Technische Universität München
Fakultät für Maschinenwesen
Lehrstuhl für Automatisierung und Informationssysteme
Prof. Dr.-Ing. Birgit Vogel-Heuser

Bachelor's Thesis

Concept and development of a human robot interface using parametrised Augmented Reality

Konzept und Entwicklung einer Mensch-Roboter Schnittstelle mit parametrisierbarer Augmented Reality (GERMAN)

Author:	Eric Vollenweider
Supervisor:	Prof. Dr.-Ing. Birgit Vogel-Heuser
Advisor:	Dr. Pantförder Dorothea
Issue date:	Februar 26, 2019
Submission date:	August 26, 2019

Table of Contents

1 Introduction

1.1 Reasons for- and problems with Human-Robot collaboration

Today's world is highly automated in each and every aspect of our lives. One of the most important advancements within the last 30 years where robots. They allow us to minimize costs, while maximizing output, quality and efficiency. With the rising need of agile production plants, classic robotic systems do not seem to fulfil all requirements anymore. Industrial robots do have numerous abilities but adapting to new situations easily is certainly not one of them. In contrast, the human's strength always was exactly that - to learn quickly. Thus, Human-Robot (HR) collaboration is an inevitable step towards the future of humanity.

Robotics needed to adapt to the new situation where humans and robots share a perimeter, thus compliant robots where developed. For machines to feel natural it is not enough to be compliant and not hurt humans. As Gary Klein et al. stated, communicating intent is a key issue in effective collaboration within teams [1].

Augmented Reality (AR) can help us to communicate with robots in three dimensions. With rapid advancements in the field of robotics during the last few years [2], innovative ways to program, develop and operate these highly complex systems need to be researched. Both robotics and AR have been heavily worked on for the last 20 years separately, but their combination has mostly remained untouched. Lately though, it has become a new focus area in research. While it is known that AR can improve communication in certain use cases [3], actually developing them is not as easy, since there is a jungle of technologies and frameworks. Since most development environments require rely on source code to be written, the reusability is quite low, which may result in re-doing the same steps over and over again, requiring software engineers along the way to do everything. Thus, the time-to-market is long and expenses are high.

According to the U.S. Department of Labour, there is a massive hunt for talent in the software industry [4]. The department reported an expected growth of over 30% within the next 10 years. One possible reason could be, that acquiring the necessary skills to develop software applications takes a long time and is not viewed as an easy path. Specifically AR applications currently require a wide variety of skills. On top of that, many processes and steps have to be repeated for each new AR-application and it is increasingly hard to build upon other peoples work.

The lack of programming talent is also seen in the robotics industry, which requires its developers to posses an even wider set of skills, including software development, mechanical and the basics of electrical engineering. Due to the lack of developers the speed of innovation and time-to-market may either be slowed down or the companies will have to pay about 20% above market value for their employees [5].

1.2 Parametrised development as an approach to solve AR-developer shortage

Today, many industrial robots are only parametrised and not actually programmed, which results in lower requirements for employees and thus in less time and money spent on training. Employees only set certain locations, points and actions utilizing an easy to understand User-Interface. Applying a similar methodology to the development of Augmented Reality applications could lead to similar effects. If non-programmers could setup, configure and thus develop Human-Robot interfaces involving AR and robot controlling all by themselves, without the need to write a single line of code, the cost efficiency and speed of innovation can be improved.

This bachelor thesis aims to research the feasibility and practicability of such an abstract development environment. First there will be a ”State of the Art” section, where current research and progress is displayed. From there some gaps will be identified, a concept proposed, implemented and tested in a real application.

2 State of the Art

The State of the Art chapter will highlight technologies and products that highly influenced this thesis. First, Augmented Reality itself will be explained and some basics will be laid out, which is followed by a paragraph about current AR-Standardisations, which is important since they help developers to speed up their processes because many decisions do not have to be made repeatedly. Later the combined field of AR and Robotics will be described in detail. How it can use Augmented Reality technology, which projects succeeded in doing so and what benefits it brought. A part about parametrised thinking and other disciplines that already work as such represents the last component of this thesis. A final remark describes how this thesis tries to combine all explained technologies or principles into an innovative way of programming AR applications.

2.1 Augmented Reality and its standardisations

This bachelor's thesis evolves around AR and its use cases. It is utterly important to clearly define some terms before using them thoroughly. AR technology must not be confused with Virtual Reality (VR). Whereas VR completely immerses users, the former still allows the real world to be seen together with superimposed 3D objects projected into the user's view. T. Azuma crafted a very general definition of AR-Systems very early on [6]. He wrote that, in order for a system to be classified as an AR system, it has to fulfil three characteristics. Namely, the system has to combine real and virtual objects, be interactive in real time and spatially map physical and virtual objects to each other. The term Augmented Reality is often used as a synonym for the also commonly used expression "Mixed Reality".

After having cited a definition about AR and answering what it is, this paragraph describes what underlying principles exist. Two types of AR technologies are in use today. Firstly, there are see through displays with strong user immersion and secondly monitor based approaches like smartphones with cameras. A large variety of producers [7] release new AR-Devices regularly. In this bachelor's thesis the term AR should be understood as head-mounted

superimposing Augmented Reality devices with see through displays, such as the Microsoft HoloLens [8], that actually create the illusion of three dimensional holograms being projected into the real world.

Standards often contribute to the development and usage of technologies substantially, firstly by unifying used tech-stacks, thus potentially reducing complexity and secondly by helping new members to navigate in a complex environment. There are not many international standards on AR since the field's popularity still is very young. Nevertheless, C. Perey et al. [9] examined existing standards and some best-practice methods for AR. They clearly identified some gaps in the AR value-chain. The researchers lay out numerous standards for low level implementations such as geo-graphic location tracking, image tracking, network data transmission etc.

Figueroa et. al [10] researched about a "Conceptual Model and Specification Language for Mixed Reality Interface Components". They want to lay a foundation for other developers to standardize 3D interface assets, for others to build upon. The team developed an xml based data structure called "3DIC" that defines the look and some smaller behaviour actions of UI control elements. Although their work goes into the right direction, it lacks certain features that robot-human AR interfaces need, and contains unwanted components like pseudo-code. "3DIC" is a specification language that cannot be fully processed automatically, since pseudo-code is not executable.

Concluding from this general overview, there does not seem to be an existing standard for HR Interfaces utilizing parametrised Augmented Reality, that allow the development of complete HR-AR-applications that can actually be processed, executed and used.

2.2 AR in Robotic applications

2.2.1 AR during development and testing

Wolfgang Hönig et al. examined three different use cases of Augmented Reality [11]. The research team primarily focused on the benefits offered by the co-existence of virtual and real objects during the development and testing phase. They documented three individual projects where the implementation of AR as a main feature resulted in lowered safety risks, simplified debugging and the possibility of easily modifying the actual, physical setup.

Another important dimension of development and testing is of financial nature: upfront investments and operating costs. For example, robotic aerial swarms tend to be quite expensive due to the high cost of drones. Hönig et al. successfully scaled up the number of objects in their swarms without adding physical hardware, thus, saving money and space [11]. However, it is stated that this approach might not be applicable to all experiments since simulations are never perfect replicas of actual systems. This small delta in physical behaviour might be enough to raise doubts regarding the correctness of the experiment results.

All projects by Hönig et al. focus on isolating certain aspects of the system to analyse and test them more flexibly, less expensively or with improved safety for all participants involved (humans and machines). Similarly, Wünsche et al. have created a software framework for simulating certain parts of robotic systems [12]. These researchers from Auckland worked on methods to combine real world and simulated sensor data and navigate a real-world robot in the combined environment.

2.2.2 Operating robotic systems with AR support

A well-known bottleneck in robotics is the controlling of and thus, the communication with robots [13]. In all previously cited cases, Augmented Reality was not used to enhance the interaction between humans and robots but to mitigate the current challenges in developing robotic systems. Very early work by Milgram et al. [14] shows that even the most basic implementations of AR technology, with the objective to improve the information exchange, enhance the bidirectional communication in multiple ways. The team proposed means to relieve human operators by releasing them from the direct control loop and using virtually placed objects as controlling input parameters. This replaces direct control with a more general command process.

As Gary Klein et al. stated, communicating intent is a key issue in effective collaboration within teams [1]. Whenever robots and humans collaborate in a close manner, it is critically important to know each other's plans or strategies in order to align and coordinate joint actions. For machines lacking anthropomorphic and zoomorphic features, such as aerial and industrial robots, it is unclear how to communicate the before-mentioned information in natural ways.

In order to solve this problem, Walker et al. [15] explored numerous methods to utilize Augmented Reality to improve both efficiency and acceptance of robot-human collaborations via conveying the robot's intent. The group of researchers defined four methods of doing so,

with varying importance being put on "information conveyed, information precision, generalizability and possibility for distraction" [15]. The conclusion was that spatial Augmented Reality holograms are received much more intuitively than simple 2D projected interfaces.

2.3 Parametrised Development

Numerous disciplines utilize parametrised development environments heavily. For example the CAD software *CADENCE* (integrated electrical circuits) offers parametrised cells to optimize the development process [16]. Users can place these cells and adjust certain parameters. For example spiral inductors can be configured via a simple UI that sets the parameters in the background. Aspects like outer dimensions, metal width, number of layers etc. can be set. The software then calculates its properties and behaviour at runtime. Generally there is no coding skill required.

In architecture Parametric Design is taking overhand since the late 2000s. New capabilities in computer rendering and modelling opened up a whole new world for architects at the time. Using parametrised mathematical formulas with boundary constrains allows architects to generate building structures easier and more efficient [17]. Adapting to a changing environment during the design process is much easier, since a substantial portion of work can be completed by algorithms and software applications. Additionally the reusability of components increases massively due to the very formal way of representation. Architecture studios also begin to include Virtual and Augmented Reality technologies actively in their design processes to further incorporate and better understand these new design styles called *Parametric Design* [18]–[20].

2.4 Current methods programming industrial robots

To understand where and how the content of this bachelor's thesis fits into the world of robotics, this chapter will describe the current workflows with robots, how they are configured and programmed. It is important to know that almost all industrial robots are programmed in three distinct ways.

1.) Teaching Pendant

2.) Simulation / Offline Programming

3.) Teaching by Demonstration

According to the British Automation & Robot Association over 90% of all industrial robots are programmed using the Teach Pendant (TP) method [21]. These devices basically are touch-tablets fitted for industrial use with emergency stop buttons, more durable materials and so forth. They allow the operator to jog (term for steering the robot manually using the TP) the robot into certain configurations, save the positions and offer a number of other possibilities. For easier tasks some manufacturers (e.g. Fanuc) offer specific User Interfaces where the operator simply enters parameters and positions. More complex goals can also be achieved with the Teach Pendant by programming in a textual manner, using each manufacturer's own language. This type of programming needs a substantial amount of training and can generally not be done by untrained people.

Teach Pendants are great for trivial and simple tasks, that do not require lots of collaboration between factory components. Reprogramming the robot using this method leads to a partial downtime, since the actual real robot has to be used. Important to note is, that parametrised programming already is an industry standard in industrial robotics for some brands of robots.

The other two methods are used for more complex tasks and production lines and each have their Pros and Cons. While Offline Programming is very precise and powerful, it does require a substantial amount of schooling time for the operator. Teaching by Demonstration is the exact opposite. Most people could succeed quickly, but the complexity of achievable tasks and the precision is limited.

2.5 Parametrised AR in Robotic applications

Many industries adopted Augmented Reality in their daily process already. It's advantages and challenges are clearly known and being worked on. There are numerous examples of AR helping in different environments and tasks [22]–[24]. Developing AR applications is generally speaking still a very costly task. On the other hand there is the design principle of "Parametrisation" that may involve increased initial investment but can lower the cost of changes and adaptions to changing environments. Combining the perks of both AR technology and parametrised architecture may bear great potential.

3 Concept

This chapter will explain what requirements were defined for the to be built system and how it will be designed and setup. For further reading the system will be called ”Parametrised Augmented Reality Robot Human Interface” (*PARRHI*).

3.1 Goal, Requirements and Use Cases

As described in section 1.2 this thesis presents a possible approach to solve the previously described problems (see section 1.1). It is the main target of this bachelor's thesis to remove the necessity of high software engineering skills to develop reasonably complex Augmented Reality Robot Human Interface applications, maintaining the quality of the outcome and even increasing the degree of resuability. This might result in lower development costs, lower struggle to gather software engineering talent and even in a shorter time to market.

To succeed in this goal, a specific set of requirements has to be defined and documented in a formal way. To gather these requirements the V-Model developed by the Federal Republic of Germany was used [25].

3.1.1 User Requirements

Before defining the User Requirements the system's end user has to be defined. The characteristics of the actual enduser of *PARRHI* might be someone who:

- Knows the basics of text editing software,
- has no qualifications in software engineering,
- wants to develop an Augmented Reality application for professionals that collaborate with industrial robots in a shared perimeter.

Having an idea of the end-user, the user-requirements can be defined. The user wants to:

- Develop a AR-applications without software engineering skills
- Have the tools necessary to create medium complex applications for use cases such as tutorials, maintenance instructions or other teaching purposes
- Build upon other people's work or projects
- Launch the AR-application on a suitable device
- Possibly use the same system on different types and brands of robots

3.1.2 System Requirements

Deriving from the previous chapter the System should:

1.) have user input in a simple and intuitive format to allow for non-software engineers.

2.) have readable and intuitive feedback on every user input.

3.) achieve reusablity by having an input in non-binary text format that allows copy and paste reproduction.

4.) support building for hand held mobile devices and head mounted Augmented Reality glasses.

5.) allow bidirectional communication with the real and virtual world in different formats

6.) allow documentation of the application's workflow

7.) be as platform independent as possible

3.1.3 Possible Use-Case: Factory Maintenance

This section depicts a possible use case example for the PARRHI system, especially how and why it could be used. One could imagine a huge auto mobile factory with hundreds of industrial robots assembling certain parts of a car. A factory like this, might want to repeatedly check on their robotic infrastructure to recognise problems and thus avoid any future downtime. The difficulties here are, that one generally does not want a robot to not be in operation due to the high costs of it but on the other hand non-collaborative robots are dangerous and can inflict damage on humans and other property, which means that the robot should be switched off or at least in a safety mode while humans are around. Thus, the

operator should be quick and precise in their task without letting down their security safe guard.

Despite being expensive, finding employees that have the necessary skills is not an easy task. Hence it would be interesting to teach new personnel how to succeed in these very specific cases. An Augmented Reality application might be the right approach for cases like these. For every specific maintenance task an employee knowledgable enough could develop AR applications, which lead a small army of less-skilled workers through their tasks step by step without ever compromising their safety.

Developing such a high number of AR-applications is not really plausible with current frameworks. Despite the knowledgable employee probably not having enough programming skill, it might simply be too expensive, slow and too hard to adapt to new changes. A system like PARRHI would allow the skilled employee to write these applications themself without the need of software engineers. Since many tasks might be similar to one another, the employee could re-use most of the applications program over and over again - further increasing development speed. PARRHI applications will be able to involve the Real World's data in their workflow, which allows the application to know where the robot currently is located and command the maintenance worker to move away. Having the possibility to also influence the Real World, the application could hinder the robot from moving, while the employee is in a certain danger area. All that, while showing instruction steps in Augmented Reality to easily understand and complete the task quickly.

In essence, using systems like PARRHI might reduce the development costs of AR-applications substantially, which could open a wide set of use cases, that were untouched before due to the high costs. This might allow for a much better economically usage and a much wider set of real world applications for Augmented Reality apps.

3.2 PARRHI Concept

The following chapter will now explain the concept of the PARRHI system and show one possible approach to reach the previously mentioned requirements.

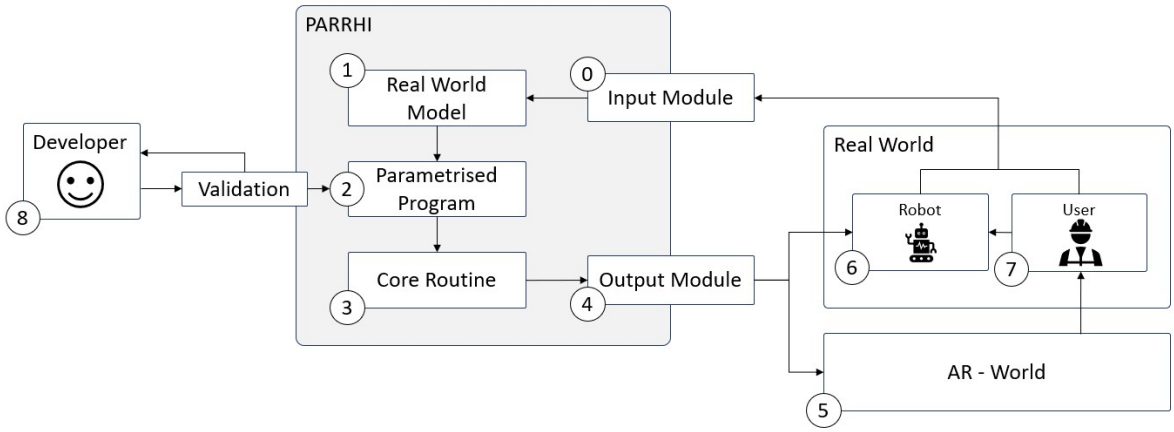

Figure 3.1: PARRHI general concept

Fig. 3.1 depicts PARRHI's concept, which consists of three main parts. The left hand side shows the developer and his workflow. The grey box in the middle is the PARRHI system itself. Lastly, the right side visualises the environment that surrounds the PARRHI system in both virtual and real space. I will now go into detail and systematically explain each component from (0) to (8), before describing the information flow between them.

0.) Input Module: The Input Module (0) is responsible for collecting data needed for the PARRHI system from the real world. In this specific case, it receives the robot's (6) joint angles and the user's (7) position.

1.) Real World Model: In order to utilize data from the Input Module (0) it has to be processed. The Real World Model (1) has a deeper understanding about the real world and helps to extract useful information from the data inflow. In this specific case, its outputs are the robot's (6) joint positions and the user's (7) location represented as parameters that are available for the Parametrised Program (2).

2.) Parametrised Program: This is a document that defines the AR application's behaviour and workflow. Its syntax is parametrised, meaning, that it makes use of placeholders (parameters) whose actual value is managed by PARRHI itself at runtime. There are two types of parameters. First, there are the ones that are provided by the Real World Model (1) (for example the discussed robot's (6) joint positions and the user's (7) location) and secondly any object defined in the Parametrised Program itself (2)

can act as a parameter for other objects in the program. These objects can be 3D AR-Hologram definitions, logical instructions or a variety of events and actions. For a more thorough explanation of the Parametrised Program see section 3.2.2.

3.) Core Routine: The Core Routine (3) is an interpreter for the Parametrised Program (2), and acts on its instructions. It generates the output of the application which can either be commands to the robot (6) and thus real world, or instructions for the AR-World (5).

4.) Output Module: This component manages the outgoing communication with PARRHI's environment.

5.) AR-World: The Augmented Reality World is the space, where a part of PARRHI's output is displayed. It can display holograms like spheres and cylinders, but also written text for instructions. The AR-World (5) augments the Real World and is thereby seen by the User (7) through an AR Device.

6.) Robot: It is a Real World object that can be controlled by the User (7) themselves, or by the PARRHI system.

7.) User: This is the person that would use the finished application to fulfil their task, which could for example be to do some work following instructions, or to simply learn different aspects about the robot/working environment.

8.) Developer: This person develops the application's workflow and behaviour. Their output is the Parametrised Program (2) as a text-based document, which is then fed into the PARRHI system after being validated.

It should now be clearer what each component is responsible for. After having laid down the fundamentals, the following paragraph will track the information flow between the components. The order will be a bit different here, since it will be built up in a chronological way. Hence, I will first start with the Developer (8) crafting the Parametrised Document and then continue with the standard runtime cycle of the PARRHI system.

After having defined what the AR-HR-Interface application should achieve, the Developer (8) crafts the Parametrised Program (2). The text-based hierarchically built document will be validated first, before it is granted access into the PARRHI system. If the validation fails, the Developer (8) receives error messages accordingly an can reiterate over his document until it validates successfully.

The runtime loop starts with the Input Module (0). As described, it not only collects the robot's (6) joint angles, location and gripper-state but also the User's (7) position. This input data is then fed into the Real World Model (1), which uses its Real World understanding

to extract/calculate useful information. In this specific case, it applies forward kinematics onto the robot's (6) joint angles to calculate their 3D position and transforms the User's (7) location coordinates into PARRHI's internal coordinate system. After transforming the input data into a usable format, the Parametrised Program (2) comes into play. The Developer (8) wrote this document in a parametrised way using placeholders. The latter are now filled with information by the PARRHI system. For example, the Parametrised Program (2) could use the robot's tool centre point (TCP) for a definition of some AR-holograms, without knowing where exactly the TCP is during development. At runtime, PARRHI inserts this information into the parameters.

The Core Routine (3) then interprets the now-filled-in Program (2). It first updates its internal state with the new parameters, before updating all AR-Holograms. At last it evaluates all events, and triggers their actions if needed. These actions might be commands for the AR-World (5) (for example to show or hide holograms, or to change UI text) or commands for the Real World Robot (6) (e.g. to move or stop the robot). These commands are passed into the Output Module (4), which is responsible for actually executing them. It has the required tools to communicate with the robot and with the AR - World (5).

At this point, the User (7) sees two things. On one hand they see the Real World in front of them. There is the Robot (6) and a surrounding environment. The Robot (6) might be instructed to do something by the Output Module (4) already. Superimposed onto the Real World they see the Augmented Reality World (5), which contains all visual information, that the Output Module (4) constructed. There could be cylindrical translucent holograms marking a danger zone around the robot's (6) axes. Now, the AR-World's (5) content influences the User (7) to do certain things. He could be instructed by holograms to move into a safe-zone. At the same time, the User (7) might control the Robot (6). At this point it is to mention, that the Output Module (4) enjoys priority over the User's (7) input when commanding the Robot (6).

As Fig. 3.1 depicts, the Input Module (0) then finally closes the feedback loop by receiving fresh data from the Real World. The information flow that was followed in the paragraphs above changed the environment either by direct commands or indirectly by commanding the User (7) via the AR World (5). These changes will now be reflected in the input data and are thus available for the next cycle. The presented loop repeats itself, where every iteration only takes a fraction of a second, until the PARRHI system is terminated for some reason.

The preceding paragraphs should give a relative detailed overview about the concept behind PARRHI. The following chapters will now explain each part in an even higher degree of detail, with the goal being, that almost no questions about the concept stay unanswered. The order

will be similar to the explanation of the information flow in the PARRHI concept, meaning, that the Input Module (0) and the Real World Model (1) will be explained first, followed by a very thorough explanation of the Parametrised Program (2) with all its objects and definitions. In the end the Core Routine (3) and the Output Module (4) with the Real and AR-World (5) will finish off the concept chapter.

3.2.1 Preparation of Parameters

One main strength of the *PARRHI* concept is its disclosure of its knowledge of the outside world to its inner components via parameters. Two steps are necessary to do exactly that. First, an Input Module (0) has to automatically retrieve real world information. Since the latter quite hard to comprehend for computer programs, some model knowledge (1) should be implemented to process the input data by reducing complexity. Generally, models are a more abstract version of the object it is supposed to describe. Next to being easier to understand, one benefit is, that one can often explain models in a mathematical way, which is of course perfect for computer programs.

The complexity of collecting Real World data strongly depends on the use case. I decided to limit my scope to the collection of three things, since it is not the main focus of this thesis. Firstly, gathering the robot's location is a key aspect. For Augmented Reality applications, it is utterly important for the AR device to know its six dimensional orientation (position and rotation). Otherwise superimposed holograms would not make any sense to the viewer, since they appear misplaced. The centre of origin for *PARRHI* applications should thereby always be the robot's base.

Then of course the robot's joint positions are needed for the application program to offer the possibility to fully integrate the robot in the application's workflow. Since most robots do not offer their individual joint positions in 3D vectors but only their joint angles, a corresponding robot model is needed to calculate each joint's position from their joint angles. This is called forward kinematics, an old problem in robotics with known mathematical tools and ways to solve it. In principle, one can calculate a robot's tool point via every joint angle and some knowledge about the robot's configuration. The latter is specified by the types of joints (degrees of freedom (DOF)) and the distance between consecutive joints. A mathematical model then outputs each joint's three dimensional location, which is offered to the Application Program (2) via parameters. To get an example of how the joint's position might be used in a parametrised way see section 3.2.2.2.

3.2.2 Parametrised Program

The Parametrised Program is the document, which the Developer ((8) in Fig. 3.1) of the *PARRHI* system crafts. It contains parametrised, hierarchically structured data, that defines the behaviour, feel and look of the final AR-HR-application. All instructions and definitions can make use of the previously explained parameters offered and maintained by *PARRHI*. Since this is the actual document that a developer writes, its syntax and structure has to be as simple as possible to fulfil the requirements from section 3.1.2 without limiting the developer's creativity. To achieve this, a rather natural way of describing the wanted behaviour was chosen.

The following chapters explain the set of tools that are available to the developer, how they work and interconnect. Table 3.1 displays the top level structure and its main parts, where Variables, Points and Holograms define assets that Events can work with. Each section will describe what exactly is parametrised in their definition and how they can be used as parameters to define other program parts.

Table 3.1: *Input Data* structure

Name	Section	Explanation
Variables	3.2.2.1	Integer variables in a traditional sense
Points	3.2.2.2	Different kinds of 3D Point definitions (fix, relative to the robot, relative to the user)
Holograms	3.2.2.3	Holograms can be mounted onto points and have a set of properties
Events	3.2.2.4	Events have certain triggers and carry Actions as a payload

3.2.2.1 Variables

Variables are storage locations for numbers and have a symbolic name. They can be used to create different steps in one's application. When using Variables as parameters, they can be the source of an event's trigger or the target of an event's action (see section 3.2.2.4) and thus take part in the application's logic. With the help of Variables an application could keep track of something by counting events, or also implement state machines that jump between modes. Variables are internal knowledge, meaning, that they are managed by the *PARRHI* system and are not imported/external information.

3.2.2.2 Points

Points are probably the best example of parametrised information in the *PARRHI* system. At runtime the data from the Real World Model (see (1) in fig. 3.1) is directly fed into the definition of all points that use the according parameters. Thus, the system updates these objects repeatedly with Real World information, that was fed through the Real World Model. Although points are defined using parameters in some way, points themselves are parameters to other objects in the Parametrised Program.

They essentially are a three dimensional vectors (X, Y, Z), with their coordinates being parameters. Depending on the type of Point, the parameters are set differently. There are three types of point definitions, with the difference between them being, what values are inserted into the X, Y and Z parameters.

1.) Fix-Point

2.) Robot-Point

3.) Camera-Point

The **Fix-Point** has static coordinates and thus fixed values are used to define their coordinate parameters (see figure 3.2). It could be used to setup holograms that visualize certain spacial environmental constraints or for different steps in a *PARRHI* AR-HR-Interface application.

Robot-Points are defined by two indexes of the robot's joints and one scalar value that linearly interpolates between them (see figure 3.3). The application's developer does not have to understand the robot's kinematics and simply uses the joint-indexes as parameters. At runtime, the *PARRHI* system retrieves the robot's joint position, uses the Real World Model to calculate each joint's position and then feeds this data into the Robot-Points. The final point's position is calculated as follows (with s being the scalar value and J_n the position vector of Joint n which is an output of the Real World Model):

$$\boldsymbol{P} = \boldsymbol{J_1} + (\boldsymbol{J_2} - \boldsymbol{J_1}) * s \tag{3.1}$$

Camera-Points are a way to involve the user's position in the application. Similarly to the Robot-Points, the *PARRHI* system retrieves the cameras coordinates via the Input Module, feeds it through the Real World Model to map it onto the internal coordinate systems and finally periodically updates the Camera-Points with the new location data of the head mounted AR-Device.

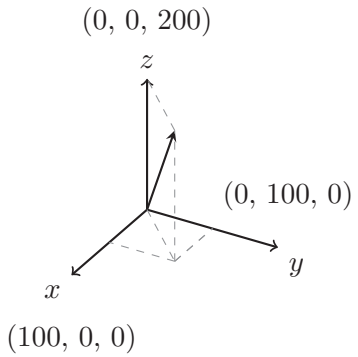

Figure 3.2: Fix-Point example

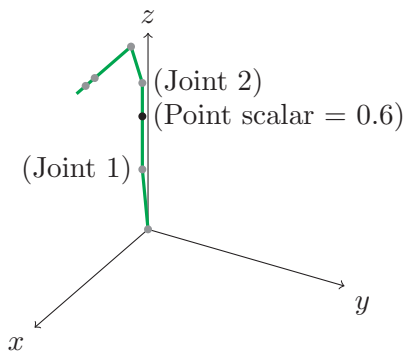

Figure 3.3: Robot-Point example

3.2.2.3 Holograms

One of the main output channels of *PARRHI* is the Augmented Reality World. The AR World can be filled with numerous objects (generally called holograms). In the Parametrised Program these Holograms are defined using parameters. In this case, Points are used to define the Hologram's location and orientation. Since Points are parametrised themselves, and are thus updated by the data from the Real World model, holograms are indirectly also strongly dependant of that feedback loop.

Holograms in the PARRHI system have a number of attributes. Holograms can be active (respectively also inactive), have a render-mode, a geometric definition and of course a location. The location is defined by a certain number of points - depending on the hologram's geometry. Currently the *PARRHI* supports two types of holograms. There are spheres and cylinders, respectively taking one or two points and a radius as input parameters to define its size and position.

A Sphere's centre is always set to the point it was defined with, whereas a cylinder always connects the two points of its definition. All described types of points in section 3.2.2.2 can be used as input parameters for the location. With a given radius and point the three dimensional figure is well defined.

Holograms have an attribute called *renderMode*, which if set to "transparent", renders the hologram in a half transparent way, allowing holograms to be used for boundary or zone visualizations. Furthermore the visibility of holograms can be changed by setting the *active* parameter, which is done by actions as described in section 3.2.2.4.

3.2.2.4 Events

All previous elements (variables, points and holograms) exist to define the scene and to set up assets that can be utilized by events, which now actually describe the application's workflow. To do so, there are two subtypes in this category. There are event-triggers and event-actions (or short *triggers* and *actions*). Triggers have a boolean expression, which is checked periodically. As soon as the boolean expression evaluates to *true*, the attached actions will be executed and the trigger will be disabled, avoiding multiple executions. One could say that if triggers the *PARRHI's* sensors, actions are its tools to act on the augmentation.

Triggers can be enabled and disabled, either from the beginning on, or toggled by an action. Every trigger has at least one action as a payload. To reach a reasonably capability numerous different but easy to understand triggers are available for the application's developer. In table 3.2 is a complete list of all defined triggers. *(Note, that the enabled/disabled flag and actions are omitted in this table)*

Distance trigger can be used for two main purposes. First the application can use the robot's movement as an input using a *Robot-Point*. An action could be triggered as soon as the user jogged the robot's TCP into wished position by using a *Fix-Point* an waiting for them to come close to each other or to monitor the robot's joint position in Joint-Space by using two *Robot-Points* as input parameters. Second the user's movement can be monitored by utilizing a *Camera-Point* as an input. The application can thus ask the user to move to a specific location.

With **Variable triggers** the application is able to use defined variables as triggers. One could implement a counter for certain events, and trigger an action when a threshold value is reached. It can also be used for workflows that need states or steps. Finally the **Time trigger** allows the application to involve timers. The user could be given a maximum time for a task or holograms can be hidden after a few seconds.

Table 3.2: Event Triggers

Name	Input Parameter	Trigger expression
Distance trigger	Two Points $\boldsymbol{P_1}, \boldsymbol{P_2}$, distance d	$\lvert\boldsymbol{P_2} - \boldsymbol{P_1}\rvert \leq d$
Variable trigger	Variable v, trigger value v_{tr}	$v = v_{\text{tr}}$
Time trigger	trigger time t_{tr}, time since enabling t_{enabled}	$t_{\text{tr}} \geq t_{\text{enabled}}$

Whenever a trigger's boolean expression evaluates to true, its actions are invoked and the trigger gets disabled. Since actions are the only way *PARRHI* can influence the augmentation, there are numerous different types of actions - each serving a general purpose to fulfil the defined requirements (see section 3.1.2).

As with triggers, actions have a set of input parameters they need to fulfil their task. As with all other input data objects discussed so far, actions have a unique ID. The table 3.3 gives a quick overview about all actions that *PARRHI* currently supports.

Increment-counter actions increment their integer variable by 1. If a developer wanted to count the number of times a user jogged the robot into a specific region, an Increment-counter action could be used as a payload of a Distance trigger. After a threshold value is reached, a Variable trigger could change the UI text and display a hint.

The **Set-Hologram-State action** enables hiding and showing holograms at runtime. If a hologram represents a region for a tutorial step, it can be hidden after the user's task is completed. The new scene can then be setup by displaying new holograms that guide the user's way. Another possible use would be, to display a warning boundary, if the user moves into a forbidden zone. This can be achieved by combining Distance trigger and Set-Hologram-State actions.

When using *PARRHI* the user is presented a GUI that shows text and some other few options. The **Change-UI-Text action** allows to change this displayed text. There are numerous obvious use-cases where this is useful. Whenever it is of value to inform the user about something that cannot be achieved by holograms, this is a simple way to do so.

Table 3.3: Event Actions

Action Name	Input Parameter	Explanation
Increment Counter	Variable v	Increments the value of v by 1
Set Hologram State	Hologram-IDs, State to set	Enables/disables all specified holograms
Set Trigger State	Trigger-IDs, State to set	Enables/disables all specified triggers
Change UI Text	Text to set	Sets the UI Text
Move Robot	Point P	Moves the robot to P
Set robot-hand State	State to set (open/close)	Opens or closes the robot's gripper

To create meaningful and longer applications, enabling and disabling triggers is an essential tool. This is what the **Set-Trigger-State action** is for. Triggers can only invoke their payload actions, if they are active. Triggers can either be defined as disabled from the beginning on, or get disabled by triggering as described above. The Set-Trigger-State action has the ability to (re)activate disabled triggers. There is a speciality in the case of *Time triggers*. Their inner timer starts ticking, whenever they get enabled. This allows for timers to be used in the middle of applications, relative to other events.

3.2.3 Core Routine

3.2.4 I/O Modules

4 Bibliography

[1] G. Klein, P. J. Feltovich, J. M. Bradshaw, and D. D. Woods, "Common ground and coordination in joint activity", *Organizational simulation*, vol. 53, pp. 139–184, 2005.

[2] C. Laschi, B. Mazzolai, and M. Cianchetti, "Soft robotics: Technologies and systems pushing the boundaries of robot abilities", *Sci. Robot.*, vol. 1, no. 1, eaah3690, 2016.

[3] M. Billinghurst, "Augmented reality in education", *New horizons for learning*, vol. 12, no. 5, pp. 1–5, 2002.

[4] U. D. o. L. Bureau of Labor Statistics. (2018). Occupational outlook handbook, software developers, [Online]. Available: `https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm` (visited on 03/28/2019).

[5] H. J. Baker. (2017). 2018's software engineering talent shortage — it's quality, not just quantity, [Online]. Available: `https://hackernoon.com/2018s-software-engineering-talent-shortage-its-quality-not-just-quantity-6bdfa366b899` (visited on 04/25/2019).

[6] R. T. Azuma, "A survey of augmented reality", *Presence: Teleoperators & Virtual Environments*, vol. 6, no. 4, pp. 355–385, 1997.

[7] A. Minds. (2019). Endgeräte für ar mr, [Online]. Available: `https://www.augmented-minds.com/de/erweiterte-realitaet/augmented-reality-hardware-endgeraete/` (visited on 04/25/2019).

[8] M. Inc. (2019). Microsoft hololens, [Online]. Available: `https://www.microsoft.com/de-de/hololens` (visited on 04/25/2019).

[9] C. Perey, T. Engelke, and C. Reed, "Current status of standards for augmented reality", in *Recent Trends of Mobile Collaborative Augmented Reality Systems*, Springer, 2011, pp. 21–38.

[10] P. Figueroa, R. Dachselt, and I. Lindt, "A conceptual model and specification language for mixed reality interface components", in *VR 2006, In Proc. of the Workshop "Specification of Mixed Reality User Interfaces: Approaches, Languages, Standardization*, 2006, pp. 4–11.

[11] W. Hoenig, C. Milanes, L. Scaria, T. Phan, M. Bolas, and N. Ayanian, "Mixed reality for robotics", in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 5382–5387.

[12] I. Y.-H. Chen, B. MacDonald, and B. Wunsche, "Mixed reality simulation for mobile robots", in *2009 IEEE International Conference on Robotics and Automation*, IEEE, 2009, pp. 232–237.

[13] R. S. Magazine. (2018). The grand challenges of science robotics, [Online]. Available: `https://robotics.sciencemag.org/content/3/14/eaar7650` (visited on 04/25/2019).

[14] P. Milgram, S. Zhai, D. Drascic, and J. Grodski, "Applications of augmented reality for human-robot communication", in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, IEEE, vol. 3, 1993, pp. 1467–1472.

[15] M. Walker, H. Hedayati, J. Lee, and D. Szafir, "Communicating robot motion intent with augmented reality", in *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, ACM, 2018, pp. 316–324.

[16] V. Grozdanov, D. Pukneva, and M Hristov, "Development of parameterized cell of spiral inductor using skill language", Apr. 2019.

[17] M. Stavric and O. Marina, "Parametric modeling for advanced architecture", *International journal of applied mathematics and informatics*, vol. 5, no. 1, pp. 9–16, 2011.

[18] H. Seichter and M. A. Schnabel, "Digital and tangible sensation: An augmented reality urban design studio", 2005.

[19] F. D. Salim, H. Mulder, and J. Burry, "A system for form fostering: Parametric modeling of responsive forms in mixed reality", 2010.

[20] X. Wang, P. E. Love, M. J. Kim, C.-S. Park, C.-P. Sing, and L. Hou, "A conceptual framework for integrating building information modeling with augmented reality", *Automation in Construction*, vol. 34, pp. 37–44, 2013.

[21] B. A. R. Associtation. (2018). Robot programming methods, [Online]. Available: `http://www.bara.org.uk/robots/robot-programming-methods.html` (visited on 04/07/2019).

[22] P. Diegmann, M. Schmidt-Kraepelin, S. Eynden, and D. Basten, "Benefits of augmented reality in educational environments-a systematic literature review", *Benefits*, vol. 3, no. 6, pp. 1542–1556, 2015.

[23] P. Salamin, D. Thalmann, and F. Vexo, "The benefits of third-person perspective in virtual and augmented reality?", in *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM, 2006, pp. 27–30.

[24] S. Henderson and S. Feiner, "Exploring the benefits of augmented reality documentation for maintenance and repair", *IEEE transactions on visualization and computer graphics*, vol. 17, no. 10, pp. 1355–1368, 2011.

[25] M. Broy and A. Rausch, "Das neue v-modell® xt", *Informatik-Spektrum*, vol. 28, no. 3, pp. 220–229, 2005.

Declaration

The submitted thesis was supervised by Prof. Dr.-Ing. Birgit Vogel-Heuser.

Affirmation
Hereby, I affirm that I am the sole author of this thesis. To the best of my knowledge, I affirm that this thesis does not infringe upon anyone's copyright nor violate any proprietary rights. I affirm that any ideas, techniques, quotations, or any other material, are in accordance with standard referencing practices.

Moreover, I affirm that, so far, the thesis is not forwarded to a third party nor is it published. I obeyed all study regulations of the Technical University of Munich.

Remarks about the internet
Throughout the work, the internet was used for research and verification. Many of the keywords provided herein, references and other information can be verified on the internet. However, no sources are given, because all statements made in this work are fully covered by the cited literature sources.

Munich, December 1, 2016

YOUR NAME