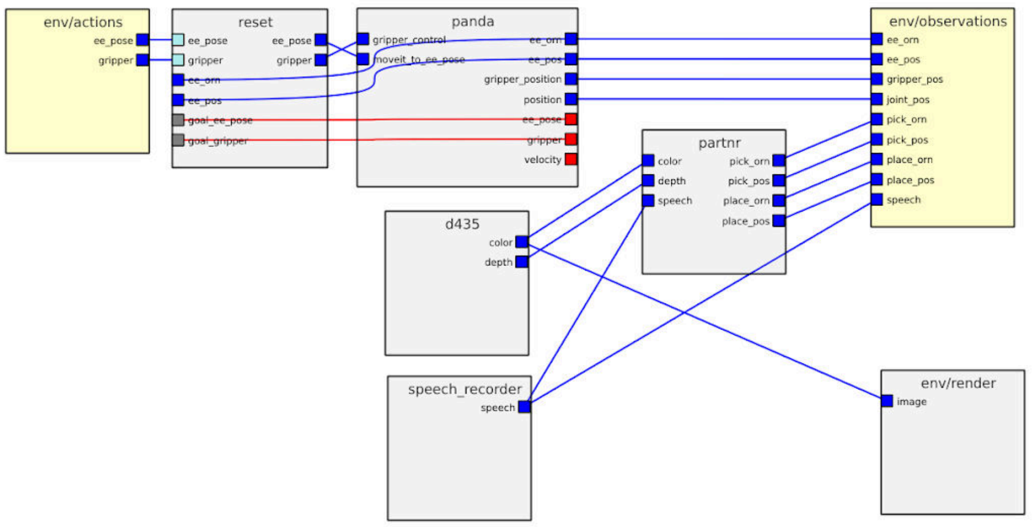


Figure 4.7.2: The agnostic graph, as visualized by the GUI of the EAGERx tool, depicts the unified pipeline that is used for both simulated and real world deployment that enables rapid prototyping.

4.7 Speech recognition

Speech recognition is utilized to provide different ways for the human to engage and collaborate with the robot, ranging from action-target pairs to command robot actions, to short sentences for coordinating robot actions and the completion of a shared assembly task, and high-level language phrases to command specific robot tasks. The Vosk speech tool is used for all speech recognition and in all cases relies on predefined input commands and word phrases. This set of words and sentences relate to available actions of the robot and locations in the scene, as well as full sentences, as follows.

<*action,target*> pairs for commanding robot actions.
Examples: <*give,tool*>, <*pick,tool*>, <*come,here*>, <*kit,bolts*>, <*place,box*>, <*take,bolt*>

https://youtu.be/SzIuLHzLYpA

Short sentences for coordinating robot actions (see Section 4.6).
Examples: <*Give me that rocker arm*>, <*Pick up the last rod*>, <*Go home*>, <*Place the rod*>, <*Continue*>
https://youtu.be/b_ISrhOlcC8

High-level commands for commanding to execute the CLIPort model (see Section 4.9).
Examples: <*put rocker arm in red box*>, <*put all long screws in brown box*>, <*put all push rods in brown box*>
https://youtu.be/hVkqQD9ASm8

4.8 Multi-modal HRI

The considered use case to demonstrate multi-modal HRI replicates an industrial assembly task that in current situations is done manually by human operators. Our solution proposes to introduce a collaborative robot as an assistive tool to the assembly station, under control of the person. This means that the assembly work is coordinated by the human, with the robot assisting in tasks that the human decides. Available robot actions are to move to certain locations in the workspace, pick objects that are detected on the table, place objects in specified locations or hand them over to the human. In addition, coordinated actions include the stopping and continuing of robot actions during execution (see Figure 4.9.1), for human visual inspection of the objects placed by the robot. Human commands can be communicated by hand gestures (see Section 4.1) and/or speech (see Section 4.8), with different levels of functionality as described in Table I (see also Figure 4.9.1 and 4.9.2).

The single-modal visual and speech perception models are also fused into a multi-modal perception model by combining speech commands, pointing gestures and object detection (see Figure 4.9.3). Several examples of these co-speech gestures are described in Table 4.9.1. The human can refer to individual objects in the scene by speech (e.g., <rod>, <rocker arm>) and pointing to them, and apply specific robot actions by speech commands (e.g., picking with <pick>, placing with <place>, robot to human hand-over with <give>). Depending on the object, different robot actions are possible, as specified beforehand. For example, objects can be picked up from the table, placed in specific locations and handed over to the person. Object detection returns a list of objects in the scene, which can be verbally referred to by their class. Pointing gesture detection allows referring to specific objects in the scene by relating the pointing gesture location to detected object locations. Robot actions are therefore commanded by specific action verbs and object classes, complemented by gestures to provide fine-grained object references.

Table 4.9.1: Perception models input and output to achieve single-modal and multi-modal human-robot interaction

Method	**Input**	**Output**
Wrist detection	RGB image of the scene (human front-facing) Human gesture by moving wrist to certain image location	Robot stop/continue actions
Speech recognitio n	Robot action commands: <*pick, place, give, go, stop, pause, continue*> Workspace commands: <*rod, home, arm, me*>	Robot motion Gripper actions Robot to human hand-over

	Human speech requests: <*place rod*>, <*go home*>, <*give me another rocker arm*>, <*pick up the last rod*>	Robot stop/continue actions
Object detection	RGB image of the scene (top-down)	Detected objects in the scene Valid target location for robot
Co-speec h gestures	<*pick rod*> + pointing gesture + object detection <*give me this rod*> + pointing gesture + object detection <*give me that rocker arm*> + pointing gesture + object detection	Robot motion Gripper actions Robot to human hand-over

Results of the different human-robot interaction functionalities are depicted in Figures 4.9.1, 4.9.2 and 4.9.3.

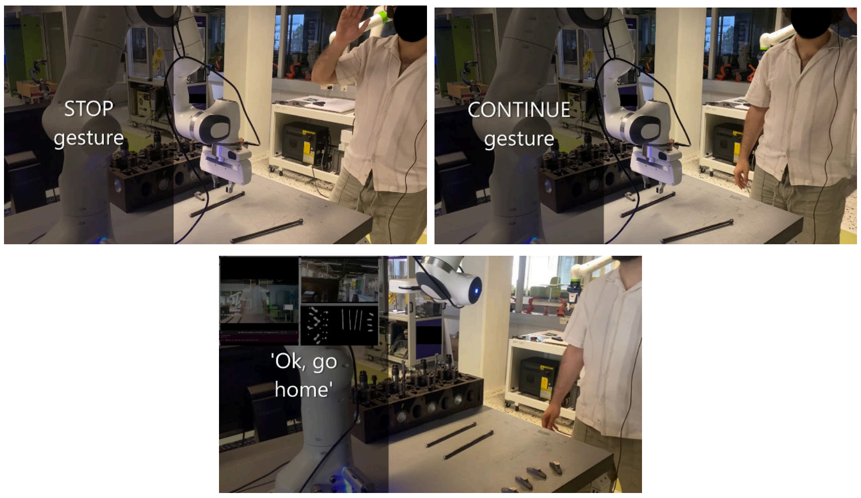


Figure 4.9.1: Single command gestures STOP (a) and CONTINUE(b) and speech commands (c).

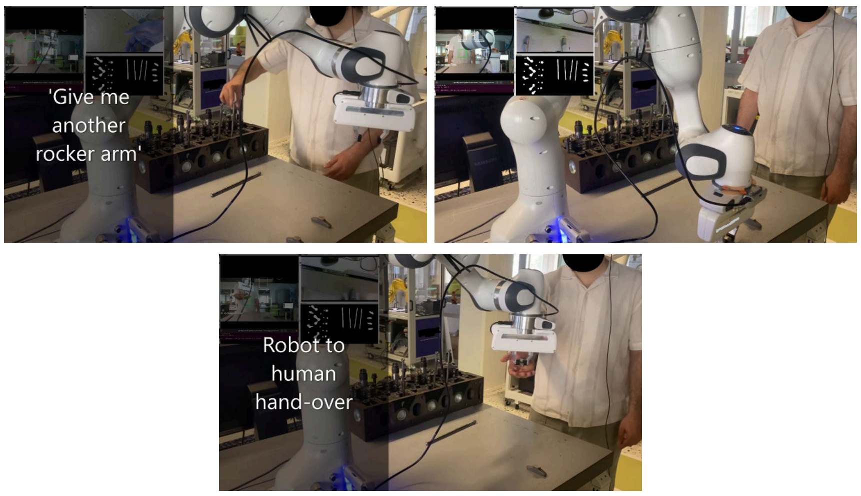


Figure 4.9.2: Speech phrase to achieve robot to human hand-over.

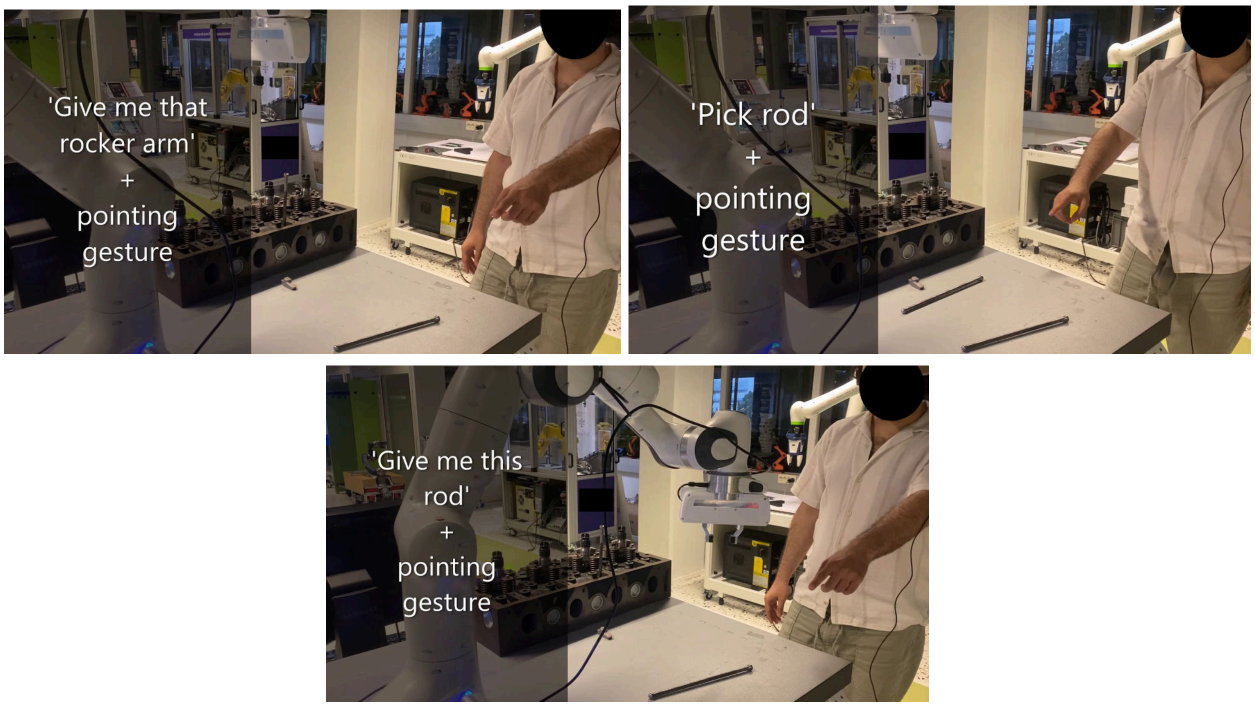


Figure 4.9.3: Co-speech gestures to achieve specified robot actions to objects.

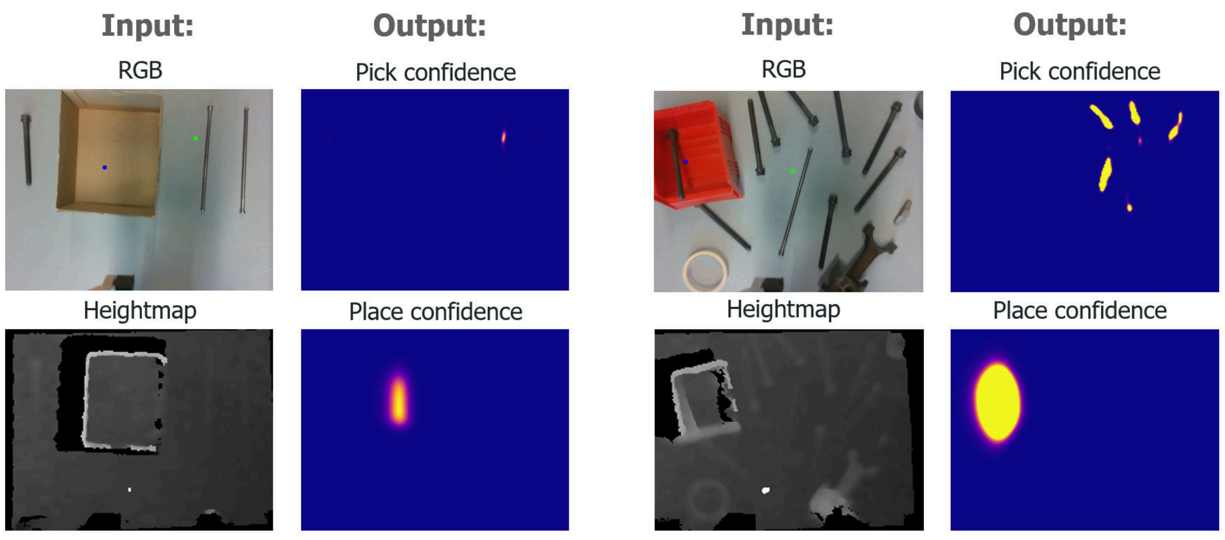
A video of the integrated experiments can be seen here: https://youtu.be/b_ISrhOlcC8

In addition, multi-modal HRI for the agile production use case was also investigated with recent developments in large language models and, in particular, CLIPort. CLIPort utilizes the combined power of visual-language models by Contrastive Language-Image Pre-training (CLIP), with millions of image-caption pairs. After pre-training, new captions can be used to describe other raw images, paving the way for zero-shot transfer learning of the model to different tasks. The architecture has the capability to understand semantic and spatial representations for robotic vision-based and language-conditioned manipulation tasks, in an end-to-end network. A large variety of tasks are demonstrated, including folding cloth, manipulating unseen objects, all without the need to learn objects' poses, structure or instance segmentations. To be relevant for the agile production use case, the CLIPort model was expanded with custom collected data (see Table 4.9.2), including images of relevant scenes and objects, as well as language phrases.

	Demonstrations			Demonstrated tasks		Pick-and-place-targets			
	Training	Validation	Total	Single-object tasks	Multi-object tasks	Known objects	Unknown objects	Goals	Training time [h]
Single-task model	88	22	110	60	50	9 bolts 4 pushrods 3 rocker arms	6	brown/ red box	89
Multi-task model	39	8	132	0	47	2 bolts 2 pushrods	0	brown box	26

Table 4.9.2: Details of the datasets that were added to the CLIPort model. The single-task model only executes one task, while the multi-task model executes multiple tasks sequentially, depending on the language input.

Results of the trained CLIPort model can be seen in Figure 4.9.4, which depicts the input and output of the model with certain examples.



Language phrase: <*Put pushrod in brown box*> Language phrase: <*Put pushrod in red box*>

Figure. 4.9.4: CLIPort takes as input RGB-D images and a language phrase and generates as output picking (green dot) and placing (blue dot) locations extracted from the confidence maps.

Results are also shown in Figure 4.9.5 which depicts snapshots of the executed experiments within the agile production scenario. As a general outcome, it was concluded that, while interesting, the CLIPort model and its functionalities require extensive data collection and fine-tuning for the limited functionalities that it achieves.

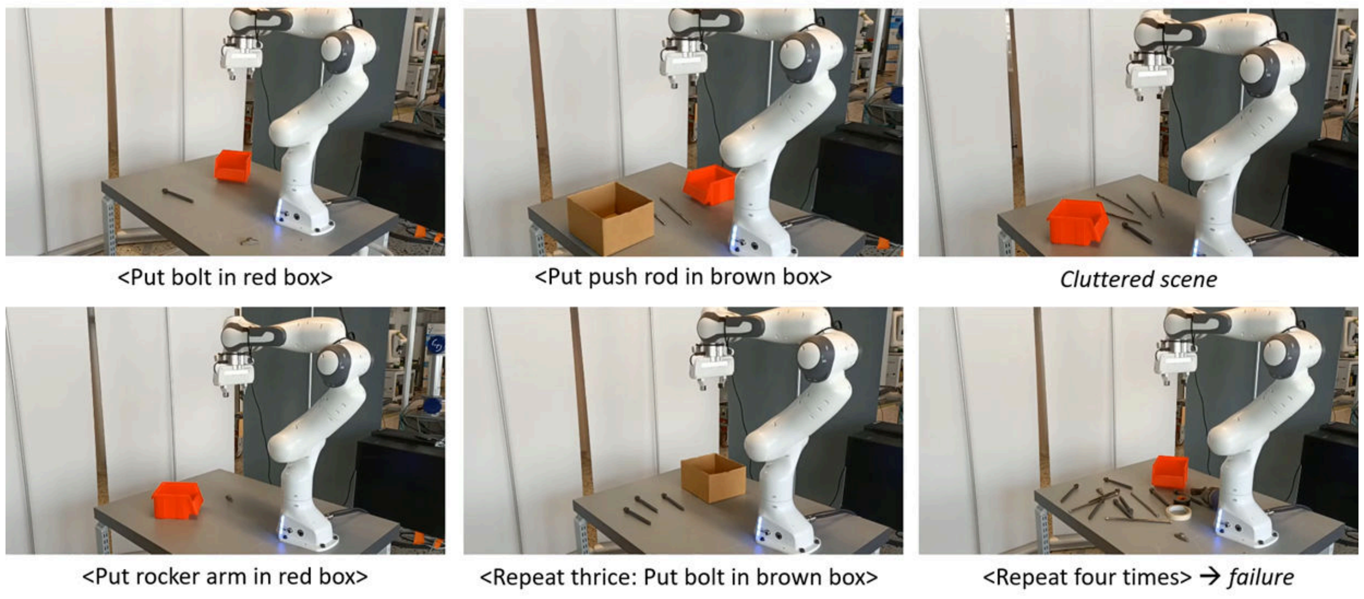

<Put bolt in red box>
<Put push rod in brown box>
Cluttered scene
<Put rocker arm in red box>
<Repeat thrice: Put bolt in brown box>
<Repeat four times> → failure

Figure 4.9.5: The original CLIPort model was adapted to include data relevant for the agile production use case. The images depict the tested scenarios with respective input speech commands.

A video of the integrated CLIPort model can be seen here: https://youtu.be/hVkqQD9ASm8

4.9 Interactive Imitation Learning

In our use case, a robot must discern the appropriate actions for a pick-and-place task based on RGB-D imagery and verbal commands from the operator. Traditional approaches necessitate a prolonged offline pre-training period with extensive operator demonstrations before achieving satisfactory task execution. To circumvent this, we employ the PARTNR tool from OpenDR, which facilitates interactive policy training. PARTNR employs an adaptive, sensitivity-based, gating function that decides if additional operator demonstrations are required. User demonstrations are aggregated to the dataset and used for subsequent training. In this way, the policy can adapt promptly and it can minimize the number of required demonstrations for a well-trained policy. The adaptive threshold enables us to achieve the user-acceptable level of ambiguity to execute the policy autonomously and in turn, increase the trustworthiness of our system. Initially, the policy's uncertainty prompts frequent operator interventions. However, as the policy becomes more certain through additional demonstrations, the robot's autonomy increases, minimizing disruptions to the operator. Fig 4.10.1 illustrates this interactive training loop. Within the context of this use-case, Figure 4.10.2 illustrates the PARTNR tool's initial input, prompting a user demonstration, followed by an interactive training phase, resulting in the system's autonomous determination of the correct pick-and-place actions.

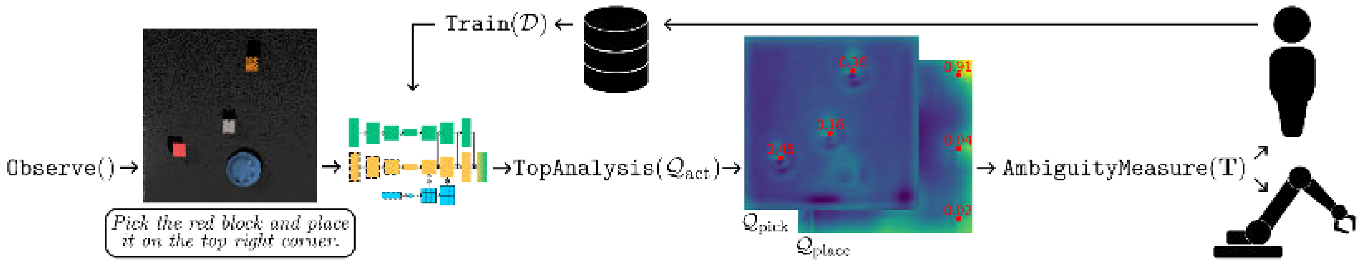


Figure 4.10.1: PARTNR framework on an example task

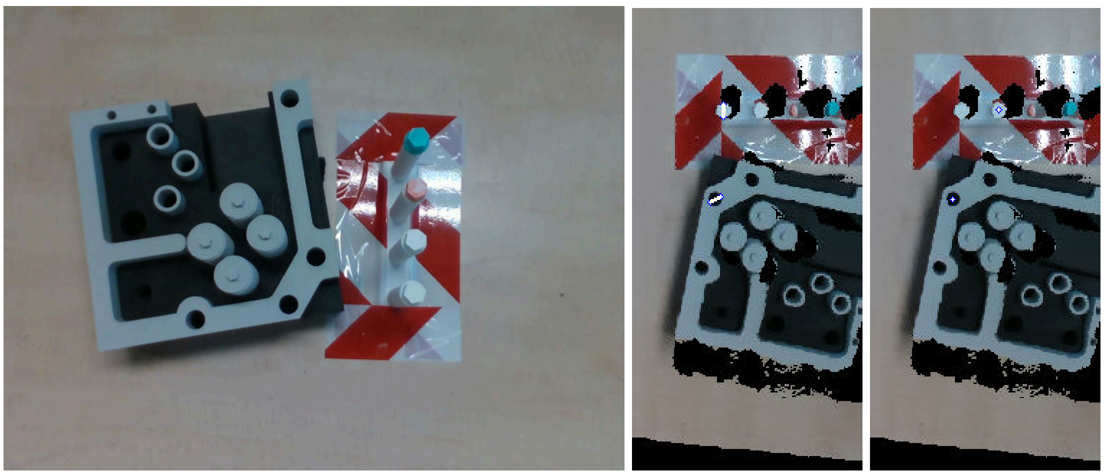


Figure 4.10.2: PARTNR Tool Demonstration for Robotic Pick-and-Place Task. Left: Input image to PARTNR depicting initial task setup. Middle: Operator demonstration of the desired action. Right: Automated pick-and-place action proposed by PARTNR after interactive training phase. This sequence illustrates the progressive autonomy of the robotic system using PARTNR, which reduces the need for extensive pre-training by employing operator demonstrations to rapidly refine the policy, thus enhancing trustworthiness and minimizing operator disruptions.

5 Conclusions

D7.5 presents the work of integration and experimental validation of the OpenDR toolkit across Agriculture, Healthcare, and Agile Production domains in WP7. The evidence presented within this document, supported by visual documentation, attests to the efficacy and versatility of the OpenDR toolkit.

As industries continue to embrace automation and robotics, the achievements outlined in this report pave the way for a future where intelligent robotic solutions play a central role in enhancing efficiency, safety, and overall operational excellence. Integrating OpenDR toolkits into practical use cases sets a precedent for the continued evolution of robotics, with far-reaching implications for industries seeking to embrace the transformative power of automation.