Development of an Augmented Reality Interface for Intuitive Robot Programming

MASTER'S THESIS

at the Human-centered Assistive Robotics
Technical University of Munich

Submitted by cand. M.Sc. Georgios Chnitidis
on 31.03.2023

First supervisor: Univ.-Prof. Dr. Dongheui Lee

Second supervisor: M.Sc. Christoph Willibald

Abstract

As the demand for advanced robotic systems continues to grow, the need for new technologies and techniques that can improve the efficiency and effectiveness of robot programming is imperative. The latter relies heavily on the effective communication of tasks between the user and the robot.

To address this issue, we developed an Augmented Reality (AR) interface that incorporates Head Mounted Display (HMD) capabilities, and integrated it with an active learning framework for intuitive programming of robots. This integration enables the execution of conditional tasks, bridging the gap between user and robot knowledge. The active learning model with the user's guidance incrementally programs a complex task and after encoding the skills, generates a high level task graph. Then the holographic robot is visualising individual skills of the task in order to increase the user's intuition of the whole procedure with sensory information retrieved from the physical robot in real-time. The interactive aspect of the interface can be utilised in this phase, by providing the user the option of actively validating the learnt skills or potentially changing them and thus generating a new skill sequence. Teaching the real robot through teleoperation by using the HMD is also possible for the user to increase the directness and immersion factors of teaching procedure while safely manipulating the physical robot from a distance.

The evaluation of the proposed framework is conducted through a series of experiments employing the developed interface on the real system. These experiments aim to assess the degree of intuitiveness provided by the interface features to the user and to determine the extent of similarity between the virtual system's behavior during the robot programming procedure and that of its physical counterpart.

Contents

Chapter 1

Introduction

1.1 Problem Definition and Motivation

Human-Robot Interaction (HRI) is a rapidly developing research field, with the programming of robots representing a fundamental aspect of its various subfields. The main goal is to understand, design, and evaluate robotic systems that can interact with humans in a safe, efficient, and intuitive manner. Collaborative robots representing an integral component of this approach, as they are designed to work alongside humans in shared workspaces, enhancing productivity and efficiency. Within the context of reintroducing the current industrial revolution wave, namely Industry 4.0 (Figure 1.1), solutions to the problem of creating more intuitive and flexible methods and techniques for robot programming have to be reinvented in order for this phase to achieve its main goal. That being said, a potential course of actions could be towards the characterisation of the aforementioned goals by enhanced Human-centered Robotics (HCR) collaboration environments with improved HRI metrics. The latter could be accomplished by integrating different cutting edge technologies in the local industrial ecosystem.
The main problem with the conventional robot programming methods, namely teaching pedant, offline programming etc., is the prerequisite from the user's side to have advanced knowledge of high-level programming, which can be difficult to acquire. In addition, some of the aforementioned methods can be unsafe and inefficient during their realisation. Robot Programming by Demonstration (PbD), being a relatively new programming method and an active research field, constitutes an easier method to learn and can enable more efficient and accurate programming. However, it still has its limitations in its current form. One drawback is the limitation of its applications, since the robot must be able to recognise and understand the task that is being demonstrated. To that end, modern data-driven exploration techniques can be implemented to train the robot for executing a certain task and learn new skills. Moreover, graphical programming and scripting languages can be used that introduce a more visualised way to program and deliver faster results in a more convenient manner. Whereas this approach positively affects the learning

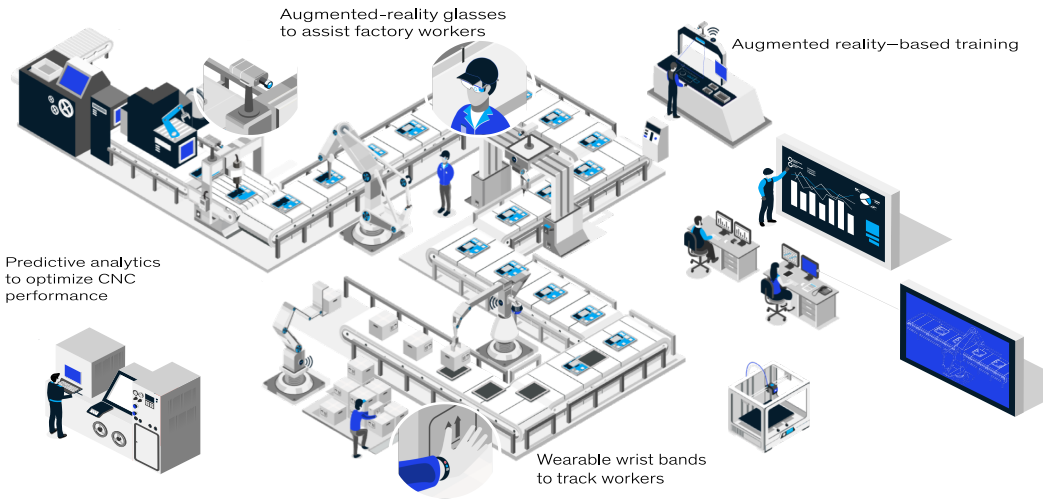


Figure 1.1: Rendering of the proposed Industry 4.0 environment by [McK]

procedure outcome from the robot's side in a more efficient fashion, it renders the user's perception of the system incomplete. Thus, it impacts negatively the intended outcome and fails to improve the overall procedure. This is due to one of the main challenges in PbD, which is the knowledge gap between the user (teacher) and the robot (learner). The user may have a deep understanding of the task being taught but may not have the technical expertise to program the robot. On the other hand, the robot may have the technical knowledge to execute the task, but may not understand the underlying concepts that the user is trying to implement.

A solution to that problem can be the addition of another rapidly evolving technology, that of Augmented Reality (AR) (or Mixed Reality (MR) for interactive frameworks that enable virtual objects to interact with the real world, creating a seamless hybrid environment) via the usage of Head Mounted Displays (HMD). PbD that utilises AR techniques, can enable robots to learn complex tasks quickly and accurately and can be used to develop new processes that can be adapted to changing or specialised conditions and constraints, ultimately improving the productivity and effectiveness of the procedure. In that regard, AR and MR are ideal for enhancing the physical reality and the surroundings of the user by means of virtual object augmentations and by providing an immersive experience that could also be applied with great benefits in the field of HRC.

As it is presented more detailed in Sec. 2, there are different techniques that are used towards the direction of utilising Augmented Reality methods and tools for achieving better performance on the programming of robots, techniques that their intuitiveness, as is claimed by the authors of the authors of those researches, can ameliorate the performance metrics for experts and non-experts alike. It is obvious that complementary to what the performance indices suggest, the main endeavour of any work that has been done in this field should focus on creating a more immersive interface that the user would not only find more intuitive and advantageous, but

also increase the enthusiasm and eagerness to experience it.
Furthermore, AR-enhanced PbD can be used to provide remote guidance and troubleshooting, allowing for faster problem-solving and more efficient maintenance procedures. Regarding the user experience (UX), combining PbD with AR methods can provide an immersive and a more interactive experience, which can further improve the safety and reliability of the system since the user works in its entirety with holograms. In this context, AR can help bridge the aforementioned knowledge gap by providing an intuitive and interactive interface that allows the user to communicate with the robot in a more natural and effective manner. With AR, the user can use physical gestures and actions to demonstrate the task to the robot, which can be captured by cameras and sensors and then mapped to the robot's control system. This way, the user can teach the robot without having to write complex code and the robot can learn from the user's demonstrations.

1.2 Research Goals and Novelties of the Proposed Approach

The ultimate goal and focus of this thesis lies at the endeavor of enhancing Human-Robot Interaction, and delving into the potential impact of merging two cutting-edge techniques within this research domain. Consequently, one could argue that the overall novelty is the design and development of an interface that aims at a captivating and fully immersive experience for users, whereby seamless interaction with robots is made possible through the integrated AR-PbD framework. One of the most important issues that need to be tackled is the ability of the external agent, namely the human user, to handle demonstrations in the best way possible. The variability of human efficiency, the demonstration quality during the teaching process and the knowledge (tasks) across human subjects, are only few of the evaluation metrics of the teacher's performance. To that end, the dependence on the teaching platform and the human factor are one of the research-aims listed for this thesis.
As mentioned above (and more extensively described in Sec. 2), a thorough literature review, reveals that there is a knowledge discrepancy between the teacher and the learner in Programming by Demonstration. Therefore, the need for providing the user with structured information online (while programming), regarding the robot's skills and knowledge, is imperative. This is another one of the main research aims of this work. Some of the questions that need to be further addressed are the following:

- How can we optimally design the AR interface to increase the intuition of the user regarding the teaching procedure?

- How do we create a robust and adjustable virtual robot control scheme for manipulation procedures?

- What is the best way to visualise the informational retrieval and its propagation/representation for the user to understand the status of the robot?

- How do we visualise the virtual content in order to render it in a more enjoyable way for the user (user friendly aspect)?

- How does the use of AR during PbD affect its usability, the user's understanding and the mental workload?

- How does the user benefit from a more intelligent PbD algorithm that also makes suggestions by being able to identify the demonstrated skills?

- Research results, such as those mentioned in the previous section and were presented in [SKG+16, QLP+18], do not yet agree on the effects of AR on teaching. A premise which begs the question in which situations is it better or worse to use an AR as a tool to accelerate and improve the learning/teaching process.

Many approaches of different AR applications on robot programming are trying to enhance the physical world with virtual objects and schemes that help the user understand the abilities of the robot and undertake the task of increasing the performance of the programming procedure. However, the incorporation of the learning procedure is being done in a more standardised way without adding some extra layers of information to visualise this too for increasing the user's intuition about the system. One possible novelty of our approach is the addition of a graph-based knowledge representation, where the previously demonstrated skills are sequenced on an abstract task-level, a tool through which the teaching process gains a significant advantage over more traditional PbD techniques like kinesthetic teaching.
In Figure 1.2 a generic example of the aforementioned graph is depicted. This task-representation graph assign its nodes as decision states and the skills are represented by the edges. The user is not only able to understand the task-levels through the graph, which depicts this sequence of logical skills and sub-goals of that task which the robot is trying to learn, but also can enable an interactive way to execute each skill and decision state individually by selecting the respective node or edge.
We also provide the user with the option of choosing different ways through which the robotic manipulation can be executed. Different options enable a more immersive interaction with the physical world, for example automatic virtual trajectory execution or individual joint manipulation can offer the user a more spherical view of the task at hand and how to choose the best way to execute the teaching/programming and thus the type of the manipulation. Another functionality that is included in the developed interface and could be used as an argument in favor of the case for the combination of AR (by means of an HMD) with PbD, is the execution of kinesthetic teaching through AR interaction modalities like gesture, gaze and touch. This

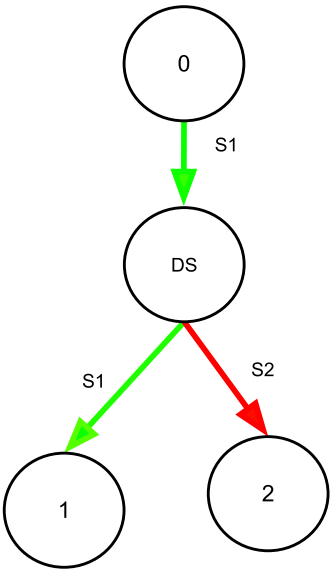


Figure 1.2: Task graph concept [WEL20]

feature enables a more sophisticated way to kinesthetically teach the robot and simultaneously minimises the physical interaction with the robot creating the perfect conditions for a safer manipulation and demonstration procedure.
The thesis is structured as follows: In chapter 2 the most important parts of the state of the art in Programming by Demonstration, Augmented Reality and Intuitive Robot Programming are reviewed. In this chapter the definitions of basic terms and methods that were used or implemented in the context of this thesis are given, like Learning from Demonstration, Inverse Reinforcement Learning, Augmented/Virtual/Mixed Reality, as well as to the general context of Intuitive Robot Programming approach. Furthermore, chapter 3 presents in detail the proposed approach, the theoretical scheme and high level of the system's architecture as well more technical details regarding implementation and hardware that was used during the development phase.It also discusses the concept of Human Robot Interaction (HRI) is discussed. chapter 4 deals with the experimental evaluation of the developed interface and presents the hypotheses and observations deduced from the procedure. In the concluding chapter 5, the report provides a short summary of the integrated system, the overall purpose of the thesis, as well as its outcomes. Additionally, some suggestions for prospective future research directions are given.

Chapter 2

State of the Art in Programming by Demonstration and Augmented Reality

Programming by Demonstration (PbD) and Augmented Reality (AR) are two rapidly evolving technologies that have gained significant attention in recent years. PbD allows users to teach a robotic system by demonstrating a task rather than programming it explicitly. On the other hand, AR provides an interactive experience by overlaying digital information on real-world objects. Both technologies have immense potential and are being explored for various applications from industrial automation to healthcare. In this chapter, a comprehensive review of previous work and recent advances in those scientific areas is provided. The foundations are set, for exploring the connecting points upon which a unification of these two methods can offer potentially useful insights on how to approach the problem of Intuitive Robot Programming, and hence the Human Robot Interaction (HRI).

2.1 Robot Programming by Demonstration

Robot *Programming by Demonstration* (PbD) or *Learning from Demonstration* (LfD) (also mentioned as *Imitation Learning*) is the paradigm in which robots acquire new skills by learning to imitate an expert [RPCB20]. PbD constitutes a method of robot programming that involves teaching the robot how to perform a certain task by demonstrating that task. In its framework, the user is relieved from writing extensive and generic controllers that would enable the robot to adapt in different scenarios. The development of such a controller would require a fair amount of advanced reasoning and knowledge implementation from the human side incorporating a very sophisticated error handling algorithm and very expensive non-pragmatic concept to materialize. After the robot has learnt from a series of demonstrations, it imitates the learnt behaviour and it repeats the task autonomously when called

upon.
The quintessence of Imitation Learning lies in the inherent ability of complex multicellular organisms like mammals to imitate certain behaviours, as part of their low-level evolutionary encoding. Its emergence originates from the bio-inspired imitation process of learning a new skill used by both humans and animals. In the human and animal world, imitation is often observed during the early stages of life, either instinctively for survival purposes or through social interaction and information exchange with other beings of the same or different species. There are two types of biologically inspired imitation systems: the conceptual, where the system mimics its biological counterpart behaviour and tries to mimic it, and the connectionist models, where some low-level artificial neural network is used [Sko09]. In robotics, this behaviour can be programmed into intuitive interfaces that can also be used by novice users without prior programming experience. Some key questions that could decipher the process of imitation learning, and determined in a sophisticated way in [ND07], are the following:

- Who to imitate? That question refers to the problem of who is the expert in the case of Human Robot Interaction.
- When to imitate? What is the timespan of the imitation and when is the best time for it to take place.
- What to imitate? What is the goal of performing a certain skill.
- How to map? The translation from a demonstration to actual imitating behaviour is not straightforward and usually some encoding procedure of the acquired information has to take place.
- How to evaluate an imitation? A question that concerns implementation of self-improvement techniques for the learnt skills by defining and using the proper metrics.

These questions play a pivotal role on how to approach the problem of Intuitive Robot Programming by means of Programming by Demonstration and should be in general thoroughly investigated during any related research conduction.

2.1.1 Skill Encoding and Generalisation

Through PbD, non experts are enabled to intuitively transfer new task knowledge to the robot through different learning modalities according to the respective situation applicable to the system. Each task can break down to individual skills in order to be demonstrated to the robot. Those skills can be imitated multiple times on alternate scenarios and the desired behaviour can be encoded. That is accomplished during the skill encoding phase of learning a skill. The other important phase is the generalisation of the learnt behaviours, previously acquired by the demonstrations.

During this phase the robot can assimilate the subtle differences of the modified task and adapt accordingly. These two concepts are the main subject of this section and are analysed below.

The skill encoding process is one of the main challenges in PbD, which is how the acquired knowledge is represented and perceived by the robot. Skill representation can be done either at low-level, namely a non-linear mapping between sensory and motor information [BCDS08], where the encoded information (i.e. a particular movement) provides part of the solution to the main question of what is important to be imitated [Cal09], or at high level, namely a skill decomposition of action-perception units that symbolically code predefined motion elements [Cal09, AC05]. These representations, also depicted in Figure 2.1 are referred to as trajectory-based approaches and symbolic approaches respectively.

In more detail, the symbolic level of encoding, also mentioned as task level, is using a sequential form of skills represented in symbolic way [EK08, PKDZ07], which is possible to be identified by means of classification techniques and can be used to create a type of hierarchical structure. Those structures usually are tree-like and represent some specific state-action as nodes and relations respectively [NOK+15]. Upon skill reproduction, that sequence is mapped on the robot's own action set.

On the other hand, the trajectory level is related to the mapping of the demonstrated trajectory to the robotic one [vLGG19] using statistical modeling and dynamical models and encoded into new robot actions or primitives.

Other levels of encoding imitation includes goal-level approach, where an inference process derives the goal after observing a specific set of actions, and model-based level where an exploration-based learning model is used. A combination between the symbolic level and trajectory level has also been implemented in an effort to fuse the benefits of both methods in [KPK+15], where a form of sensor information are synced with movement primitives in order to create the concept of Associative Skill Memories (ASM) previously introduced in [PKRS12]. These are based on a small set of stereotypical movement primitives but suffice in order to create large manipulation skill-sets, in an effort to imitate the auto-associative memory function of the human brain too and process various sensory readings. They postulate as well that a manipulation graph can be incorporated into the system in order to encode in a state machine with imposed constraints in the manipulation sequence.

The second important phase during learning in PbD is the generalisation of a skill. Generalisation refers to the ability of a robot to adapt and apply the learned skill to new situations, task conditions, or environments that were not encountered during the initial demonstrations. In other words, generalisation is the process of extending the acquired knowledge to perform the skill effectively under varying conditions. The objective of generalisation in PbD is to enable robots to handle novel circumstances without requiring explicit demonstrations for every possible scenario and it contributes to the robustness, and flexibility of the learned behaviors.

The generalisation of symbolic level encoding involves sequential organization of predefined action elements, an instance that enables the learning of hierarchy, rules and

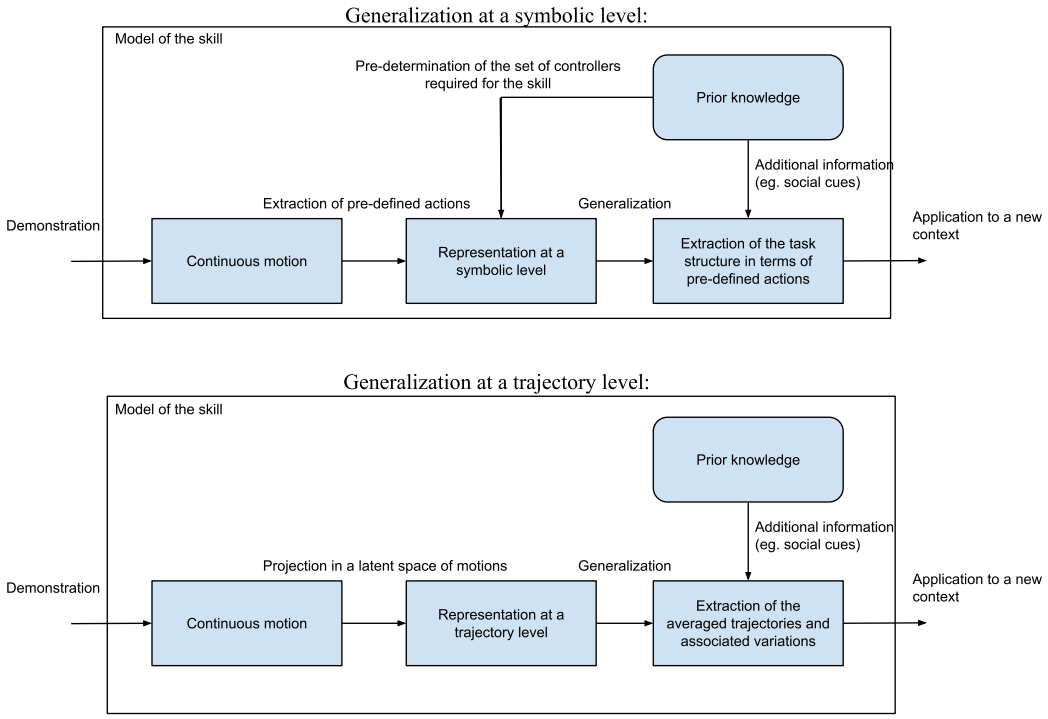


Figure 2.1: Different levels of representation for describing the skill [Cal09]

loops but also requires pre-defined set of basic controllers for reproduction [Cal09]. On the other hand, on trajectory encoding level, the generalisation span includes the movements generalisation. This generalisation could be averaged trajectories and related variations, with advantageous behaviour when it comes to generic representation of motion. The advantage lies on the capability to encode very different type of signals or gestures, but with the drawback of the inability to reproduce high-level skills with high complexity [Cal09]. Generalization can be achieved through various methods, depending on the representation used for encoding the skill. Some approaches include:

- Dynamic Movement Primitives (DMPs): DMPs can be adapted to new task conditions by changing the goal position, adjusting the movement duration, or scaling the amplitude of the movement. This allows the robot to perform the skill under different circumstances.

- Gaussian Mixture Models (GMMs) and Gaussian Mixture Regression (GMR): GMMs can be used to model the joint probability distribution of the demonstrated trajectories and their corresponding task parameters. GMR can then be employed to estimate the most likely trajectory for a given set of task parameters, facilitating generalization to new situations.

- Neural Networks (NN): When using NN to learn a skill, the network can be trained to predict the appropriate actions or trajectories for different task

conditions or environments. Once trained, the network can generalize to new situations by adjusting its internal weights based on the input conditions.

- Reinforcement Learning (RL): PbD can be combined with RL to allow the robot to refine and adapt the learned skills through trial and error. This enables the robot to generalize its behavior and improve its performance in varying conditions. This method is analyzed more in section 2.2.

A typical generalisation technique is the use of Dynamic Movement Primitives (DMPs) introduced in [INS02]. DMPs represent complex motor movements by decomposing them into temporal (canonical system) and spatial (transformation system) components. The robot observes a demonstrated skill, extracts the trajectory, and learns a DMP by fitting basis functions to the trajectory. The learned DMP can be modified to adapt to new task conditions or environments by changing the goal position, adjusting the movement duration, or scaling the amplitude of the movement. DMPs capabilities have been extended in many ways with Gaussian Processes (GP) [MEO+17]. More specific, the generalisation for the trajectory encoding case is effectively decreasing the inherent uncertainty factor of the human demonstration which can be imperfect in some cases. Chernova et. al conducted researches [CV07, CV08], where the aforementioned problem is addressed by introducing a different representation of the learning policy as a set of GMMs, with multiple GMM components of each model which corresponds to a single action. In their approach the training data for the GMMs set were basically the demonstrations examples that were incrementally received, ideally in a way that minimises the number of demonstrations needed to acquire the policy.
At this point a succinct definition of GMM framework should be given. A Gaussian mixture model is resulting from a combination of several Gaussian components [TSM85]. A 1-d Gaussian component or a 1-d Gaussian probability density function (pdf) described by the following distribution with mean μ and variance σ^2:

$$\mathcal{N}(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-(x-\mu)^2}{2\sigma^2} \tag{2.1}$$

A GMM is characterised by the vector Θ of the means, variances, and weights of its C components. The weight of each component is the portion of samples belonging to that component [CV07],:

$$\Theta = \{\mu_1, \sigma_1, \omega_1, ...\mu_C, \sigma_C, \omega_C\} \quad \ni \quad 0 < \omega_c \leq 1 \quad \text{and} \quad \sum_{c=1}^{C} \omega_c = 1 \tag{2.2}$$

The pdf of a GMM parameterised by the Θ vector, is defined as a weighted sum over the Gaussian distributions:

$$p(x|\Theta) = \sum_{c=1}^{C} \omega_c \mathcal{N}(x;\mu_c,\sigma_c) \tag{2.3}$$

The most popular method to estimate the parameters of the GMM, provided that a set of data-point is given, is the Expectation-Maximisation algorithm [DLR77] (EM) where maximum likelihood estimates are used to iteratively optimise the GMM. In Figure 2.2 an illustration of GMM is provided.

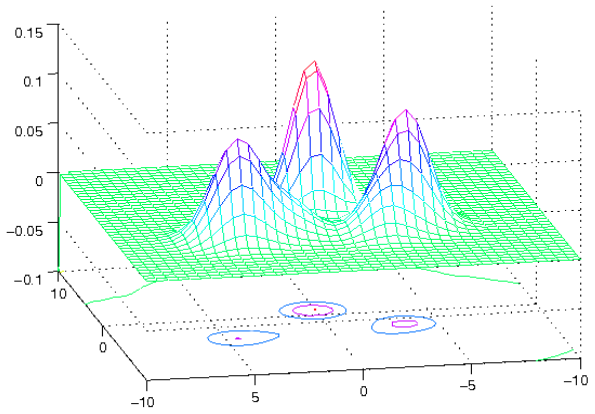


Figure 2.2: Example of 2D Gaussian Mixture Model with three components [CV07].

Gaussian Mixture Regression (GMR), which are complementary to GMMs [CAL16], is used to retrieve the conditional distribution at each time step and obtain the generalised reproduction [YYD22]. Given a new input, the trained GMM is used to compute the conditional probability distribution of the target variable, taking a weighted average of the Gaussian components conditioned on the input. This step results in a predicted output (e.g., desired joint angles, positions, velocities) for the robot to execute the skill in a new situation. The GMR provides a generalised trajectory, as well as a covariance matrix, for every time step

2.1.2 Input Modalities and Demonstration Procedure

As previously mentioned, at the beginning of this chapter, the demonstration procedure can be performed in various forms, with the main attributes of defining a form being the used modality, through which the necessary information, in order to construct the knowledge base, is obtained. The information retrieval can be done through various sensor inputs and with the implementation and usage of different interfaces, that function as a semantic translation for the user to understand and use that information basis accordingly. Input modalities in PbD refer to the different ways in which the user can demonstrate the desired task to the robot, including physical actions, such as gestures or movements, or verbal descriptions of the task. Effective interface design is critical in PbD, as it directly impacts the user's ability to communicate the desired task to the robot. A well-designed interface should be intuitive, easy to use, and provide real-time feedback to the user. There are several types of interfaces that can be used in PbD, including:

- *Immersive Interfaces*: This VR/AR interfaces use immersive technology to create a virtual environment that can simulate real-world scenarios, in the VR case. On the other hand, AR interfaces use computer-generated images and information to augment the user's view of the real world, thus providing an intuitive and interactive interface for the user to demonstrate the task to the robot by enabling many different layers of functionalities. Users can interact with virtual objects and demonstrate tasks to the robot in a more natural and intuitive manner. VR interfaces are particularly useful for tasks that require a high degree of precision or for tasks that are difficult to simulate in the real world, whereas AR interfaces can be particularly effective in PbD as they allow the user to directly interact with the robot in a more natural and immersive manner while the user interacts with the physical reality as well. This topic is more extensively addressed in the section 2.3

- *Graphical User Interfaces (GUIs)*: These interfaces use visual representations, such as buttons or menus, to enable the user to interact with the robot. GUIs can be effective in PbD as they provide a familiar and intuitive interface for the user. A notable example is that of FRANKA EMIKA's GUI, which is depicted in Figure 2.3, enables the use of many different processes through one Application.

Figure 2.3: The GUI used for FRANKA EMIKA products, that utilizes a workflow-based robot programming approach [EMI].

- *Natural Language Interfaces (NLIs)*: NLIs use natural language processing (NLP) techniques to enable the user to communicate with the robot using

natural language. Natural language interfaces for robot control aspire to find the best sequence of actions that reflect the behaviour intended by the instruction [HTR14]. NLIs can be useful for users who may not have the technical expertise to use more complex interfaces.

- *Voice User Interfaces (VUIs)*: VUIs, or Spoken Dialogue Interfaces (SDIs), leverage spoken language, which is the most natural and potent mode of communication between humans, to enhance the usability of human-machine interfaces. The degree of flexibility and robustness in handling spoken input and output depends on the specific type of Spoken Dialogue Interface (SDI). Consequently, Spoken Dialogue Systems (SDSs) may range from relatively basic finite-state systems designed to handle a restricted set of commands to more sophisticated systems that can perform inference and planning as part of a collaborative interaction approach.[SSK20]. VUIs can differ from NLIs mainly in the modalities, where the later can incorporate text-based communication and speech input.

- *Tangible User Interfaces*: Tangible User Interfaces (TUI) use physical objects, such as blocks or toys, to represent the task being taught. The function of these blocks is to provide annotations for objects, locations, or regions, and to define actions and their sequence. The robot, on the other hand, utilises these blocks to identify objects and blocks in its workspace and to arrange them into instructions by resolving any associated constraints [SAC17]. Users can manipulate these objects to demonstrate the desired task, providing a tactile interface for the user. Tangible interfaces are particularly useful for users who may have limited technical expertise or who prefer a more hands-on approach. In [SD13] the authors compare the effects that a tangible system has to a group of children of various ages who have to follow the instructions of a robot programming task, with the performance on a GUI. The results indicate that the tangible system is more cumbersome for older children who are familiar with computers and not the easiest interface paradigm to use. It is also more enjoyable as a learning media in both cases of the older and the younger ones. For younger children that are not used to computer interaction the TUIs were easier to use. In this case the TUI was applied on physical blocks without a projected table. An example of a TUI is depicted in Figure 2.4

- *Brain-Computer Interfaces (BCIs)*: BCIs use electroencephalography (EEG) or other techniques to measure brain activity and translate this into commands that the robot can understand. In this type of interface, commands can be conveyed using various modalities such as screens, speech, or manual marking of target objects with a laser pointer [KVBT20]. The problem that arises in this case, concerns individuals with disabilities who may face difficulties in reliably controlling the robot using conventional modalities. To address this issue, brain signals have been proposed as a feedback modality in command-and-

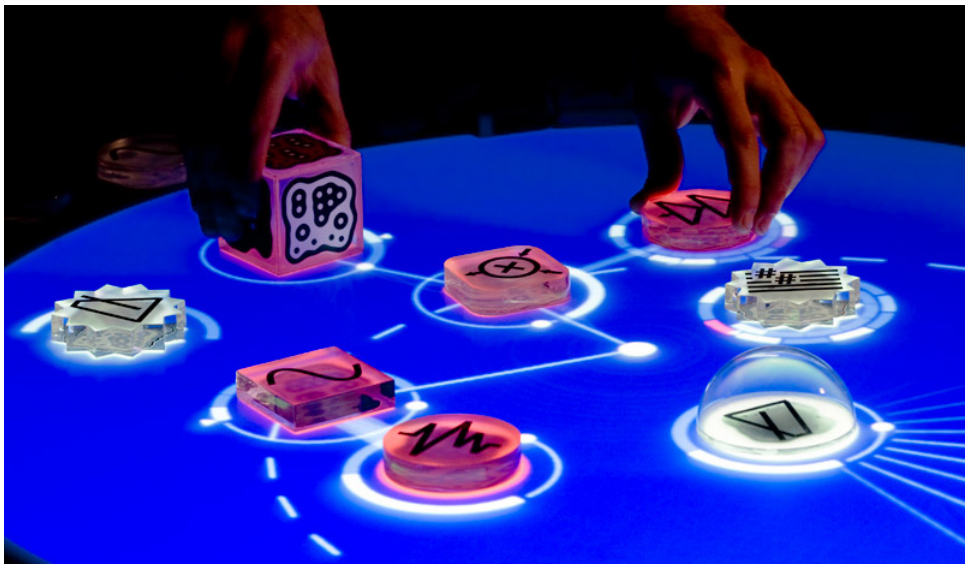


Figure 2.4: Tangible User Interface on a projected table [vLGG19].

control scenarios within robotics environments. These approaches typically employ screens for stimulus presentation, while approaches based on mental imagery do not necessarily require stimulus presentation [KVBT20]. BCIs are particularly useful for users with physical disabilities or for tasks that require a high degree of precision.

The choice of different interface, and therefore the selection of input modalities, can significantly affect the demonstration procedure in PbD. For example, using a tangible interface may be more effective for tasks that involve physical manipulation or dexterity, whereas using an AR interface may be more effective for tasks that involve complex spatial reasoning or visualisation. Similarly, the choice of interface type can significantly impact the user's ability to communicate the desired task to the robot, and may depend on factors such as the user's technical expertise, familiarity with the interface, and level of interactivity required. The biggest challenge yet is of course the development of an interface trough which a more generic and adaptive type of robot programming could be realized.
The most notable learning modalities used in PbD are directly related to human social interactive concepts of teaching and most widely used techniques to provide demonstrations to the robot, like observational learning, kinesthetic teaching and teleoperation settings [Cal18].

1. **Observational learning**, as the term indicates, is the type of learning based on the visual input of the learner on the demonstrator performing a certain task [Cal18]. The main input modalities in that case are found in the domain of vision systems and motion recording devices like gyroscopes and accelerometers. The use of three-dimensional motion capture systems enables the replication of a user's motion onto a robot, which can widely be used in

applications of humanoid robots. The system is comprised of three main components: measurement of human motion, mapping of motion from a human to a robot, and control of robot motion [CL19]. There are two types of learning categorized based on the way the observations are done: direct and assisted type. In the direct case, the video modality is observed without help whereas in the second case the learner is provided with extra tools, like [LGAL18] Deep Neural Networks for optimisation to identify and track the observed interactions [Pau21]. The assisted methods are using sensing modalities like trackers, visual detectors, motion capture and more visual inputs.

2. **Kinesthetic teaching** is the mode where the user guides the robot through a specific trajectory by using physical contact and haptic guidance as inputs. In this modality, the robot is set in gravity compensation mode for easy control [AS11]. By means of physical interaction, is introduced a more natural way to propagate the necessary knowledge for demonstrating and reproducing or refining a skill, thus enabling the user to interact in a more tactile way and enhance his/her perspective on the robot's capabilities [CL19]. Kinesthetic teaching as a method to provide demonstration, poses a good solution to the correspondence problems which refers to different mapping challenges, like human to robot mapping function, and capability of adapting to different scenarios. Drawbacks include, hindrances on teaching tasks via demonstrations, that require good synchronization between multiple limbs [BG13] . Through kinesthetic teaching, one can also generate a good primitives generation source. When learning in kinesthetic mode, the possibility for the robot to incrementally learn a task is possible to be exploited as well. Approaches for incremental learning through the use of multiple learning modalities provide an intuitive way of teaching natural movements and ensure the synchronization of complex whole-body motions. The learning process typically begins with observation learning, which involves the transfer of whole-body motion from a human demonstrator to a robot. Subsequently, the process involves refining kinesthetic motion [CL19].

3. **Immersive teleoperation** where haptic interfaces can be used and the robot's sensory and effectors can be exploited [BG13]. The user is limited on that regard but on the other hand one of the advantages is that of the training from distance. The other advantage is again the solution to the correspondence problem. In contrast to kinesthetic teaching, which confines the user to the physical body of the robot, the learning modality of teleoperation aims to restrict the user's perception to that of the robot as well. Teleoperation can be accomplished using joysticks or other remote control devices, including haptic devices. In most of the cases, teleoperation is a method majorly used to transmit the kinematics of motion, and it is highly correlated with the ability of the teacher to operate the remote control device effectively which naturally requires training. Another more contemporary approach on the teleoperation

scenarios, is the use of AR and VR which is analysed in more detail in section 2.3.

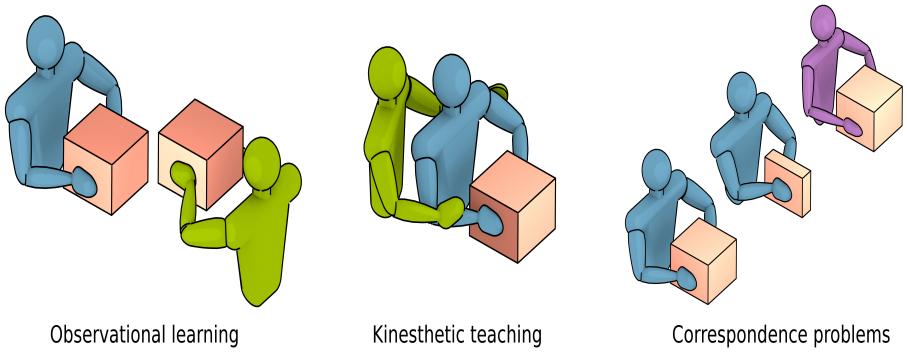


Figure 2.5: Input modalities and the correspondence challenges in robot Programming by Demonstration [Cal18].

Each learning modality and hence teaching interface presented above, performs better at some levels comparing the rest thus being suitable to applied in specific tasks, and each one has limitations. Research has commenced to explore how these interfaces can be combined to take advantage of the complementary information provided by each modality separately [BG13, SAMB12].

2.2 Intuitive Robot Programming

The term Intuitive Robot Programming (IRP) is mainly referred to the paradigm of Programming by Demonstration and the effort to develop interfaces that would demand no prior knowledge or high expertise, thus making them accessible to a broader range of users, including those who do not have specialized programming knowledge or related experience. One of the main challenges in traditional robot programming is the complexity of the programming languages and the need for precise, explicit instructions. IRP aims to simplify this process and make it more accessible by allowing users to program robots through more natural means.

2.2.1 The Teacher's Role in Programming by Demonstration

As stated in previous sections, a crucial aspect of making LfD feasible in real-world applications is to guarantee the robot's ability to learn skills efficiently by adapting to uncertainty during operation and reducing the teaching effort for the instructor. Therefore, it does not suffice for the robot only to learn to perform a demonstration but an indispensable part of the teaching process is the generalisation as well if it were to perform better.

In contrast to what was presented in previous section [CV07, CV08], where the subject of Programming by Demonstration was approached from the robot's perspective and how it learns, a more holistic approach on the improvement of the complete robot programming concept should include a thorough investigation on the teacher's perspective and understanding of the system. One notable argument on that regard, presented initially from Sena et. al [SH20], is that the discrepancy between the teacher's belief of the system's knowledge and its actual capabilities, created by the poor quality of the input data mentioned previously, is a significantly unexplored area that can be potentially problematic and lead to the following major issues:

- undemonstrated states
- ambiguous demonstrations
- unsuccessful demonstrations.

These issues can have a negative effect on the assimilation of the robot and its familiarisation to the task during demonstration and thus the skill behaviour reproduction. It is important for the users, in order to minimise this effect, to share some core mental elements. This would enable a behaviour from the teacher's side, during the complete teaching phase, with epicenter the understanding of the questions of *how* and *what* the robot learns [HB18, CT14, SH20].
In their approach Sena and Howard are emphasising on the feedback design for the PbD and by setting the right evaluation metrics, are trying to enhance the teacher's role and contribution to the learning procedure. Figure 2.6 depicts an imperfect PbD model where the teaching framework and the interaction between the central factors of the procedure, are represented in graph form with the relationships of the individual parts also being mapped.
The model's framework maps the learning processes into three different spaces:

1. $\mathcal{J}_h$ is the information history space.
2. Π is the policy space,
3. $\mathfrak{M}$ is the teacher's belief system.

Within their framework the information history space is containing elements like initial conditions, observations, and action sequences, including incorrect elements. The policy space represents all the possible policies learnt. Finally, the teacher's belief space contains information related to the extension of the robot's knowledge base and skill acquired to perform a specific task. For evaluating the teacher's intuition two metrics were defined, the first one is that of *teacher's efficacy* which is related to the primary teacher's objective, that of accurately teaching a skill. That is satisfied if the Π set (the policy space) contains a policy π that can generate close

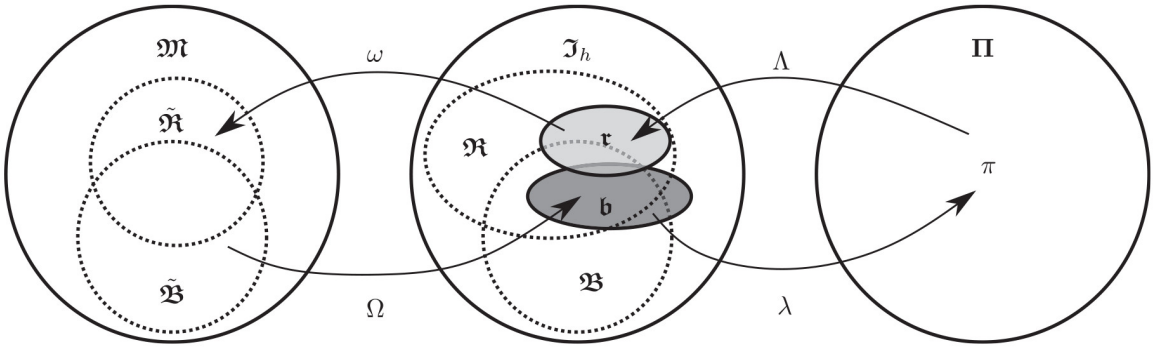


Figure 2.6: Space set representation of the discrepancy between the user's understanding of the system, the task realisation and the learning policy [SH20].

to the goal skill trajectories. The second metric is that of *teacher's efficiency*, which is defined by the minimisation of the total number of demonstrations required.
The most important conclusion of the quantification and evaluation conducted in [SH20], is the significance of the feedback of a system which not only improves the teaching procedure and the learner's performance on the task at hand, but also leads to greater consistency. With the feedback system focusing on the needs of the learner, it operates as a determinant factor on what should be demonstrated by the teacher, excluding that way, the need to understand how learning is realized [SH20]. Some interesting results and insights that emerged from their experiments are summarized to the following points:

- the form of feedback (4 different feedback conditions: No Feedback (NF), Replay Feedback (RF), Batch Feedback (BF), Selected Feedback (SF)) is beneficial to the learning process by improving the teacher's efficiency, which is evaluated by the ability of the learner to successfully generalise.

- there are common pitfalls novice teachers encounter, like premature stopping of the teaching procedure or inability to provide demonstrations for adequate time period.

- participant's guidance is essential to retain a pragmatic sense of the quantitative and qualitative nature of the required demonstrations,

- the often falsely derived teacher's confidence while determining the learning ability of the robot.

2.2.2 Task-Level Learning and Collaborative Programming

Another approach to developing intuitive interfaces for robot programming, is the addition and combination of methods that exploit high-level behavioural techniques to represent knowledge and skills for a task. Some recent researches are tackling

the skill handling problem with visual programming languages and task-level approaches, as for example, in [SNS19] a combination of PbD with Task Level Programming (TLP) is presented. By means of TLP, the robot is programmed to perform a task in a manner similar to how a human would perform the same task. Instead of specifying individual motor commands, the programmer defines a series of abstract actions that the robot must perform to complete the task. These abstract actions can be defined using natural language, graphical interfaces, or other high-level programming languages. The advantages of the TLP make PbD more:

- intuitive with robust results, since the error-handling can be incorporated into individual skills
- safe, since the skills can be certified,
- generic, the skills can be parameterised for different task variations.

According to Steinmetz et. al, combining the TLP with PbD into Task-Level Programming by Demonstration (TLPbD) the advantages of both methods are the only remainders. TLPbD enables experts to semantically annotate robot skills with their conditions and effects by means of Planning Domain Definition Language (PDDL) descriptions of the available skills, allowing that way online recognition from demonstrations to be realized by non-experts.
This is achieved through the core component of the system, the *semantic skill recognizer*, which sends parameterised skills to the TLP interface, when the latter sends a signal to verify that the current status of the system is the correct one. The recognizer creates the skills based on the *world model* which contains more abstract representations than only raw trajectories, updated by the *world observer*, and the PDDL descriptions of the available skills [SNS19].
In an effort to make a more interactive robot programming interface, Willibald et. al combined in [WEL20] an incremental framework where the user and the robot are able to program complex tasks in a collaborative way which is called Collaborative Incremental Programming (CIP). Two of the main characteristics that this framework enables is an on-line anomaly detection algorithm and the ability to refine an existing skill, or demonstrate a new one. A high-level graph which represents the complete sequence of actions and states of the task at hand, is used for visualising the whole procedure with purpose to increase the intuition of the user for the system.
Low-level statistical skill encoding is used to balance out any local perturbation, and the model's inherent probability distribution can identify deviations from intended behaviour [WEL20]. The workflow of the algorithm is presented in the flow chart of Figure 2.7. The core functionalities of the CIP framework are the following:

1. Probabilistic encoding of demonstrations, where dynamics of the system can be integrated without the need for user guidance. This is achieved by having the robot repeat the trajectory of the user's demonstration and recording a

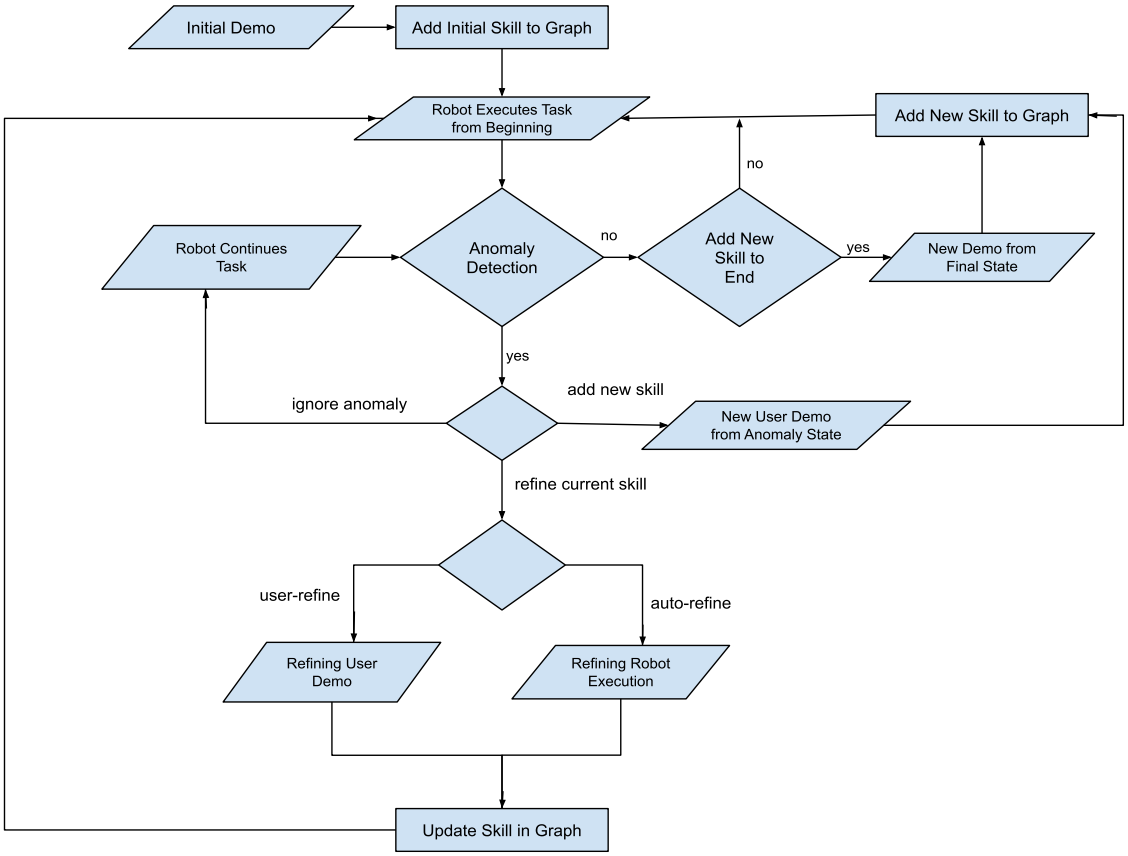


Figure 2.7: Collaborative programming with on-line anomaly detection for interactive teaching where the user can incrementally program the robot by adding new skills or refining existing ones. [WEL20]

second sequence and using the sensor readings as means to learn variability through several demonstrations and robot reproduction. While exactness is imperative for parts with less variability, higher deviations are tolerated during the execution of parts with high variability. This strategy makes the procedure more robust.

2. On-line anomaly detection, through which the measured sensor modalities are continuously compared to the commanded ones through a real-time monitoring segment. If the latter is detected, it is classified to a sensor category (end-effector position/orientation, force/torque, gripper opening and grasp status), a feature which allows the anomaly detection algorithm to make use of the robot's proprioceptive sensory systems and create a bidirectional dependency between the learning model and the actual robot.

3. Incremental graph generation, enabling the user to interact with the graph by choosing between two different actions of either adding a new skill to the graph or refining a current skill enabling a more interactive way to adapt the task-graph into any possible changes.

4. Execution of learnt behaviour is combined with the teaching process in a collaborative framework. Insertion of the decision states is done automatically at critical skill transition timesteps of the task. This inclusion decreases the necessary computing time of the decision-making process and eliminates perceptual aliasing. Therefore, the risk of selecting an incorrect skill is minimised.

2.2.3 Imitation Learning and Reinforcement Learning

In order to develop intuitive robot programming techniques and incorporate robotic systems more into our everyday life, Imitation Learning, as a collaborative learning approach based on human-robot interaction, offers the promising solution of learning to imitate the behaviour of a teacher by observing a demonstration of the task. The consensus would agree that learning only from demonstrations adds a limitation in the performance of PbD techniques and the abilities of the teacher. Those limitations could stem from one or more of the following:

- the expert's abilities to demonstrate a skill,
- an incomplete demonstration,
- ambiguity in demonstration (inconsistent or conflicting demonstrations),
- Teacher's ability to convey knowledge and the task's objective clearly.

It is self-evident that the user has a major role to play in this, since most of the limitations are a product of the teaching process. Clear, consistent and comprehensive demonstrations must be provided at any time. Ensuring the user's high level of expertise is also something recommended as well as the additional provision of demonstrations or feedback to correct any misconceptions or inaccuracies in the learnt behaviour.
To tackle this problem, PbD methods can be combined with exploration-based methods [RPCB20]. Although in some cases, for example the systems presented in [SPK02, CV07], it has been shown that in the context of Imitation Learning, statistical supervised learning significantly reduces learning time compared to exploration-based methods, there are still many that favor the latter approach [AN04, KP08]. Those cases includes the work of [GHCB07] where the described system uses a dynamic system generator modulated by a learned speed trajectory and reinforcement learning to enable the robot to adapt its trajectory when faced with new situations, such as obstacles. Also in [KP08], a framework that derives both policy gradient methods and Expectation-Maximisation (EM) inspired algorithms is presented. A novel EM-inspired algorithm is introduced, specifically suited for dynamical system motor primitives. Another approach in [KCC10] is presented, which utilizes an Expectation-Maximisation based Reinforcement Learning (RL) for modulating the mixture of dynamical systems derived from user demonstrations. The work in [AN04] explores learning in a Markov decision process without an explicit reward

function, relying instead on observing an expert demonstrating the task. The expert is assumed to maximise a reward function represented as a linear combination of known features. The proposed algorithm uses Inverse Reinforcement Learning (IRL) to recover the unknown reward function. The algorithm converges quickly, and although it may not precisely recover the expert's reward function, it outputs a policy with performance close to that of the expert, measured with respect to the expert's unknown reward function.

In exploration-based methods, the agent learns through trial and error by taking actions in the environment and observing the outcomes of those actions. The agent uses this information to update its knowledge of the environment and adjust its actions accordingly. This process is repeated iteratively until the agent learns to navigate the environment effectively.

One of the exploration-based methodologies that could be used for the improvement of the learning process is Reinforcement Learning (RL). Conventional RL is formulated as a Markov Decision Process (MDP) defined by a 5-tuple (S, A, T, R, γ) where S, A, γ are the state space and action space, and the discount factor respectively, $T : S \times A \to P(S)$, with $P(S_{t+1} = s' \mid S_t = S, A_t = A)$ is the stochastic transition function and $R : S \times A \to \mathbb{R}$ the reward function. A policy π, that is a mapping between state space S and action space A, is what the RL agent tries to learn with motivation and guidance through the cumulative reward function. Agent and environment are constantly interacting and exchange information on an action-reward feedback loop. The evaluation of the policy can be done with various methods, such as Monte Carlo (MC) or Temporal Differences. Policy improvement is also an important part of the learning process and can be done also with different techniques such as Greedy policy improvement or "Vanilla" policy gradients improvements. In addition, methods have to work in continuous state and action space and also be easily parameterised through function approximation techniques [SMSM99, KT99, PVS03]. With RL another important problem can be tackled, that of the limited performance of the robot which can only be as good as the performance of the human demonstration. To that end, RL can be used to explore new control policies through iterative search in the state-action space, which makes the method computationally expensive since the convergence rate can be of high numerical order.

Various approaches have been developed to combine Imitation Learning and RL with the goal of leveraging their strengths to overcome their limitations. Specifically, demonstrations can be utilised to initiate and guide the exploration process undertaken in RL, which ultimately reduces the time required to identify an enhanced control policy that may deviate from the demonstrated behaviour [BG13]. Demonstrations in the RL context, can be used as an initial point from which a first estimate of the iterative optimal policy search is computed [KP08, KCC10, JT13], or to generate an initial set of primitives [KCC10, BAC04, MKKP13]. When using primitives for task execution, RL can be employed to learn how to select among them. Demonstrations can potentially be used as a mean to constrain the search

space covered by the RL algorithm [GHCB07, PVS03], or to estimate the reward function [AN04, ZMBD08]. Furthermore, enabling the demonstrator to assume control over a part of the process during a single trial is a possible occurrence during the combined use of RL and Imitation Learning according to [RGB11].
In contrast to RL where the reward function is assumed to be known and the exploration process is guided by that assumption, Inverse Reinforcement Learning (IRL) is trying to derive the reward function of a Markov Decision Process (MDP), i.e. a discrete state-action space, given the transition function and a set of observed demonstrations in the form of state-action pairs [MH12, AN04]. The Generic IRL algorithm [MH12] reads:

1. Initialize reward function parameters w^0
2. Iterate from t = 1 to T:
   (a) Solve for optimal MDP value function V^* corresponding to reward function $\hat{R}(s \mid w^{(t-1)})$
   (b) Use V^* to define a policy $\hat{\pi}$
   (c) Choose parameters $w^{(T)}$ to make $\hat{\pi}$ more similar to demonstrations $O_{1:N}$ in the next iteration.
3. `Return` Reward function given by $\hat{R}(s \mid w^{(T)})$

Many IRL implementations are aiming in inferring a single reward function which describes the complete observation set.
The conventional IRL formulation in an MDP environment is used in [MH12] where the Bayesian Nonparametric IRL model is defined. In this framework, the robot observes a set of demonstrations represented as a time-ordered set of observed unique state-action pairs $O = \{O_1, O_2, ...\}$, where an observation $O_i = (s_i \in S, a_i \in A)$, and then it models the observed demonstration as an MDP but the partitioning of the observations is being done into smaller groups and the learning of a set of simple reward functions [PNP+20]. They also use the Chinese Restaurant Process model of the Dirichlet Process in order to define the probability distribution over the space of possible partitions. The inference part is applied for the most likely set of partitions and the corresponding goals.
In [PNP+20] the authors introduced an enhanced IRL formulation as the Constraint-based Bayesian Nonparametric IRL (CBN-IRL) framework, which consists an extension of the model presented in [MH12]. The reward and transition function are not specified in this MDP environment, instead the subgoal reward R_g and the constraint transition function T_c are given. Hence, the problem is to infer the decomposition of the demonstrated trajectory into smaller partitions and for each partition estimating the subgoal and associated constraints that maximise the likelihood of the observed trajectory [PNP+20]. This formalization occurs in a descretized state-action space. A set of predefined generic spatial features is used for representing the

world state, assuming the robot knows the geometry of environment and includes coordinate-free features. They also use subgoals and constraints as latent variables $\theta = \{(g_1, c_1), (g_2, c_2), ...\}$ which are inferred by the demonstration. That is achieved by inferring the MAP estimate of the likelihood of subgoals, active constraints, and assignment variables related to the demonstration.
In the context of intuitive robot programming, the research presented in [WL22] is incorporating an approach towards Bayesian Non-parametric GMM in combination with the Inverse Reinforcement Learning paradigm (BNGMM-IRL), a fusion of two different modellings of Bayesian Non-parametric GMM [MH12] and Bayesian Non-parametric IRL [Ras99]. The concept of interactive teaching introduced in [WEL20], where user and robot collaborated with goal to program task by demonstration, is the foundations to the updated version illustrated in Figure 2.8.

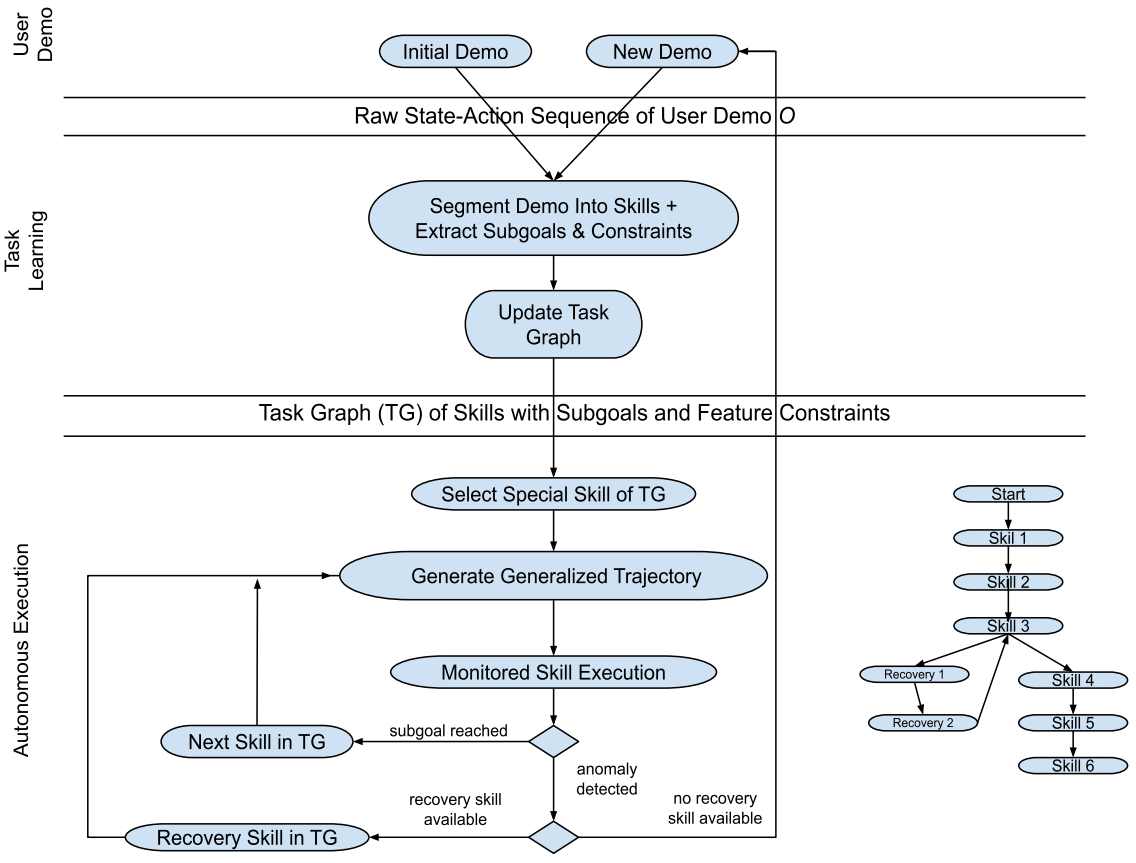


Figure 2.8: Updated interactive framework. The new framework provides the flexibility of switching between demonstration, learning and execution phase. The high level Task Graph (TG) in the right corner of the figure is updated from a data-driven segmentation algorithm which infers skills from the user demonstration.After the update the TG is provided to the robot for execution [WL22].

The approach in [WL22] is leveraging the IRL technique to infer feature constraints fitted individually to the skills of the task, which is segmented during the early stage of the procedure into simpler skills, through which the high-level task graph is updated, and use those features to detect anomalies during the autonomous execution

phase. Any anomalies detected can be dealt with, using the recovery behaviour of the task graph. The novelties are summarized in the following points:

- The segmentation algorithm which infers the low-level encoding of a task from the demonstration by combining intention recognition and feature clustering.
- The feature constraints of a segmented skill is inferred, thus enabling the anomaly detection to operate multimodally even after a possible generalisation of a skill.
- Learning process where represented by a high-level task graph, in which the skills are automatically parameterised by the low-level skill encoding.

2.3 Immersive Computing Technologies and Augmented Reality

In this section of Chapter 2 and before proceeding with the state of the art work on the field of Augmented Reality and Robot Programming, some essential definitions and context on the wider scientific area and spectrum of these technologies is covered, in order for the reader to comprehend the significance of the chosen methodology that is presented in Chapter 3.
Immersive Computing Technologies is a rapidly expanding field that encompasses many different technological paradigms of the wider Extended Reality (XR) area, with utter goal to create digital environments that not only mimic but also enhance our physical reality. At its core, immersive computing seeks to provide users with a fully interactive and immersive experience that engages all of their senses, including sight, sound, touch and even smell.
It refers to any technology that creates a virtual experience that fully or partially engages the user in a digital environment that befalls in the Virtuality Continuum [MK94]. This can include everything from Virtual Reality (VR) headsets that create fully immersive 3D environments, to Augmented Reality (AR) applications that overlay digital content onto the real world. A combination of technologies and techniques is required for that to be accomplished, including advanced graphics processing, 3D modeling, and sensory input devices such as haptic feedback, motion tracking, and even smell generators.
As mentioned above, different methods are included in the set of Immersive Computing Technologies, each with its own unique focus and technology stack. The most common subcategories of Immersive Computing include:

- **Virtual Reality (VR)**: VR is perhaps the most well-known subcategory of Immersive Computing. It creates a fully immersive digital environment that is completely separate from the physical world. This is frequently accomplished by wearing a VR headset that tracks the user's head movements and renders a 3D environment in real-time [CGRR18].

- **Augmented Reality (AR)**: AR, as mentioned in section 2.1, overlays digital content onto the physical world, enhancing or augmenting the user's experience of their surroundings by means of computer graphics and virtual objects [MK94]. This is usually achieved through a mobile device's camera, which recognizes and tracks objects in the real world and overlays digital content onto them.

- **Mixed Reality (MR)**: MR encompasses the entire spectrum between the real environment and the completely virtual environment, including both Augmented Reality (AR) and Augmented Virtuality (AV). In MR digital content is overlaid onto the real world in a way that makes it appear to be part of the environment and enables more realistic interaction concepts [MTUK94]. Instead of simply overlaying objects, like AR, it integrates digital content with the physical environment in a more interactive and context-aware manner. In MR, the digital content is aware of and can interact with the real world, allowing for a more immersive and seamless blend of the virtual and physical realms.

The framework of VR is closely connected with the term of immersion as well, since in [BF92] VR is defined as: *"real-time interactive graphics with 3D models, combined with a display technology that gives the user the immersion in the model world and direct manipulation"*.Furthermore, the VR applications can be categorized into 3 different types depending on the level of immersion they introduce [CGRR18]: non-immersive, immersive and semi-immersive. In [ABB+01], it is stated that an AR system should combine real and virtual objects in a real environment, operate in real-time in an interactive way, maps and registers (aligns) real and virtual objects with each other. Regarding the definition content for MR, it was given initially in [MK94] but with the marketing policies of today's big companies the term can be confused with a slightly diverged one. More information on this topic is presented in the following subsection 2.3.1.
There are several different emerged technologies used for creating content by means of immersive computing. The main characteristic of those technologies is a form of display through which the user interacts with the projected virtual features for extending reality. Some of those display frameworks are the following:

- Head-Mounted Displays (HMDs), are the most common way of creating XR interfaces. HMDs consist of a visual display and a set of sensors that track the user's head movements in real-time. The visual display typically utilises stereoscopic or monoscopic images to create a 3D or 2D virtual environment that is rendered in real-time. The tracking sensors use various technologies, such as inertial measurement units (IMUs), optical tracking systems, or magnetic tracking systems, to detect the user's head movements and adjust the virtual environment accordingly. State of the art hardware of that category includes Microsoft's HoloLens 2, Oculus Quest 2, HTC Vive Pro 2 and more.

- Head-up Display (HUD), which consists of a transparent display that is mounted on a vehicle's windshield, helmet visor, or eyeglasses, visualises information without distracting the user from their viewpoints. HUDs originated in military aviation as a way to provide pilots with critical flight data, such as airspeed, altitude, and heading, without having to look down at the instrument panel. Today, HUDs are used in a variety of industries, including automotive, aviation, and gaming. More and more applications of that kind of displays are making their way to the industry as automotive giants like BMW, Mercedes, Tesla and more, are investing a lot for R&D in that area. HUD are widely used in aviation as that kind of technology has many military applications, from data display semi-autonomous guidance systems for pilots, targeting systems and more.

- Room-Scale Immersion, which involves the use of sensors and cameras to track the user's movements within a physical space, allowing them to move around and interact with the digital environment in a more natural way. This technology is often used in conjunction with VR headsets to create a more immersive experience. A hardware example is Oculus Rift S, a mid range VR headset with room mapping capabilities up to 400 square feet of playing area.

- Mobile Devices, such as smartphones and tablets are increasingly being used to create immersive experiences.

2.3.1 The Reality-Virtuality Continuum and Extended Reality (XR)

Reality can be perceived as the collective sensory and perceptual experiences that we, as humans, receive and interpret from the world around us. These experiences are shaped by our biological and cognitive capacities, as well as our cultural and social contexts, and are filtered through our senses of sight, hearing, touch, taste, and smell. On top of that, immersive extensions of this reality, achieved through advancements in more than one scientific areas (computer science, electronics, neuroscience, psychology etc.), is the aggregated effort that embodies the set of Extended Reality (XR) paradigms like VR and AR.
Oftentimes, the terms Virtual Reality (VR) and Augmented Reality (AR) are confused and in cases misused as well, which is why it is important to clearly set the boundaries between the various terms included in the general category of Immersive Computing Technologies and XR. To that end, a major effort to identify a spectrum which will set boarders between different versions of Realities, was made in the of work of Milgram et al. [MK94, MTUK94]. A significant advantage of AR systems over their VR counterparts is the exploitation of the physical world [BS23]. In [MK94], Milgram et al. first gave an official definition for the Mixed Reality framework and tried to identify the individual cases and create a taxonomy

of the respective types of Visual Displays. Mixed Reality (MR) visual displays are a specific subset of Virtual Reality (VR) technologies that combine real and virtual worlds along the "virtuality continuum." This continuum ranges from completely real environments to entirely virtual ones. Augmented Reality (AR) is arguably the most popular MR technology, wherein the display of a real environment is enhanced through virtual computer graphics. The opposite scenario along the virtuality continuum is known as Augmented Virtuality (AV).

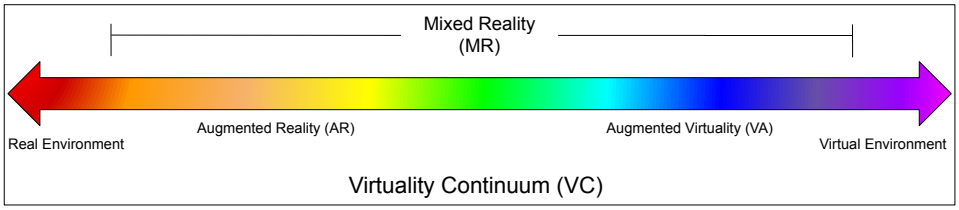


Figure 2.9: The Virtuality Continuum as was presented in [MK94].

The Viruatlity Continuum hyperspace has three main attributes that are defined in [MTUK94]:

- *Reality*: there are the computer generated virtual environments that were created artificially and the primarily real ones.
- *Immersion*: the need for the observer to be completely immersed within the interacted environment should not determine the ability to display both virtual and real environments.
- *Directness*: this concerns the ability to represent the world objects, if they are viewed directly or by using an electronic image synthesis process.

In the same work [MK94] Milgram et. al are trying to classify the different MR displays into 6 different categories. These classes are the following:

1. Non-immersive monitor-based AR displays, upon which Computer Graphic (CG) images are overlaid or in more modern displays holographic units.
2. Similar to previous using video displays but this time using Immersive HMD-based AR displays.
3. HMD-based AR systems, incorporating Optical See-Through (OST-HMD) and video see-through (VST-HMD).
4. Monitor-based Augmented Virtuality (AV) systems, with CG world substratum, employing superimposed video reality.
5. Completely graphical immersive or partially immersive systems, employing superimposed video or texture mapped reality.

6. Partially immersive AV systems with completely graphical like large screen displays, enabling as well real-object interactions.

These three factors define also the taxonomy presented in [MTUK94, MK94], where the three dimensional structure is basically illustrated in Figure 2.10 and Figure 2.11. The components of the taxonomy are the following:

- **Extent of World Knowledge (EWK)**, this dimension describes the real world level of modeling, with questions like *where* (locations of objects) and *what* (identification of objects), included in the MR environment.
- **Reproduction Fidelity (RF)** is the level of accuracy of a virtual world to recreate the real one.
- **Extent of Presence Metaphor (EPM)** concerns the level of world-conformal graphics and viewpoint experienced by the user in the MR environment.

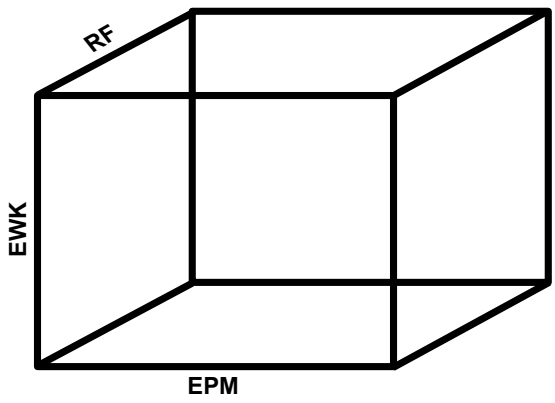


Figure 2.10: Relationship of the three-dimensional taxonomy proposed by [MTUK94] for classifying MR displays.

2.3.2 Augmented Reality and Application Domains in Robotics

Augmented Reality, as an Immersive Computing Technology paradigm, behaves as a contemporary mean of interaction and information exchange between human and autonomous systems in the context of HRI, in an effort to ameliorate the overall performance of the system. Within the AR framework, a variety of technological mediums have been developed, including mobile displays (e.g., tablets and smartphone screens), computer monitors, Head-Mounted Displays (HMDs), and projecting systems for Spatial Augmented Reality (SAR) [GBEL15, MV20].
An important factor to the emergence and incorporation of AR techniques in various areas of robotics is the synchronous advancements in hardware but also the development of algorithmic methods to utilise that hardware in optimal way. Regarding the

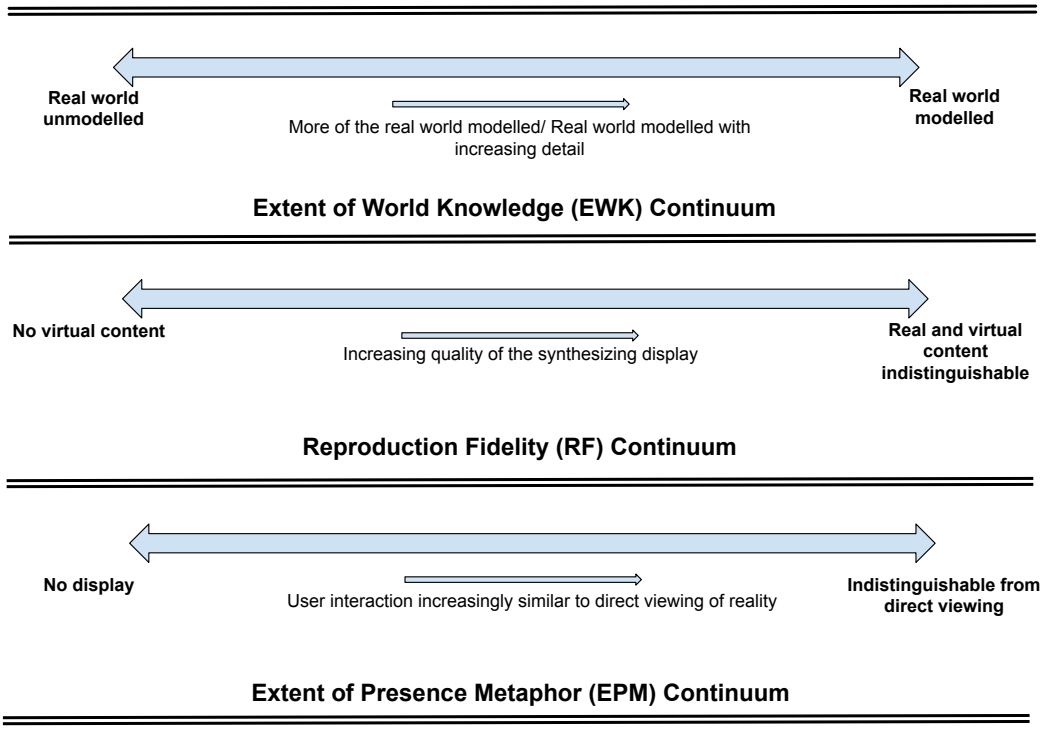


Figure 2.11: Three-dimensional taxonomy of Virtuality Continuum structured by the Extent of World Knowledge, the Reproduction Fidelity and the Extent of Presence Metaphor Continuum that were introduced in [MK94]. Illustration presented in [CGRR18].

hardware, modern AR headsets utilise highly advanced display technologies, such as OLED and MicroLED displays, that provide high-resolution and high-contrast images with low latency. Additionally, recent advances in headsets technology enable hardware architectures that integrates highly sophisticated sensors and instrumentation that read from a wide area of values (from magnetic behaviour to inertial, acoustic, mechanical but also electromagnetic radiation up to infrared wavelength), with algorithmic methods that implement highly accurate and responsive tracking, navigation and registration of real world objects and surfaces. More specifically, vision-based tracking algorithms, such as feature tracking and model-based tracking, are used to track the motion of objects and surfaces in the real world by analysing video feeds from one or more cameras. Camera localization algorithms, such as Structure from Motion (SfM) and asynchronous localization techniques like SLAM (Simultaneous Localisation and Mapping), use data from multiple images to estimate the camera's position and orientation relative to the scene providing also significant computational power and efficiency. Registration methods, such as point cloud registration and surface registration, are used to align virtual objects with real-world surfaces and objects, allowing for accurate placement and interaction of virtual content in the real world.

Numerous displays, have emerged over the last decade that indicate the immense technological progression on the hardware of AR Headsets. Different categories of headsets include Optical See-Through (OST-HMDs) displays, such as the HoloLens 1 and HoloLens 2 from Microsoft, nVisor ST60 from NVIS Company, VuzixTM Star 1200XL, Google Glass, Atheer One Smart Glasses, Recon Jet Eyewear, and the HTC Vive HMD with ZED Mini AR stereo video passthrough camer. Some of these are illustrated in Figure 2.12. but also the VST-HMDs headset category, which commonly employs i-Glasses SVGA Pro technology, enabling stereoscopic-view 3D capabilities. More recently, a new class of AR glasses has emerged that leverages small projectors, such as Moverio BT-200 Smart Glasses, where images are are projected onto transparent display [MV20].

Figure 2.12: A collection of different Augmented Reality Headsets incorporating the Display class defined at [MK94] of OST-HMDs and VST-HMDs [Cop]

Augmented Reality Approach in Robotics

When designing an AR interface, one should not only be aware of how to approach any prospective use case, but also of the complete set of benefits that this technology can offer if integrated properly into a specific framework. That being said, the proper approach should make the right use of the respective input modalities, and examine thoroughly the methodology context that should be implemented in order for the aforementioned benefits to be exploited in each application. To that end, the detailed examination of the taxonomy created in [SKX+22] is providing a comprehensive structure of the necessary information that one could use as a base to build a strategy on developing an AR interface for HRI. This taxonomy is comprised of eight different dimensions which are depicted in table 2.1. Some of the challenges presented in [SKX+22] could also be reference points for future work such as, the feature of

01	Approaches of AR for HRI	• Location of the augmentation device (on-body, on-environment, on-robot) • Target location of visual augmentation (augmented surroundings and augmented robot)
02	Characteristics of augmented robots	• Form factor/ type of robot (robotic arms, humanoid, drones, mobile robots etc.) • Relationship with regard to the number of users and robots incorporated in the interface • Scale of the robot (small, tabletop, body-scale, large) • Proximity to objects (near, co-location from distance, semi-remote, far)
03	Purpose and benefits	• Programming and control (facilitate programming, real-time control) • Understanding, interpretation, communication (improve safety, increase expressiveness, communicate intent)
04	Types of information	• Internal information • External information • Plan and activity • Supplemental content
05	Design elements and strategies of the interface	• User Interface and widgets (menus, information panels,, annotations and labeling, controls and handles, displays and monitors) • Spatial references and visualisation (points and location, path and trajectories, areas and bounds etc.) • Embedded visual effects (Anthropomorphic effects, virtual replicas and ghost effects texture mapping effects)
06	Interaction modalities	• Tangible input • Touch input • Controllers • Proximity by implementing trackers • Gaze input • Voice commands • Gesture input
07	Applications of AR interfaces in Robot Programming	• Domestic and everyday ese • Design and creativity tasks • Medical and health • Remote collaboration • Education and training • Mobility and Transportation • Industry • Social interaction • Data physicalisation • Entertainment • Search and Rescue
08	Evaluation Techniques	• Demonstration • Technical evaluation • User studies

Table 2.1: AR interfaces and they use in robotics [SKX+22].

reducing the cognitive load for an interface, the AR safety trade-offs, some hardware and technological limitations, bridging the gap between studies and systems, new techniques that could be utilised in that context.

An elaborated review of the most recent advancements in robotical AR, is conducted in [MV20]. In this review of AR applications in Robotics, research papers from three major application areas of robotics are mentioned, namely medical robotics, motion planning and control, Human-robot interaction (HRI) and multi-agent systems.

In the medical domain, Qian et al. in [QDK18] introduced ARssist: an augmented reality system designed to enhance robotic surgery by providing visualisations of robotic and hand-held instruments in real time. The system uses a combination of fiducial markers, robotic kinematics data, and an OST-HMD, i.e. Microsoft HoloLens, for tracking and visualisation. It employs a hybrid tracking scheme to ensure accurate and reliable instrument localization. ARssist offers various visualisation options for stereo endoscopy, including head-up display, virtual stereo monitor, and endoscopy registered with the endoscope frustum. The system is built on the da Vinci Research Kit (dVRK) platform and incorporates display and pivot calibrations for precise operation.

The study conducted in [PDS+14], described the use of AR in assisting with liver resections for three patients with varying liver conditions. A 3D virtual model of each patient's abdominal cavity was generated using VR-Render and Virtual Surgical Planning (VSP) software. The model was then used during the surgery to guide port placement and liver resection. Laparoscopic ultrasonography was also used to check the position of the tumor and delineate resection margins. The results showed that the AR-guided liver resection allowed for correct dissection of the tumor with precise and safe recognition of major vascular structures.

The emergence of Industry 4.0 has stimulated the adoption of AR in networks of connected physical systems and human-machine communication [GSLZ14, MV20]. There is an always increased research interest towards the direction of integration of AR interfaces in the domain of Industrial Robotics, some of which worth to be mentioned. In the domain of manufacturing, AR can be used for various purposes, like robot programming, robot teleoperation, workspace inspection, maintenance, training of personnel and more.

In [LS15], the authors introduced a telematic control system for tele-maintenance and inspection of a work environment via an AR interface. The telematic user interface consists of a window displaying different views of the work environment and another showing system status information and control elements for data display, customization, work process analysis, and remote robot control. This allows teleoperators to gain situational awareness and control the robot remotely with AR modalities. The interface also enables intuitive data display with AR, using lines, box-plots, or directional symbols to represent data with positional and value information. Some of the visualised data are shown in 2.13.

The system presented in [OK18], describes the development of a virtual robot system that enables human interaction with robots through gestures and virtual tools, using HoloLens. Virtual robot models KRAgius and iiwa were employed, with their geometric models accurately representing real-world parameters. The system consists of Application Manager, Geometrical Path Planner, Trajectory Planner, and Simulations. It allows the programming of the robot for various tasks, taking into account the robot's kinematic model and limitations. Robot interaction is done through a combination of gestures and virtual objects, such as menus and spatial maps. The system can also handle path modification and planning, with minimum-time trajec-

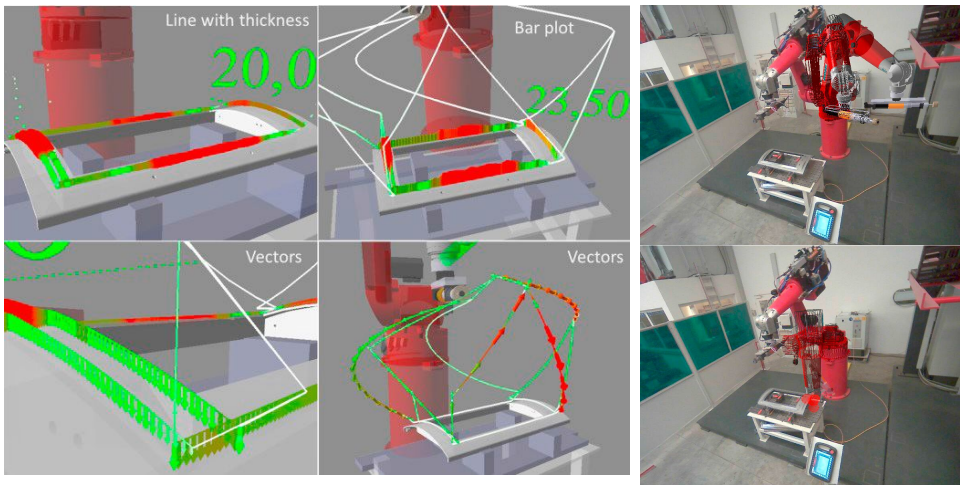


Figure 2.13: Spatial data visualisation (right) and pre-visualised movement with overlay with possible collision (left)[LS15]

tory planning and smoothing. The application allows for easy replacement of the robot model and trajectory planning, Figure 2.14.

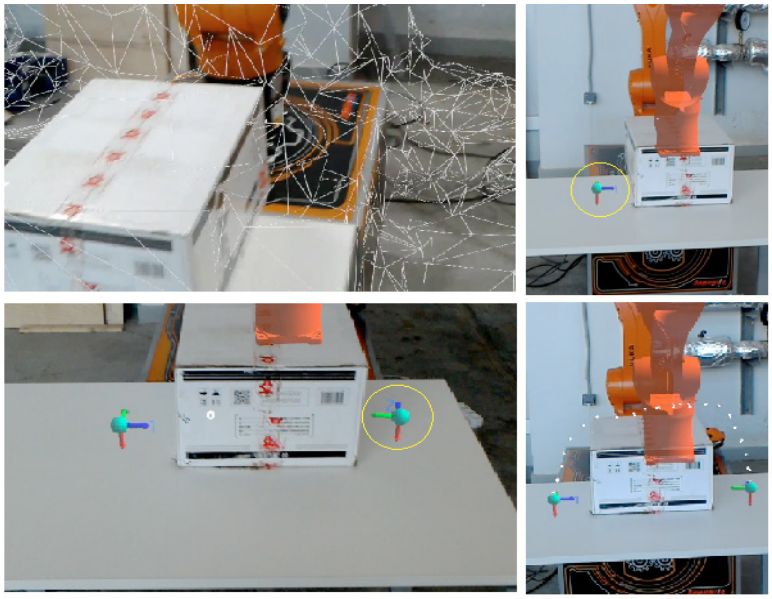


Figure 2.14: User path represented with pointer and point to point with collision avoidance, scanning, goal, points setting and path planning [OK18]

2.3.3 Intuitive Robot Programming and Augmented Reality

Another approach of providing information to the robot in PbD, without relying on symbolic/verbal cues, is by including the element of XR paradigms like AR/VR in

the teaching process and enhancing the system with virtual elements for visualisation and simulation [BG13, BCDS08, Cal09, Cal18]. This helps also to address the problem of knowledge/skill generalisation, with the introduction of explicit request to the teacher for additional information [BCDS08].

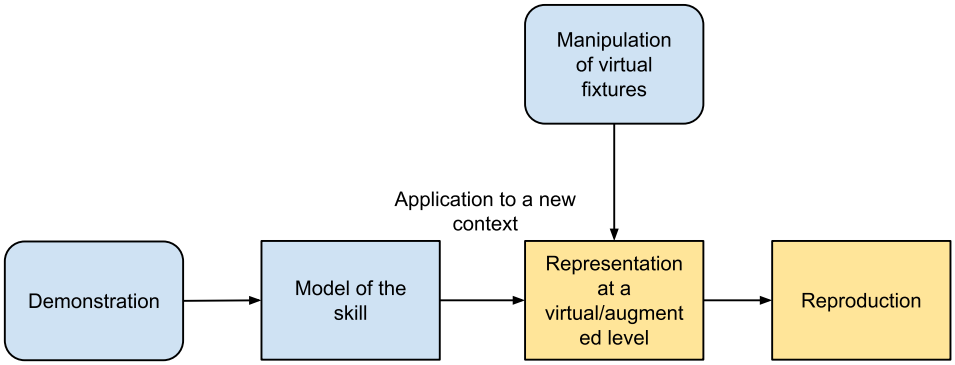


Figure 2.15: Learning a skill by Demonstration in Augmented Reality setup[Cal09].

AR has found a significant role in robot programming, enhancing human-robot interaction, robotic perception, and decision-making capabilities. To achieve a better HRI experience and ultimately build a sophisticated interface that would enable multiple use cases, various input modalities are employed to communicate with and control robots. Although the nature of the modality heavily depends on the application, aimed hardware and task at hand, some of the key modalities that can generically be used in different AR displays and mediums are mentioned here:

- *Visual Input*: Cameras and depth sensors are used to capture images or videos of the environment, allowing the AR system to analyse the scene and overlay relevant virtual objects or information. This is crucial for tasks like object recognition, navigation, and motion planning.

- *Gesture Input*: Gesture-based inputs allow users to interact with robots using natural hand and body movements, which are captured by sensors like depth cameras, infrared cameras, or motion tracking devices. This enables intuitive, touch-less control of robots which could potentially create a more immersive experience.

- *Voice Input*: Voice recognition systems enable users to give verbal commands to robots, which are then processed and executed by the AR system. This offers a hands-free and accessible method of control, particularly for users with limited mobility or in situations where manual input is inconvenient or unsafe.

- *Haptic Input*: Haptic devices provide tactile feedback to users, allowing them to "feel" virtual objects or interactions. In robotics, this can be used to enable

or enhance teleoperation, giving operators a more realistic sense of touch and force when controlling a robot remotely.

- *Sensor Fusion*: Integrated data from multiple sensors, such as accelerometers, gyroscopes, and magnetometers, to enhance the accuracy and robustness of the systems software. Sensor fusion algorithms combine this data to provide a comprehensive understanding of the robot's position, orientation and motion in real-time.

- *Touch Input*: Touchscreen interfaces on tablets, smartphones, projection tables or control panels can be used to interact with AR content in robotics applications. Users can manipulate virtual objects, access menus, or issue commands to the robot through touch gestures.

As mentioned previously, these input modalities are not mutually exclusive and can be combined in various ways to create a rich and effective AR experience in robotics. The choice of input modalities depends as well on the desired level of interactivity and immersion, and is specified in regards with the hardware limitations and capabilities. For example, in Chapter 3 we describe in a more elaborate way the used input modalities in the specific case of the interface we developed for HoloLens 2 HMD.

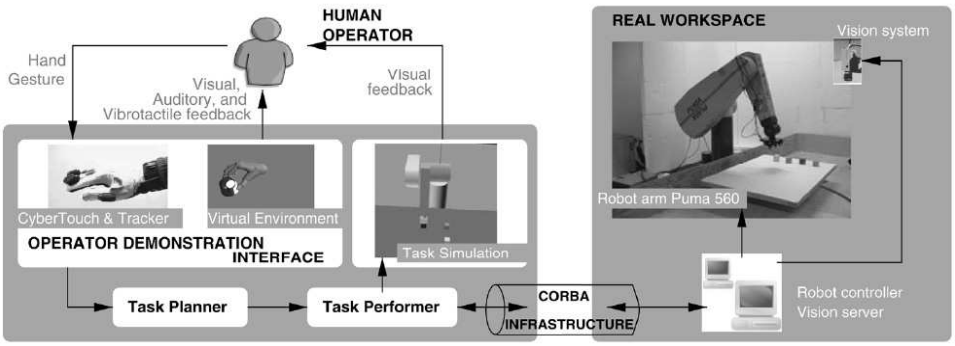


Figure 2.16: Depiction of a Learning from Demonstration scenario where virtual features are enabling an augmented layer on top of the physical demonstration [AC05, BG13].

Various implementations of AR interfaces developed in the context of HRI explicitly for improving the Programming of robots by Demonstration the past few years with the majority of them utilizing the capabilities of Microsoft's HMD, HoloLens 1 and 2. In the work that is presented in [BWP+18] the authors created an application that converts virtual assembly steps into a robot program for pick-and-place motion profiles. They developed the interface for the HoloLens HMD and linked it with the robot, enabling the worker to interact with virtual components that are displayed at the same position as the physical ones. The Vuforia SDK is used for object localization and marker tracking, allowing the overlay between real and virtual elements. In more detail, the virtual components of the assembly are moved with gestures and

placed in the environment. If they are close to the target position then a green indicator which informs the user that the object can be placed. Then the assembly step is carried out in an augmented manner by the user, with start and end coordinates saved in the internal coordinate system of the HoloLens (CSH). The importance of marker tracking accuracy and precision is highlighted and its evaluation is carried out by analysing the tracking process under varying conditions. While the accuracy achieved is not as high as initially expected, it is sufficient for the demonstrated use case, allowing the robot to grip and place components correctly.

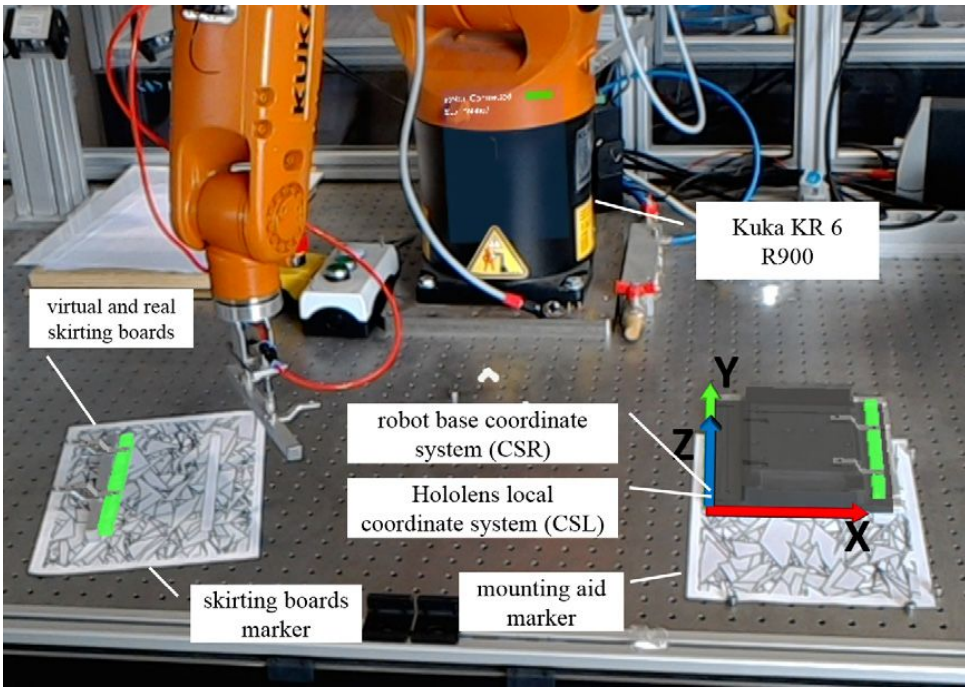


Figure 2.17: The assembly workstation setup with the markers used for positioning and verification of the components and assembly process respectively [BWP+18].

The research conducted in [OME+20], presents a Mixed Reality (MR) interface for interacting with robotic manipulators. The proposed system includes three main modules: the visualisation and interface on HoloLens, the computing part on ROS Kinetic, and the real-world robotic manipulators UR10e and KUKA LBR iiwa 14. The Unity3D game engine and Mixed Reality Toolkit are employed for HoloLens development. The system includes an interface, geometrical path planning, spatial mapping, and virtual models, with communication between components provided by ROS Kinetic. The HoloLens device is responsible for user interaction, action recognition, and translation to ROS. Users can specify control points for the trajectory, which can be connected using PTP, line, or arc. Unique MR features include automatic positioning of the virtual robot model, shortest path construction with obstacle bypass, path scaling, and path drawing. The system employs algorithms such as Jarvis algorithm for removing internal point in the model point cloud by constructing a convex hull; random sample consensus algorithm (RANSAC) for

the image planes initialization; density-based spatial clustering of applications with noise (DBSCAN) for defining objects and comparing volumes of real objects by initializing places of point accumulation; iterative closest point algorithm (ICP) for robot localization; rapidly-exploring random tree algorithm for obstacle avoidance in joints space (RRT) and A* algorithm for shortest path and obstacle avoidance search in Cartesian space; and path scaling which is the increase and decrease of the virtual path through augmentation. The latter was realized by exploiting the Mixed Reality Toolkit (MRTK) capabilities for virtual object manipulation, with which the scaling is being done by setting a bounding box around the virtual object. The MR-based system provides an intuitive and efficient way for industrial robot programming, reducing errors and improving trajectory accuracy. In Figure 2.18 some of the core elements can be found.

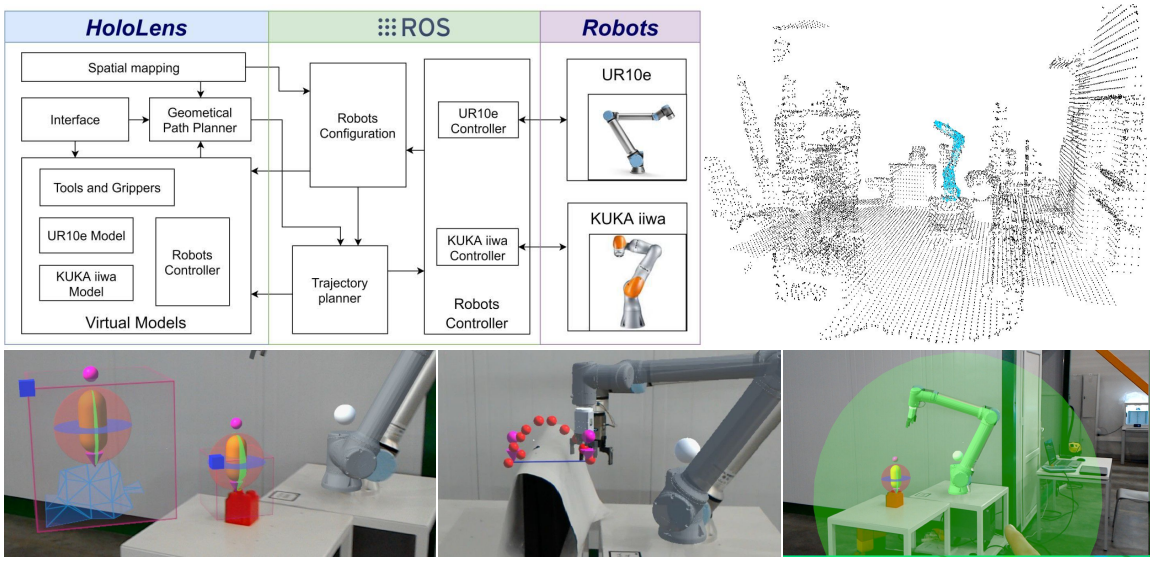


Figure 2.18: From top right to bottom left: the system architecture, robot recognition feature with sparse point clouds, the path scaling functionality, obstacle avoidance, and workspace visualization of the robot presented in [OME+20].

The research of Stadler et al., [SKG+16], investigates the impact that a tablet-based AR support interface has on industrial robot programmers (experts and non-experts). The online teaching process they designed was based on three repetitive tasks: Tool center point (TCP) teaching, through which they calculated the robot base frame by brushing the TCP against a fixed point, trajectory teaching and overlap teaching where they used a trajectory plus the overlapping of one point. An AR session using the interface follows the previous tasks in the respective context but with a virtual fashion, namely in the first case where the edge of the tool reached the fixed point it was colored green, in the second case an overlayed virtual trajectory connected the initial and end point, and in the third case again the start and end-point were visualised with the addition of the overlap radius of the fly-by point, as shown in Figure 2.19. The main difference between the AR session and

the one without AR was that the first one visualised task-based support parameters which could potentially help the user.

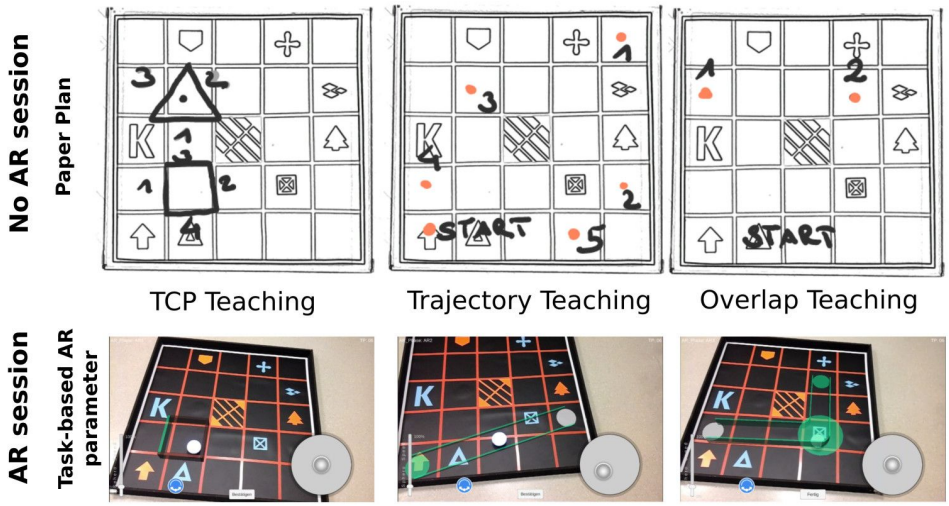
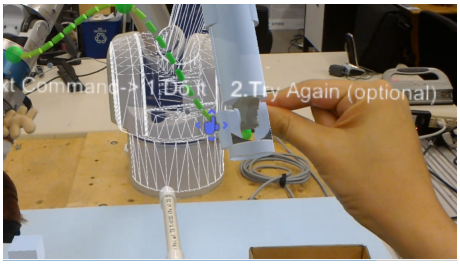
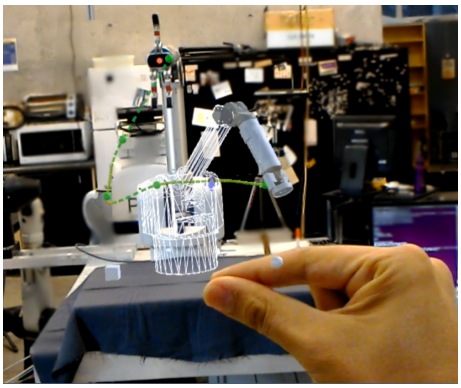


Figure 2.19: Task-based parameters per task, without and with AR session [SKG+16].

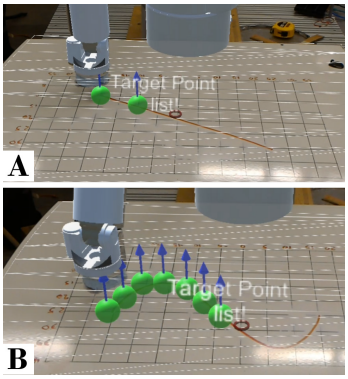
A more sophisticated approach, in terms of AR technology, is proposed by Quintero et al. in [QLP+18], where they present a future-based approach for robot programming based on augmented trajectories, as illustrated in Figure 2.20. They use



(a) Participants edit the virtual path.



(b) User's point of view while executing the task.



(c) Specifying the virtual trajectory.

Figure 2.20: Augmented trajectory planning for a pick-and-place task. [QLP+18].

an MR HMD, namely Microsoft's HoloLens, and a 7-DOF robotic arm with four interactive functions: trajectory specification, virtual previews of robot motion, parameters visualisation and online reprogramming during simulation and execution. They exemplified two robot functionalities: free space trajectory and contact surface trajectory. Furthermore, they compared the system' functionalities with kinesthetic teaching. The conclusion was contradicting with the one mentioned in [SKG+16], i.e. although the performance was better, the whole teaching was physically easier and faster with a trade-off on higher mental workload, which the authors claim to be neglected when working with larger and heavier labor-intensive robots.
The work presented in [LZS+18] is addressing the tasks of diagnosing, teaching and patching interpretable knowledge of a robot. By means of AR and Temporal And-Or Graph (T-AOG), the knowledge representation is structured hierarchically which enables the end-user not only to understand and oversee the whole decision making process of the robot in real-time, but also interact with it and provide the robot with correction-instructions [LZS+18]. The system's architecture is shown in Figure 2.21. The interpretability on that kind of knowledge representation is being done in three stages, as mentioned before:

- structuring the knowledge by compositional models (T-AOG),
- visualising the decision making process through the T-AOG in the holographic interface, in order to make it interpretable,
- enabling the users to interactively patch knowledge gaps by adding/deleting/updating nodes on the T-AOG.

This consists also an evidence that by incorporating a behavioural graph of temporal decision making for task segmentation into skills, can increase both the total intuition of the user for the capabilities of the system but also improve the teaching procedure by constantly updating the knowledge base of the learner, namely the robot.

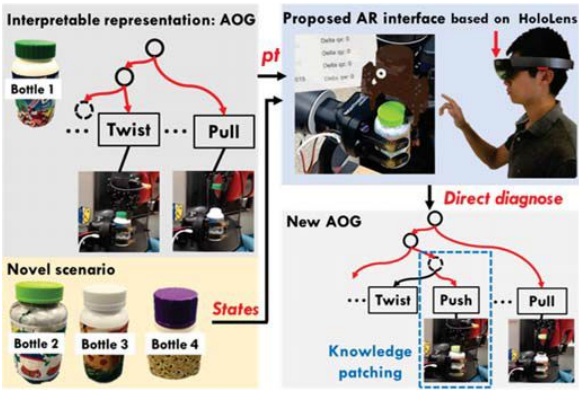
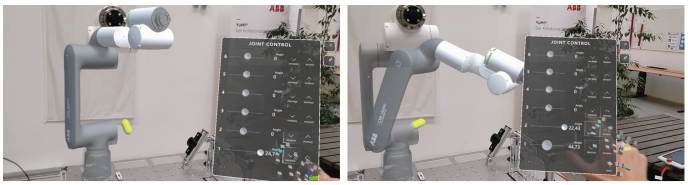
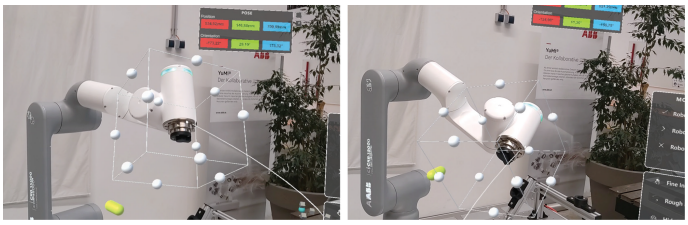


Figure 2.21: System's architecture with the incorporated TAOG task graph, that was presented in [LZS+18].

Finally, the framework presented in [HO22] includes some very notable features that can potentially improve the robot PbD and thus the HRI. They developed an MR interface for Microsoft's HoloLens 2, through which the digital-twin of an ABB GoFa robot was used to simplify the collaboration between the human operator and the aforementioned cobot. The application includes a data model based on the robot's sensor data and enables real-time communication between the virtual model and the physical robot. The robot provides data for the digital twin and responds to commands, while the interface manages communication between the robot, virtual controller, and the HoloLens. The MR application, developed using Unity, PTC Vuforia engine for marker-based tracking, and Mixed Reality Toolkit (MRTK), allows users to interact with the digital twin via gesture control. Their MR application provides clear information about the robot's state, including visualisations of axis positions and torque measurements. This approach offers the potential for easier and more intuitive programming and control of cobots, ultimately improving efficiency and reducing personnel costs.



(a) Individual joint manipulation of the virtual robot.



(b) AR teleoperation of the physical robot.

(c) Use-case on industrial robot.

Figure 2.22: The developed application in [HO22], which enables individual joint manipulation, end-effector manipulation and teleoperation, and visualise torques and position values.

Chapter 3

Interactive Augmented Reality Interface for Intuitive Learning of Conditional Tasks

3.1 Methodology and Design of the Interface

In this chapter, the development process of the proposed Augmented Reality Interface is described in an elaborate way. This chapter consists of three different sections. In the first section, the high level architecture and methodology is analysed, and the input modalities that are used for the robot programming procedure are discussed. In section two, the implementation of the AR subsystem is described and which tools are utilised, hardware and software-wise. The last section defines the interaction concept between the robot (also the virtual one) and the user, in the proposed framework and how, not only the user but rather the complete PbD process, could benefit from the fusion of AR with PbD. In more detail, the holographic concepts and virtual modalities are listed, how the manipulation of the virtual robot and the real one through AR features benefits the process, and in what ways the incorporation of the behavioral task graph can affect the intuition of the user about the overall system.

The proposed approach of this work is the development and integration of the AR paradigm into the existing PbD framework and the Bayesian Nonparametric Gaussian Mixture Model - Inverse Reinforcement Learning (BNGMM-IRL) algorithm that was presented in [WL22]. The high level architecture of the system is depicted in Figure 3.1. The robot programming procedure, in order to be realised, needs three indispensable components. A user that operates the programming interface and by extension the robot. The interface, which in this case is an Augmented Reality Interface (ARI) that acts as the intermediate component that binds the

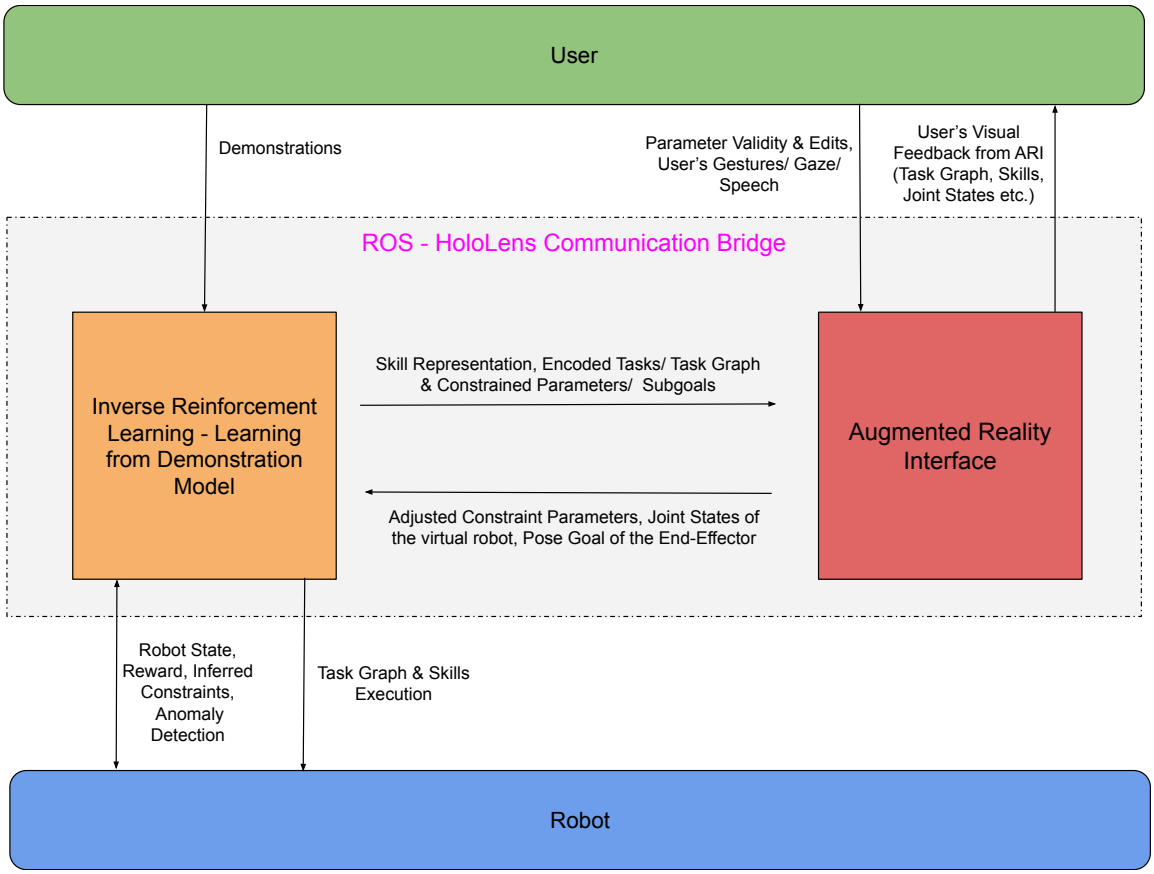


Figure 3.1: High-level architecture of the proposed integrated system.

interaction process and conveys the necessary information from the user basis to the programming basis of the robot. Finally, the actual robot which provides the necessary reaction and a behaviour that acts as a feedback for the user and autonomously conducts the task in the end. The integrated system is comprised of the two main programmatic elements, namely the BNGM-IRL LfD subsystem and the developed ARI subsystem. The user's input modalities to the system include the demonstrations and the modalities that the Head Mounted Display utilises, like gesture's, gaze, speech and more. In return, the visual feedback from the AR interface increase the intuition of the user about the process by visualising the information, which can be joint states of the virual robot, the task graph, skills and many more. The task graph is generated from the learning model, and is retrieved from the AR in order to reduce the knowledge gap and hierarchically decomposes the task, which would render the system more intuitive for the user to utilise. Since the need for flexibility from the user's side is vital, one important feature of the proposed method is the ability to switch between demonstrations, learning and execution phases. As an input exchange basis from the user to the integrated system, the following could be defined: the constraint parameters and the kinesthetically provided trajectory demonstrations for the learning model. A major issue that need to is tackled is

the communication between the learning model and the AR subsystem which is implemented as ROS communication bridge to the AR interface. The information exchange between the two subsystems from the LfD subsystem to the AR interface, includes the following:

- Skill representation/ Encoded task/ Task graph
- Updated constrained parameters/sub-goals

Finally, the sequence of the skills executed during the task, is provided to the robot by the learning model through the derived task graph. Through the learning algorithm, the anomalies can be detected and assigned to a sensor category (end-effector position/orientation, force/torque, gripper opening and grasp status), a feature which allows the anomaly detection algorithm to make use of the robot's proprioceptive sensory systems.
The integrated architecture with the implemented software and hardware components is presented in Figure 3.2. Further programmatic concepts depicted in the

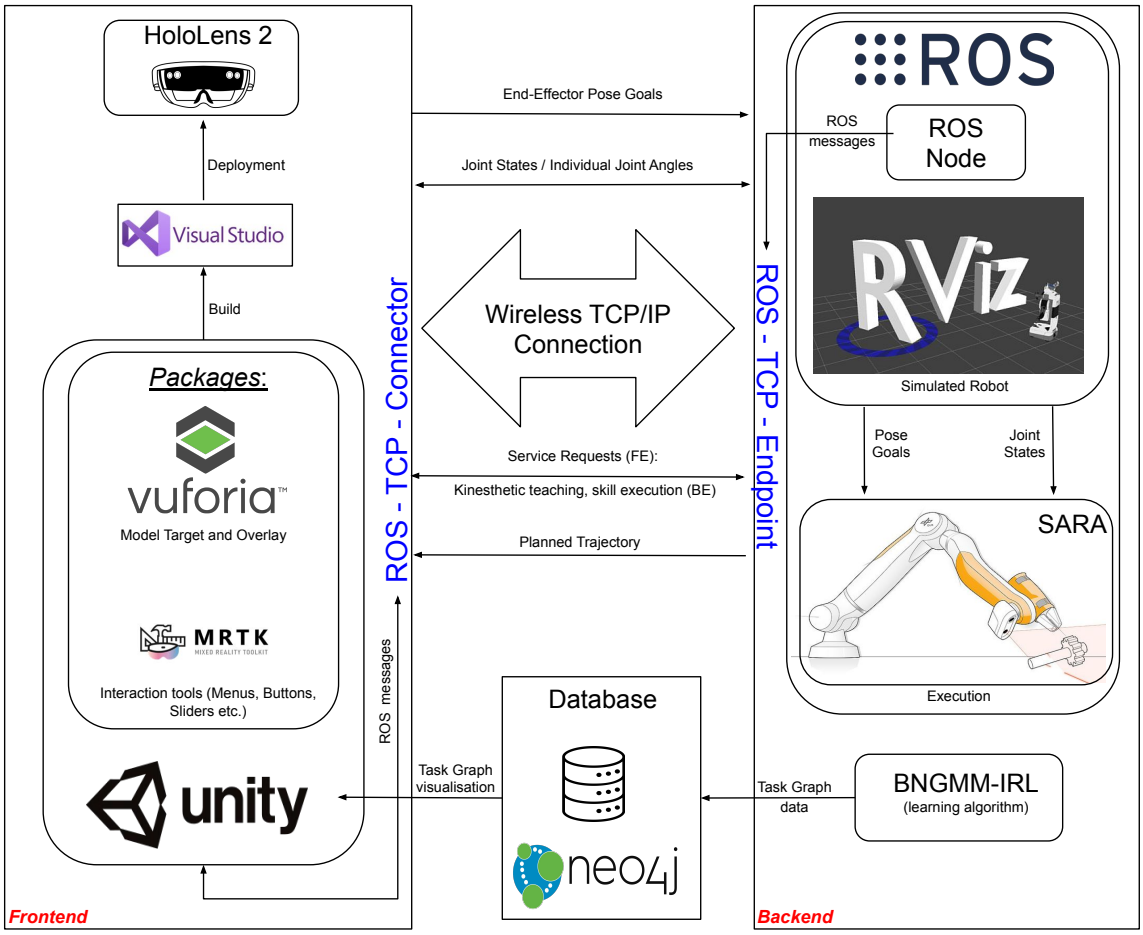


Figure 3.2: An overview of the Frontend and Backend of the implemented system architecture.

Figure above, are explained in the next sections.

3.2 Hardware Capabilities and Specifications

3.2.1 Microsoft HoloLens 2 Head Mounted Display

As mentioned above, the HMD that is used in the context of this work, is Microsoft's HoloLens 2 Mixed Reality Headset [Mic]. An exploded view diagram is depicted in Figure 3.3. It is an advanced and improved version of the original HoloLens, designed to project digital 3D images onto the user's view of the real world. HoloLens 2 is a fully untethered holographic computer which enables interaction between user an augmented/holographic content in the real world. The device features see-through holographic lenses (waveguides) with a 2K 3:2 resolution from its light engines, resulting in a holographic density of over 2.5k radiants (light points per radian). Eye-based rendering optimises the display according to the 3D position of the user's eyes.

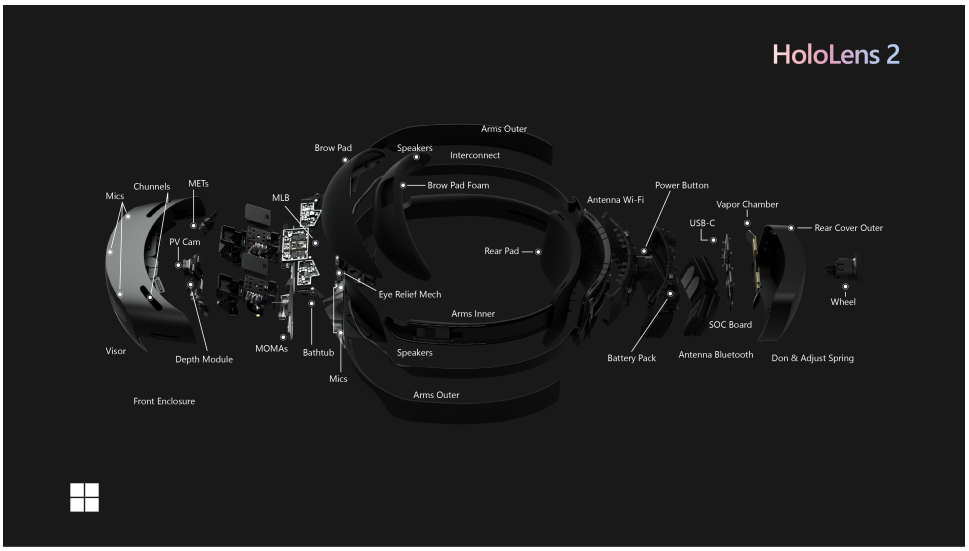


Figure 3.3: Exploded view of Microsoft's HoloLens 2 architecture [Har].

A variety of sensors are built into the HoloLens 2, including four visible light cameras for head tracking (stereo and periphery) which are coupled with IMUs for motion and pose estimation in an effort to mimic the human perception of motion that uses the inner ear system in a similar way. It is also comprised of two infrared cameras for eye tracking, and a 1-MP time-of-flight (ToF) depth sensor. An accelerometer, gyroscope, and magnetometer are included in the IMU, and an 8-MP camera allows for still images and 1080p30 video capture.
The headset's audio and speech capabilities are enhanced by a five-channel microphone array and built-in spatial sound speakers. Human understanding is achieved through hand tracking, eye tracking, and voice commands, with Windows Hello providing enterprise-grade security through iris recognition. The HoloLens 2 is designed to understand the user's environment, featuring 6 DoF tracking for world-

scale positional tracking, real-time spatial mapping, and Mixed Reality capture that integrates holograms with physical surroundings in photos and videos. Powered by a Qualcomm Snapdragon 850 Compute Platform SoC and a second-generation custom-built holographic processing unit (HPU), the device has 4 GB of LPDDR4x (Low Power Double Data Rate 4X) system DRAM and 64 GB of UFS 2.1 storage. Connectivity options include Wi-Fi 5 (802.11ac 2x2), Bluetooth 5, and USB Type-C. The custom-built HPU is an integration of different components that are combined to create a dedicated processor designed to handle the complex tasks associated with MR experiences. The HPU's main components are depicted in Figure 3.4 and are the following:

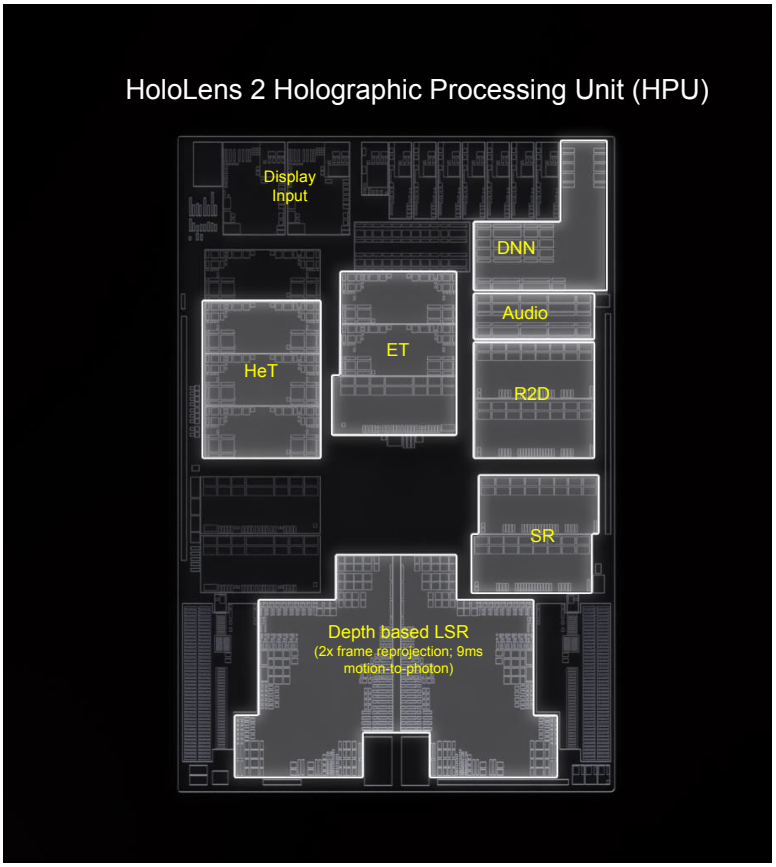


Figure 3.4: Holographic Processing Unit architecture

1. *Head Tracking (HeT) module*: HeT refers to the process of determining the user's head position and orientation in real-time. The HPU utilises data from multiple sensors, such as accelerometers, gyroscopes, and magnetometers, to estimate the head's pose, allowing the device to render holograms accurately and maintain their position as the user moves.

2. *Eye Tracking (ET) module* : ET is the process of monitoring the user's eye movements and gaze direction. The HPU processes data from infrared cameras

to track the eyes in real-time, enabling more natural and efficient interactions with holographic content by adapting the display based on the direction of the user's gaze.

3. *Surface Reconstruction (SR) module* : SR refers to the process of creating a 3D mesh of the environment based on sensor data. The HPU processes depth information from the time-of-flight (ToF) sensor and combines it with data from visible light cameras to generate a real-time 3D representation of the physical space, allowing the device to understand and interact with the surrounding environment.

4. *Raw to Depth (R2D) module*: R2D is the process of converting raw sensor data into depth information. The HPU processes the raw data from the time-of-flight (ToF) sensor to generate a depth map, which is then used for various tasks such as spatial mapping, gesture recognition, and surface reconstruction.

5. *Audio Chip*: The audio chip in the HPU is responsible for processing spatial audio data, enabling the device to provide a 3D audio experience that complements the visual content. It processes audio signals to deliver accurate positional audio that matches the user's perspective and interactions with the holograms.

6. *Deep Neural Network (DNN)*: The HPU includes a DNN module that leverages ML algorithms to enhance various features, such as gesture recognition and hand tracking. The DNN module enables real-time inference and allows for more accurate and efficient processing of complex sensor data, contributing to a more immersive and natural user experience.

7. *Display Input module*: Display input refers to the process of receiving and processing visual data from the device's display system. The HPU handles the synchronization and optimisation of the holographic content, ensuring that it is accurately displayed in the user's field of view.

8. *Depth-based LSR (Late Stage Reprojection) system*: Depth-based LSR is a technique used to correct the final rendered image based on the user's head movements, ensuring that holograms remain stable and properly aligned in the environment. Depth-based LSR takes into account depth information from the environment, allowing for more accurate reprojection and minimizing visual artifacts, resulting in a more immersive and realistic mixed reality experience. In order for Depth LSR to function properly, the client application must provide a depth buffer that includes all relevant geometry to be considered during LSR. This technique utilises the supplied depth buffer to stabilize the video frame. As a result, any content that has not been rendered to the depth buffer, such as transparent objects, cannot be stabilized by LSR and may exhibit instability or reprojection artifacts [LSR].

The HoloLens 2 is designed for a comfortable fit, with a single size that accommodates glasses and weighs 566 grams. The Windows Holographic Operating System comes preloaded. As a self-contained device it has to be passively cooled and cannot physically dissipate the heat generated by the computing units. This poses a fundamental limitation of the device and a major constraint on its performance after extended use.

3.2.2 SARA DLR Light-weight Robot

The robot that was used for the experimentation and the implementation, is the SARA (Safe Autonomous Robotic Assistant) lightweight DLR (German Aerospace Center) robot presented in Figure 3.5. SARA demonstrates novel functionalities in force controlled robotics for a smooth human-robot collaboration [SAR].

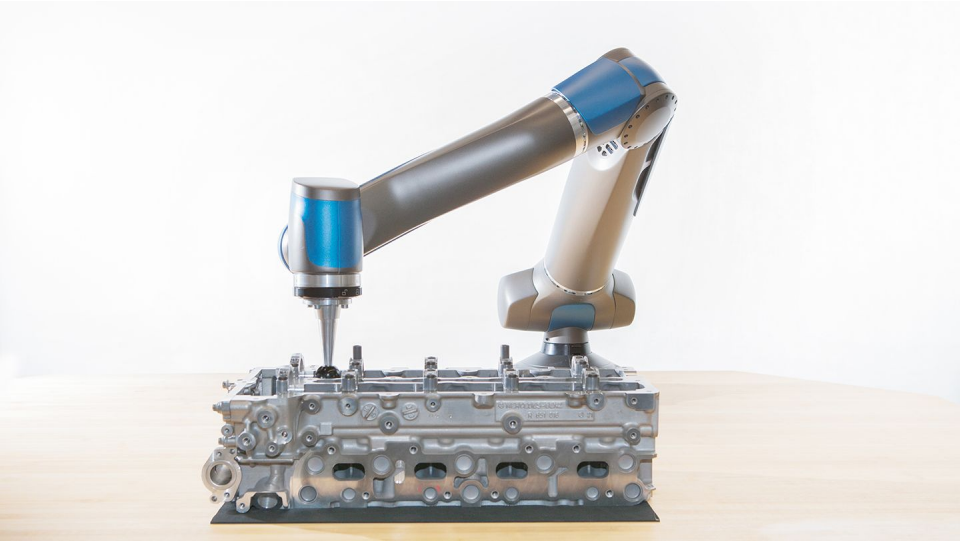


Figure 3.5: The DLR's new generation of lightweight robots, SARA, developed at the Center of Robotics and Mechatronics (RMC) [SAR].

Its goal is the functional enhancement of the output device robot to an interconnected sensor-actuator system in the Factory of the Future (FoF) [FoF], which will be strongly characterized by digital supervision and control procedures. Another integral part of the FoF is the use of AR for digital twin manipulation, task supervision and intuitive robot programming. The SARA project by DLR has developed a new generation of lightweight robots with the goal of functional enhancement of the output device robot to an interconnected sensor-actuator system in the factory of the future. The torque controlled robot arm with seven axes and 12 kg payload has precise torque sensors and optimised actuators for high dynamics, allowing swift manipulations and a wide range of applications. More of the technical data presented in Table 3.1.
A scientific novelty is the redundant force sensor-kinematics that enable high-resolution force and torque measurement at the robot flange and measuring contact forces dur-

Size:	Length 1250 mm for stretched robot arm
Weight:	22.6 kg
Degrees of freedom:	7 (roll-pitch-roll-pitch-roll-pitch-roll)
Nominal payload:	12 kg
Force resolution:	Better than 0.1 N
Axis speed:	Up to 400 °/s
Sensors:	• Torque sensors in each joint • Force-Torque-sensor at the base and the wrist • 2 IMUs
Communication:	• Links and Nodes Middleware • Ethernet via PCIe
Workspace:	Spherical hull with r = 297 mm, R = 1024 mm (wrist)
Features:	• Recuperation via a capacitor bank • Sensor-Port (PCIe) for optical sensors • Limitation of collision torques • Option for force-teaching in contact

Table 3.1: The technical specifications of the SARA light-weight robot [SAR].

ing manual guidance of the robot. SARA can record position and force trajectories simultaneously, allowing for accelerated programming by demonstration and more intuitive programming. High frequency supervision of force directions and values during assembly operations enables the detection of errors in real-time, increasing safety in human-robot collaboration.
SARA features an infinitely rotatable seventh axis with energy and communication transmission, an integrated quick-change mechanism with tool recognition, energy and data interface, and an operator input with buttons and a display at the robot arm for on-site inputs. The joint controllers are connected via PCI Express to the control computer, which performs whole-arm control at an 8 kHz clock rate, enabling real-time supervision of all internal values and the integration of additional components via mPCIe-slots.

3.3 Input Modalities and Technical Approach

3.3.1 Input Modalities for the AR Interface

Depending on the range that the developed interface belongs to, in the Virtuality Continuum spectrum, the interaction of the user with the environment is closely related to the modalities that are going to be used as input methods. the user can then interact with the interface through the input modalities in order to start

the robot programming. These input modalities can vary depending on the type of system or device being used and the task at hand. This work is realized within the AR/MR framework and used Microsoft's HoloLens 2 device. Therefore, it would be only logical to elaborate more on the most commonly used input modalities for the environment of the HoloLens 2 headset, which we also incorporated in the context of the created interface so that the user will exploit the full capabilities of the device:

- **Eye tracking**: The eye-tracking modality uses four embedded sensors in the headset to track the user's eye movements. The data is processed by software algorithms to adjust the display focus, select objects, and provide feedback. Achieving high accuracy and precision in the data is a technical challenge due to various factors such as distractions and incompatible glasses, physiology and external factors like smudges or intense sunlight. To address this the sensors capture a high-resolution image of the user's eye to track even small movements accurately.

- **Head tracking**: The device's sensors and software track the user's position and orientation, which is crucial for accurate rendering of virtual content in the user's FoV. Head-tracking algorithms analyse changes in eye position and device orientation to estimate the user's position and orientation in real-time, while taking into account head and device motion.

- **Speech Input**: HoloLens 2 uses Cognitive Services Speech SDK by Microsoft to implement speech input. The microphone array utilises beamforming to capture user's voice and isolate it from surrounding noise for better accuracy. The captured audio is processed by the speech recognition engine, which transcribes speech into text and allows the device to execute a range of commands. Text-to-speech functionality is also provided, generating natural-sounding speech using neural TTS technology. Although the speech input modality is determinant for immersion, it was not used as a primary modality in the developed interface.

- **Gaze**: Gaze input combines eye and head tracking to determine the user's point of gaze and translate it into input commands for interacting with virtual objects. By taking into account both the movement of the user's eyes and head, the system provides a more accurate and responsive interaction experience. This modality allows users to select or activate virtual objects simply by looking at them, making it particularly useful in hands-free situations, such as in industrial or medical settings. The system uses data from depth sensors and infrared cameras to determine the user's point of gaze, and responds accordingly based on that information.

- **Gestures**: HoloLens recognises, through its sensory, various hand gestures for performing actions such as selecting, dragging, and resizing holograms. This

gesture recognition is accompanied by visual feedback in the form of holographic responses that correspond to the user's physical cues. For example, a pinch gesture can be used to zoom in on an object or select an item.

- **Hand tracking**: Hand tracking is a crucial aspect of the Gesture modality The depth camera employs active infrared (IR) illumination to determine depth through phase-based time-of-flight, enabling high-framerate (45 FPS) near-depth sensing mode for articulating hand tracking. By visualising a virtual hand over the real hand, the user can experience a high level of immersion. Hand tracking is a widely used modality in MR applications as it can be applied to any virtual object. Without hand tracking, the Gesture modality cannot be fully utilised.

- **Spatial awareness**: Spatial awareness generates a set of mesh models representing the environment's geometry facilitated engaging interactions between holograms and the physical world. It encapsulates the headset's ability to understand and interpret the surrounding physical environment in real time, allowing it to place digital objects and information within that environment in a way that looks and feels natural to the user. An array of depth cameras captures images of the environment, while the SLAM algorithm combines the sensor data with a model of the environment to estimate the headset's position and orientation. This allows for the accurate placement of digital content within the environment, ensuring it remains in the correct position as the user moves around.

3.3.2 Development Process and Workflow

The development of the AR interface for robot programming using the HoloLens 2 HMD, employs several software tools and frameworks. These tools facilitate the creation, testing, and deployment of the application, ensuring smooth integration between the HMD and the robotic systems. In this subsection, the details of the used software tools for the development of the interface are given. The main software for programming HoloLens applications is the Unity game engine, and Microsoft's Visual Studio for deploying and script editing. The same methodology was followed during the software development phase of our interface.

Unity Game Engine

In the context of this thesis the Unity 2020.3.41f1 LTS game engine was used for developing the interface. The Unity game engine is a powerful and widely-used development platform that allows creators to build 2D, 3D, VR, AR, and MR applications, games, and interactive experiences. It provides a rich set of tools, features, and a scripting API, making it a popular choice for developers of varying skill levels and backgrounds. Among others, Unity is also compatible with Universal Windows

Platforms (UWP) such as HoloLens 2 HMD, which is designed for a variety of applications, including gaming, industrial training, and medical simulations. To create AR/ MR applications for HoloLens 2 using Unity, developers can utilise several characteristics and features:

1. Built-in support for UWP: Unity has native support for building UWP applications, which means, one can create HoloLens 2 apps without the need for external plugins or tools.

2. Extended Reality (XR) Software Development Kit: Unity's XR SDK enables developers to create MR, VR, and AR experiences with a unified framework. This SDK simplifies development for various platforms, including HoloLens 2, by providing a common set of tools and features

3. Visual scripting: Unity's visual scripting system allows developers to create and edit scripts using a node-based interface, making it easier for non-programmers to create interactive MR frameworks for HoloLens 2.

Below, some of the main terms that were used for the Unity development and will be mentioned in the next sections, are being enumerated:

1. **Hierarchy window**: The *Hierarchy window* in Unity is a panel that displays the list of all *GameObjects* present in the current scene. It allows users to view, select, and manage the objects within the scene in a hierarchical tree-like structure.

2. **Game and Scene view**: In Unity, there are two primary views to work with - the *Game view* and the *Scene view*. The *Game view* is used to preview and test the game, as it will appear to the player during runtime. The *Scene view*, on the other hand, is an interactive workspace for designing, editing, and arranging *GameObjects* within a scene.

3. **Main Camera**: The *Main Camera* is the default camera in a Unity scene that renders the visual content of the game. It determines the perspective from which the player views the game world and captures the final image displayed on the screen.

4. **GameObject**: In Unity, a *GameObject* is the basic building block of a scene. It represents any object or entity that can be placed in the game world, such as characters, props, or lights. *GameObjects* can have various components attached to them to define their behavior and appearance.

5. **Prefab Asset**: A *Prefab Asset* is a reusable template for creating instances of a *GameObject* with predefined components and properties. It allows developers to create, configure, and store a *GameObject* with its components and settings, which can then be easily instantiated multiple times throughout the game.

6. **Rigidbody**: A *Rigidbody* is a component in Unity that adds physics properties to a *GameObject*. It enables the object to be affected by forces, gravity, and collisions, allowing it to interact with other objects in the game world according to the rules of physics. It is also a mandatory component in order to use the Hinge Joint

7. **HingeJoint**: A *HingeJoint* is a physics component in Unity that constrains the movement of a *GameObject* to rotation around a single axis, like a hinge. This can be used to create realistic joint-based interactions between objects, in our case for representing the dependency between the joints of the robot.

8. **Articulation Body**: The *Articulation Body* component in Unity is used for creating complex, articulated, and physically accurate body simulations. It offers precise control over joints and their limits, enabling the creation of realistic and stable robotic systems.

9. **Colliders**: In Unity, *Colliders* are components that define the shape and boundaries of a *GameObject* for the purpose of detecting collisions and interactions with other *GameObjects*. They can be added to *GameObjects* to enable physics-based interactions, such as collisions or triggers, with other *GameObjects* in the scene. There are Colliders of standard shape like *Box Colliders* or *Sphere Colliders* etc., and there are non-standard colliders that are generated from the geometry of the 3D model imported in the scene. The latter type of *Collider* is called *Mesh Collider*.

10. **Scripting**: *Scripting* in Unity refers to the process of writing code (usually in C#) to define the behavior, interactions, and logic of *GameObjects* and their components. Through scripts, developers can create custom functionality, immersive mechanics, and complex interactions between objects in the virtual world.

In Figure 3.6 the Unity *Scene View* can be seen, where the user can edit the different elements, *GameObjects* that are going to be displayed in the HoloLens 2 device. This is an early stage development where a near-menu object with some functionalities, is used as the main control panel for the activation and deactivation of *GameObjects* in the *Scene* and *Game View*. Moreover, the 3D model of the robot along side with a task graph are depicted. The robot model was imported in the Unity Scene by using the Unity Package of URDf Importer. URDF Importer allows the user to import a robot defined in URDF (Unified Robot Description Format) format in a Unity scene. URDF defines the geometry, visual meshes, kinematic and dynamic attributes of a robot. Importer parses a URDF file and imports it into Unity using *Articulation Body* Unity classes. In our interface, the *Articulation Body* Unity classes were removed due to some errors that emerged originated from the *Mesh Colliders* of the joints of the imported robot model. Instead we used *Rigidbody* classes in order to enable the *HingeJoint* class that was representing each joint

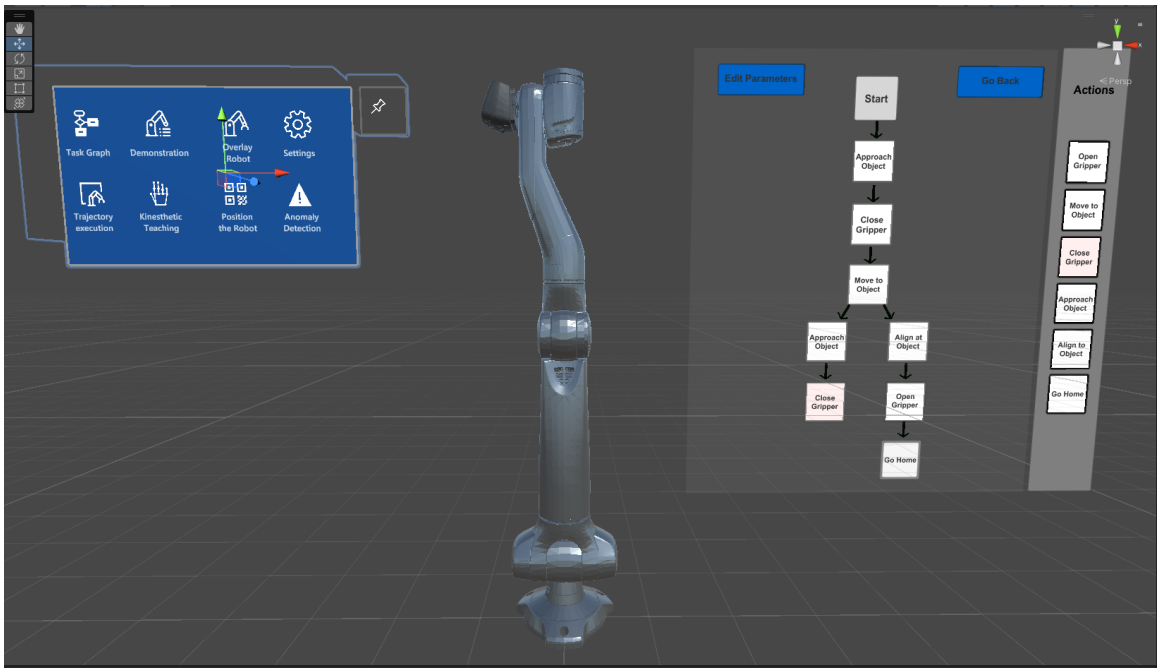


Figure 3.6: Example of an early stage of the development phase of the interface in the Unity Scene view.

of the 3D model of the robot. The *HingeJoint* classes enabled the properties of stiffness, damper, velocity and acceleration as well as joint angles limits to the individual joints of the robot model. Through Unity, the developer can build the Unity project and set as target device in order to deploy in the HoloLens 2 HMD. Then it generates a Visual Studio solution file (`.sln`) with the necessary configurations and dependencies. The deployment to the device is being done through Microsoft's Visual Studio environment by compiling the solution and selecting the appropriate compilation option, namely "Release", and target platform, namely ARM64 for HoloLens 2. The compilation process converts the Unity-generated code and assets into a UWP app package (`.appx`) that can run on the HoloLens 2 device.

Mixed Reality Toolkit

The Mixed Reality Toolkit (MRTK) is an open-source, cross-platform development framework provided by Microsoft, specifically designed for building MR applications on devices such as the HoloLens and HoloLens 2. MRTK aims to simplify the development process by offering developers a collection of scripts, components, and prefabs to support common interactions, spatial awareness, and input handling in MR contexts. In our interface MRTK modalities and X features from this platform, were the core building element of the functional modules. Some of the User Experience (UX) building blocks are depicted in Figure 3.7. Some functionalities that are enabled with the use of the MRTK, including the cross-platform input system and building blocks for spatial interactions and UI (User Interface). Furthermore,

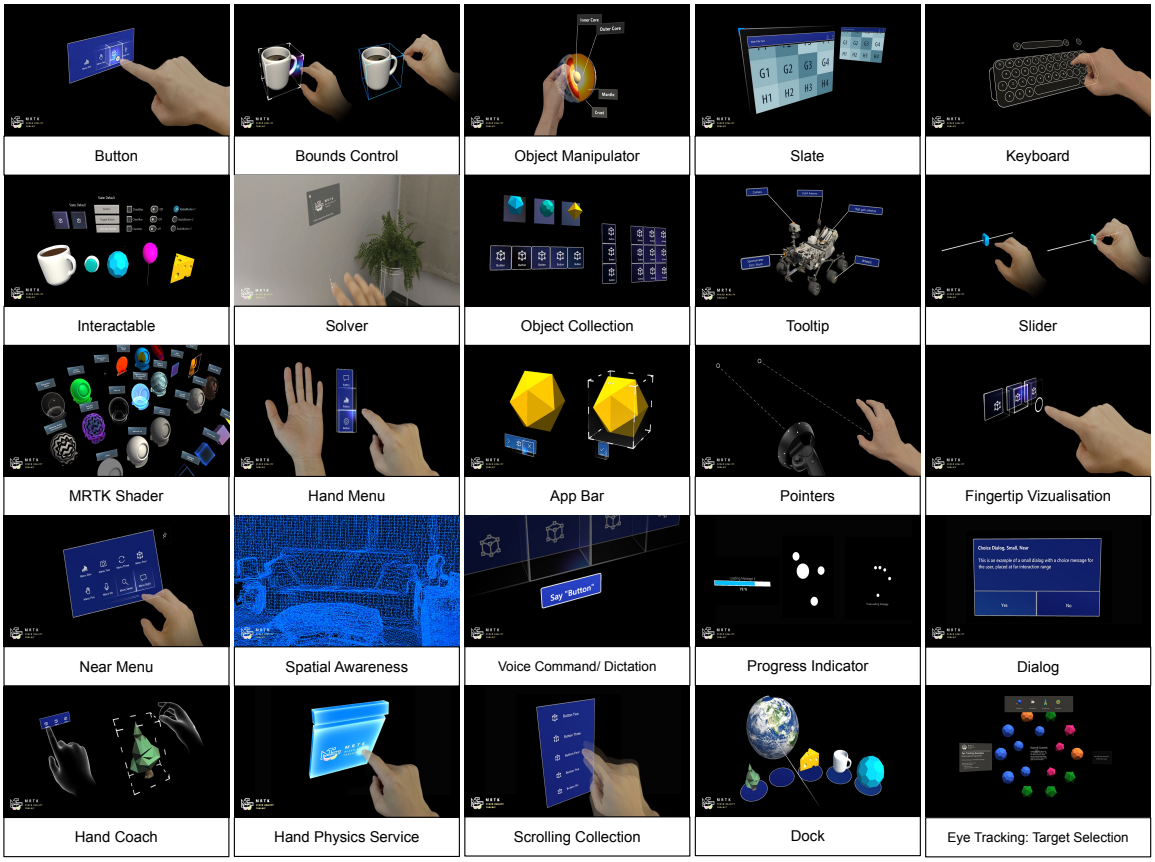


Figure 3.7: The UX building blocks provided by MRTK for development of AR/ MR interfaces [MRT].

accelerated prototyping through the in-editor simulation where the changes can be easily monitored and supports a broad range of devices. In the proposed system the most common building block that is used for the core functionalities of the interface, were buttons. *Buttons control* support different input methods with the major one and easiest to implement/ use the device's articulated hand. *Bounds control* is used for making the object manipulation, which is provided through a script, more flexible by giving to the user the ability to manipulate the objects in 3D space. *Slate* features were used for creating control panels that overlaid different buttons for various minor functionalities. The main menu in our scene that includes the core of the interface, namely the five main functionalities, is an Object collection which is visualised in a cyclic *Grid* form menu. More details on that in section 3.3

The main menu appears when using the *Hand Menu*, a hand-locked UI for quick access using the *Hand Constraint Solver*. Among others, the *Dialog* feature is exploited for service execution, which is an integral part of the communication process as it is stated next in this section, but also for increasing the user's intuition about the nature of the application and providing text-based instructions of how the user should use the interface. Scripts like *Interactable* and *Solver* are used for providing the user with the ability to interact in additional ways with the objects with visual

states and automatic positioning behaviour in the scene, such as tag-along, body-lock, constant view size and more. Each functionality block spawns a *Tooltip* where the user can read which is the functionality of each module.

Model Tracking and Overlaying

Immersion and *directness* are the two metrics we mentioned in section 2.3 and define the level of intuitiveness and realistic representation of an AR interface, which locates it along the Virtuality Continuum spectrum. In order to increase the values of those metrics in our interface, we introduced the ability to overlay and position the virtual objects, i.e. holograms, on top of the real ones. With the term objects, we are referring to the robot model, the workbench of the robot in addition to different objects that can be used in various tasks, like tools, small box containers, grippers and more. That way the digital twin of the robot can visualise a simulated scenario and the user is able to verify a certain behaviour, like an executed skill or a specific pose that the robot could potential get to. This also helps in the overall creation of a more immersive environment without taking out of the picture elements existed on the physical reality. The Vuforia Engine library was utilised for the overlay functionality in the augmented HoloLens scene. This library was developed for the robotics community as the framework for image- or model-based tracking of objects. Vuforia Library can be integrated with external sensory devices, such as gyroscopes and accelerometers [SCLO21, MV20]. Vuforia's Model Target feature was implemented for tracking the physical objects and overlaying the virtual models on top of the real ones in the coordinates that are located in physical space. Unlike traditional marker-based tracking, Model Targets use the actual 3D geometry of an object to recognise and track it. This makes it particularly suitable for applications like robotic programming, where complex 3D objects like robots or robot parts are involved.

First, a 3D model of the target object is imported into the Vuforia Engine. The model is then used to generate a database containing features and descriptors of the object. These include the orientation of the object in the real scene, the scale of the object, any color or surface texture mismatch that needs to be defined, how accurate the 3D model is to the real object, and the target recognition range that the object is expected to be tracked in the scene. During runtime, the Vuforia Engine processes the RGB camera input to extract features and matches them against the database where several references are stored. This corresponds to the case of the use of certain "Guide views" of the object as input for tracking. These "Guide views" are some specific pictures of the object that are set by the user in the Model Target Generator application. Additionally, the user has the choice of training the an Advanced Model Target database for a complete 360° recognition view without requiring the user to align an outline of the model with the physical object to start tracking. Once a match is found, the object's position and orientation are estimated, allowing for the augmentation of the object in the AR scene.

In our interface the Model Target was used to overlay both the virtual robot and the virtual model of the complete workbench where the robot operates, onto the physical ones. That way the immersion factor is increased. In Figure 3.8 the three different phases of the Model Target are depicted. Initially he contour of the 3D model is derived and used from the Vuforia SDK for positioning the Unity SARA Game-object that is rendered on the HoloLens scene. Then, the model of the tracked object that is going to be overlaid on the physical robot is generated but is not visible in the Unity or HoloLens scene it is only used as reference. Finally the actual virtual object is activated and placed on top of the reference after the tracking is done and the contour is matched to the real object. When Vuforia detects the object

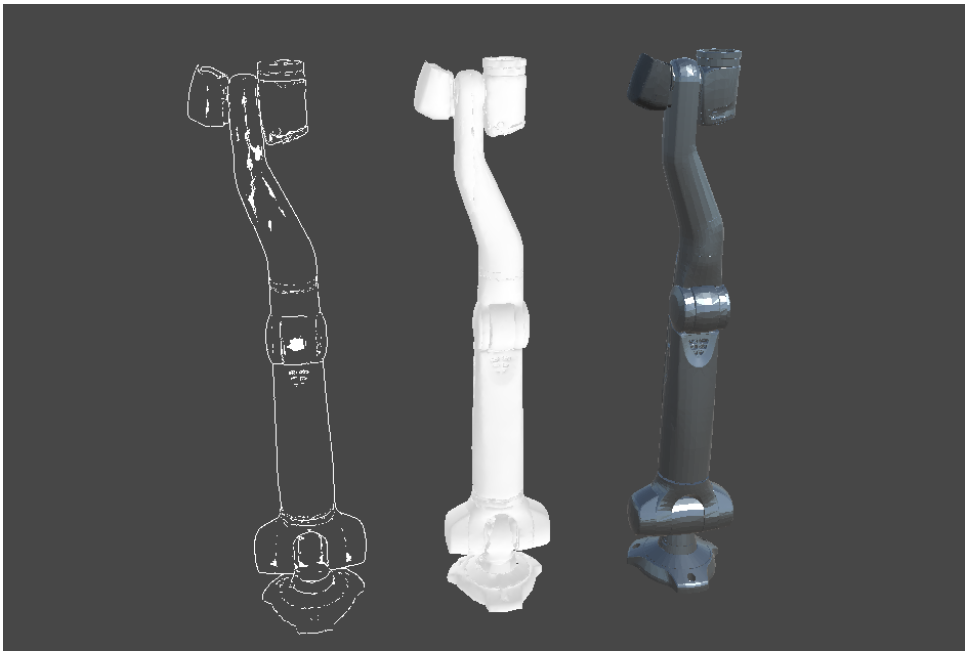


Figure 3.8: The three different stages of the Model Target tracking and overlaying.

in the camera feed of the HoloLens in our case, it computes the object's position (translation) and orientation (rotation) relative to the camera. This information is represented as a 4x4 transformation matrix, which encodes both rotation and translation information. The transformation matrix is used to convert points from the object's coordinate system to the camera's coordinate system or vice versa. This process enables the virtual robot to be aligned with the real-world robot correctly. The transformation process is visualised in Figure 3.9. In Unity, the coordinate system adheres to the DirectX convention, which is a left-handed Y-up system, and poses are automatically assigned to objects within the scene. The world is a left-handed, Y-Up system with gravity alignment. The camera or device uses a left-handed, Y-Up system. For the Model Target, the coordinate system depends on the original model. One of the benefits of Vuforia's Model Targets over the more traditional marker-based positioning and overlaying, is that the first one can track

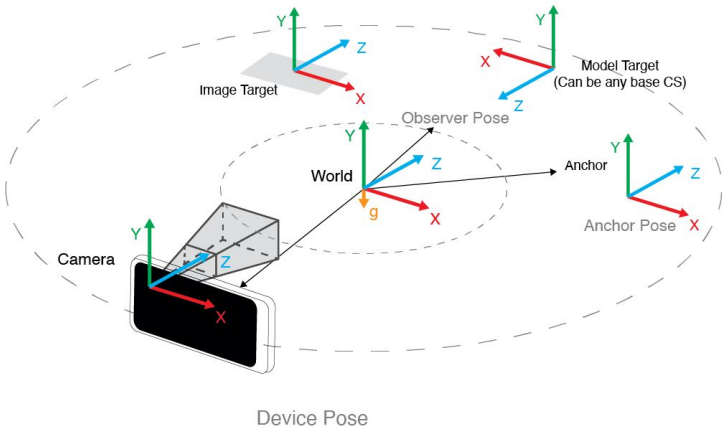


Figure 3.9: The coordinate transformation between the device's camera and the World coordinate system [Tar].

objects without the need for attaching marker images to them or the area around them. This can be particularly useful when dealing with objects that cannot be easily modified, such as machinery. Furthermore, Model Targets can handle partial occlusions better than markers, as they rely on the 3D features of the object for tracking. This means that even if a part of the object is blocked, the virtual robot's position in the world frame can still be maintained.

ROS and TCP Communication

An integral part of this implementation is the communication layer between HoloLens 2 and the Backend, which includes the robot, the ROS node (including the RViz vizualisation environment of the simulated robot) and the BNGMM-IRL learning algorithm [WL22], that was previously mentioned in section 2.7. Robot Operating System (ROS) is an open-source, flexible framework for robot software development that provides libraries, tools, and conventions for creating complex robotic systems. In the context of development of AR applications, ROS can play a crucial role in facilitating communication and integration between the application and the robotic system. That can be achieved by integrating ROS with HoloLens 2 and Unity through third-party plugins and libraries, such as ROS#, a set of libraries and tools for ROS-Unity communication. RViz (ROS visualisation) is a 3D visualisation tool that comes with ROS, designed for displaying sensor data, robot state, and planned trajectories in a simulated environment. RViz allows users to simulate and visualise various aspects of the robotic system, such as point clouds, odometry, and joint states, which can be useful for understanding the robot's current status and behavior. In our case it is used to simulate different trajectories and skill executions. After the simulation is accurate these values can be sent to the Unity side. That way the

behaviour of the virtual robot can be tested and it can be verified whether or not the Unity configuration is representative of the real case by identifying possible errors in the scene. A preview of the simulated SARA robot is shown in Figure 3.10. Rviz provides also the option of visualising many different objects, as it can be seen in the Figure. The complete workbench that the robot is operating on is visualised, and the coordinate system of the base and flange that can be used to understand positioning and orientation of the robot's joints.

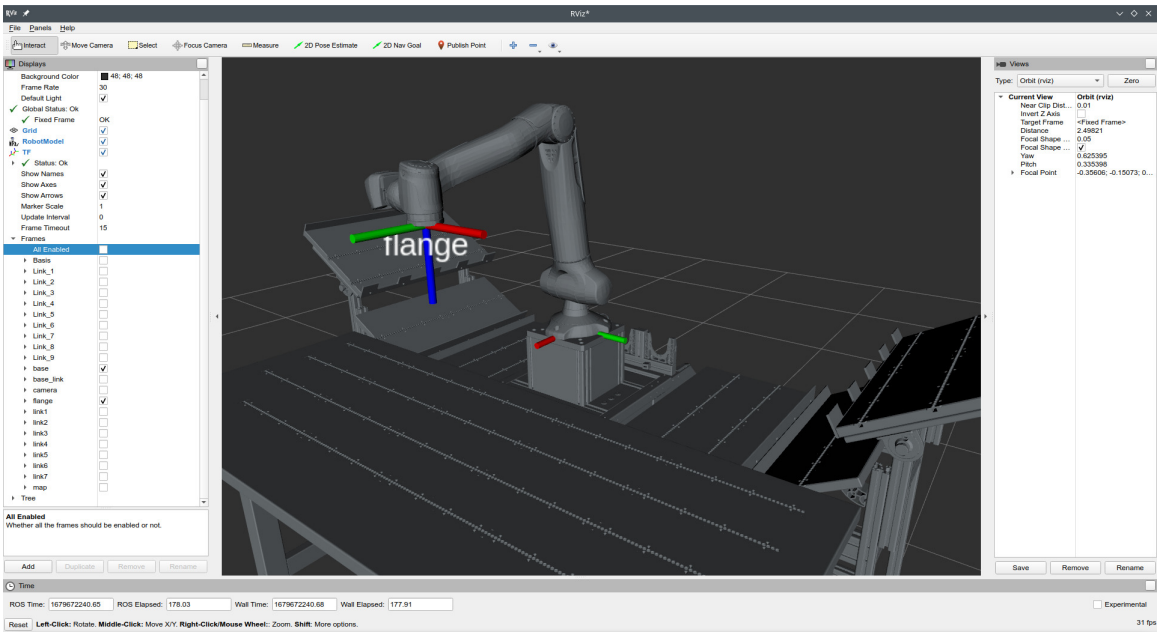


Figure 3.10: The simulated robot movement as it is shown in the Rviz visualisation environment.

The communication consists of three major parts: The protocol, the messaging architecture pattern and the mean of communication, namely the used hardware. Transmission Control Protocol (TCP) was favoured over other protocols because it is more suitable for ordered data delivery. An alternative solution could be the User Datagram Protocol (UDP), which is more appropriate for applications where speed and low latency are prioritised over reliability. TCP is a fundamental communication protocol used on the internet and falls under the category of transport-layer protocols. It ensures reliable, ordered, and error-checked delivery of data between devices. In Fiure 3.11, the communication layers of the proposed system are represented as an Open Systems Interconnection (OSI) model.
Another important decision regarding the communication, is the messaging architecture patterns through which the connectivity layers are defined. These patterns determine the type of messaging and the efficiency of the final communication protocol. Different messaging patterns in the context of the respective communication protocol, enable different kind of communication and increases the system's robustness, reliability, performance, and scalability. Some of these patterns are:

1. Publish-Subscribe (Pub-Sub): In the publish-subscribe pattern, messages are sent to "topics" or "channels" instead of directly to individual recipients. Subscribers express an interest in one or more topics and only receive messages that are sent to those topics. This pattern allows for decoupling of message producers (publishers) from message consumers (subscribers), enabling greater flexibility and scalability.

2. Point-to-Point (Queue): In the point-to-point pattern, messages are sent to a specific message queue and processed by a single consumer. This pattern ensures that a message is consumed by only one recipient, allowing for load balancing and fault tolerance. Queues can be used to implement asynchronous communication between components, where the sender and receiver do not need to be available simultaneously.

3. Request-Reply (Request-Response): The request-reply pattern is a synchronous messaging pattern where a sender (client) sends a request message to a receiver (server), and the receiver processes the request and sends back a response. This pattern is often used in Remote Procedure Call (RPC) scenarios, where the client needs the result of a server-side operation before it can continue.

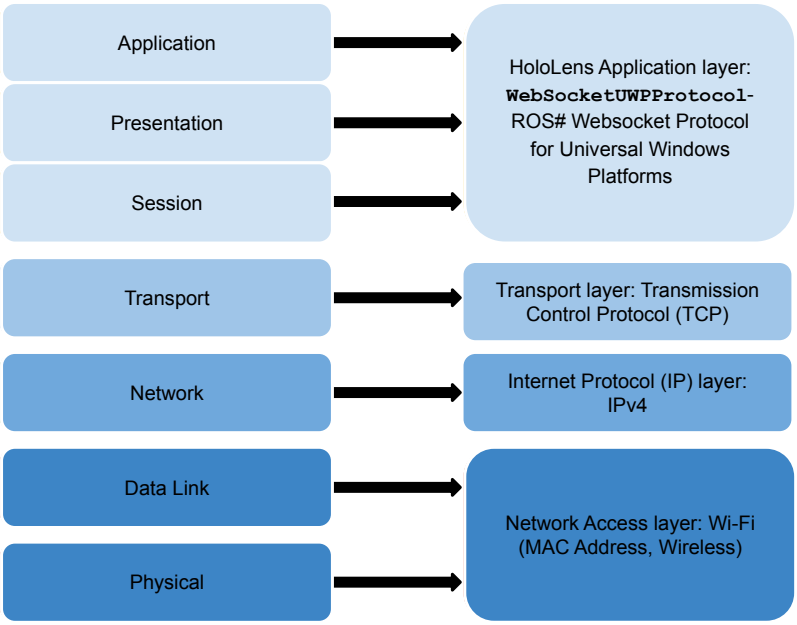


Figure 3.11: The OSI abstraction layers and the respective ones on the developed system.

The type of messaging pattern was determined mainly by the respective user case we implemented. In our case, we used both request/reply pattern, and the publisher/subscriber pattern for transferring sensor values, joint states and pose goals. In order to establish robust and secure communication, the ROS TCP Connector package from Unity Robotics Hub was used [URH]. The ROS TCP Connector is

a communication package that enables integration between Unity and ROS. It uses TCP as the underlying transport protocol for communication between the client, which in this case is the HoloLens application, and a ROS-based system.

The ROS TCP Connector sets up a TCP server on the ROS side and a TCP client on the Unity side. The package communicates using a custom message format built on top of TCP, which allows Unity and ROS to exchange data efficiently, such as sensor data, robot control commands, and other relevant information. The package facilitates the development and testing of robotics applications in Unity by providing a way to interface with ROS-based robots. A TCP endpoint runs on a ROS node, which facilitates message passing to and from Unity and ROS. The passed messages between Unity and ROS are expected to be serialized as ROS would internally serialize them from ROS `.msg` files. To achieve this, the `MessageGeneration` plugin can generate C# classes, including serialization and deserialization classes in order for the data to be handled efficiently and to be easily transmitted, stored, or used by other components. The `ROS Connection` plugin provides the Unity scripts necessary to publish, subscribe, or call a service. In Figure 3.12 the high-level architecture of the TCP client-server system is depicted.

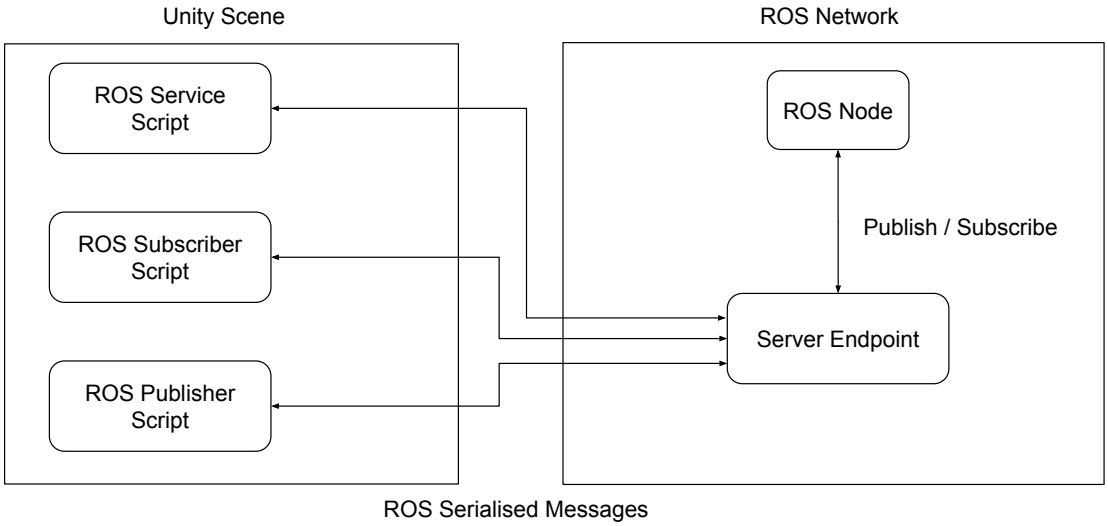


Figure 3.12: The high-level architecture of the TCP connection package in [URH].

After careful consideration, the chosen hardware to establish communication was a router for Wi-Fi connection that operates as an access point for the HoloLens. Wi-Fi is a common choice for data exchange also in the literature, due to its widespread availability, reliability, and relatively high data transfer speeds. It can be used for both local network communication and internet-based communication. In our case it is used for local network communication.

3.4 Human-Robot Interaction Concept

In this section the concept and details of the implemented functionalities are thoroughly analysed. It intended interaction framework between the holographic objects, the user and the robot, are elaborated, in an effort to convey the purposes and original ideas of the design and development process.

3.4.1 Framework of the Holographic Interaction

The interface uses at its foundation, almost all the core input modalities of the HoloLens 2 HMD (3.3.1), as well many of the UX building tools provided from the MRTK (3.3.2). In addition, the holograms are in most of the cases movable with respect to the scene. By that, the user can be independent when the need to move the objects to a different configuration or compare different working stations emerges. Starting the application the user is interacting with a message-based window that provides some guidance and information regarding the workflow and how the main modules are activated. To minimise the virtual objects in the scene, the *Hand Menu* tool, depicted in Figure 3.13 from the MRTK UX toolbox is implemented as the origin of the hierarchical structure upon which the interaction process with the holograms is initiated. Through the *Hand Menu* feature, a more minimalistic

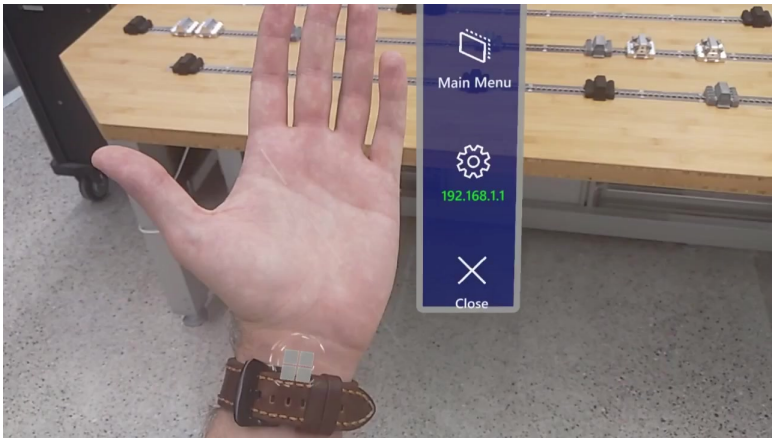


Figure 3.13: The *Hand Menu* that the user can use to initiate the main interaction instance, check for the status of the connection or terminate the app.

design is implemented for a more smooth transition of the user into the MR scene. This type of menu, as stated previously, is activated only with immediate track of the hand by the cameras of the HMD. Two more buttons are implemented as well, the connection, which if enabled, the displayed number of the ip address, that the HoloLens is connected to, turns green, otherwise it turns red. Also a termination that for the user to close the application.
We already established that the interface is built in a modular way, visualised as a cyclic grid of a characteristic radial area analogous to the area of interest. By

area of interest is meant here the area where the robot the workbench and the tools are placed. That way, the user is able to identify the functionalities immediately in a more immersive way, as each functionality is represented by a virtual block, positioned in a convenient place in the scene at the field of view of the user, slightly above the robot. The block functionalities are depicted in Figure 3.14.

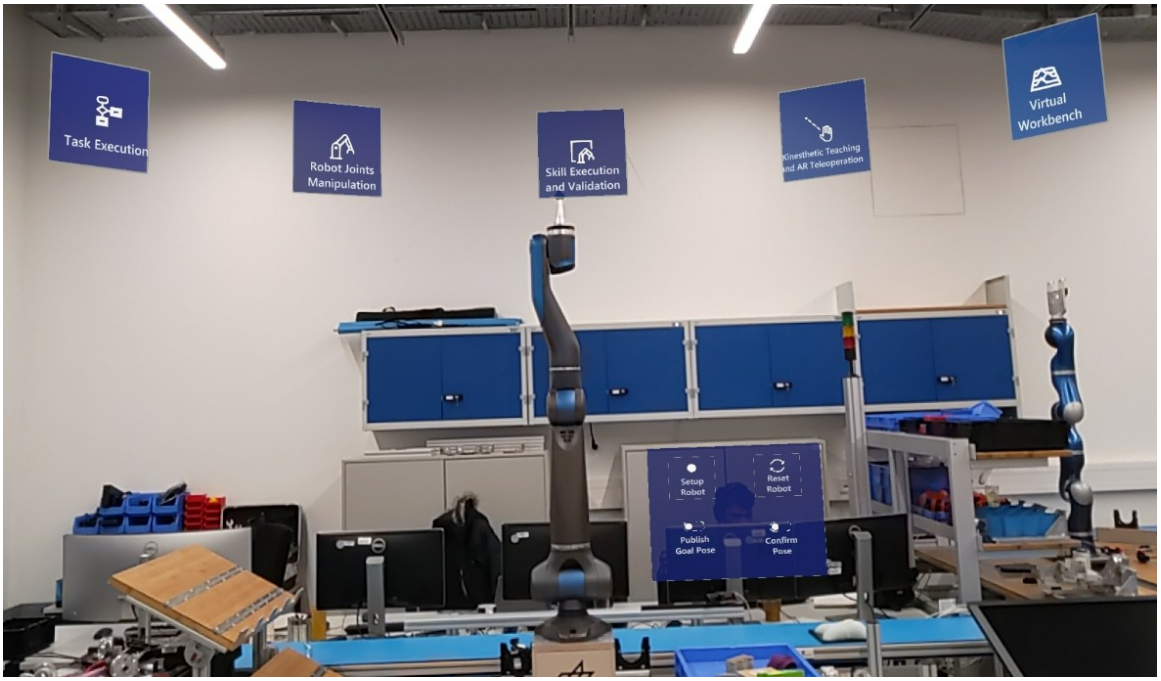


Figure 3.14: The Grid menu that enables modular function over the interface and its core functionalities.

The *Tooltip* feature from the MRTK, is also implemented on each block with which a tip-message is spawned when the user points at it with the virtual pointer for five seconds. That way, a brief explanation on what the pointed block's main operation is, is provided to the user. This grid menu can be activated through the main *Hand Menu*, also an MRTK feature, that is activated when the user tracks his/her hand, and located in the inner hand-side. That way the choice of controlling the amount of virtual objects in the scene, is given to the user. The five different blocks of the grid menu represent the following functionalitiess:

1. Task Execution.
2. Robot Joints Manipulation.
3. Skill Execution and Validation.
4. Kinesthetic Teaching and AR Teleoperation.
5. Virtual Workbench.

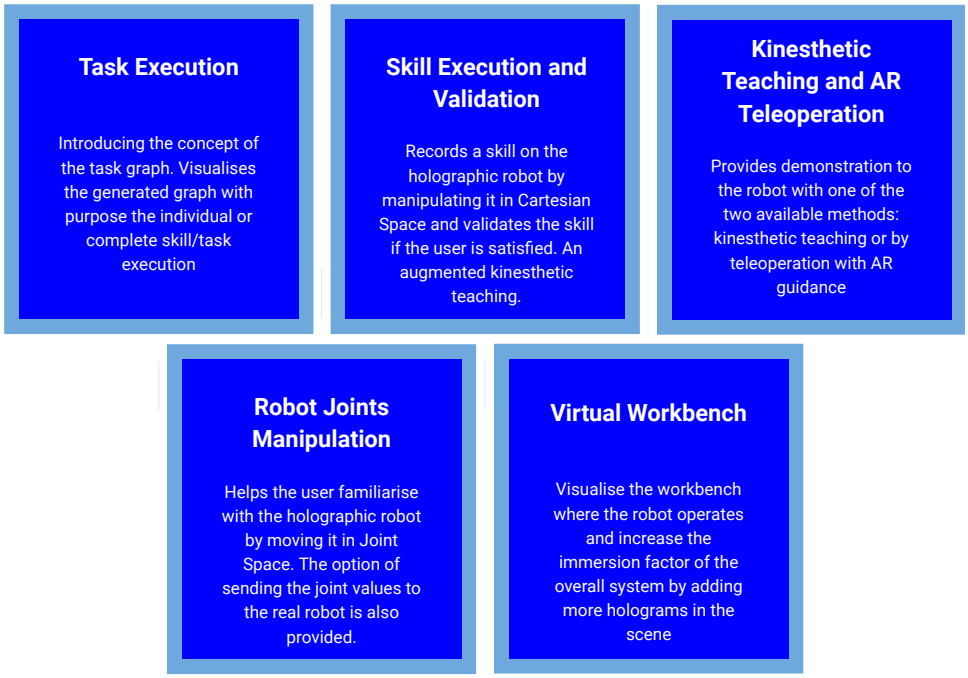


Figure 3.15: The outline of the main functionalities of the interface.

The aforementioned functionalities are summarised in the Figure 3.15 The intention behind the design of the depicted outline, was for the AR interface to be more modular and flexible which is an important characteristic in Robot Programming. The theoretical conceptualisation of this architecture is mainly to provide the user with the ability to combine or use separately the individual functionalities. In Figure 3.16 the manifestation of this theoretical concept of a commutation scenario between the use of the functional modules is represented. These individual functionalities are more thoroughly analysed in the following sections.

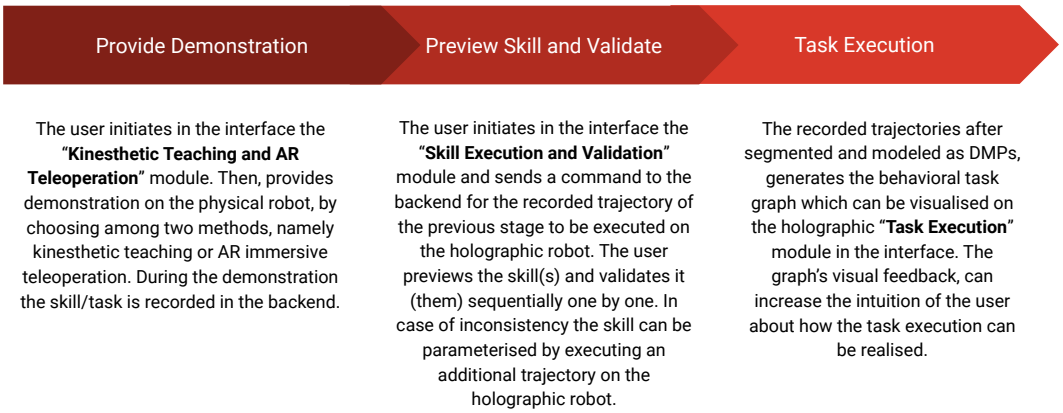


Figure 3.16: A theoretical concept of the combination of the different modules of the interface.

3.4.2 Behavioural Task Graph Representation

The first module of the core functionalities is called "Task Execution". The notion behind this feature is the initiation of a behavioural task graph that represents the skills of the task. The concept behind the function of this module is the integration of the interface with the PbD framework that was previously mentioned in section 2.2 and introduced in [WEL20, WL22].
The task graph's (Figure 3.17) theoretical concept incorporates a modular way to execute skills and potentially correct the executed skills by editing the constraint parameters of each skill separately. This proposed feature enables the visualisation of the task graph, generated by the learning algorithm in the backend, within the HoloLens 2 scene, thereby accurately representing the intended behavior. By creating the separate nodes of the graph as *Interactable* objects in the scene, the user would be able to preview and verify the individual skills and the interaction concept would transition towards an immersive experience. Alternatively, the user could initiate the complete task in the given sequence. A feature of the task graph concept in the AR framework could also be the correction of a specific pose or point on an executed trajectory. The communication between the HoloLens and the robot

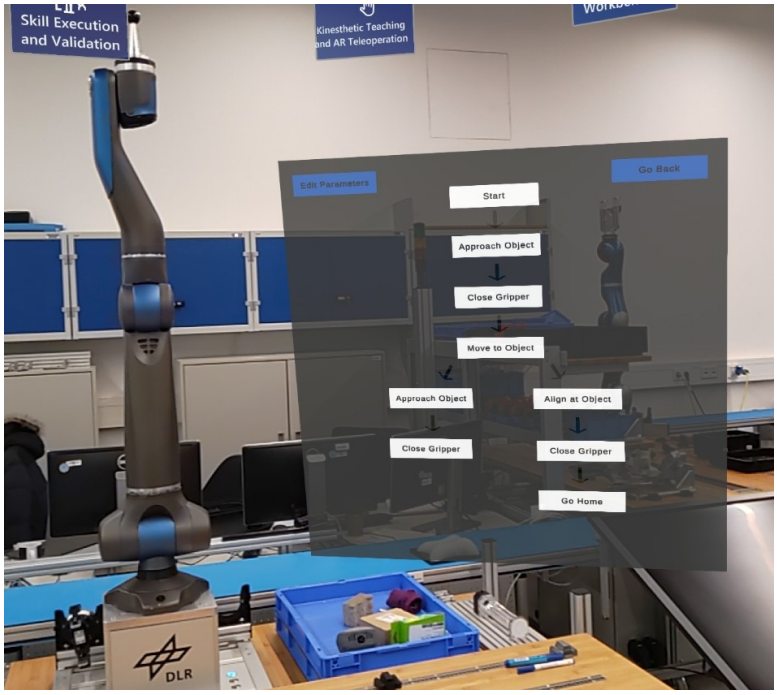


Figure 3.17: The proposed solution for the task graph feature in the HoloLens 2 scene.

would enable basically the preview of the task execution on the holographic robot. This process could involve, within the current system architecture and setup, a ROS service call that requests the initiation of the task execution skill-by-skill or in its entirety. Then the joint states are being published to the subscriber that is activated in the interface at the beginning of the phase. The individual skill trajectories

or actions are executed in the augmented robot so that the user would understand and oversee the complete task. Through that process, the user's intuition about the actual behaviour of the robot during the execution phase is increased. This could be beneficial for the robot programming procedure, by combining it with the other functionalities of the interface too, like the individual skill validation.

3.4.3 Robot Joints Manipulation

The "Robot Joints Manipulation" module's main purpose is to enable the user to move the robot in Joint Space and manipulate individual joints of the robot. Through a control panel, called "Joint Control Panel" as it is shown in Figure 3.18, the user is able to move any of the joints in order to bring the robot in a specific configuration, by moving the *Sliders* that represents the respective joint. These

Figure 3.18: The "Robot Joint Manipulation" module that enables individual joint configuration of the robot.

Sliders are also implemented through the MRTK UX features mentioned in **??**. This functionality introduces more flexibility and direct control of the joints, and since it is a more simple concept of manipulation it also increase the user's intuition for how the robot's joints behave. Furthermore, this programming way offers more simplicity since configuring individual joints it can be in some cases easier to realise than programming a robot to execute a trajectory.
Additionally, in the lower part of the control panel some extra modalities have been added, namely a "Reset Robot" button which brings the robot to the initial position in case the user wants to restart the programming or execute a new session, and two more switches for adding two different grippers on the robot's end-effector,

one static box gripper that is meant to be used for moving box objects around the workstation, and a 2F-140 adaptive Robotiq gripper for pick-and-place tasks. An "Overlay SARA" switch activates the Model Target algorithm and the holographic robot model can be place on top of the physical one for adding a sense of realism in the programming process. Lastly the user has the ability to send the joint configuration to the real robot by enabling the "Publish" switch. The Joint Space manipulation of the robot can be also effective during the robot programming stages. For example, some of the benefits that can offer are the following:

- Simplicity: In some cases, it is easier and more intuitive to control each joint separately, especially for simple tasks or when programming a robot for the first time.
- Direct control: Manipulating individual joints allows direct control over the robot's configuration, which can be helpful in avoiding singularities, joint limits, or other constraints.
- Reduced computation: Compared to trajectory planning, controlling individual joints usually requires less computational resources since it doesn't involve generating complex trajectories and solving inverse kinematics.
- Easier calibration: It can be easier to calibrate the robot by manipulating individual joints, as one can directly relate joint angle changes to corresponding changes in position or orientation.

Overall, this module can be used in different ways for various reasons. For example, through this case the users can familiarise themselves with each joint's capabilities, both virtually and physically. Another interaction concept related to this feature is the correction of a specific pose or point on a trajectory or adjustment of a point on the executed trajectory, into a specific configuration of interest. Although the decision regarding the exploitation of the functionalities, is up to the user to make in a free way, a flowchart in Figure 3.19 displays a suggested workflow that could optimise the robot programming process.

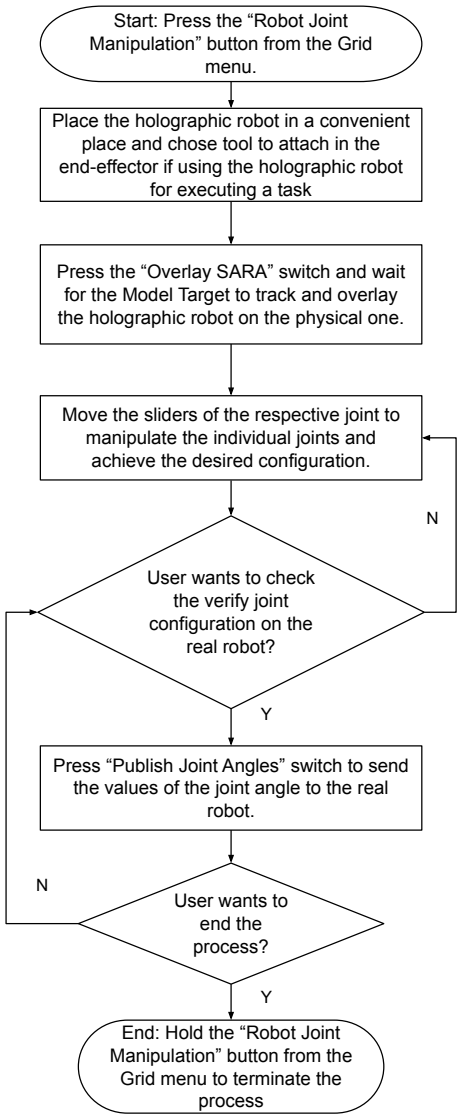


Figure 3.19: The flowchart of a step by step guide for the "Robot Joint Manipulation" module.

3.4.4 Skill Execution and Validation

This feature is meant for individual skill execution and validation. Trough that functionality, the user moves around the end-effector of the holographic robot. Then the demonstrated trajectory, which is executed simultaneously in the Rviz simulated environment. In this case as well the modularity of the interface could be showcased by potentially having the demonstration recorded with the possibility to be added as a new skill in the task graph. The user can observe how the holographic version of the robot behaves when executing the trajectory of the respective skill, and if the behaviour fulfils the requirements then it can be validated through a text-based *Dialog* sequence.

A small panel, depicted in Figure 3.20 is used to control the workflow, and for enabling and disabling the extra functionalities appears at the beginning of the process when the user activates the block functionality, i.e. presses the "Skill Execution and Validation" button from the Grid menu. The button and switch options that are located on this panel include:

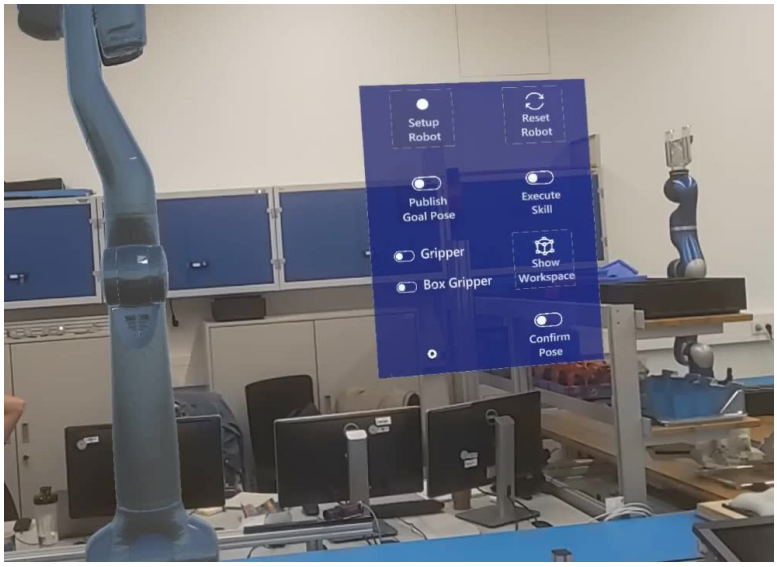


Figure 3.20: The panel for controlling the "Skill execution and Validation" block functionality.

- A "Setup Robot" button which moves the virtual robot into a specific configuration. The purpose of this button is to avoid the occurrence of any possible singularities in the starting pose of the robot's joints configuration.
- A "Reset Robot" button where the user can reset the robot into the initial position, in case a restart is necessary.
- A "Publish Goal Pose" switch, which when enabled, publishes continuously, as a ROS topic, the position and orientation of the end-effector to the backend in order for the inverse kinematic of the robot to be calculated.
- An "Execute Skill" switch where the user is able to record and execute the trajectory of the skill and validate it, if behaving as intended. The visualisation of the skill's trajectory is achieved through subscribing to the calculated by the inverse kinematic joint state topic, namely a ROS topic for the standard sensor message (`sensor_msgs/JointState.msg`) for joint states values.
- A "Show Workspace" button for the user to be aware of the robot's workspace and capabilities, as it is displayed in Figure 3.21. An option of alternate between different grippers as presented in the previous functionality of the "Robot Joints Manipulation" block, where the user can change between an adaptive gripper and a static box gripper.

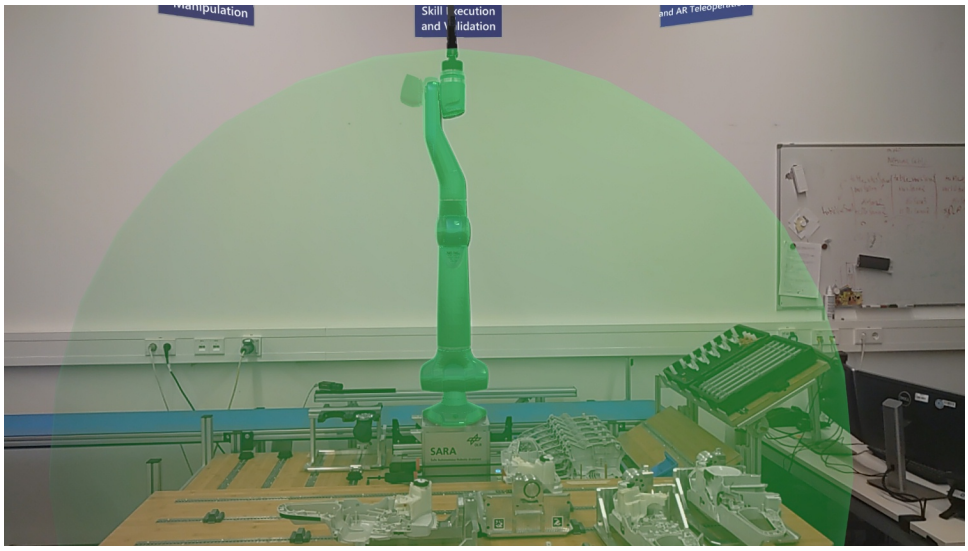


Figure 3.21: The projected workspace of the robot.

- A "Confirm Pose" button, where the user can verify a valid overlay pose of the robot's virtual robot on top of the physical one. The positioning of the holographic robot model on top of the physical one is happening continuously with the Model Target algorithm constantly trying to overlay the robot model. This can have an effect to the actual value of the position, by creating an offset between physical and holographic object and orientation of the holographic robot that is being sent to the control algorithm that runs on the backend. This can also create some perturbations on the calculation of the inverse kinematic, thus producing an unwanted behaviour.

One major issue that had to be resolved, was the transformation between the coordinate systems of HoloLens, and thus Unity, and ROS coordinate system, which uses the real-world coordinate reference. The mathematical formulation of the transformation between the base of the robot and the end-effector is the following:

$$ {}_{B}T^{EE} = ({}_{Gl}T^{B})^{-1} \cdot {}_{Gl}T^{EE} \tag{3.1} $$

where ${}_{B}T^{EE}$ is the transformation matrix of the end-effector relative to the base reference of the robot, ${}_{Gl}T^{B}$ is the transformation matrix of the base relative to the global coordinate frame and ${}_{Gl}T^{EE}$ is the transformation matrix of the end-effector relative to the global coordinate frame.
In Figure 3.22 the coordinate systems are illustrated. Unity uses Left-handed Cartesian Coordinates, namely the X axis points right, Y up, and Z forward. ROS, on the other hand, supports various coordinate frames: in the most commonly-used one, X points forward, Y left, and Z up. In ROS terminology, this frame is called "FLU" (forward, left, up), whereas the Unity coordinate frame would be "RUF" (right, up, forward). A respective transformation had to be applied in order for the robot trajectory to be executed correctly.

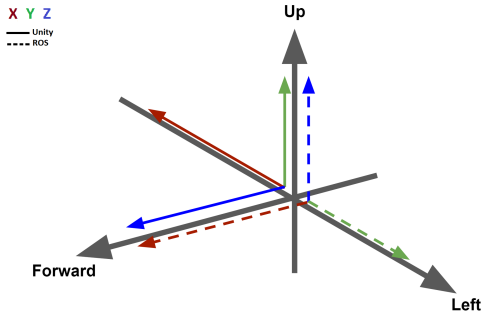


Figure 3.22: The Cartesian coordinate system of the Unity scene and the ROS coordinate system [ROS].

In Figure 3.23 the flowchart of the intended optimised workflow is presented, although again it this case the user is not limited to it for using that module in the actual robot programming procedure.

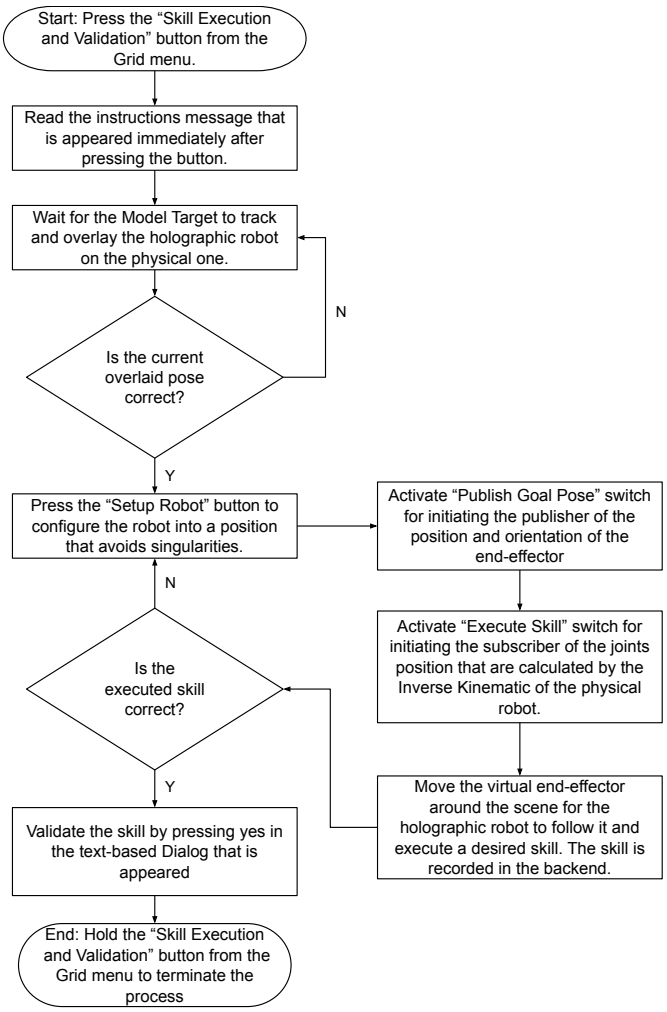


Figure 3.23: Suggested workflow of the ”Skill Execution and Validation” functionality.

3.4.5 Kinesthetic Teaching and AR Teleoperation

The function of this module is based on the decision of the user, regarding the chosen method for the teaching of a new skill. The user interacts with an initial text-based *Dialog* window (Figure 3.24) on which there are two options, namely the activation of the kinesthetic teaching mode and the use of teleoperation through holographic elements. In the first case a command is sent to the backend as a ROS service

Figure 3.24: The text-based message appears immediately after the user chose the fourth functionality.

call where it requests for the robot to be set on demonstration recording mode and the system responds by enabling the kinesthetic teaching mode on the physical robot. Moreover, a command is sent back on the interface from the backend which enables another text-based *Dialog* that informs the user that the robot is ready to record the kinesthetic teaching. In the second case, the user is manipulating the physical robot, whereas in the previous module the interaction was only with the virtual one, in Cartesian Space by moving a holographic version of the end-effector. The holographic unit is represented as a *Bounds Control* cube (MRTK UX tools 3.3.2) around the position of the end-effector (see Figure 3.25) and it uses an *Object Manipulator* (MRTK UX tool as well, section 3.3.2) element through which the user can grab and move the hologram around the scene. A holographic model of the adaptive Robotiq gripper is located inside the cube and is used for the orientation reference of the real end-effector in order to increase the intuition for the user. The position and the orientation of the virtual object are constantly published to the real robot for calculation of the inverse kinematic. The same communication instances

that were used in the previous functionality are used here too. Namely, the interface sends publishes the pose goals of the end-effector to the robot in order to calculate the inverse kinematic and move physically to Cartesian space.

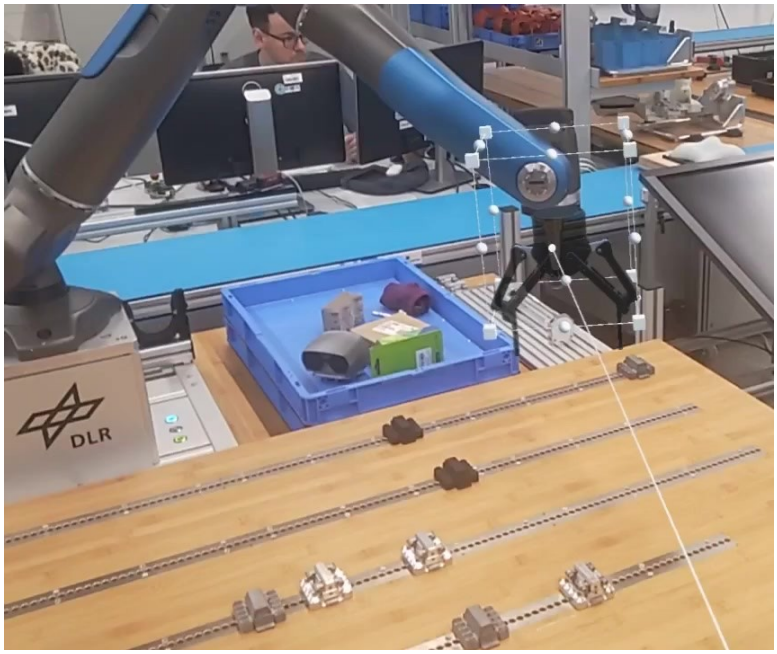


Figure 3.25: The holographic *Bounds Control* object around the end-effector and a virtual gripper is used as orientation reference.

Through this feature, the user is able to exploit an immersive teleoperation scenario, compare it with kinesthetic teaching in different instantiations of he module, and choose the most suitable teaching method to the requirements of the task. That way the physical interaction with the robot is minimised and the safety factor increase as well. Another advantage of the minimisation of physical contact with the robot during demonstration is the addition of an extra layer between the robot and the user. This could be proven beneficial to the teaching process, as described in [SH20], since the human factor could be prone to errors also during demonstrations. This limitation was previously mentioned in 2.2, as a limitation on the expert's abilities to demonstrate a skill. With employing means as the holographic end-effector the user's task is only to drag and drop the virtual object to a desired position. Consequently, the workflow of the fourth feature can be seen in the Figure 3.26.

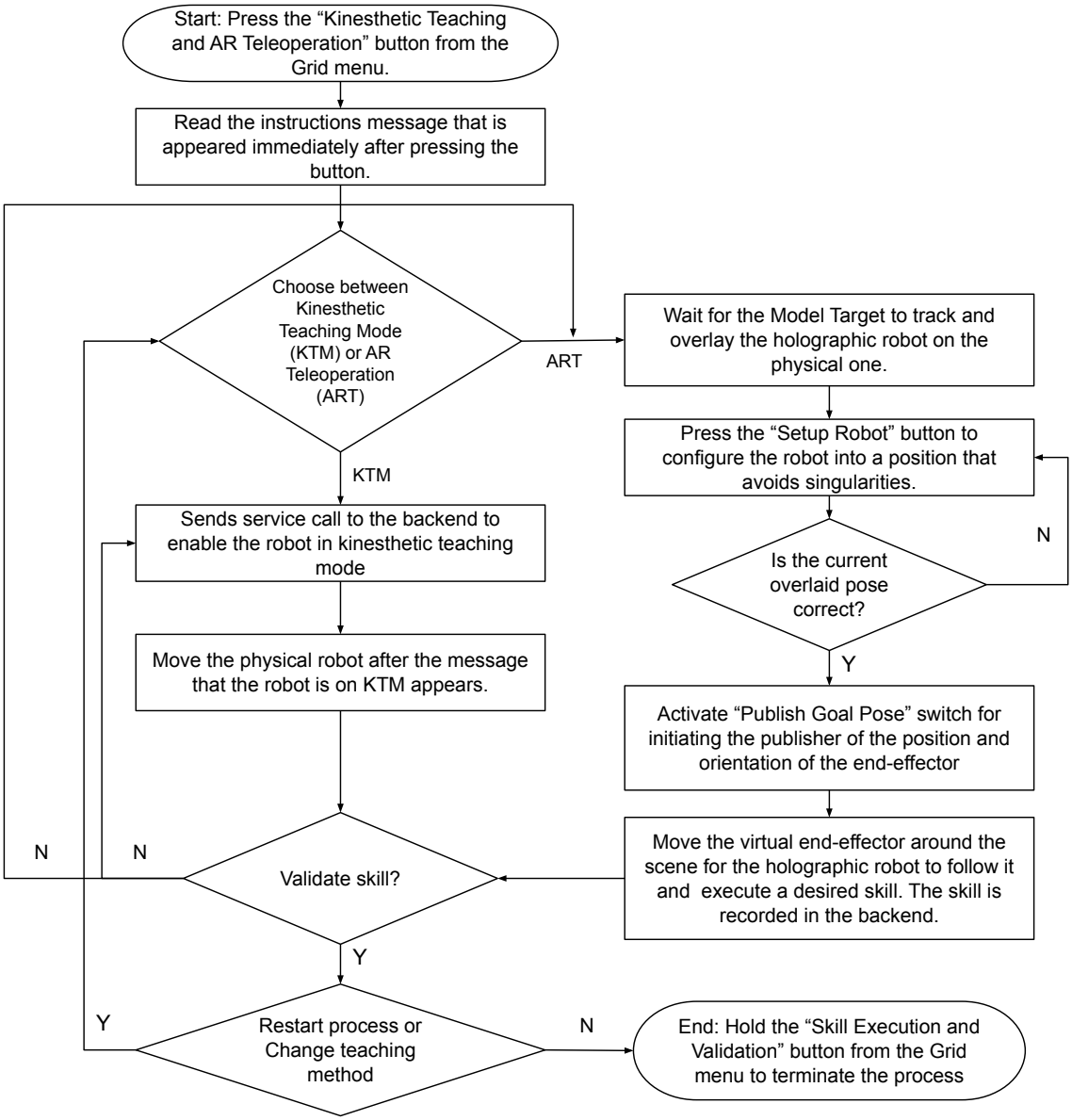


Figure 3.26: The flowchart of the module ”Kinesthetic Teaching and AR Teleoperation”.

3.4.6 Virtual Workbench

The final functionality block can be identified as ”Virtual Workbench” in the Grid menu, and the motivation behind its creation is the scaling possibility of the robot programming procedure. Furthermore, its addition to the main functionalities, transitions the system more towards the Augmented Virtuality area of interest in the Virtuality Continuum previously mentioned in 2.3 and introduced in [MK94]. This scaling and transition is accomplished through the virtualisation of the entire workbench upon which the robot operates, including the various tools and objects used for various tasks, such as container boxes, parts of a product in an assembly line, or

different tools for the end-effector.

In this module, as with the previous ones, there are two modes that the user can change to. The first mode includes the overlay of the workbench and the robot on top of the real ones. The second one is an unconstrained mode where the virtual objects can be moved around the scene according to the user's preferences. Both modes enable an identical joint control panel as seen in section 3.4.2 and 3.4.3 in order for the user to control the robot in Joint Space. In this case, the control panel incorporates more features such as a switch that subscribes to the joint states of the simulated robot in Rviz, or the real robot, for individual skill execution or a complete task. For example, it could be use to validate a skill of a pick and place task, or the complete task as well, and preview the execution on the holographic workbench with the virtual model of the item that is being picked and placed as well. An example of how the combined holographic workbench with the robot on top looks like, can be found in Figure 3.27. The user is able to interact with the

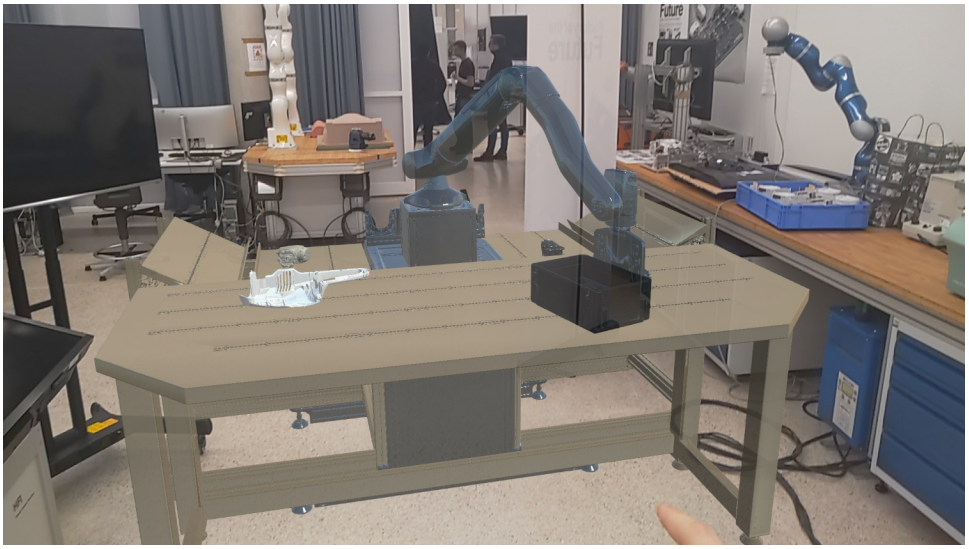


Figure 3.27: The holographic robot and workbench in a free-to-move mode.

objects around the workbench, as well as move around the entire workbench and place it in a convenient spot where it would be possible to preview a set of skills or a task on the holographic robot.

As previously, the control panel provides the possibility to the user to activate the Model Target and position the holographic workbench onto the physical one. When the "Overlay" switch is activated, a text-based Dialog appears to verify that the user indeed wants to overlay the complete workbench. As soon as the workbench is overlaid another Dialog appears to ask whether or not the user wants to overlay the robot as well, as it is depicted in Figure 3.28.

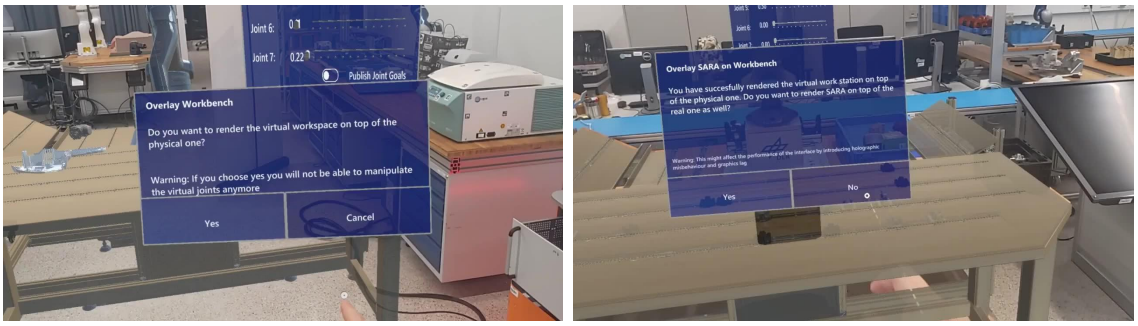


Figure 3.28: The text-based Dialogues for enabling the Model Target activation on the real robot and on the workbench.

If the answer is no, there is still the possibility to overlay the robot afterwards by activating the "Overlay SARA" switch from the control panel. Then the complete overlaid system can be seen in the Figure 3.29. The possibility to overlay individual

Figure 3.29: The overlaid workbench and SARA robot on top of the real one.

objects on the virtual workbench is also presented here. This feature could be potentially exploited in other ways as well, by including the ontology framework into an integrated version of the production line. Through these implementation, the immersive factor could increase significantly by potentially exposing the user into a complete digitalised and virtual setup.

Chapter 4

Evaluation of the Integrated System

4.1 Experimental Design and Scenarios

As it is presented in the taxonomy introduced in [SKX+22] and mentioned in 2.3, an AR interface that is developed for utilisation in the HRI context, can be evaluated through three different methods: a demonstration, technical evaluation or a user study. To determine the potential advantages of the developed AR interface for IRP, a technical evaluation took place. The experiments were designed as simplified versions of possible tasks or skills that could be performed in a manufacturing setting, with the aim of replicating specific trajectories, that could for example be part of a pick-and-place task commonly carried out by robotic systems.
The technical evaluation was realised in the workstation laboratory of the *Factory of the Future* [FoF] project in the Robotics and Mechatronics Center of the German Aerospace Center (DLR). The SARA DLR lightweight robot, which was mentioned in 3.2.2, was used for experimenting with a real robotic system. The key success factor for this application was identified as time savings with similar precision, as well as reduced workload compared to conventional methods. As a side-evaluation for our interface and in order to include extra feedback, a small sample of untrained users tried and tested the different functionalities that the interface offers, in a more abstract way of experimentation. Four out of five different functionalities were tested, including:

1. "**Robot Joints Manipulation**": In the first case, the main goal of the experimentation was meant for the user to establish a first contact with the system and try to change the joint configuration of the holographic robot to a specific position where a box is positioned. The user could perform the task twice, once with the overlaid robot and once with a free of constraints holographic robot that could be positioned anywhere in the scene by the user and not rendered automatically by the interface.

2. "**Skill Execution and Validation**": The technical assessment of this functionality module involved positioning the end-effector in a specific configuration by moving it between two different points and returning to the starting position. The setup was designed in a more abstract way, without specific anchors for the two points in the scene, in order for the demonstration phase to be more flexible. It utilises a text-based *Dialog* validation system too, for the virtually executed trajectory. The evaluation component in this case was accuracy of the systems behaviour to the intended one, mental workload because for the user for the manipulation of the holographic robot and how much time does it take for the process.

3. "**Kinesthetic Teaching and AR Teleoperation**": The experimentation with this functionality included a teleoperation task where the user could move the real robot on the robot's workspace in a unconstrained way. Moreover, a precision task where the tip of a feature tool was passing inside a ring area took place. For safety reasons a virtual box constrained the robot from reaching singularities or hitting with the workbench and other objects. The main goal was to move the Tool Center Point (TCP) to multiple positions around the workbench and compare it with previous experience they had with kinesthetic teaching. Evaluation components of that setup were mainly the mental workload, the precision of the executed task and an overall method preference.

4. "**Virtual Workbench**": The evaluation procedure on this module was more abstract than the others. For experimenting with this functionality, a free-to-play approach was followed, allowing the user to interact freely with the holograms. The primary objective of examining a fully holographic environment lies in delving into the potential opportunities arising from a more immersive system. This exploration facilitates discussions around possible implementation scenarios and how they may be harnessed within the context of human-robot collaboration. The evaluation metrics that were used for this case included the sense of immersion, essentially meaning to ask the user how close this setup is to physical reality and how distinguishable could be while operating in a workstation with the robot. Another important factor that this functionality employs is the safety factor.

The experimental setup that includes, the objective of the experiments, the interaction concept, the robot control method and the sequence of demonstration, is summarised also in Table 4.1.

Table 4.1: The different functionalities that were used during the experiments

	Functionalities			
	Robot Joints Manipulation	**Skill Execution and Validation**	**AR Teleoperation**	**Virtual Workbench**
Objectives	Familiarisation with the robot	Skill validation based on the increased teacher's intuition	Skill/task teaching	Increase immersion and safety factors
Interaction modality	Slider panel for individual joint control	TCP Manipulation, Dialog	TCP Manipulation, Dialog	Slider panel for individual joint control, Dialog
Robot control space	Joint Space	Cartesian Space	Cartesian Space	Joint Space
Sequence of demonstration	execution and demonstration are asynchronous	execution and demonstration are asynchronous	demonstration initiates execution	execution and demonstration are asynchronous
Evaluation metrics	mental workload, physical workload, time, precision/ performance, frustration	mental workload, physical workload, time, precision/ performance, frustration	mental workload, physical workload, time, precision/ performance, frustration, sense of immersion, safety factor	sense of immersion, safety factor, scaling

4.2 Hypotheses

In this section, we will discuss the hypotheses that our technical evaluation aims to prove. Through a series of experiments and tests, we seek to validate the potential benefits and advantages of our developed AR interface. Specifically, we aim to demonstrate the effectiveness of the interface in terms of performance, efficiency, ease of use, immersion factor, safety and scaling. The following hypotheses are presented for the evaluation of the functional features of the developed interface, with the aim of testing and providing supporting evidence.

H1: The individual robot joint manipulation concept, can familiarise the user with the system and increase awareness of the robots capabilities.

H2: The execution of a skill, or a task, and its previewing in a holographic form, can occur iterative without the physical restrictions of the system, thus optimising the process.

H3: The AR teleoperation interaction, when used during the teaching phase, can significantly reduce the physical workload compared to kinesthetic teaching.

H4: The AR-based teleoperation adds an extra layer between the human factor and the robot, thus minimising the error-prone human factor during demonstration.

H5: The AR teleoperation interaction is more appealing to the user by creating an immersive experience, which can also increase the focus and thus the performance of the user during the teaching phase, compared to kinesthetic teaching.

H6: The use of a holographic version of the workstation can enhance the sense of immersion and provide increased safety when it comes to heavy physical workload tasks, and increase the overall intuition of the execution phase in a more holistic manner.

By establishing the validity of these hypotheses through the following experimental results, we can provide valuable insights and evidence for the potential of AR technology and contribute to the development of more advanced and effective human-robot interaction systems.

4.3 Experimental Results

Robot Joints Manipulation

In the first case, the users could easily move the robot from position A to position B (see Figure 4.1) as soon as they familiarised with the control panel that the *Sliders* were located. It was not, however, very appealing for the users to operate, since the manipulation would require precise moving of the *Sliders*. The interaction with the holograms can sometimes be affected by different misbehaviour due to conflicting physics-related components that governs the virtual objects (see section 3.3.2 for *Colliders*, *Rigidbody* and MRTK modalities, i.e. *Object Manipulators* etc.), especially when untrained users are operating it. In our case, this had as an effect to increase the mental demand of some users, for completing the joint manipulation of the holographic robot and an emerged sense of frustration. In the cases of successful attempts, and when the users could easily grasp the notion behind the operation of the control panel and its components, they stated that the mental workload was not significant. Precision and performance, for the successful attempts, were also high

(a) Start of the execution from position A

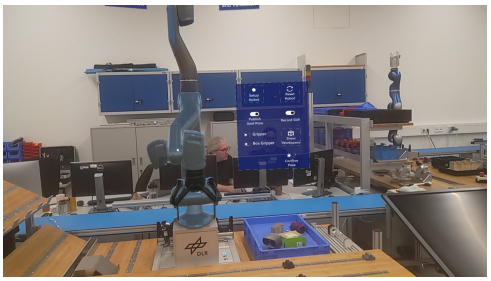
(b) Finish of the execution by moving into position B

Figure 4.1: Manipulation and evaluation of the manipulation of the robot in Joint Space through the "Joint Control Panel".

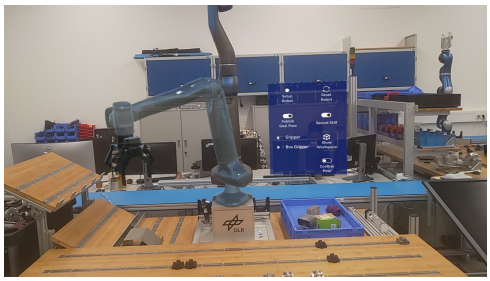
for this simple manipulation case. The two latter evaluation metrics might change values for more complex task that could realised in a complex robot programming procedure.
In a more suitable scenario, the manipulation of the robot in Joint Space using the developed control panel that employs the *Sliders* modality, could be applied more effectively for educational purposes, enabling untrained users to gain a better understanding of the robot's capabilities.

Skill Execution and Validation

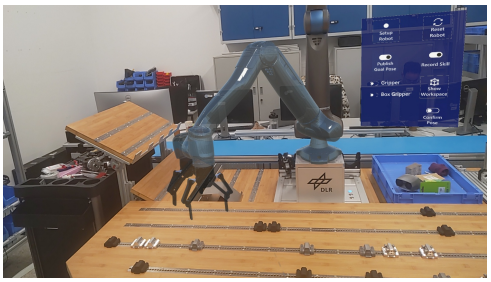
During the evaluation of the third feature, a simple trajectory execution was recorded. The task included moving the holographic robot in Cartesian Space into two different positions and then returning to the initial position. The setup is visible in the Figure 4.2.
According to the users' opinions, the offset that is created after some time of receiving the joint values due to accumulation reasons, was a disadvantage of the feature. The mental workload and precision/ performance of the executed task were negatively affected in some cases. That also increased the frustration of the unsuccessful cases. The completion of task was realised relatively fast for the successful cases The received feedback regarding the possible use of this functionality is also of big importance for the integration process to the real robot.
Users experienced in robot programming, suggested that the examined functionality could be exploited more effectively for previously recorded skills as well and preview any possible parameters editing on the holographic robot. Through that feature the user can understand better the individual steps of the procedure and repeat the skill recording and execution in an iterative way for ameliorating the demonstration process and thus minimising the discrepancy between the robot's learning and the user's knowledge about the system efficiently.



(a) Start of the manipulation, move to position A



(b) Move into position A



(c) Move into position B

(d) Return Home

Figure 4.2: An execution of a skill trajectory that is demonstrated for the robot.

(a) Start of the execution

(b) Move into position A

Figure 4.3: Text-based Dialog modality for skill validation.

Additionally, further experimentation had to realised to determine the physics of the holographic robot to compensate for a possible bottleneck that emerged from the data exchange and the bandwidth limitations of the implemented communication protocol and the network connection.

Kinesthetic Teaching and AR Teleoperation

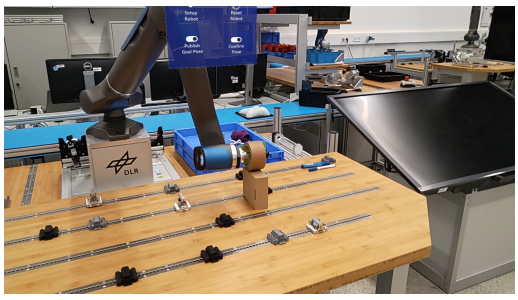
For the AR Teleoperation functionality, abstract trajectories were executed, where the TCP was moving around the scene. The users were manipulating the robot in Cartesian space. When moving it freely, they were instructed to bring it in different poses so that the behaviour of the robot could be determined if it was the intended one, even in complex configurations. The robot's behaviour was indeed very accurate and responsive to the user's command. An example of such a configuration can be

seen in Figure 4.4.

Figure 4.4: An example of a more complex pose.

The users noticed a small offset that although it didn't affect the positioning and the manipulation in the end, but instead caused some small confusion at the beginning a of the process. The precision of the manipulation is very accurate as it is visible in Figure 4.5. They also expressed a sense of enthusiasm when using the app and eagerness to use such a feature. it offered them an improved sense of immersion, too. The former outcome has as an immediate consequence the minimisation of the frustration factor, too.

(a) Start of the execution



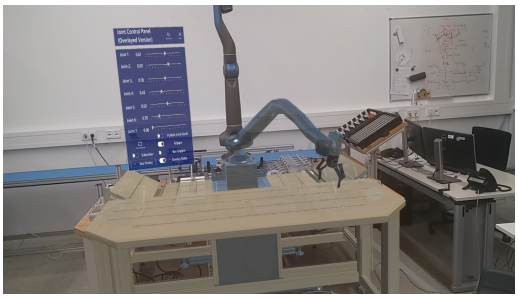
(b) Move into position A

Figure 4.5: AR Teleoperation for a precision task.

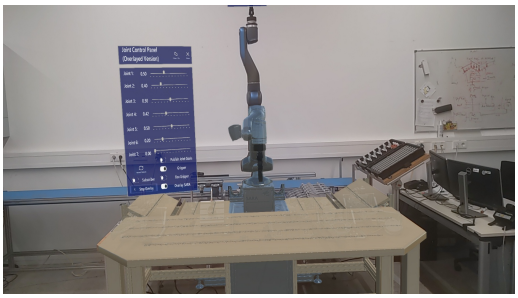
A disadvantage of the immersive teleoperation funcitonality, is the lack of haptic feedback which could affect the effectiveness of the demonstration and the user's intuition of the process. This effectively means that the mental workload is increased in comparison with the physical workload where there is no physical contact with the robot. Although, in our case the absence of haptic feedback did not affect the task execution, for more complex robot programming procedure this should be taken into consideration. Lastly, since the teleoperation is by definition a method operating at distance, the safety factor is unequivocally increased with the utilisation of AR features as implemented in our case.

Virtual Workbench

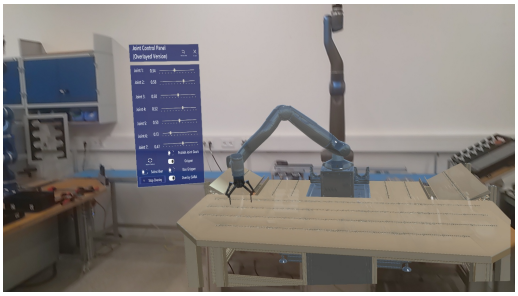
The experiments on the overlaid virtual workbench included also some individual joint manipulation in Joint Space, to show the concept of the complete and operational workstation. As it can be seen in Figure 4.6, the robot was moved into three different configurations before returning in home position. Although the setup is quite similar to the one in the first case, the logic behind the inclusion of this experimental case is aiming more to unravel the levels that the immersion factor can reach through this interface. This can be achieved by incorporating additional objects into the scene, which could potentially be utilised for a task and manipulated accordingly.



(a) Start of the execution



(b) Move into position A

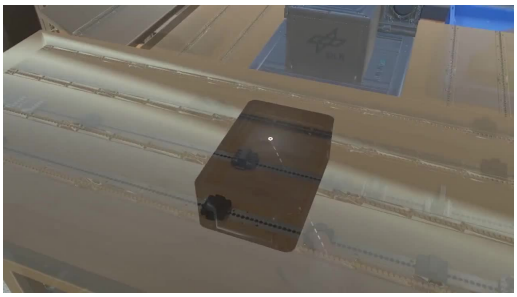


(c) Move into position B

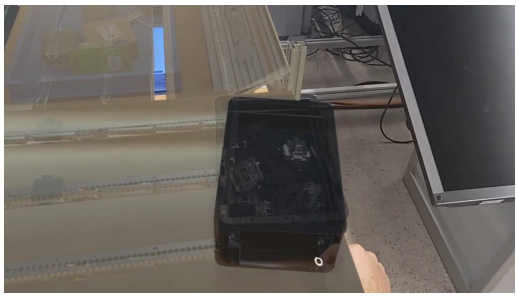
(d) Return Home

Figure 4.6: The overlaid workbench creating a more immersive sensation of the scene.

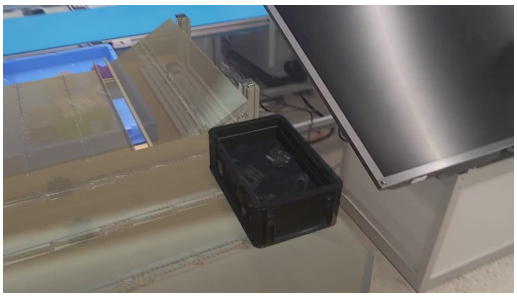
The concept of positioning an object in the virtual workbench is depicted in the Figure 4.7. The user can either overlay the virtual object on top of the real one or freely position the object at a specific location in the workbench. This location can be used as utilised in a scaling scenario were it is stored in the ontology's database in order to be used in a way that could digitalise the workstation and hence the manufacturing process. The latter is also the most effective way to accomplish immersive awareness in the scene in a such a manner that can be productively exploited. The possibility that the feature could integrate the ontology of a workstation into the interface, can be used as a metric for postulate that the scaling factor of the proposed functionality is effectively increased. This concepts of scaling are comprehensively elaborated in the next section.



(a) Virtual box positioned on the overlaid workbench



(b) Box moved to the position of the real one



(c) Positioning the box on top of the real one

Figure 4.7: The ontology concept and how it can be utilised through the interface with the example of box positioning.

4.4 Discussion

The technical evaluation revealed that the use of the proposed AR interface in conjunction with Intuitive Robot Programming has significant potential for improvement, especially for users with prior experience with AR/MR headsets. However, a steep learning curve was observed for users with limited or no experience with similar headsets, resulting in limited observed temporal improvements. Nevertheless, most of the participants showed improved success rates in the usage of some functionalities after a short period of self-exploration and practice. These findings suggest the need for further investigation to determine the optimal training and on-boarding methods for new users of AR interfaces in industrial settings. Furthermore, the interface's capabilities, especially features like the AR teleoperation, could be applied in a manufacturing line where heavy industrial robots are operating by humans, thus increasing the safety factor, which it is crucial in such setups.

Increased Intuition: With hypotheses **H2** and **H3** to be supported by the results mentioned in the previous section, more specifically during the results mentioned in the results of the AR teleoperation. **H1** could also be observed for users that are aware of the robot's capabilities without being limited to them. Also during the experiments of the precision task that took place, an observed behaviour that is described in **H4** was displayed and deduced by the result. It goes without saying that the hypothesis **H5** was obseved for every case of a user that tried the system since it is a technique that is rarely used by individuals.

Increased Immersion The tendency of hypothesis **H6** is also discussed next. The safety factor and remote operation is a concept also demonstrated during the experiments of the "Virtual Workbench" module. By setting up a holographic working station the processes can be simulated and safely observe if their behaviour is the intended one, before executing in the real system. Another interesting outcome of the experimentation on this feature, is the conceptualisation of a more digitalised industrial environment. A feature of this setup on a real laboratory-manufacturing environment, could be the inclusion of the ontology on an integrated system. An ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. It utilises database systems, in order to serve as a level of abstraction for modeling knowledge about individual objects and their respective attributes and relationships, analogous to hierarchical and relational models [Gru]. In the Figure 4.8 the concept and setup of the Factory of the Future laboratory is depicted, which is designed to simulate various workstations of a manufacturing line and conduct research on future manufacturing, with the aim of developing a manufacturing paradigm that aligns with and surpasses the limits of Industry 4.0.

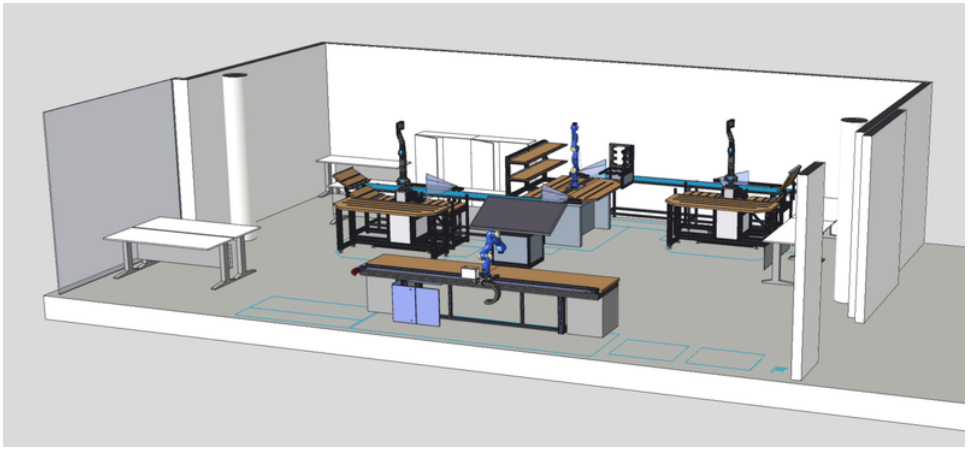


Figure 4.8: The main setup of the Factory of the Future laboratory at DLR Robotics and Mechatronics Center [RMC].

The position of each object on the top of the workbench is mapped and stored in the ontology and any prospective task that involves some object (a tool, a box a product in the assembly line etc.), can be executed in a flexible way. The visualisation feature of the ontology through the interface, can help the user in various ways, some of which are listed below:

- With means of gesture interaction provided by the interface, the user can drag and place around the workbench the different components and items.
- The user can verify if the arranged layout is valid.
- Potentially the user can submit the self-arranged physical layout to the database that all the related information is stored and start the reconfiguration process.

The attainable integration and implementation of ontology in the interface could enable universal awareness and the creation of a holographic version of the current setup of the workstation and the operations that can be carried out on them. Potential advantages of such feature include but not limited to, data integration from different sources (ERP, sensors), decision support, intuitive collaboration between the human operator of the HoloLens and the robot and knowledge discovery in order to make the teaching more efficient by introducing an extra input to the system, i.e. some predefined patterns and a common vocabulary that could be identify from the robot, HoloLens and other devices.

Chapter 5

Conclusion

In the concluding chapter, we ascertain the implications of the analytical and experimental outcomes, as well as review the objectives accomplished by the developed interface. Furthermore, we define the limitations inherent to our methodology and outline an outlook for potential improvements and expansion of our work.

5.1 Summary of the Proposed Approach

The main objective of the framework we conceptualised within the scope of our research, is the manifestation of an Augmented Reality interface that enables intuitive robot programming through demonstration. The premises for the argumentation in favor of the validity of the aforementioned claim, can be deduced from the confirmed hypotheses supported by the experimental results on the proposed interface. Through the use of the AR interface, the robot programming introduces immersive characteristics that could potentially increase the user's intuition about the capabilities of the robot, how to efficiently perform programming and manipulation tasks. Furthermore, the theoretical concept of five different functionalities is introduced, within a modular implementation framework. The integrated concept utilises demonstrations provided by the user in different modes. The interface provides flexibility to the user on the demonstration choice between virtual teaching where the holographic robot is executing the trajectory provided by the user, kinesthetic teaching and immersive teleoperation through augmented elements. The recorded demonstration could be used to generate a behavioural task graph containing the segmented task into a sequence of skills. Each skill could be previewed in the AR interface in order to verify that the behaviour of the robot is the intended, then dynamically edited by the user, if necessary. After editing the constraint parameters, the user could validate the skill and renew or extend the graph.
A key feature of our interface is the provision of demonstration through immersive teleoperation scenario that utilises AR elements. The main motivation behind this feature is the intention to create an interface that enhances the user's intuition by minimising the knowledge discrepancy between the robot's capabilities and the

user's understanding about the robot's capability range during demonstration. A secondary motivation that is hypothesised to affect the robot programming procedure could potentially be an increasing immersion factor. To that end, another feature was developed, that of a completely virtual workstation which potentially could exploit ontology capabilities in a manufacturing environment in order to visualise, simulate and consequently optimise the processes.

5.2 Limitations

The use of HoloLens 2 for developing an Intuitive Robot Programming interface, has shown promise, however, several limitations must be considered when adopting this technology not only in the scientific domain but most importantly in the actual manufacturing environment as well. In this response, this section discusses the limitations in terms of hardware, software, user experience, and environmental factors.

Hardware-wise, the device has inherent limitations in terms of processing power, field of view (FOV), and battery life. These constraints may lead to slow rendering of AR content, limited interactivity, and reduced usage time. Furthermore, the need for high precision in robot programming may be hindered by the device's tracking accuracy, potentially affecting the efficacy of AR-assisted tasks.Moreover, the system's underlying software may lack compatibility with existing robot programming frameworks or require additional development effort for integration. This could create a barrier to adoption and limit the extent to which AR can be utilised in robot programming tasks. Regarding the user's, the effectiveness of the headset depends on their ability to provide intuitive and seamless experiences. However, the system may suffer from usability issues such as increased complexity of the interface, visual discomfort, or cognitive overload, which could negatively impact the learning curve and adoption of the interface in a robot programming environment.

The performance of similar interfaces using HoloLens 2 or similar devices, can be affected by external factors such as lighting conditions, physical space constraints, and occlusion. These factors may limit the applicability of AR in certain robot programming scenarios or necessitate additional setup and calibration efforts. An important factor when developing an interface that employs AR capabilities is the ability to generalise and adapt across different robotic platforms, programming languages, and tasks. However, achieving such generalisation may be challenging, given the diversity of robotic systems and the rapidly evolving landscape of robot programming methodologies. Acknowledging these limitations and addressing them through continued research and development will be crucial for the successful integration of AR into the robot programming domain for the future.

5.3 Future Work

Further research and implementation needs to be conducted towards the AR interface. New features could be added to the system, including for example an integrated version of the task graph functionality. It could be applied to a specific use-case along with a meticulously structured user study that could prove the benefits of using the AR interface. Another feature that was discussed extensively in section 4.4, is the bidirectional integration of the ontology visualisation on the interface. Through the AR interface, the virtual paths of the objects could be rendered and the components could be placed by the user in either a predetermined or more adaptable manner. Furthermore, the user could validate the positioning of the holographic objects in the scene and reconfigure them accordingly in a self-arranged layout.
As a theoretical concept, the paradigm of the manufacturing line mentioned in, could scale-up even more in a virtual setup. An advanced concept could be the incorporation of multiple virtual workstations simulating the complete production process of a specific product from start to end, in an effort to optimise the manufacturing process by finding error prone stages. Multiple HoloLens devices could be also connected to a network and utilise a swarm like like control to achieve synchronisation during the process executions. Cloud capabilities could render the development phase of such a system also easier to realise. This concept could also introduced the theoretical notion of *intersubjectivity* into the system. *Intersubjectivity* is related only to the human factor and how the different users, while operating in immersive Mixed Reality environment, would share understanding, interpretation and meaning that emerges through communication, social interaction and shared experiences. This emergence, combines elements of both virtual and physical environments, and could serve as a platform for fostering *intersubjective* perception not only between humans, but also during human-robot collaboration. Potentially, it could also build up a workforce of robot programmers that would develop advanced skills on process involving in a industrial environment, thus leading to the next wave of industrial revolution.

List of Figures

Acronyms and Notations

HRC Human-Robot Collaboration

IRP Intuitive Robot Programming

PbD Programming by Demonstration

AR Augmented Reality

VR Virtual Reality

MR Mixed Reality

HMD Head Mounted Display

VC Virtuality Continuum

DOF Degree of Freedom

LfD Learning from Demonstration

T-AOG Temporal And-Or Graph

RL Reinforcement Learning

IRL Inverse Reinforcement Learning

CBNIRL Constraint Bayesian Non-Parametric Inverse Reinforcement Learning

TG Task Graph

HRI Human-Robot Interaction

DMP Dynamic Movement Primitives

GMM Gaussian Mixture Model

GMR Gaussian Mixture Regression

TCP Transmission Control Protocol

TCP Tool Center Point

TLP Task Level Programming

TLPbD Task-Level Programming by Demonstration

PDDL Planning Domain Definition Language

MC Monte Carlo

Bibliography

[ABB+01] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, 2001. doi:10.1109/38.963459.

[AC05] J. Aleotti and S. Caselli. Trajectory clustering and stochastic approximation for robot programming by demonstration. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1029–1034, 2005. doi:10.1109/IROS.2005.1545365.

[AN04] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 1, New York, NY, USA, 2004. Association for Computing Machinery. doi:10.1145/1015330.1015430.

[AS11] Baris Akgun and Kaushik Subramanian. Robot learning from demonstration : Kinesthetic teaching vs . teleoperation. 2011.

[BAC04] Darrin Bentivegna, Christopher Atkeson, and Gordon Cheng. Learning tasks from observation and practice. *Robotics and Autonomous Systems*, 47:163–169, 06 2004. doi:10.1016/j.robot.2004.03.010.

[BCDS08] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. *Robot Programming by Demonstration*, pages 1371–1394. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-30301-5_60.

[BF92] Gary Bishop and Henry Fuchs. Research directions in virtual environments: Report of an nsf invitational workshop, march 23-24, 1992, university of north carolina at chapel hill. *SIGGRAPH Comput. Graph.*, 26(3):153â177, aug 1992. doi:10.1145/142413.142416.

[BG13] A. Billard and D. Grollman. Robot learning by demonstration. *Scholarpedia*, 8(12):3824, 2013. revision #138061. doi:10.4249/scholarpedia.3824.

[BS23] István Barakonyi and Dieter Schmalstieg. Exploiting the physical world as user interface in augmented reality applications. 03 2023.

[BWP+18] Sebastian Blankemeyer, Rolf Wiemann, Lukas Posniak, Christoph Pregizer, and Annika Raatz. Intuitive robot programming using augmented reality. *Procedia CIRP*, 76:155–160, 2018. 7th CIRP Conference on Assembly Technologies and Systems (CATS 2018). URL: https://www.sciencedirect.com/science/article/pii/S2212827118300933, doi:https://doi.org/10.1016/j.procir.2018.02.028.

[Cal09] S. Calinon. *Robot Programming by Demonstration: A Probabilistic Approach*. Engineering sciences. CRC, 2009. URL: https://books.google.de/books?id=7l65QwAACAAJ.

[CAL16] Sylvain CALinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9, 01 2016. doi:10.1007/s11370-015-0187-9.

[Cal18] Sylvain Calinon. *Learning from Demonstration (Programming by Demonstration)*, pages 1–8. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018. doi:10.1007/978-3-642-41610-1_27-1.

[CGRR18] Pietro Cipresso, Irene Alice Chicchi Giglioli, Mariano Alcañiz Raya, and Giuseppe Riva. The past, present, and future of virtual and augmented reality research: A network and cluster analysis of the literature. *Frontiers in Psychology*, 9, 2018. URL: https://www.frontiersin.org/articles/10.3389/fpsyg.2018.02086, doi:10.3389/fpsyg.2018.02086.

[CL19] Sylvain Calinon and Dongheui Lee. *Learning Control*, pages 1261–1312. Springer Netherlands, Dordrecht, 2019. doi:10.1007/978-94-007-6046-2_68.

[Cop] Matt Coppinger. Unlocking workforce productivity with spatial computing. (accessed 12-03-2023). URL: https://octo.vmware.com/unlocking-workforce-productivity-spatial-computing/.

[CT14] Maya Cakmak and Andrea L. Thomaz. Eliciting good teaching from humans for machine learners. *Artificial Intelligence*, 217:198–215, 2014. URL: https://www.sciencedirect.com/science/article/pii/S0004370214001143, doi:https://doi.org/10.1016/j.artint.2014.08.005.

[CV07] Sonia Chernova and Manuela Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of*

the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '07, New York, NY, USA, 2007. Association for Computing Machinery. doi:10.1145/1329125.1329407.

[CV08] Sonia Chernova and Manuela Veloso. Multi-thresholded approach to demonstration selection for interactive robot learning. In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction*, HRI '08, page 225â232, New York, NY, USA, 2008. Association for Computing Machinery. doi:10.1145/1349822.1349852.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1977.tb01600.x, arXiv:https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1977.tb01600.x, doi:https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.

[EK08] Staffan Ekvall and Danica Kragic. Robot learning from demonstration: A task-level planning approach. *International Journal of Advanced Robotic Systems*, 5(3):33, 2008. arXiv:https://doi.org/10.5772/5611, doi:10.5772/5611.

[EMI] FRANKA EMIKA. (accessed 19-03-2023). URL: https://www.franka.de/interaction.

[FoF] DLR RMC FoF. (accessed 22-03-2023). URL: https://factory-of-the-future.dlr.de/.

[GBEL15] Hind Gacem, Gilles Bailly, James Eagan, and Eric Lecolinet. Finding objects faster in dense environments using a projection augmented robotic arm. In *Human-Computer Interaction INTERACT 2015*, pages 221–238, Berlin, Heidelberg, 2015. doi:10.1007/978-3-319-22698-9_15.

[GHCB07] Florent Guenter, Micha Hersch, Sylvain Calinon, and Aude Billard. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, 21:1521 – 1544, 2007.

[Gru] Tom Gruber. (accessed 27-03-2023). URL: https://tomgruber.org/writing/definition-of-ontology.

[GSLZ14] Dominic Gorecky, Mathias Schmitt, Matthias Loskyll, and Detlef Zühlke. Human-machine-interaction in the industry 4.0 era. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 289–294, 2014. doi:10.1109/INDIN.2014.6945523.

[Har] Microsoft Hardware. (accessed 22-03-2023). URL: https://learn.microsoft.com/en-us/hololens/hololens2-hardware.

[HB18] Thomas Hellström and Suna Bensch. Understandable robots - what, why, and how. *Paladyn, Journal of Behavioral Robotics*, 9(1):110–123, 2018. URL: https://doi.org/10.1515/pjbr-2018-0009 [cited 2023-03-10], doi:doi:10.1515/pjbr-2018-0009.

[HO22] Jakob Hörbst and Horst Orsolits. Mixed reality hmi for collaborative robots. In Roberto Moreno-Díaz, Franz Pichler, and Alexis Quesada-Arencibia, editors, *Computer Aided Systems Theory – EUROCAST 2022*, pages 539–546, Cham, 2022. Springer Nature Switzerland.

[HTR14] Thomas Howard, Stefanie Tellex, and Nicholas Roy. A natural language planner interface for mobile manipulators. pages 6652–6659, 05 2014. doi:10.1109/ICRA.2014.6907841.

[INS02] A.J. Ijspeert, Jun Nakanishi, and Stefan Schaal. Learning attractor landscapes for learning motor primitives. volume 15, pages 1523–1530, 01 2002.

[JT13] Nikolay Jetchev and Marc Toussaint. Fast motion planning from experience: Trajectory prediction for speeding up movement generation. *Autonomous Robots*, 34, 01 2013. doi:10.1007/s10514-012-9315-y.

[KCC10] Petar Kormushev, Sylvain Calinon, and Darwin G. Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3232–3237, 2010. doi:10.1109/IROS.2010.5649089.

[KP08] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Mach. Learn. J*, 84:171–203, 01 2008. doi:10.1007/s10994-010-5223-6.

[KPK+15] Daniel Kappler, Peter Pastor, Mrinal Kalakrishnan, Manuel Wuthrich, and Stefan Schaal. Data-driven online decision making for autonomous manipulation. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, 2015.

[KT99] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[KVBT20] Henrich Kolkhorst, Joseline Veit, Wolfram Burgard, and Michael Tangermann. A robust screen-free brain-computer interface for robotic object selection. *Frontiers in Robotics and AI*, 7, 2020. URL: https://www.frontiersin.org/articles/10.3389/frobt.2020.00038, doi:10.3389/frobt.2020.00038.

[LGAL18] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation, 2018. arXiv:1707.03374.

[LS15] Florian Leutert and Klaus Schilling. Augmented reality for telemaintenance and -inspection in force-sensitive industrial robot applications. *IFAC-PapersOnLine*, 48(10):153–158, 2015. 2nd IFAC Conference on Embedded Systems, Computer Intelligence and Telematics CESCIT 2015. URL: https://www.sciencedirect.com/science/article/pii/S240589631500991X, doi:https://doi.org/10.1016/j.ifacol.2015.08.124.

[LSR] Microsoft Hardware LSR. (accessed 22-03-2023). URL: https://learn.microsoft.com/en-us/azure/remote-rendering/overview/features/late-stage-reprojection.

[LZS+18] Hangxin Liu, Yaofang Zhang, Wenwen Si, Xu Xie, Yixin Zhu, and Song-Chun Zhu. Interactive robot knowledge patching using augmented reality. pages 1947–1954. IEEE Press, 2018. doi:10.1109/ICRA.2018.8462837.

[McK] McKinsey. (accessed 01-03-2023). URL: https://www.mckinsey.de/capabilities/operations/our-insights/industry-40-reimagining-manufacturing-operations-after-covid

[MEO+17] Guilherme Maeda, Marco Ewerton, Takayuki Osa, Baptiste Busch, and Jan Peters. Active incremental learning of robot movement primitives. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 37–46. PMLR, 13–15 Nov 2017. URL: https://proceedings.mlr.press/v78/maeda17a.html.

[MH12] Bernard Michini and Jonathan P. How. Bayesian nonparametric inverse reinforcement learning. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 148–163, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Mic] Microsoft. (accessed 22-03-2023). URL: https://www.microsoft.com/en-us/hololens/hardware.

[MK94] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE Trans. Information Systems*, vol. E77-D, no. 12:1321–1329, 12 1994.

[MKKP13] K. Mülling, J. Kober, O. Kroemer, and J. Peters. Learning to select and generalize striking movements in robot table tennis. *International Journal of Robotics Research*, 32(3):263–279, 2013. doi:10.1177/0278364912472380.

[MRT] MRTK. (accessed 22-03-2023). URL: https://github.com/microsoft/MixedRealityToolkit-Unity.

[MTUK94] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: A class of displays on the reality-virtuality continuum. *Telemanipulator and Telepresence Technologies*, 2351, 01 1994. doi:10.1117/12.197321.

[MV20] Zhanat Makhataeva and Huseyin Atakan Varol. Augmented reality for robotics: A review. *Robotics*, 9(2), 2020. URL: https://www.mdpi.com/2218-6581/9/2/21, doi:10.3390/robotics9020021.

[ND07] Chrystopher L. Nehaniv and Kerstin Dautenhahn. *Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions*. Cambridge University Press, 2007. doi:10.1017/CBO9780511489808.

[NOK+15] Scott Niekum, Sarah Osentoski, George Konidaris, Sachin Chitta, Bhaskara Marthi, and Andrew G. Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015. arXiv:https://doi.org/10.1177/0278364914554471, doi:10.1177/0278364914554471.

[OK18] M. Ostanin and A. Klimchik. Interactive robot programing using mixed reality. *IFAC-PapersOnLine*, 51(22):50–55, 2018. 12th IFAC Symposium on Robot Control SYROCO 2018. URL: https://www.sciencedirect.com/science/article/pii/S2405896318332233, doi:https://doi.org/10.1016/j.ifacol.2018.11.517.

[OME+20] Mikhail Ostanin, Stanislav Mikhel, Alexey Evlampiev, Valeria Skvortsova, and Alexandr Klimchik. Human-robot interaction for

robotic manipulator programming in mixed reality. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2805–2811, 2020. doi:10.1109/ICRA40945.2020.9196965.

[Pau21] Leo Pauly. Seeing to learn: Observational learning of robotic manipulation tasks. March 2021. URL: https://etheses.whiterose.ac.uk/29169/.

[PDS+14] Patrick Pessaux, Michele Diana, Luc Soler, Tullio Piardi, Didier Mutter, and Jacques Marescaux. Towards cybernetic surgery: robotic and augmented reality-assisted liver segmentectomy. *Langenbeck's archives of surgery / Deutsche Gesellschaft fur Chirurgie*, 400, 11 2014. doi:10.1007/s00423-014-1256-9.

[PKDZ07] Michael Pardowitz, Steffen Knoop, Ruediger Dillmann, and Raoul D. Zollner. Incremental learning of tasks from user demonstrations, past experiences, and vocal comments. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):322–332, 2007. doi:10.1109/TSMCB.2006.886951.

[PKRS12] P. Pastor, M. Kalakrishnan, L. Righetti, and S. Schaal. Towards Associative Skill Memories. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 309–315, Osaka, Japan, November 2012. IEEE. URL: https://ieeexplore.ieee.org/abstract/document/6651537/, doi:10.1109/HUMANOIDS.2012.6651537.

[PNP+20] Daehyung Park, Michael Noseworthy, Rohan Paul, Subhro Roy, and Nicholas Roy. Inferring task goals and constraints using bayesian nonparametric inverse reinforcement learning. In *Proceedings of the Conference on Robot Learning*, pages 1005–1014, 2020.

[PVS03] Jan Peters, Sethu Vijayakumar, and Stefan Schaal. Reinforcement learning for humanoid robotics. *Proceedings of the third IEEE-RAS international conference on humanoid robots*, pages 1–20, 01 2003.

[QDK18] Long Qian, Anton Deguet, and Peter Kazanzides. Arssist: augmented reality on a head-mounted display for the first assistant in robotic surgery. *Healthcare Technology Letters*, 5(5):194–200, 2018. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/htl.2018.5065, arXiv:https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/htl.2018.5065, doi:https://doi.org/10.1049/htl.2018.5065.

[QLP+18] Camilo Perez Quintero, Sarah Li, Matthew KXJ Pan, Wesley P. Chan, H.F. Machiel Van der Loos, and Elizabeth Croft. Robot programming through augmented trajectories in augmented reality. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1838–1844, 2018. doi:10.1109/IROS.2018.8593700.

[Ras99] Carl Rasmussen. The infinite gaussian mixture model. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper/1999/file/97d98119037c5b8a9663cb21fb8ebf47-Paper.pdf.

[RGB11] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

[RMC] DLR RMC. (accessed 21-03-2023). URL: https://factory-of-the-future.dlr.de/.

[ROS] Siemens ROS#. (accessed 30-03-2023). URL: https://github.com/siemens/ros-sharp/wiki/Dev_ROSUnityCoordinateSystemConversion.

[RPCB20] Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):297–330, 2020. doi:10.1146/annurev-control-100819-063206.

[SAC17] Yasaman S. Sefidgar, Prerna Agarwal, and Maya Cakmak. Situated tangible robot programming. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17, pages 473–482, New York, NY, USA, 2017. Association for Computing Machinery. doi:10.1145/2909824.3020240.

[SAMB12] Eric L. Sauser, Brenna D. Argall, Giorgio Metta, and Aude G. Billard. Iterative learning of grasp adaptation through human corrections. *Robotics and Autonomous Systems*, 60(1):55–71, 2012. URL: https://www.sciencedirect.com/science/article/pii/S0921889011001631, doi:https://doi.org/10.1016/j.robot.2011.08.012.

[SAR] DLR RMC SARA. (accessed 22-03-2023). URL: https://www.dlr.de/rm/sara#gallery/29681.

[SCLO21] Gian Maria Santi, Alessandro Ceruti, Alfredo Liverani, and Francesco Osti. Augmented reality in industry 4.0 and future innovation programs. *Technologies*, 9:33, 04 2021. doi:10.3390/technologies9020033.

[SD13] T. Sapounidis and Stavros Demetriadis. Tangible versus graphical user interfaces for robot programming: Exploring cross-age children's preferences. *Personal and Ubiquitous Computing*, 17, 12 2013. doi:10.1007/s00779-013-0641-7.

[SH20] Aran Sena and Matthew Howard. Quantifying teaching behavior in robot learning from demonstration. *The International Journal of Robotics Research*, 39(1):54–72, 2020. arXiv:https://doi.org/10.1177/0278364919884623, doi:10.1177/0278364919884623.

[SKG+16] Susanne Stadler, Kevin Kain, Manuel Giuliani, Nicole Mirnig, Gerald Stollnberger, and Manfred Tscheligi. Augmented reality for industrial robot programmers: Workload analysis for task-based, augmented reality-supported robot control. pages 179–184, 2016. URL: https://uwe-repository.worktribe.com/output/906007, doi:10.1109/ROMAN.2016.7745108.

[Sko09] Alexander Skoglund. *Programming by demonstration of robot manipulators*. PhD thesis, Ãrebro University, School of Science and Technology, 2009.

[SKX+22] Ryo Suzuki, Adnan Karim, Tian Xia, Hooman Hedayati, and Nicolai Marquardt. Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces. CHI '22, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3491102.3517719.

[SMSM99] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

[SNS19] Franz Steinmetz, Verena Nitsch, and Freek Stulp. Intuitive task-level programming by demonstration through semantic skill recognition. *IEEE Robotics and Automation Letters*, 4(4):3742–3749, 2019. doi:10.1109/LRA.2019.2928782.

[SPK02] W.D. Smart and L. Pack Kaelbling. Effective reinforcement learning for mobile robots. In *Proceedings 2002 IEEE International Conference on*

Robotics and Automation (Cat. No.02CH37292), volume 4, pages 3404–3410 vol.4, 2002. doi:10.1109/ROBOT.2002.1014237.

[SSK20] Pepi Stavropoulou, Dimitris Spiliotopoulos, and Georgios Kouroupetroglou. *Voice User Interfaces for Service Robots: Design Principles and Methodology*, pages 489–505. 07 2020. doi:10.1007/978-3-030-49282-3_35.

[Tar] Vuforia Model Target. (accessed 25-03-2023). URL: https://library.vuforia.com/device-tracking/spatial-frame-reference.

[TSM85] D.M. Titterington, A.F.M. Smith, and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions*. Wiley, New York, 1985.

[URH] Unity-Technologies/ Unity-Robotics-Hub. (accessed 19-03-2023). URL: https://github.com/Unity-Technologies/Unity-Robotics-Hub/blob/main/tutorials/ros_unity_integration/README.md.

[vLGG19] Tanja von Leipzig, Pieter Gouws, and M Greeff. Game-based learning and virtual reality: Innovation in performance improvement. 02 2019.

[WEL20] Christoph Willibald, Thomas Eiband, and Dongheui Lee. Collaborative programming of conditional robot tasks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5402–5409, 2020. doi:10.1109/IROS45743.2020.9341212.

[WL22] Christoph Willibald and Dongheui Lee. Multi-level task learning based on intention and constraint inference for autonomous robotic manipulation. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7688–7695, 2022. doi:10.1109/IROS47612.2022.9981288.

[YYD22] Congcong Ye, Jixiang Yang, and Han Ding. Bagging for gaussian mixture regression in robot learning from demonstration. 33(3):867–879, March 2022. doi:10.1007/s10845-020-01686-8.

[ZMBD08] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, pages 1433–1438, 2008.

License