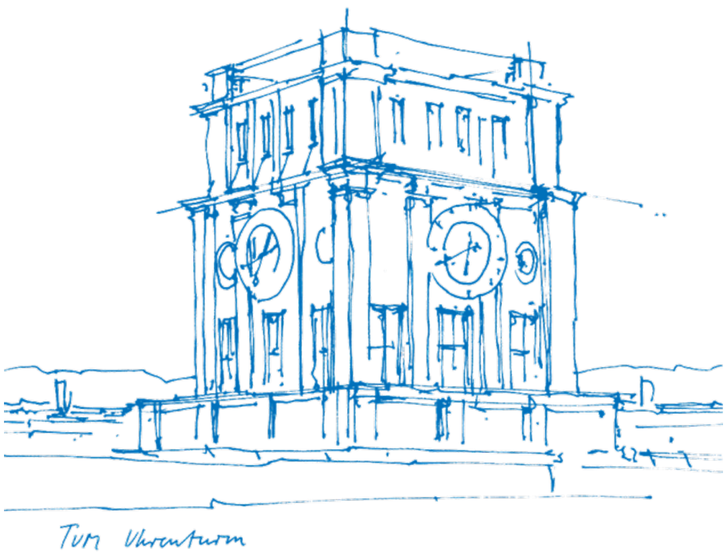
Sensor Based Robotic Systems and Intelligent Assistance Systems
School of Computation, Information and Technology
Technical University of Munich

Cognitive Load Reduction in Commanding Heterogeneous Robotic Teams

Creating Executable Command Sequences and Command Prioritization based on Few-Shot Historical Data Learning

Nicole Grabner



Cognitive Load Reduction in Commanding Heterogeneous Robotic Teams

Creating Executable Command Sequences and Command Prioritization based on Few-Shot Historical Data Learning

Nicole Grabner

Sensor Based Robotic Systems and Intelligent Assistance Systems
School of Computation, Information and Technology
Technical University of Munich

Cognitive Load Reduction in Commanding Heterogeneous Robotic Teams

Creating Executable Command Sequences and Command Prioritization based on Few-Shot Historical Data Learning

Nicole Grabner

Thesis for the attainment of the academic degree

Master of Science (M.Sc.)

at the School of Computation, Information and Technology of the Technical University of Munich.

Examiner:
Prof. Alin Albu-Schäffer

Supervisor:
Peter Schmaus, Dr. Neal Lii

Submitted:
Munich, 01.04.2021

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Munich, 01.04.2021 Nicole Grabner

Abstract

With the proliferation of multi-robot systems, the interfaces required to operate them have become increasingly complex compared to those used for single robot systems. This can present challenges for operators who need to extract relevant information in order to make informed decisions about how to operate the robots. To address this issue, this thesis explores a variety of strategies aimed at improving the intuitiveness and usability of such systems. These strategies encompass a range of approaches, from designing user interfaces to integrating physical input devices, knowledge representations, and other modalities to assist operators. In this context, the thesis proposes a decision support system that provides operators with additional information in an intuitive way, focusing specifically on handling a set of distinct commands for a heterogeneous robotic team. A key constraint during the development of this system was the lack of historical data available to train the modules on. As a result, the proposed system was tested in a few-shot environment and was specifically designed for this circumstance. The support system comprises two modules: one that probabilistically classifies the next command using a data mining approach called sequence prediction, which is used to reorder the available commands in the interface; and a second that creates higher-level commands by mining frequent sequences from the historical dataset. These command sequences are presented to the operator, who can add them as additional executable commands. To evaluate the advantages and disadvantages of this novel approach, a user study was conducted, which showed that both modules increased the efficiency and usability of the system, while also identifying opportunities for further improvement.

Contents

1 Introduction

1.1 Motivation

The use of robotic systems has experienced a significant surge in popularity in recent years, with their applications ranging from personal assistance to industrial use. These systems offer a multitude of ways for humans to interact with them, including simple start and stop buttons as well as more interactive options. To enhance the user experience, different modalities for interaction have been created, such as computer, laptop, tablet, and smartphone interfaces, as well as game controllers, gamepads, and joysticks. Furthermore, some interfaces are designed to provide haptic feedback, while others display all the necessary information on a screen, and some offer a combination of both.

In this text, the focus will be narrowed down to on-screen interaction with robotic systems, while omitting additional feedback provided on non-screen interfaces. Nonetheless, some of the most interesting new input systems with on-device feedback that are used in conjunction with a screen-based interface will be highlighted. As application interfaces become increasingly complex, inexperienced users may find them challenging to understand, making interaction difficult. Overcrowded interfaces can also increase workload and reduce efficiency, ultimately leading to accidents [76].

Most robotic systems are based on a set of distinct commands [86][81] that involve executing a single action, such as moving an arm to a specific position or driving to a particular location. Alternatively, a command might involve planning and executing a complete set of actions based on a specified goal. The available commands or actions may vary depending on the type of robot and its level of autonomy. Interaction with robotic systems can be carried out using more widespread interfaces, such as a mouse or touchscreen, or more advanced devices, such as the haptic input device sigma.7 by Force Dimension. The latter provides haptic feedback, allowing the user to feel the robot's movements and exert force on the device to control the robot more accurately.

When determining the level of autonomy for different commands, certain criteria must be taken into account. The literature proposes various definitions to categorize the autonomy level of a system or command. However, the level of autonomy can be broadly defined in several subsections. For example, one subsection pertains to robotic autonomy, which is defined by the number of interactions required to achieve a specific goal [27]. Commands with low extent and time affordance are considered to have a low level of autonomy. On the other hand, if a goal consists of a combination of different actions and there is no need for intermediate inputs, the level of autonomy can be considered high.

Another subsection pertains to system autonomy, which concentrates on data extraction and processing. This approach is based on concepts like unification and evaluation of the entire system's data. It is not limited to the scope of single commands or actions to control robots, but rather encompasses the entire interaction and computation process. It is also applicable as a general information system. The complexity of each command depends on the degrees of freedom the robot has and the already implemented interaction possibilities. However, the level of autonomy is independent of complexity. Hence, it is easier to attain a high level of autonomy on systems supporting a narrow set of tasks.

For some systems, users can utilize commands on various levels of autonomy [48], resulting in a relatively high number of possible commands. This can sometimes lead to confusion, as some commands may result in a similar or even the same goal state. Therefore, it would be beneficial to have a system that supports the user in deciding which command to choose next or a way to make the robot automatically choose the next move. This would require a higher level of autonomy on the holistic side to develop reasonable suggestions.

1.1.1 Multi-robot Scenarios

The preceding discourse has been limited to the examination of the capabilities of single robot systems. However, with the current advancements in technology, interfaces have been developed that enable the commanding of multiple robots at once, also known as robotic teams. These teams can either be composed of homogeneous robots, i.e., robots of the same type, or heterogeneous ones, which consist of a blend of different robots.

When commanding a heterogeneous robotic team, there are numerous possibilities for interaction, as each robot has its unique abilities and fields of application. Consequently, an increase in the number of commands required to manage the team effectively is inevitable. Additionally, there may be instances where multiple robots must work in unison to accomplish a task, thereby necessitating more dynamic control mechanisms.

Resource management is also a crucial consideration when working with a team of robots. For a group of mobile robots, safety during any locomotion task is of the utmost importance, and prioritizing collision avoidance becomes imperative.

1.1.2 Non expert Operation of Complex Systems

When it comes to operating a team of robots, it's essential to provide thorough training to users due to the complex nature of the system and the potential problems that may arise. This puts additional requirements on human factors, which could be reduced or avoided with an effective interface. An effective interface should either be self-explanatory or provide enough information to guide users with no prior training.

If the system is highly advanced, a recommendation system can serve as an expert system to suggest following steps or provide an overview based on stored knowledge. The interface may also include a help page for available interactions. The system may include a ontology about the known world, which stores information about objects and their relationships for each robot individually, in a combined world state, or both. This ontology could be used to infer on reasonable next steps or help the operator decide on the robots' next steps by providing him with an overview of the current state.

To gain a deeper understanding of human-robot interactions, it's possible to record interactions and evaluate the operator's focus and commonly used interaction possibilities. This information can help developers extract the needs of users and develop further functionalities based on the gained knowledge. It's also essential to evaluate data based on user experience levels for effectively training support systems.

Manual improvements can also be made based on stored and gained knowledge. Training support systems based on experienced users' data can guide inexperienced users towards intuitive usage. Ultimately, a thorough understanding of the system and its users is an essential next step for efficient and safe operation of a team of robots.

1.1.3 Support Systems to decrease Mental Load

As mentioned, opportunities to increase levels of autonomy are divided into robotic command-based and holistic approaches. Both of those can be handled on each robot and the complete system of all robots combined in the robotic team. The supporting systems that can be created for each of these opportunities are sheer endless. Here, the focus lies on a data acquisition and processing approach that covers a combined view of the whole robotic team and will only brush on the topics mainly used for single robots. Depending on which data is recorded during the operation, the possible systems that can be built on top of it vary. This already filters the possible approaches one might take for each given system.

The recordings of the sessions here will be shown as a structured text file containing time-stamped natural language text and parameterization values. Those can be processed afterward. Hence, there is no data acquired to create a speech recognition system for example. Still, there are many opportunities to work with the data given.

First, the recorded data can be used to extract knowledge about the system. Therefore, one might extract semantic states of the given environment and store them in a world representation or ontology. Based on that, one can build a rule-based reasoning system. If the information is unclear and needs some further processing, one can fall back on natural language processing and data mining approaches to extract, classify, or translate the data to be processed further [87].

Another chance to create support for the operator lies in processing the sequences of actions and parameter changes within the system. Here, one might extract rules and common item sets of any form or create predictions [29]. Those processing steps can be implemented based on data mining methods or with any neural network approach. The core difference here is that neural network approaches work best if a lot of data is available to train the systems.

Those approaches can be used to gain and create knowledge about the system that can support the user. Depending on the system requirements, each of the proposed approaches might be better suited for one given environment than another. Hence, creating a general supporting system should be based on a modular approach that can quickly adapt to fit the corresponding system's needs.

1.2 Core Challenges in creating a supporting system

Providing support is a complex task that requires a system capable of processing data such that robots or users can understand the output. To achieve this, several key factors must be considered:

- Create a data storage that contains valuable information:
  - store detected objects within the given environment
  - store semantic state
  - store system and environmental parameters
  - store system knowledge
  - store user/operator interactions
- Process the stored data to extract useful information or improve the system
  - create a knowledge base
  - train artificial intelligence / neural networks
- Feed the gained knowledge or information back into the system
  - enhance the user interface
  - enhance the reasoning processes of the robot(s)

By following these steps, one can develop a comprehensive support system that caters to the needs of robots and users alike. It's worth noting that each robotic system may require unique data acquisition and processing steps, but this overview provides a solid foundation to build upon.

1.3 Explored Use Case

For the successful implementation of the proposed system, it is crucial to take into consideration the environment in which it will be utilized. Specifically, for the purpose of this thesis, the system will facilitate Surface Avatar experiments that have been outlined in [48]. These experiments entail teleoperating a team of robots from an orbital point of execution. The user interface employed for this setup will be operated by an astronaut stationed at the International Space Station (ISS), who will be tasked with controlling four distinct robots located on Earth: Rollin' Justin, the Interact rover from the European Space Agency (ESA), Bert, and the Lander Mounted Arm LAMA.

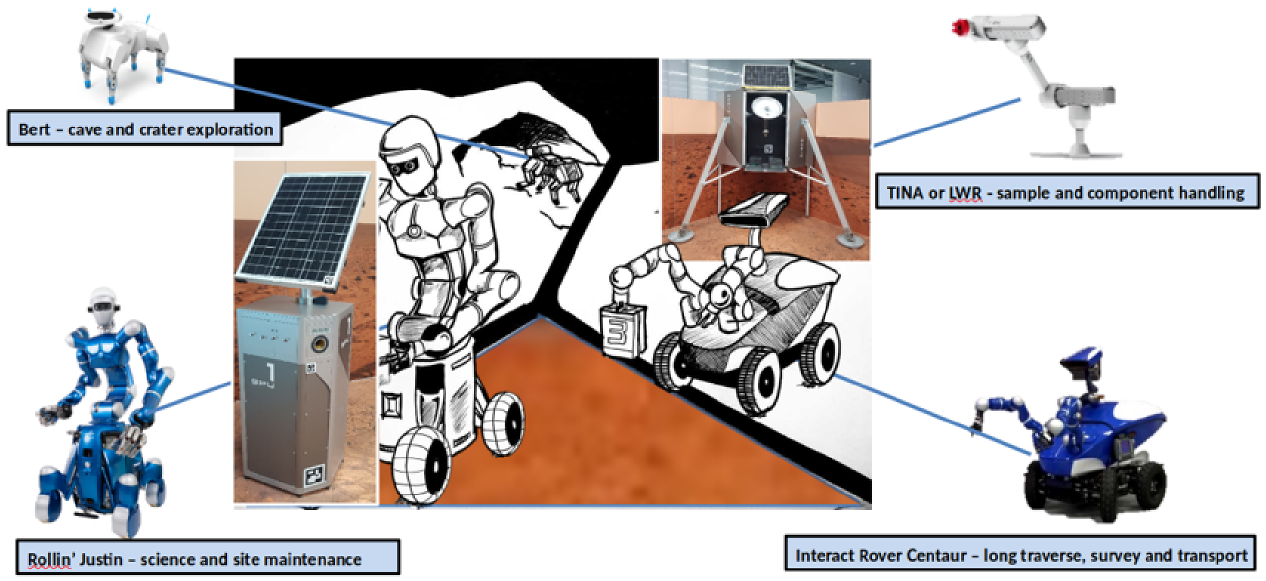

Bert – cave and crater exploration
TINA or LWR - sample and component handling
Rollin' Justin – science and site maintenance
Interact Rover Centaur – long traverse, survey and transport

Figure 1.1 Available Robots in Surface Avatar Setup

As depicted in figure 1.1, each robot has a specific role to play. Justin is responsible for the maintenance of the base camp and the handling of scientific workflow. On the other hand, LAMA is tasked with managing the storage of all scientific equipment and sample probes within the lander. The rover is equipped to make long traverses, while Bert is better suited for narrow spaces such as caves and craters.

The robots can be controlled through specific commands or direct teleoperation, with inputs from a haptic 3D device and joystick that translate into joint movements. The user interface displays camera views, available commands, and input configurations for the direct teleoperation modes of each robot, with the flexibility to switch between them.

1.4 Thesis statement and Overview

The primary objective of this thesis is to introduce a new data acquisition and processing system that is specifically designed for robotic teams. The system is highly modular and can be easily adapted to the requirements of most command-based user interfaces that are used for controlling either a single robot or a team of robots.

The challenges that are discussed in section 1.2 serve as the foundation for the proposed system, and each challenge is addressed through a process that is suitable in the test environment. The modular approach employed in the system allows for the addition or exchange of algorithms in each module, thereby making it more flexible and adaptable. Since this thesis also entails the creation of a data acquisition system, it is imperative to test the feasibility of the system through the use of a limited amount of training data. This few-shot learning approach is extending the current set of requirements for the proposed system.

The thesis is divided into several chapters, each with its own unique focus and purpose. Chapter 2 includes a thorough bibliographic research and state-of-the-art overview of the individual parts, as well as an overview of other existing systems. It covers various topics such as GUI layout optimization, knowledge representation, natural language processing, command recommendation in terms of sequence prediction, multi-robot handling, and a basic overview of UI testing. Chapter 3 describes the fundamental concept of the approach and its formal definition, while chapter 4 provides the test scenario definitions and requirements, an overview of the robotic team, and the user interface. Chapter 5 delves into the system's implementation and its interfaces, including a simple testing environment that was used during the development phase.

Chapter 6 presents an overview of the collected run-time data on the system's performance in terms of efficiency, speed, and usability. It also includes the results of a conducted user study. Finally, chapter 7 discusses the system's advantages and disadvantages and provides an overview of possible future work in this area.

In summary, this thesis presents a comprehensive and adaptable data acquisition and processing system that can be used to aid the operator in controlling either a single robot or a team of robots, thereby addressing the challenges that arise in such scenarios. The chapters provide a detailed overview of the concepts, implementation, and evaluation of the system, and discuss the potential for further research in this area.

2 Bibliographic Research

The operation of commanding robots can be quite demanding on the operator in many ways. The complexity of robots has significantly increased, and it is becoming more common to operate not just one robot but multiple ones at the same time. As the variables in the scenarios increase, the possible inputs to the operating system also increase, making the interfaces more and more crowded. This issue becomes even more critical in a multi-robot scenario where the efficiency and usability of robotic systems need to be enhanced.

To address this problem, automatic processes can be incorporated into interfaces to aid the operator in making reasonable decisions to complete tasks. The effectiveness of each method depends on the given goal and scenario. One way to improve the interface is by adapting its layout to make it more transparent and easier to navigate. Another method is to extend the current input and output devices to remove the full range of information that still needs to be displayed on the screen.

The operation of commanding robots can be quite demanding on the operator in many ways. The complexity of robots has significantly increased, and it is becoming more common to operate not just one robot but multiple ones at the same time. As the variables in the scenarios increase, the possible inputs to the operating system also increase, making the interfaces more and more crowded. This issue becomes even more critical in a multi-robot scenario where the efficiency and usability of robotic systems need to be enhanced.

To address this problem, automatic processes can be incorporated into interfaces to aid the operator in making reasonable decisions to complete tasks. The effectiveness of each method depends on the given goal and scenario. One way to improve the interface is by adapting its layout to make it more transparent and easier to navigate. Another method is to extend the current input and output devices to remove the full range of information that still needs to be displayed on the screen.

Adding processes that help to structure the displayed information is also possible. This includes intelligent structures that can store knowledge about the current state of the scenario, which can be stored in ontologies and world representations. Within those structures, one can create a comparably easy way to handle, store, and reason with the data at hand. The information can also be used to create recommendation values for the most likely following action. Predicting a following action is just one of many ways to create recommendations, as another commonly used method is based on a data mining point of view. It involves processing the history of interactions with the system and additional metadata to infer the connection between different events and actions within the system.

One crucial area that is frequently utilized to create a more natural interface is Natural Language Processing (NLP). NLP helps bridge the gap between robotic and human knowledge patterns since robots generally only comprehend what they have been programmed to understand. When the system can automatically translate data presented in a human-readable format to machine-readable data, the interaction between humans and machines becomes more accessible.

Given this brief overview of the intricacy of the subject, it is evident that there are numerous possibilities for creating a system that supports the operator. As a result, the next chapter will provide a fundamental overview of current and critical literature in the context of this topic and elaborate on the specific applications it would be helpful in. It focuses on the processes utilized in developing the given thesis, and while many more research areas can be utilized to support the user, the topics covered here will include information display and control, knowledge handling, natural language processing, command prediction and recommendation, a combination of different classification methods, and an overview of evaluation techniques for a user study at the end.

2.1 Information Display and Control

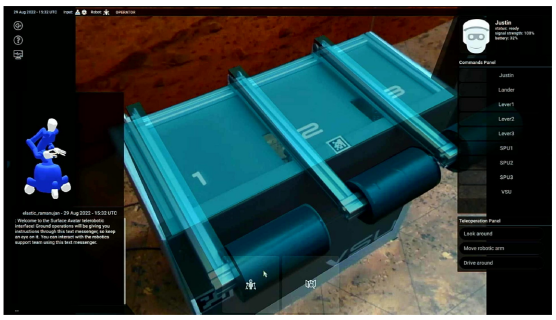
In order to effectively coordinate and control multiple robots operating simultaneously, a well-designed interface is essential. An optimized interface should provide a comprehensive system overview, including real-time status updates, task allocation, and sensor data visualization. It should enable operators to easily monitor and command individual robots or groups of robots, while also facilitating seamless communication and information sharing among the robots. Moreover, the interface should be intuitive and user-friendly, minimizing cognitive load and allowing operators to respond quickly to changing situations. By optimizing interfaces for multi-robot scenarios, one can unlock the full potential of these systems, promoting seamless coordination, enhanced decision-making, and improved overall performance. In the following sections, one will explore current research on projects that address these challenges.

2.1.1 GUIs Available

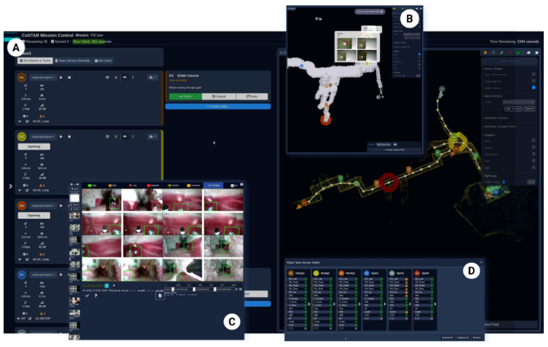
In this section, one will delve into the current implementations for multi-robot UIs and provide a more detailed explanation. Many graphical user interfaces for multi-robot scenarios primarily focus on high-level commands for each robot and provide an overview of the telemetry data of the robots. One specific area of the interface is dedicated to assigning available commands to the available robots [36] [81] [82] [85].

The multi-robot interfaces typically consist of multiple views that allow the operator to view the information available for the given system. The basic layout includes a map of the known surroundings, an overview of the basic robot information, such as availability, error messages, and available commands, and an overview of each robot's currently scheduled commands. Furthermore, an overview of the available camera streams from the robots is also part of the basic layout, as illustrated in figures 2.1c and 2.1b. It should be noted that most interfaces allow for the simultaneous execution of tasks for multiple robots. The commands are defined in such a way that the command specifies a goal, and the robots are responsible for planning all the necessary sub-tasks. Therefore, if one commands a robot to collect a sample at a specific point on the map, it autonomously plans the path, drives there, and collects the given sample. This approach eliminates any different levels of autonomy that may be present.

The design of interfaces for multi-robot systems is a crucial aspect of developing efficient and effective systems. As shown in Figure 2.1, these interfaces are typically designed to separate information into different screen layouts, which can either require multiple screens or an efficient way to switch between them. One interesting layout, presented in Schmaus et al.'s paper [86], focuses on a single robot for the



(a) Multi-robot Interface in robot focused view (source: [86])



(b) Copilot (source: [36])

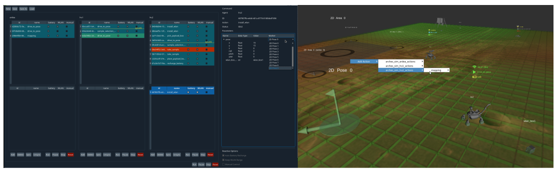

(c) ROS-MC (source: [81])

Figure 2.1 Current GUIs for multi-robot Scenarios

base layout (as seen in Figure 2.1a), allowing the user to have all the information of that single robot on the screen while also providing the ability to switch between robots to command each one of them. This focus on individual robots is particularly helpful for a lower level of autonomy, as the robots require more interactions for commanding. It is plausible that this system also allows interaction with the robots on different levels of autonomy.

It is worth noting that the components for multi-robot interfaces are stable in their combination, but the layout varies depending on the robotic team's autonomy level. In contrast, on a lower level of autonomy, the information is structured so that each robot is the center of attention by itself. On the other hand, robotic systems based on a higher level of autonomy tend to have a layout that provides an overview of all the robots at once. This highlights the importance of designing interfaces that meet the specific needs of the robotic team's autonomy level.

Overall, the design of interfaces for multi-robot systems is a complex task that requires careful consideration of the robots' autonomy levels. The layout of the interface must be tailored to meet the needs of the user, with a focus on providing efficient and effective ways to command and control each robot. With the right interface design, multi-robot systems can be highly effective tools for achieving complex tasks and improving overall efficiency.

2.1.2 Multi-Robot Handling by Single Operator

The use of multiple robots in a scenario can be quite complex, especially when it comes to commanding them, which is often the responsibility of a single operator. This can become overwhelming and potentially lead to errors. However, there are several options available to manage the workload and reduce the operator's stress levels.

As mentioned in the introduction, robots can operate at different levels of autonomy [27][86], offering a wide range of possible interactions. To keep the operator's workload manageable, one approach is to filter the commands based on their current mental state. This means that the operator's situational awareness, stress levels, and trust level are assessed in some way. Based on this assessment, the available commands and interface are adapted. This approach was highlighted in an article by Gomez et al. [76].

Another interesting idea was presented in a 2016 article [75], which stated that the data recorded from the robots and the operator can be used to create a selection model. This model determines which parts of the UI are most likely to be needed and used, taking into account the current mission state and the operator's mental state. Consequently, the robot's layout and commanding method are adapted according to these parameters. Furthermore, depending on the state, it should be considered to adjust the level

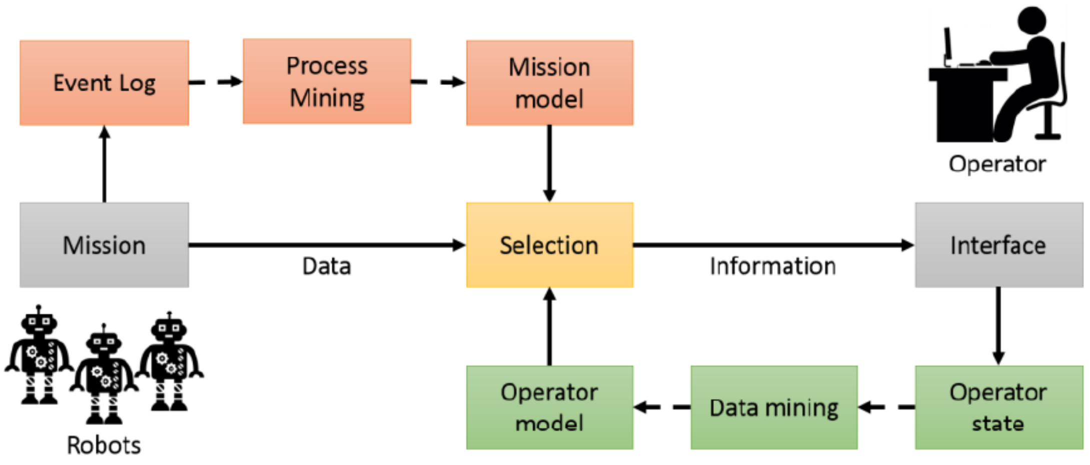


Figure 2.2 Mining Information from Robot and Operator-Side (source:[75])

of autonomy and the type of command provided for each robot. This method can save time and reduce errors. The states are retrieved from the user and mission data and collected in the selection algorithm, as shown in Figure 2.2, to decide which interface parts should be currently visible.

These approaches can make the commanding of multiple robots more manageable and less overwhelming for the operator. By adapting the interface and commands to the operator's mental state, the operator is more likely to make the right decisions and avoid errors.

2.1.3 Supporting Devices for Information Display and Teleoperation Modalities

Given the increasing diversity of interfaces and available input and output devices, it is important to consider how these factors can impact immersion and task workload. With numerous options available, including devices currently in development and already on the market, it is essential to carefully evaluate their potential benefits.

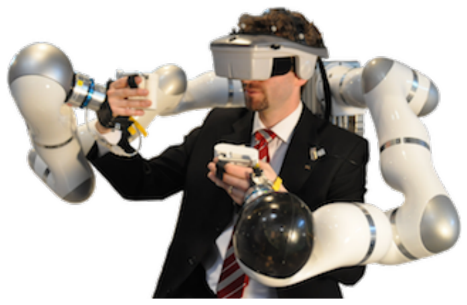
In addition to traditional input methods such as mice, keyboards, and touchscreens, newer control devices are being developed to provide output information to the device or an additional output device. This is known as haptic feedback and is a growing area of research. Haptic feedback devices can reduce workload and help manage information overload by providing additional resources [5].

One example of a haptic feedback device is the e-Vita haptic tablet [52]. This device, which resembles a touchscreen, can adapt its texture to provide physical feedback when the state of the commanded robot changes. The result is an increased level of immersion and a decrease in task workload.

Another device that provides both input and output capabilities is the sigma.7 joystick device from Force Dimension [93]. This seven-degree-of-freedom device offers force output on each of the translational and rotational dimensions, making it ideal for tasks that require collision detection. The force output can mimic collisions on the input device and guide the user in a specific direction.

Next, the GUI robot [30] is a unique spherical device that can detect hits, shaking, spinning, movement, and pick-up and put-down events. It can also use these actions as output events, such as simulating a jumping motion, shaking, or driving. The GUI robot was used to improve the usability of desktop GUI applications, including the 3D modeling application Blender, by emulating keyboard and mouse inputs

(a) Brain-embedded system to control robotic team (source: [38])


(b) Exoskeleton as input device with haptic feedback (source: [80])

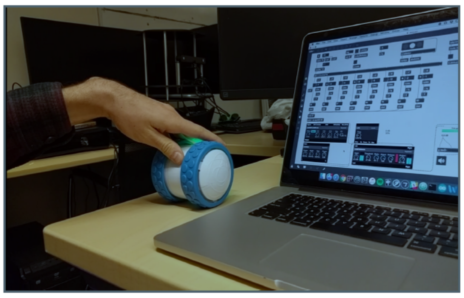

(c) GUI Robot Ollie with haptic feedback (source: [30])

Figure 2.3 Overview of in- and output devices

(see figure 2.3c). The result is an improved user experience and greater efficiency in non-scriptable applications.

An exoskeleton is one of the most complex haptic devices currently available in the field. Exoskeletons come in various forms, from arm-only to full-body exoskeletons, and are primarily used for complicated tasks that have yet to be autonomously implemented on robots [88][80][98]. Due to their complexity and infrequent use, the implementation of exoskeletons would be of little priority in most cases. However, they do offer increased granularity in which one can operate the robot or robots. An example for such a system can be viewed in figure 2.3b.

Finally, the traction cylinder [53] is another great haptic device option, which is designed to provide haptic feedback to the user of an unmanned ground vehicle (UGV). The traction cylinder outputs different motion patterns, depending on the current traction state of the vehicle, and deflects the information from the screen accordingly.

While most examples discussed here are primarily based on haptic feedback devices, it is worth mentioning that research in robotics is going in different directions. For example, in Kirchner et al.'s paper [38], robots are controlled using an embedded brain reading device, showing a promising future of using brain signals to control robots (figure 2.3a). On the other hand, Jeffri and Rambli's work [35] showcases the advantages of reducing mental load and task completion time with the help of an augmented reality system.

These technologies have varied fields of application, from exploration missions to assistive robotics. The potential for their implementation in different sectors is still being explored, and with the rapid advancement in robotics technology, one can expect more exciting developments in the field in the coming years.

2.2 Knowledge Representation and Handling

As previously mentioned, decision-making is an essential aspect of managing a team of robots. However, the operator's workload can be reduced if some autonomous processes can assist in this aspect. To achieve this, the system requires knowledge representation to make logical decisions within those processes. This is typically achieved by creating a "world model" or "world representation."

In simple terms, a world model is a machine-readable database that provides information about the current state of the environment surrounding the robots. It includes the states of all objects, including the robots themselves. Depending on the level of autonomy and the task space in which the robots operate, the world model may consist of different variables. For example, it can be a purely geometric model that handles collision avoidance for any movement. Alternatively, the model can contain semantic information and interaction instructions. It is often a shared data store that each of the robots or components can access. In other words, the world models can be seen as the system's current belief state.

The robots can share their world information and sensor data with the world model, as well as request information from the world model to plan their next task. Based on these world states, the robots can execute different tasks, which can be represented in a planning framework like the planning domain definition language (PDDL) [26][43][87][63]. This framework defines the world in a set of finite states, and the actions within are defined in a way where the prerequisites, the required tools, and the outcome of each action are defined. With this information, the robot can plan its actions accordingly.

2.2.1 Ontologies

By continuing to advance in the field of robotics, it is becoming increasingly important to develop a proper way to extract and process information from these systems. One newer approach to defining world models is through the use of ontologies, which are representations of a domain of knowledge consisting of classes with different properties and their relation to one another. While the concept of an ontology is not new, its application as a knowledge base in robotics is a rising trend, as it offers a more holistic approach to representing the world and solving problems without needing specific robots to perform tasks [63][7][60][55]

Unlike traditional world models, ontologies provide a better framework for the relationship between parts of the world, creating a network of world component classes, their properties, capabilities, functions, and relations. Each component can be represented by an ontology individually, providing a specification of the concept for each of the components. According to Olivares et al. [63], the cognitive capabilities implemented on current ontologies include recognition and categorization, decision-making, perception, prediction, problem-solving, belief modeling, execution handling, interaction with the system, remembering historical data, and learning from it. Although not all ontologies cover all of these points, the list of options alone shows the power and potential of such a system. Ontologies can create, monitor, and execute applications for a robotic team, depending on the circumstances.

One example of a robotic ontology is Know-Rob [8][91], an open-source knowledge processing ontology and one of the most full-featured ontologies currently available. It can perform most tasks described in Olivares et al.'s paper, taking an action-centered approach that models the available tasks of a robot. Another example is the Socio-physical Model of Activities (SOMA) [4], a project by the collaborative research center Everyday Activity Science and Engineering (EASE), which focuses on the physical and social context in which the actions are executed in and tries to model human reasoning on the robots in that scope. A query-based shopping assistant is one of the newest applications of this research project [41]. Each example is based on a semantic web representation of the given data, a widespread way to implement an ontology as it can represent the ontology's classes and its relations [90].

In summary, by continuing to develop autonomously operating robots, it is essential to have a proper way to extract and process information from the system and store it for future use. Creating an ontology that handles all the relations between acquired and given knowledge about the system is an excellent approach to achieving this goal. With the power and potential of ontologies in robotics, one can look forward to even more innovative and advanced systems in the future.

2.3 Handling Natural Language

Natural language processing (NLP) is a critical task in handling knowledge, given that the majority of information on our planet is presented in human-readable form. Any information generated by autonomous systems would be useful if it could be presented in an understandable form. Therefore, NLP is not only essential to understand the meaning of each word, but it is also important to understand the relationships between the words, as the same set of words can mean vastly different things if their order is changed. As a result, the development of large language models (LLMs) is on the rise. OpenAI's Generative Pre-trained Transformer (GPT) [12][70][71] is the most well-known LLM, and it is currently in its fourth version [64].

GPT, like many other LLMs, is based on the transformer architecture [97] and is trained on a massive amount of data. Transformers are attention-based recurrent or convolutional neural networks with an encoding and decoding block that can transform one sequence of tokens into another. This technique is commonly used in natural language processing, but transformers can also be trained to process audio and video data. As the resources required to fully train such a model are relatively high, one can choose from various pre-trained models. To fine-tune the model for one's specific needs, a limited amount of data-points need to be prepared, but using at least a couple of thousand data-points to fine-tune one of those pre-trained models is still recommended to create a valuable new model. Hence, it is not the new miracle weapon to deal with all NLP problems without the corresponding data available.

It is important to note that the output of these models is not always one hundred percent reliable, given that they are based on neural networks. The output is computed based on trained weights that provide a probability distribution based on the training data. Due to the high amount of training data, the output might sound correct in many requests, but it should not be seen as a single source of truth [79]. This is because it is difficult to define a system in a way that the truth value of each of the outputs can be determined and verified. Additionally, the models may hallucinate made-up information that appeared as fragments when combining parts of the used training data. Moreover, using those models is computationally expensive and, therefore, might not be an appropriate solution for any NLP requirement.

2.3.1 Pretrained Models available

The platform huggingface.co [100] is a well-known research platform where you can access pre-trained models and data sets to train those models. It provides various options for different training model presets, application domains, and languages in which the models were trained. This thesis will discuss a few general-purpose text processing models in English that you should be aware of. Along with the GPT2 model, which is also available on this platform, there are multiple versions of the Bidirectional Encoder Representations from Transformers (BERT) [15][42][33], LUKE (Language Understanding with Knowledge-based Embeddings) [103], CANINE (Character Architecture with No tokenization In Neural Encoders) [13], and LLaMA [96][95].

BERT is an extensively researched model that covers many languages [69] and can also be used for translation tasks. Depending on the specific model used, it tokenizes the input text on a word or sub-word level, meaning it translates the text in a sequence of tokens or identifiers for each word or sub-word. This sequence of tokens is then passed to the model to process, where the sequence will be mapped to a multidimensional space that is supposed to map semantically similar phrases and words close to each other. Each unknown word or sub-word will not be given a valid token, as the model works on a word basis, meaning that those will be defined as 'UNKNOWN' and processed as such. Pre-trained BERT models often only come with the encoder part of the model, as the decoding is dependent on the application field of the model.

LUKE is initially based on the BERT model and hence has the same restrictions in terms of word tokenization. However, this model has an advantage over BERT as, in addition to the usual training data, it allows you to define a finite set of entities, which creates the possibility to model relations between those specified entities. This works similarly to the token attention space in other tokens in usual transformers. The type of token is also considered when computing the model output, which leads to better results in entity-related reasoning and classification.

Natural Language Processing (NLP) is a rapidly growing field with many exciting developments. One of the most significant advances in recent years has been the creation of sophisticated language models that can understand and analyze text with remarkable accuracy. These models are powered by complex algorithms that enable them to process vast amounts of data and extract valuable insights from it.

The CANINE model is trained without tokenization of words, which means that there is no preprocessing step required before feeding information to the model. While the input length is limited to 2048 characters per query, the model is less sensitive to writing errors and can map unknown words within the usual dimensions without any issues. However, as the number of input parameters is higher without tokenization, the model works with three internal encoders to embed text for further processing.

Finally, the LLaMA model was created entirely using open-source data and offers a range of models with varying parameters. While the model can be used for personal projects, it is strictly prohibited for any commercial or research use in biometric processing or military areas. The biggest challenge with this model is that all available parameters need to be simultaneously loaded in float16 precision into CPU RAM, which requires at least 130 gigabytes of RAM for the 65B model. However, the smallest of the models should be able to run on a 14GB RAM system. The second version of this model is fine-tuned for chat applications.

For those looking to leverage language models for NLP tasks, the model storage of huggingface.co is an excellent resource. However, it is essential to check the licenses of the models before use, especially for commercial and research purposes. Additionally, users need to log in to their account for each use of the models, and the models from huggingface.co may not be appropriate for all applications. Alternatively, users can explore the models offered by the PyTorch and TensorFlow libraries, which offer a smaller selection of models but are still powerful. Some of the BERT models also have independent libraries that work without middleware.

Overall, the field of NLP is continually evolving, and new models are being developed and refined. As such, it is essential to stay up-to-date with the latest advances and choose the right model for the task at hand. Whether it's one of the previously mentioned models, or another model altogether, there are many exciting possibilities for those looking to leverage NLP to analyze and understand text.

2.3.2 Application Areas in Robotics

Now that a primary understanding of what types and models for NLP are available are present, this section is moving on to its application. There are many fields in which these models can be beneficial, but one of the leading research areas in this direction is task planning in a user-interactive way. For example, the user inputs a request or goal for the robot [92][105], which will try to plan a sequence of actions to reach this goal. In that scope, it is also essential to translate the input into an action that the robot is capable of, as stated in other papers [34][2].

As in traditional systems, the robot usually assumes unclear or unknown parameters, the difference here is that the robot asks the user for clarification. This helps the robot so that the plan's outcome better resembles what the user had in mind. It can also be used to create PDDL actions based on a prompt given some sample actions [87]. Additionally, these systems can improve the operator's trust in the system itself [31] and enhance the interaction with the system compared to a conventional control setup.

Furthermore, NLP models can be applied to any text classification task, where, for some instances, only the encoder part of the transformer is used for embedding the words or phrases in the feature space. Other neural network-based methods exist for embedding the words, such as global vectors (GloVe) [68] or Word2Vec [59].

However, these methods can be computationally expensive. Therefore, one of the simplest ways to compare text strings is by computing the Damerau-Levenshtein distance [14]. This distance algorithm focuses on character overlap, meaning that the output of the algorithm is the minimum operations needed to change one string into the other. While it is mainly used for spelling error correction, it can also be used here as a preprocessing step in cases where the definitions have similar descriptive strings for most instances of the same class.

Given these embeddings, the text can be classified with any appropriate classification method like k-means clustering, k-nearest neighbors (KNN), or any neural network-based classification method, where it is also possible to train the transformer decoder to output the classes. By doing so, one can achieve highly accurate and efficient text classification results that can significantly improve a wide range of applications, including chatbots, sentiment analysis, and more.

2.4 Command Recommendations and Decision Support Systems

In the field of decision support systems, the application of knowledge extracted from data can provide operators with significant advantages. According to Rosati et al.'s work [77], the main components of creating support systems for predictive support systems are data collection, feature extraction, setting up a predictive model, cloud storage, and data analysis. This applies to any knowledge-based recommendation and support system [62][65].

Recommendation systems are widely used in various fields, covering many areas of support systems and their information display. Therefore, any Decision Support System can also be called a part of a general recommendation system. However, this definition might only be supported within some scientific literature. In the software development interface, the term recommendation system often refers to a help page system and command recommendation system that introduces unknown commands to the operator [25][47]. This is especially useful in environments with at least a hundred, if not thousands of commands, where it can improve command awareness of the operators. It also highlights that help pages about the usage of commands can improve the general understanding of what impact the execution of a command can have on the system. Another type of recommender system works on a similarity check, as it recommends similar content or commands to the operator [74].

Furthermore, many recommendation systems cover some form of prediction within the system [25][47][1]. In many cases, the usage of the system covers a few reoccurring sequences of commands, which can be extracted from the recorded data and used to improve the overall usage of the interface. Especially in robotics, those sequences can also be used to extract specific goals that can be achieved within the current setup. Aggarwal et al. [1] proposed a system where the sequences are classified for specific goals.

Therefore, the most probable goal is classified first, which ideally fits the one the operator has in mind. This goal is then used as an additional input to the sequence prediction algorithm and improves its performance. This is useful as this system filters unwanted recommendations with similar sequences but different goals.

Given this first overview and definition of recommender systems, the following part of this thesis will review a couple of examples of sequence mining, sequence prediction, and its application field. For a basic overview of the whole topic of time series forecasting, reading Masini et al. [57] is recommended.

2.4.1 Data mining scope

In the scope of data mining, sequence handling plays a vital role, encompassing various algorithms involving sequential pattern mining, clustering, sequential rule mining, sequence prediction, and many more. These conventional machine learning algorithms are primarily associated with the processing of big data but can also offer value on small data sets in many cases.

Given the limited space of this thesis, only the sequence pattern mining and prediction algorithms implemented in the system will be covered. It is essential to note that, unlike time series mining and prediction, these algorithms do not predict a single value's occurrence but rather define the possibility of an item occurring after a sequence of items or item sets. An item can refer to anything from an event, object, command, string, or any other identifying value representation. Additionally, an item-set is defined as a combination of finite items. However, most algorithms only consider a sequence of single-item item-sets as a sequence of items.

The system used in this thesis does not have implemented data collection, limiting the available data at the point of testing. Therefore, it was decided to use loss-less sequence prediction algorithms. The CPT+ algorithm [28][29], which has higher accuracy compared to other prediction algorithms, and the SuBSeq algorithm [40], which works on a similar approach, are presented. The significant difference between these algorithms is that CPT+ does not consider the order of items in its prediction, while SuBSeq does. Both algorithms extract all sub-sequences of the training data and then create a tree structure based on them. During prediction, they compare the branches of the tree with the input and output the most likely next item based on that.

In the realm of sequence pattern mining algorithms, it's essential to have an algorithm that supports different parameterizations so that the model can be easily adapted to the needs of other applications afterward. That's why the algorithms within the Python library prefixspan were chosen, as they include three different specific algorithms: Jian et al.'s PrefixSpan [67], Yan et al.'s BIDE [99], and Chunacong et al.'s FEAT [24]. Additionally, two other algorithms were used, namely the non-overlapping sequential pattern (NOSEP) algorithm [102] and the Quantile-based Cohesive Sequential Patterns (QCSP) algorithm [20].

The basic implementation of pattern recognition recognizes any sub-sequence within a set of sequences where the items of the sub-sequence occur in the same order within those training sequences. The number of times these sequences occur in the training set is called the support of the sub-sequence. Each sub-sequence can only be detected once within each sequence present in the training set for the basic prefix-span algorithm. QCSP follows the same approach, but with the option to define a cohesion value, which means that the span over which the subsequence ranges can be limited.

In the case of the NOSEP algorithm, an additional requirement that the extracted sub-sequences do not overlap is stated. The detected subsequences are cohesive based on a maximum gap between the items. It can detect a subsequence multiple times within a given training sequence. If one wants to learn more about current advances in this field, Nagori and Kumar's paper "Issues and Research Challenges in Sequential Pattern Mining" [61] provides an excellent overview of this topic.

2.4.2 Neural Networks (RNN, LSTM, GRU)

One of the available machine learning options is a neural network (NN) based approach. For time series prediction, the most common models are based on recurrent neural networks (RNN) [49][104]. RNNs can store and handle a recursively updated internal memory that includes previous states of the model. As there are many different implementation specifications of RNNs, the two most commonly used RNNs in time series forecasting are the long short-term memory (LSTM) and the gated recurrent units (GRU).

LSTMs can learn long-term dependencies, which helps deal with the vanishing gradient problem. During training on big timesteps, the gradient of gradient-based approaches vanishes or becomes very small, and LSTMs can handle this issue. It is based on an input, forget, and output gate that handles what part of the data to learn, save, forget, or remember. Based on that, the output will be computed.

GRUs are more straightforward models compared to LSTMs. They lack an output gate, which is replaced by an update and reset gate. These gates are represented as vectors and define which part of the stored information should be given to the output of the GRU cell.

However, most NNs require data to be discretized in regular intervals for time series forecasting. This makes it difficult to use for data with missing information or arriving at random intervals. This issue has been handled in a paper on GRUs, which shows that they are effective in handling missing patterns for sufficient imputation.

In addition, due to the previous occurrence of the topic, the use of transformers is also possible for sequence prediction. Transformers have gained popularity in natural language processing and have shown promising results in sequence prediction tasks.

In conclusion, the use of RNNs, specifically LSTMs and GRUs, has become a popular approach for time series prediction. However, the issue of missing data and random arrival intervals can be challenging to handle. The use of transformers also presents a promising option for sequence prediction tasks. With these advances in machine learning, it is possible to improve time series prediction accuracy and make better decisions based on data-driven insights.

2.4.3 Monte Carlo Methods

Markov chains, also known as Markov processes, are a commonly used method for sequence prediction [6]. This modeling approach utilizes a set of states to represent the different items of a sequence and aims to define distinct probabilities based on Bayesian inference to the occurrence of an item given the current state. The state is defined by the sequence of a finite number of last observed events. Since this is a stochastic model, algorithms are necessary to create predictions from these constructs.

One well-known example of such algorithms is the Markov Chain Monte Carlo method (MCMC). The Metropolis method and the Gibbs sampler are among the most well-known algorithms for MCMC. However, a transformer based on the MCMC method has also been developed more recently [56].

2.5 Combining different Classifications and Predictions

The previous section discussed various methods for predicting and classifying the occurrence of an item. While each algorithm has its own strengths and weaknesses, many systems benefit from combining multiple algorithms to achieve the best possible results. In this section, some examples of how to achieve this will be covered.

The two most common methods for combining algorithms are importance sampling [94] and bootstrap aggregating (bagging) [10]. Importance sampling works by identifying the algorithm that performs best for each state and using its output. Meanwhile, the output of the other algorithms is discarded. Bagging, on the other hand, combines the outputs of the different algorithms. In this approach, each prediction value is weighted and summed for each item. The weights can be equally distributed or based on a previously calculated probability distribution. For distinct classification methods that do not return distributions, one can either work with one-hot encoding or pass the class with the highest classification occurrence number.

Importance sampling and bagging can be combined to create hybrid algorithms that leverage the strengths of both methods. This approach is particularly useful when dealing with complex systems that require multiple algorithms to achieve the best possible results.

Another approach is to use ensemble methods [51], which combine the predictions of multiple base models. Ensemble methods can be grouped into two categories: homogeneous and heterogeneous. Homogeneous ensembles use the same base model with different parameters, while heterogeneous ensembles use different types of base models. Ensemble methods can also be further divided into three types: bagging, boosting, and stacking. Bagging and boosting were previously discussed, while stacking involves combining the outputs of multiple models using a meta-model.

In conclusion, there are many ways to combine algorithms to improve the accuracy of predictions and classifications. By leveraging the strengths of various algorithms and combining them in innovative ways, it is possible to create highly effective models that can handle even the most complex systems.

2.6 User Study

To assess the effectiveness of a system in enhancing the usability of a User Interface, it is crucial to conduct one or more user studies to confirm that the additional system adds value to the interface in terms of operation speed, perceived ease of use, and various other parameters.

The user's behavior can be evaluated with regards to engagement, responsiveness, attractiveness, and performance. However, not all of this information can be automatically evaluated within a basic setup. Therefore, various questionnaire types are used to extract the perceived usability, which can be assessed afterward with all the automatically logged data.

There are many different combinations of user data evaluation types, and as such, the most commonly used ones will be reviewed in the following sections. By considering these evaluation types, one can effectively assess the performance of a system and make informed decisions to improve the usability of a User Interface.

2.6.1 Run-time Data to be stored

One common practice is to log all interactions with an interface or a specific set of them. These interactions can range from clicking on different items to executing specific tasks or opening menus. The amount of data needed depends on the hypothesis that one wants to prove. If there is no specific hypothesis, adding as many data points as possible can be beneficial as long as the overall system's performance and operation speed are not negatively impacted.

2.6.2 Standard Questionnaires

Questionnaires can be divided into two groups: those that evaluate the overall system performance and those that evaluate the system on a task level. In both groups, users rate their satisfaction and usability of the system using a corresponding rating scale, such as the Likert scale.

Overall performance questionnaires are completed after the proposed system has been tested by the user in a user study. These questionnaires typically have a broader range of questions, covering aspects such as speed, interface, usability, and error management within the system. One example is the System Usability Scale (SUS) [11], which consists of ten questions about satisfaction with the system and the ability to learn to use it. The post-study system usability questionnaire (PSSUQ) [46] covers additional questions about the interface layout and whether users were able to find the information they needed. Other questionnaires, such as the QUIS [83] and SUMI[37], cover a wider range of topics, including display layout, terminology, system-feedback, learning, system properties, technical instructions, and more.

Questionnaires are also commonly used after each task to measure user satisfaction, expectation vs. reality, or perceived task load. These questionnaires typically contain only one or two questions, such as the single ease question (SEQ) [83], the subjective mental effort question (SMEQ) [84], or between two to three questions [46][3]. NASA developed a more extensive version for task load measurement, called the task load index [32], which divides task load into mental, physical, and temporal aspects.

Overall, post-task questionnaires tend to focus on user satisfaction and ease of use, while post-study questionnaires take a more holistic approach to identifying areas where the system and its interface can be improved.

2.7 Summary

By delving deeper into the world of User Interfaces, it becomes apparent that there is always room for improvement. One of the primary goals of UI design is to make the interface as easy to handle as possible, reducing the mental load that the operator has to exhaust. There are several ways to achieve this, including improving the layout of the Graphical User Interface (GUI), deflecting the mass of information in the GUI to other devices, or working on the knowledge handling of the system.

Speaking of knowledge handling, there are numerous options available, ranging from natural language processing (NLP) that can handle both spoken and written input to different kinds of knowledge representations. The most common method, however, seems to be an ontology that represents knowledge in different classes with dependencies. Data mining and machine learning techniques like sequence prediction can also create further knowledge. When it comes to machine learning, one can choose from neural network options like the Recurrent Neural Network (RNN) and conventional methods like prediction-tree-based options. It is important to note that only certain options are feasible for each project. For example, neural network options require a comparatively high amount of training data to create reliable output, whereas some conventional methods can work with a lower amount of training data. Another important point is that NN approaches focus on predicting time series and not prediction of categorical items, although with a one-hot encoding, that could be achieved as well.

Given the plethora of options available, one may find that for an exact prediction or classification, more than one option would be feasible, and hence one would like to combine the outputs of those options. Therefore, different options to do so have also been covered, ranging from particle filters to bagging.

Furthermore, it is essential to evaluate the usefulness of such a system, which can also determine how user-friendly the given system is. This can be done through user studies, where the basic structure of user studies and the data that is usually recorded have been discussed. There are several options available to evaluate a system's usefulness, and they have been elaborated upon as well.

In conclusion, the world of UI design is vast and ever-changing, and there is always something new to learn. With the plethora of options available, it is essential to choose the right one for each project, keeping in mind the constraints and limitations. However, with the right approach, it is possible to create an interface that is easy to handle, efficient, and user-friendly.

3 Concept

The proposed framework aims to enhance the user experience, particularly for those who are new or inexperienced in the system. With the use of a multi-robot commanding user interface, the framework aims to improve the ability to execute the given task quickly and with ease. This chapter lays out the basic concept of the proposed system and provides a formal description and necessary definitions.

It is important to note that this system was created without significant historical data from the given system, as there has yet to be an implementation for logging the data. Additionally, the corresponding team has a pending project deadline (not involving this thesis' work), which restricted access to the robots and limited the number of logged missions. However, an approach has been utilized to improve operator performance, even on a low count of run-time files, making the approach a few-shot-learning problem. The performance would significantly increase the more logged runs are given to train the model.

To achieve this goal, a modular approach was employed, allowing recorded data to be processed in different ways, with each individual part being adapted according to the specific needs of the system. Additionally, other processing algorithms can be added to it for further customization. The focus is on a data-centric approach using historical data logs from the specific application system to create recommendation data for the user.

The concept presented here provides a basic description of the proposed system and the necessary definitions. It is expected to provide significant improvements in user experience, particularly for new or inexperienced users, and to offer a more efficient and effective way of handling commanding tasks within a UI.

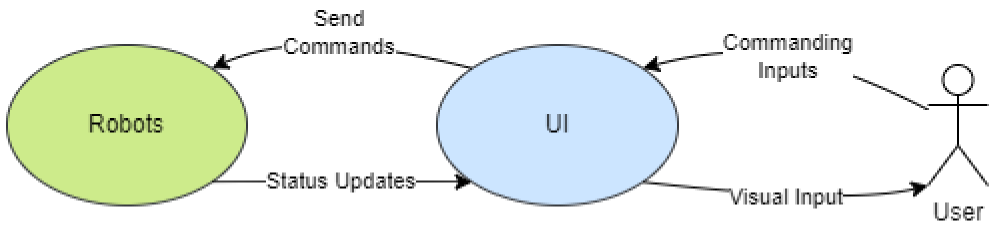


Figure 3.1 Overview of current communication components

The current system, which will be explained in more detail in the following chapter, offers a wide range of options for interacting with robots, including teleoperation with a force feedback input device and a joystick. However, due to the varying levels of autonomy and framework definitions, the decision was made to focus on sequence prediction. This allows for a more universal approach as the definition of a sequence is not dependent on any specific framework or action definition used by a particular robot. In addition, a plethora of interaction events can be recorded and used as training data, enabling the system to support any robot that can provide a set of commands in natural language form.

Moreover, any additional event information that can be formalized in a finite set of state or action definitions can also be added to the sequence definition. This distinction makes the proposed system enhancements usable in most scenarios with state-of-the-art robots, facilitating seamless integration with a wide range of robotic platforms. With the ability to work with preprogrammed commands or actions, this system can execute a variety of prediction and classification tasks, making it highly adaptable and versatile. The system's focus on sequence prediction ensures that it can efficiently and effectively predict and extract a sequence of tasks from the historical data, streamlining robotic operations and facilitating greater efficiency in executing complex tasks.

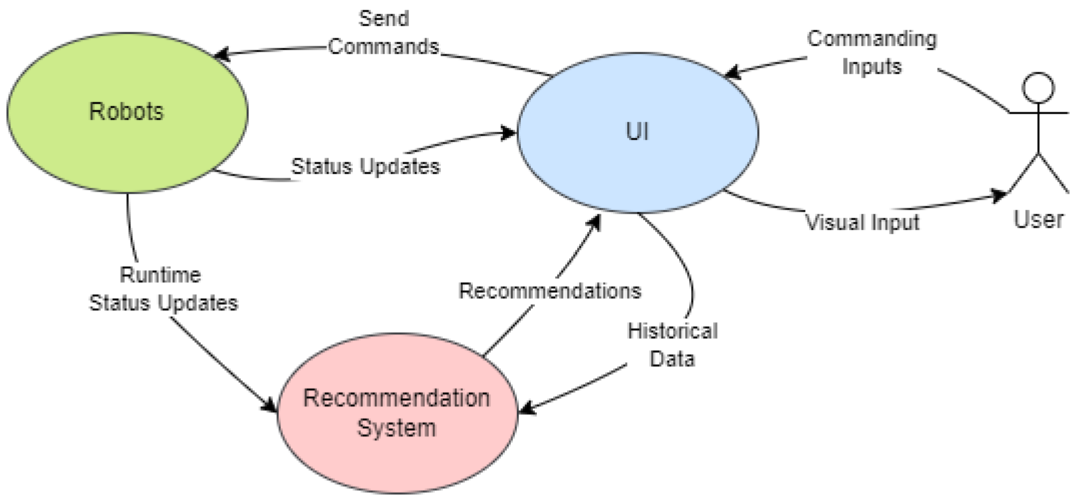


Figure 3.2 Overview of the proposed setup's components

In the current setup, the robots establish communication through a reader-writer setup with dynamic message definitions. The framework will be equipped with specific message definitions for each of the published topics, allowing it to be tailored to the requirements of the project. The framework will handle all real-time communication between the UI, the robots, and the recommendation system. As a result, the proposed system will receive all real-time data from the published topics and must ensure that all necessary topics are subscribed to and available in the system. Otherwise, the process will not be able to make any accurate predictions. This also enables seamless integration for this system in any similarly structured control system for robotic teams.

3.1 Modular Approach

The system in question is expected to be as general as possible, and as such, a modular approach was chosen. This decision allows for each process within the system to be extended or replaced with similar or entirely different processes as required. It is important to note that each of these processes has a train and a run-time mode as the data format may differ between the two. This requirement is necessary because the training data will be extracted from log files, while the run-time data will be directly extracted from the communication system between the robots and the user interface (UI).

These modules can be categorized into five groups based on their function in the process. A summary of these categories can be seen in picture 3.3. This block represents the processes that run within the red component, which is the recommendation system in picture 3.2. The first process, called 'Preprocess,' is a preprocessing step that is essential since most processes cannot work directly with raw data available in different formats. Therefore, the data must be processed to ensure uniformity irrespective of the source.

Next, the data will be filtered to suit the needs of each specific model within the system. This step is necessary because not all the data will be needed for every model. For the system presented in this thesis, the data will be filtered in multiple ways based on the event type to prepare different sets of sequences for the recommendation-generating models. This module is called the 'Filter' module.

The third module is the 'Classify' module, which is implemented to assign a specific aspect of the given event to one of several predefined classes. This additional information is then passed to the different models, making it particularly helpful when an unknown command or event is encountered. This classification enables the calculation of outputs for such instances.

The fourth module is the 'Models' module, which is the most complex and diverse module. This module can contain any function that processes the given sequence data into something useful. For the given system, these models contain two different data processing types. Firstly, it evaluates the sequences of events given and calculates priorities for each available robot command. Secondly, it uses the pure sequence of issued commands, evaluates which sub-sequences occur frequently, and sends those to the operator to add them as higher-level commands.

For the last step, it is essential to ensure that all the data is in the right format before sending it to the UI or the robots, depending on the data processed. The UI will then process all the generated command couples for the system. However, it would be interesting to try to process the data directly on the robot side, especially for the higher-level commands. This functionality would be particularly helpful for commanding robots from orbit as it would reduce the waiting time that occurs as a direct consequence of the data transmission lag as the robot is able to execute the full sequence and does not have to wait for the next commanding input.

For each category, as many processes as desired can be implemented for each module. However, it is crucial to ensure that the transferred data type within these modules matches for each of those instances. If the exchanged module differs significantly from the previous one, it would be beneficial to check whether the other components are compatible with the desired change in the system. This additional check ensures that this system's modules can be exchanged on demand.

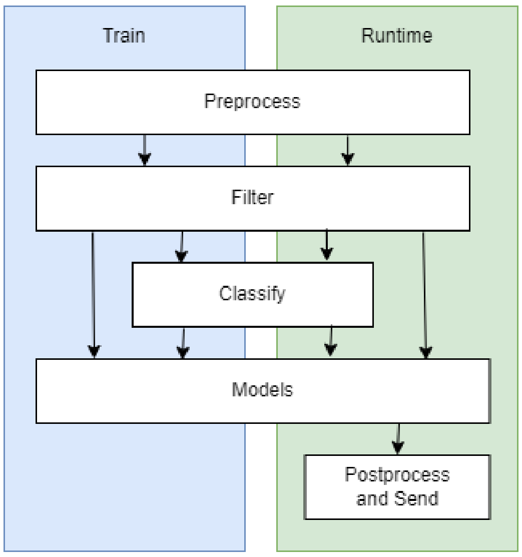


Figure 3.3 Overview of the Modules

3.2 Formal Description

In order to better understand the nature of a given system, it is often useful to formalize the basic setup in a mathematical way. This can be achieved by considering a database D of distinct sequences, each of which is defined by the available recorded sequences. These sequences, denoted by $s \in S$, consist of specific events e that define a particular action or state change within the system. These events belong to a defined set of events E, which may include actions such as "Send command," "Speak," "Add object," and "Table," among others.

To formalize this structure, one can define the sequence database D as a collection of all possible sequences that can be generated from the available events in E. Specifically, one can represent each sequence s in D as a tuple $(e_1, e_2, ..., e_n)$, where e_i denotes the i-th event in the sequence. Furthermore, one can define the length of a sequence s as the number of events it contains, denoted by $|s|$. With this baseline in place, it can be summarized by the definitions for the historical data as follows:

Definition 1: Overview of the Sequence Data Sets as retrieved from the historical data:

- D_l: This set contains all non-filtered sequences in the historical data. It is represented by $D = \{(s_k)_{k=0}^n | s_k \in S, n \in \mathbb{N}_0\}$.
- S_l: This set represents the sequence of events for a recorded instance. It is defined as $S = \{(e_k)_{k=0}^n | e_k \in E, n \in \mathbb{N}_0\}$.
- E_l: This set includes all possible events in the historical data. It is represented by $E = \{(p, s, e, t) | p \in P, s \in S, e \in T, t \in T_s\}$.
- P: This set contains all possible event participants or objects affected by the event. It is usually a set of available objects and can be defined as $P = \{"Justin", "Panel3", "Operator", "Seismometer"\}$.
- S: This set contains all the sources of the event or who executed it. It can be defined as $S = \{"Justin,""LAMA,""Rover,""Bert"\}$.
- T: This set defines the overall event type, which is logged. It usually corresponds to a published event or specific function within the program. It can be chosen as needed and might contain $T = \{"Send\,Command", "Command\,Reply", "Add\,Object, "Add\,Robot", "Change\,Robot"\}$.
- T_s: This set contains the sub-type of the event and doesn't have to be defined for each event. It might contain the specific commands issued on the event type "Send Command," which would correspond to $T_s = \{"pick\,Seismometer,""update\_firmware,""drive\_around"\}$.

In order to effectively process and analyze the items in the event set E_l, it is necessary to perform a preprocessing step. This step will enable the events to be brought into a more manageable form and will also provide additional classification information for each event. Although the database and sequence definitions will remain unchanged, the corresponding events will be redefined to facilitate the analysis process.

It is important to note that this definition will not cover the processed database D_p and its sequences S_p. Therefore, proceeding with the following definition to ensure that the data is handled accurately and efficiently is imperative:

Definition 2: Mapping the event so they are processable

- Pre is a way to map an event e_l from the set E_l to a processed event e_p from the set E_p.
- E_p: This set consists of possible processed events defined by the tuple (i, t, d, c), where i is a unique identifier, $t \in T$ represents the event type, $d \in Def$ stands for the representing string representation, and $c \in Class$.
- Def: This set consists of unique string identifiers created from the event participants, source, and sub-type
- $Class$: This set contains the class identifiers. The event class c is assigned by a classifying process in Pre.
- F is the filter applied to E_p to retrieve the corresponding subsequence, depending on the specified event type t of the events.
- E_f This set contains a subset of the processed events E_p after applying filter F.

The definition presented above already encompasses the mapping process that occurs in the 'Preprocess' and 'Classify' modules discussed earlier. These modules are merged into a single definition, and since filtering can be understood as defining a subset of the given set, it is also handled by the definition of E_f. Moreover, this definition applies to each preprocessed sequence, and it is possible to create multiple filters, resulting in different sequence types for further processing. Consequently, the fully prepared data can be passed to the modules. In subsequent sections, one can delve into more detail about this definition for both implemented models. The following are the fundamental definitions for the processing of the data within the modules:

Definition 3: Processing of the prepared data by the Modules

- M: This object contains all processing up until the model's data is sent.
- $M.train(s)$: This function trains the model using historical data $s \in S_p$.
- $M.predict(x) = rec$: This function processes the current sequence of run-time data ($x \in S_{in}$) and returns recommendation data ($rec \in Rec$).
- S_{in}: This is the sequence of events processed within the recommendation system, defined as $S_{in} = \{(e_k)_{k=0}^{n} | e_k \in (E_p \cup E_{new}), n \in \mathbb{N}_0\}$.
- E_{new}: This is a set of unknown commands stored in the same way as processed events E_f.
- where Rec: This is the data published and sent to the UI or robots, which is the output of each model.

Next, the specifics of both implemented models will be elaborated on, starting with the Command Prioritization model, which will be referred to as model M_1. In the interest of simplicity, the input to the functions will not be included, as it should be clear from the previously defined process and context definitions. Our ultimate objective is to generate item predictions from the sequence database D_f that are consistent with the available recorded sequences. This means that any item prediction generated can only contain predictions concerning events that are present in E_f. To provide a comprehensive overview and delve deeper into this topic, the following definitions are provided:

Definition 4: Specifications of the "Command Prioritization" model

- M_1: This contains a model to derive the command priority based on the historical data
- $M_1.algo$: This is a list of algorithm descriptions and parameters. Multiple instances of these algorithms can be combined to create the model, resulting in the notation $M_1.algo = \{(t_k, s_{tk})_{k=0}^{n-1} | t_k \in \{CPT+, subseq\}, s_{tk} \in T_s, n \in \mathbb{N}\}$, where n indicates the number of algorithm specifications used.
- T_s: This is the type of sequence present. It is detemined by the filters used to extrac the subsequence. As an example, it could be defined as $T_s = \{"sendCommand, ""classified, ""fullsequence"\}$.
- $M_1.train = (M_1.algo[n].train)_{k=0}^{n-1}$, where n is the number of implemented algorithms. This represents the combined training process on each implemented algorithm.
- $M_1.predict = M_1.send(M_1.filter(M_1.bag((M_1.algo[n].predict)_{k=0}^{n-1})))$: This function generates a vector with a probability distribution or one-hot encoded prediction. It results from executing probabilistic classification with the implemented algorithms, followed by postprocessing.
- $M_1.bag$: This is a function that combines the predictions of the individual algorithms. Further explanation can be found in chapter 5.

- $M_1.filter$: This function removes non-command predictions and predictions for inactive commands, and normalizes the remaining values so they sum to one.
- $M_1.send$: This function assigns a rank to each command and sends the data to the user interface in the form of lists containing the robot's name, command name, and assigned rank.

One crucial definition that has been missing so far is that of the "Command Coupling" model. The primary objective of this model is to extract frequent item sets from the data present in D_f so that higher-level commands can be created with ease. Model M_2 encapsulates this particular model, and its definitions can be found below:

Definition 5: Specifications of the "Command Coupling" model

- M_2: This model is using a data mining approach that employs a parameterized sequence mining algorithm. Three different algorithms are available to choose from, but only one can be used at a time.
- $M_2.algo$: This is the algorithm that extracts the most common sub-sequences based on the defined parameters. It currently supports three options: $\{NOSEP, pefix\_span, QCSP\}$ and returns a list of the found sub-sequences.
- $M_2.train = M_2.predict = M2.update$: Here is a special case where the predict and train functions are defined identically.
- $M_2.update = M_2.get\_unseen(M_2.algo)$: This function updates the found sub-sequences on each new input to the corresponding sequence. It reruns the algorithm and extracts any unknown sub-sequences.
- $M_2.get\_unseen$: This function compares the sent command couples with the new input and returns the unseen couples.

The previous explanation provides a comprehensive definition of the system, which includes the possibility of exchanging, adapting, and expanding the modules for future development. An advantage of this approach is that each module can be utilized individually if there is no need for the entire set of modules or if the available bandwidth cannot support the transmission of the whole set.

In the examples above, the set elements have been defined in natural language. However, if enumeration identifiers for each type better suit the requirements of the system, this can be implemented as well. For this thesis, the human-readable definition was chosen as it is easier to explain the concept.

3.3 Summary and Comparison to other approaches

The approach described above presents a unique opportunity to apply gained knowledge from historical data directly to the user interface or robots. This approach creates a modular system that can be easily adapted to one's needs and is more effective than current systems that rely on either a rule-based ontology system or neural networks. One key advantage of this approach is that the system can create meaningful recommendations even with a lower number of training data, which is contrary to using a neural network that typically requires thousands of data points to train. However, an additional rule-based ontology would be an excellent addition to the current system.

When creating a system for robots, it's crucial to take into account the variations in operating systems, which can make it challenging to establish a universally applicable system. Moreover, using item sequence prediction as a basis for recommendations is not a common approach, as most models usually predict specific function parameters. Therefore, this approach represents a new avenue of development in this direction.

Moreover, it is worth noting that recommendation systems are typically interface-specific, as they are more commonly implemented on websites or other non-robot-specific interfaces. They are often used to make the user aware of unknown commands, and for frequently executed tasks, alternative commands are presented. These presentations include information on the specific command and example usage.

Data Mining Approaches are frequently utilized in the realm of big data, where the given system presents a rather uncommon option. Consequently, this approach provides a novel way to view run-time predictions in a robotic environment with a low run count, enabling the developer to evaluate its use. Further specifications regarding the test environment for the proposed system can be found in the forthcoming chapter. All things considered, this approach has significant potential to enhance the efficiency and effectiveness of user interaction with a graphical user interface, particularly in the context of robots and other unique operating systems.

4 Robotic System Architecture

Within the context of DLR and ESA's exploration mission, a robotic team of four robots has been set up. The aim of this setup is to control a heterogeneous robotic team from an orbiting satellite, with an astronaut situated at the ISS controlling the team on earth [18][48]. These efforts are part of the Surface Avatar Project, which is a component of the ExPeRT (Exploration, Preparation, Research, and Technology) projects under ESA's Terrae Novae European Exploration Envelope Programme [17]. The Surface Avatar Project is specifically designed to conduct robotic martian and lunar exploration campaigns.

The robotic team is composed of four robots: Justin, the humanoid robot, LAMA, the Lander Mounted Arm, Analogue-1 Interact Rover, and Bert, a quadrupled robot mimicking canine locomotion. Each robot has unique capabilities that make them an integral part of the team. In this chapter, a thorough overview of each robot and their current capabilities will be provided.

Furthermore, the collaboration scenario will be discussed that outlines the different responsibilities of each robot in the current setup. Details about the commanding UI that is currently available and used will also be provided. This information will help to provide a better understanding of the robotic team's capabilities and how they function together to achieve their mission goals.

4.1 Heterogeneous Robotic Team

4.1.1 The humanoid robot "Rollin' Justin"

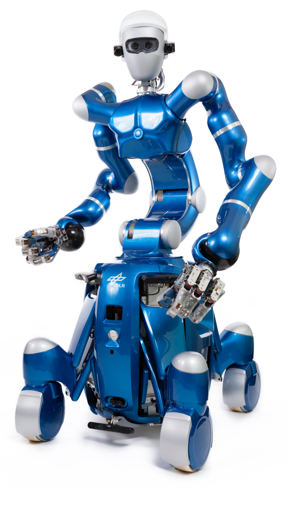


Figure 4.1 The humanoid robot Justin

Rollin' Justin is a humanoid robot that is mounted on a wheeled moving platform, as shown in figure 4.1. With 19 joints that can move the torso, arms, and head, Justin has the ability to interact with its environment in a variety of ways. The robot's head is equipped with a set of cameras that can be used to localize it within its surroundings. To aid with this, an April-Tag-based system has been implemented on the robot. This system uses visual reference markers, similar to QR codes, which are placed around the robot's environment. By recognizing these markers, Justin can determine its position in space.

The robot's reasoning and planning system is based on a two-step mechanism [44], which includes geometric and symbolic planning. When given a goal state to reach, the planning system uses Justin's internal knowledge about the world to find a feasible solution. This involves searching for a plan based on semantic knowledge of the environment. The robot's knowledge is based on a PDDL representation of all available actions it can execute. If a plan is found, it is then simulated in a 3D environment, taking into account all of the objects that have been detected. The robot will execute the plan if the simulation finishes collision and error-free, otherwise it will backtrack to the last successful part of the planning.

During execution, the robot uses a collision avoidance system based on distance-measuring sensors. If any issues occur, the robot can be manually manipulated to return to an error-free state. This can be done with the help of the robotic team or by using the implemented teleoperation systems, which allow the operator to control Justin's base and arm with a force-feedback controller.

An accurate representation of the environment is essential for the robot's planning to work effectively. This means that the robot needs to be aware of most of the objects within the environment it operates in.

Additionally, planning adaptation for dynamic objects necessary in a multi-robot environment will require an update on the planning. However, if there are no moving objects or robots within a known environment, Justin can autonomously navigate. The Surface Avatar project is working on solving these problems. As part of the METERON experiments, Justin was able to learn how to work in a solar farm environment. Here, the robot was responsible for maintaining individual solar panels and their processing units [86].

4.1.2 ESA's "Interact Rover"

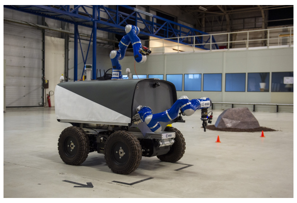


Figure 4.2 ESA's Analogue-1 Interact Rover

As an integral part of the robotic team, the Analogue-1 rover from ESA is introduces, which is shown in Figure 4.2. This rover has been developed as part of ESA's moon and Mars exploration program [16], making it perfectly equipped to navigate rough terrain and cover long distances [101][39]. The rover comes equipped with two robotic arms, one of which has a camera as the end-effector, while the other serves as a gripper. The arm-mounted camera allows the robot to adapt the given view in many ways, thus improving situational awareness. The arm with the gripper can be controlled using a force feedback system, just like Justin's arm.

The Analogue-1 rover has undergone rigorous testing in sample return missions. Notably, the rover was tested when the orbit-to-moon teleoperation was being tested on Mount Etna [19].

4.1.3 Robotic arm on Lander "Tina"

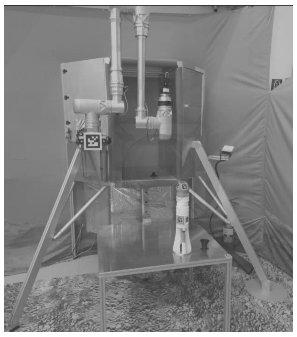


Figure 4.3 The Lander Mounted Arm (LAMA)

The Lander Mounted Arm is a crucial component of the team's site [48], serving as the primary reference point based on a static point in the camp [48]. The lander itself is not movable and acts as a router for the Wi-Fi, handling all the data transfer between the robots and the non-camp-based participants of the network.

This mounted arm is based on the TINA-system [54], a small modular torque-controlled robotic manipulator designed based on the specific requirements of the Martian mission. Its primary function is to handle all the objects stored within the lander, including sample tubes and seismometers.

To store and handle these objects effectively, the lander is equipped with an electrically powered drawer and additional storage space within the center of the lander. The objects are made available by placing them on a tray mounted in front of the lander, allowing Justin to extract them with ease.

4.1.4 Bio-inspired robot "Bert"

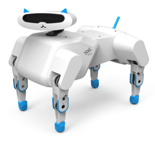


Figure 4.4 The quadrupled robot Bert

Bert's development is a work in progress and not yet ready to be implemented as a part of the team. In order to be successful, it needs to be autonomous and able to follow predefined commands without any issues. As of now, a teleoperation system is not available, which limits the possibilities for the robot. However, despite these challenges, the potential of Bert is exciting.

The project aims to explore the implementation of canine locomotion patterns on robotic systems [72], which is an interesting avenue to pursue. The spring-based actuator that Bert is equipped with should allow for energy-efficient movement, which is a crucial aspect of robotic design. Furthermore, due to its small size, it can reach areas that other robots might not be able to access.

To increase the capabilities of Bert, he will be equipped with a camera for our experiment. This will enable him to record and map small spaces like caves and caverns, which has exciting implications for exploration and research.

4.2 Collaboration Scenario

The collaboration scenarios planned for the given setup are crucial for the proper functioning and maintenance of a scientific exploration camp on Mars. This camp is primarily powered by solar panels, which ensure energy sufficiency, and the LAMA, which is responsible for handling storage and managing all the scientific equipment required for the expedition. Within the camp, Justin plays a critical role in maintaining all the objects on site. As the camp's caretaker, he is responsible for keeping the solar panels clean, updating the solar panel software, and handling all handover actions between the rover and the LAMA.

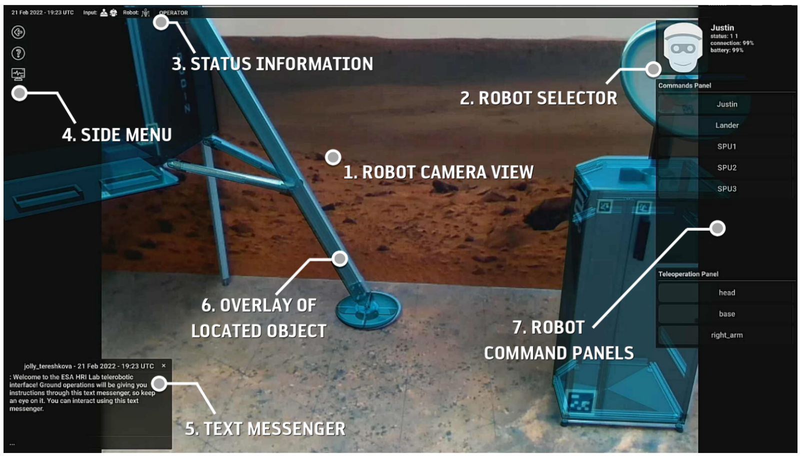
The rover's responsibilities are equally significant, mainly focusing on tasks that demand long traverses and require the collection of soil samples and other necessary objects from the surrounding environment. Additionally, the rover is capable of transporting Bert to any cave that requires exploration, making it an essential asset to the team. Bert's purpose in the team is to explore narrow areas and caves, as it is the only small robot available. Furthermore, it can serve as a Wi-Fi booster when the rover is too far from the lander to have access to the network, making it a valuable asset to the team's communication system.

4.3 User Interface

In Schmaus et al.'s work [86], the presented interface to control the previously presented robots is presented (figure 4.5b). It is based on a layout that focuses on the robot that is currently being controlled. However, the interface lacks an overview of the available robots. Figure 4.5 provides an overview of the current setup. The robots can be controlled using a set of predefined commands, provided by the available robots, listed on the right of the interface. Above the commands, the current robot is displayed, and the operator can switch to control another robot. The teleoperation options available for the current robot are located below the commands. It is important to note that this system supports the parameterization of commands, which means that for some of the commands, an operator must define the value of those parameters before sending them. These values can be either strings chosen from a set of options, strings defined by text input or numerical values. A command can also support multiple values of different types.

The main view displays the camera image sent from the robot. The operator can use the drop-down menu on the top left to switch to other robot's cameras to gain better awareness of the surrounding objects.

(a) Usual Setup of the Interface



jolly_tereshkova - 21 Feb 2022 - 19:23 UTC ×
: Welcome to the ESA HRI Lab telerobotic interface! Ground operations will be giving you instructions through this text messenger, so keep an eye on it. You can interact using this text messenger.

(b) overview of the current interface

Figure 4.5 Current State of the User Interface

An additional overview is provided in the form of a map of the currently known objects, located on the bottom left, above the chat window. The chat window enables the operator to contact the robotics team if any problems arise.

Commands can be filtered by clicking on any object in the scene. After that, only commands for the corresponding object can be viewed. To reverse this action, the operator can either click somewhere else on the screen or use the cancel button on top of the list of commands.

4.3.1 Communication System

The communication between the user interface (UI) and the other components of the system is achieved through broker software. For this purpose, the rti DDS framework [78] has been employed, which provides standard protocols for publishing, subscribing, and services in a hybrid form. The message types are defined using the Interface Description Language (IDL) format, which is specified in a file. An IDL definition example is provided below. With the message definitions, the topic name, and the transmission protocol definition, data can be transmitted between points. The transmission protocol is defined by a quality of service (QoS) definition, which specifies whether packages should be sent with best effort or if each package should be checked for delivery. Rti offers several default options for QoS definitions.

```
// Temperature data type
struct Temperature {
    // ID of the sensor sending the temperature
    string<256> sensor_id;

    // Degrees in Celsius
    int32 degrees;
};
```

In this framework, each robot shares and transmits valuable information about its current state. This information includes telemetry data, such as current joint angles, the robot's current pose, available commands, and detected objects and their poses. Once this received information is processed in the UI, it can be easily added to the screen. Additionally, the selected command for execution is sent over this framework in this system, where the robot returns a message to the UI after execution. The standardized communication protocol used in this project's scope makes it easy to add other robots given these definitions. Overall, this framework promotes efficient and effective communication among all robots involved.

4.3.2 Teleoperating robotic team on different levels of autonomy

As robots are integrated into different controlling and commanding structures, it is logical to assume that each robot operates on a slightly different level of autonomy. Furthermore, the different modalities in which one can operate the robots can involve multiple levels of autonomy as well. In particular, when it comes to the rover and Justin, there are options available to control them using commands or directly control their movements using the implemented teleoperation modalities. Command-based controlling can be seen as a form of supervised autonomy, which is considered a mid-tier level of autonomy, while teleoperation is considered one of the lowest levels of autonomy.

The teleoperation modes available for these robots include moving their arms with the help of the sigma.7 or locomotion using the joystick. By utilizing these different levels of autonomy, one can work with the robots on a finer granularity. For long and complex tasks, the robot can be instructed to execute the task autonomously, with the operator assuming a supervisory role. At the same time, it is also possible to use teleoperation commanding to fix errors, exemplifying the flexibility and versatility of these robots.

Overall, the varying levels of autonomy available for these robots allow for a more tailored approach to controlling and commanding them, catering to the specific needs of each task or situation. As technology continues to advance, it will be interesting to see how these levels of autonomy evolve and what new possibilities will arise in the realm of robotics.

5 Implementation

The previous chapters have discussed the fundamentals of the thesis and the environment setup. In this chapter, the implementation of the given concept in the presented environment will be delved into. To provide a clear understanding, an overview of all topics and their connection will be provided. Following this, a detailed description of each part will cover the logging of historical data, necessary preprocessing steps, support systems currently at play, and post-processing of data.

It is important to note that the data itself is insufficient to help users understand the interface and deal with it effectively. Therefore, the implementations made on the UI to visualize the given data will be discussed. This will include how one could interact with the data and handle it in the UI.

In addition to the recommender system implementations, the implementations for simulating the real robots will also be explained. These implementations have been used to conduct the user study, as the actual system is still under development and only available with limits. Moving forward, the next chapter will then focus on the user study itself and the evaluation of it.

5.1 Distinct Parts of the Recommendation System

As described in chapter 3, the recommender system is developed in a modular way, consisting of a preprocess, filter, classify, and post-process step. Those steps must be executed for each implemented model, where, ideally, more than one model can use the given processing steps. The models implemented for this thesis include creating a probability distribution for the available commands and a system to couple multiple commands into one higher-level command. The prioritization should ease the navigation on the

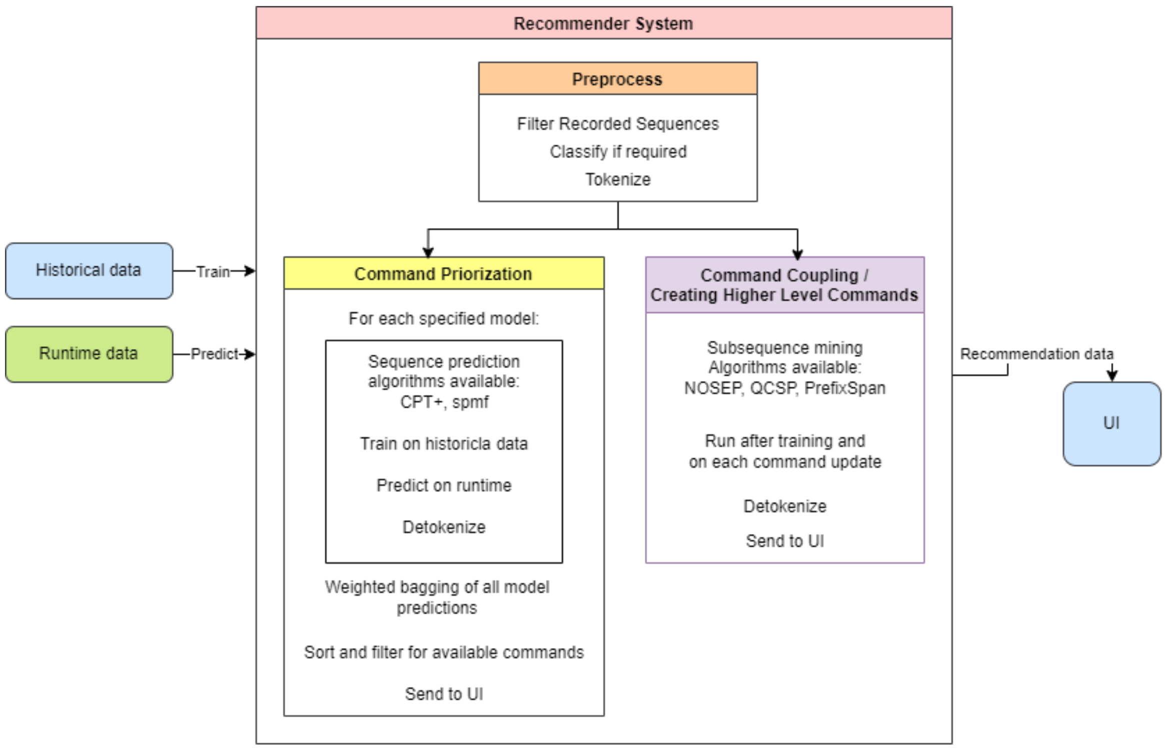
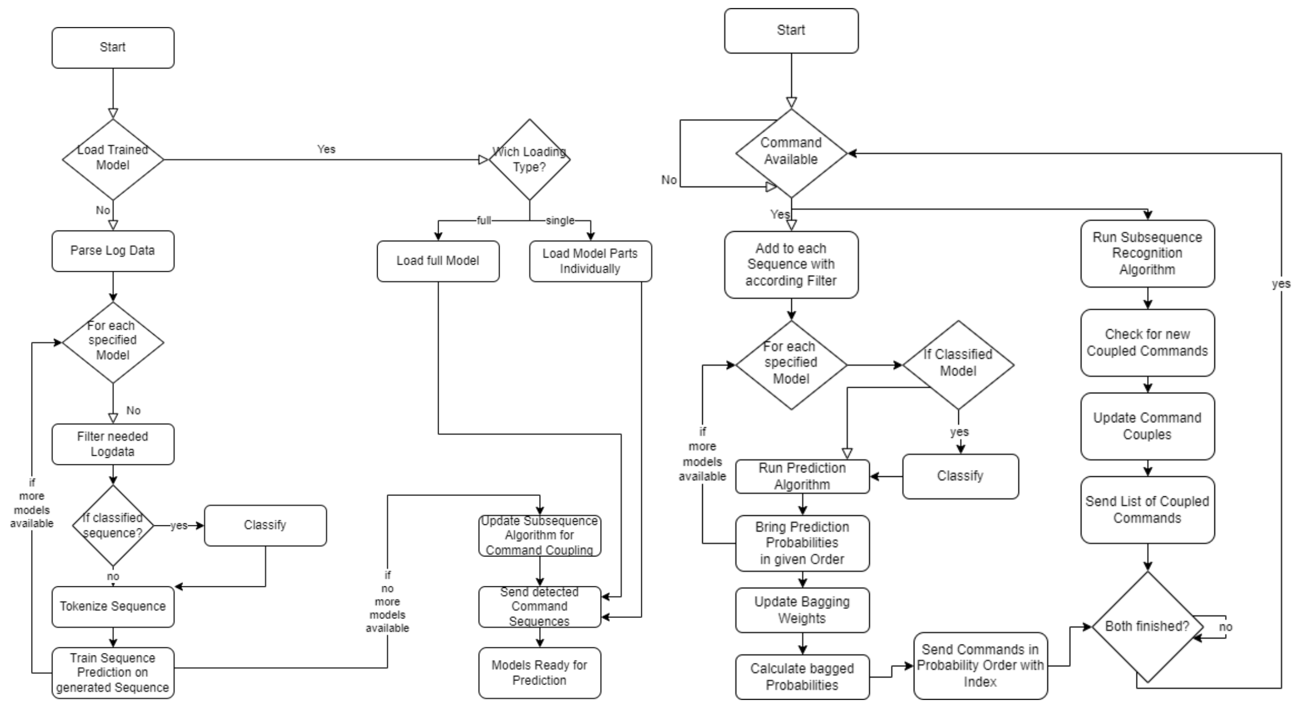

Recommender System
Preprocess
Filter Recorded Sequences
Classify if required
Tokenize
Historical data
Train
Runtime data
Predict
Command Priorization
For each specified model:
Sequence prediction algorithms available: CPT+, spmf
Train on historicla data
Predict on runtime
Detokenize
Weighted bagging of all model predictions
Sort and filter for available commands
Send to UI
Command Coupling / Creating Higher Level Commands
Subsequence mining Algorithms available: NOSEP, QCSP, PrefixSpan
Run after training and on each command update
Detokenize
Send to UI
Recommendation data
UI

Figure 5.1 Basic Structure of full proposed System


Start
Load Trained Model
Yes
No
Wich Loading Type?
full
single
Load full Model
Load Model Parts Individually
Parse Log Data
For each specified Model
No
Filter needed Logdata
If classified sequence?
yes
Classify
no
Tokenize Sequence
Train Sequence Prediction on generated Sequence
if more models available
if no more models available
Update Subsequence Algorithm for Command Coupling
Send detected Command Sequences
Models Ready for Prediction
Start
Command Available
No
Yes
Add to each Sequence with according Filter
For each specified Model
If Classified Model
yes
Classify
Run Prediction Algorithm
Bring Prediction Probabilities in given Order
Update Bagging Weights
Calculate bagged Probabilities
Send Commands in Probability Order with Index
if more models available
Run Subsequence Recognition Algorithm
Check for new Coupled Commands
Update Command Couples
Send List of Coupled Commands
Both finished?
no
yes

(a) Flowchart of training/loading prediction models **(b)** Flowchart of run-time prediction generation

Figure 5.2 Flowcharts of the prediction models

interface, while the coupling should allow for faster command execution. The recommender system, as described in chapter 3, is developed in a modular way, consisting of several steps. These steps include preprocess, filter, classify, and post-process and must be executed for each implemented model. Ideally, more than one model can use the given processing steps. The models implemented for this thesis include creating a probability distribution for the available commands and a system to couple multiple commands into one higher-level command. The prioritization should ease navigation on the interface, while the coupling should allow for faster command execution.

To provide a quick overview of the system, the recommendation system takes input from logged historical data for the training phase and run-time data from the communication system on run-time, as shown in 5.1. This data is preprocessed within the recommender system by filtering, classifying, and then tokenization. The preprocessed sequences are then passed to both models, command prioritization and command coupling. The recommendation system is based on a Python program, which is suitable for fast implementation.

The implementation of the recommender system involves the installation of a bagging model for command prioritization, comprising as many individual prediction models as needed. These models are trained on preprocessed sequences, and users can choose from two implemented algorithms - the CPT+ algorithm or the Subseq algorithm. The Command Coupling algorithm extracts relevant sub-sequences based on predefined parameters, and users can select from three algorithms - NOSEP, QCSP, and PrefixSpan. The extracted data is processed and sent back to the user interface.

Now that a basic overview of the implementation is given, the next part goes over how the processing for the training and run-time works. These processes are also shown in figure 5.2. Each time the recommender system models are trained, the models will be saved to a binary file at the end of execution. They can be loaded individually or as the complete recommender system on a later run.

In order to improve the functionality of the training phase, an option called "Load Model" has been added, which allows for both options to be utilized. However, once the models have been saved, it is not possible to fine-tune them. If these options are not chosen, the recorded data is extracted from the log file and filtered for each specific sequence of event tokens. These event tokens represent distinct events and are equal to the concept's outcome of mapping Pre, therefore being a part of the set E_p. Each model is then trained on the corresponding set of sequences. In the case of a classified model, the sequence is translated to a class sequence, which creates an output for unknown commands that do not appear in the training data. After the event tokens are assigned an identifying number or class number, they are ready for further processing.

Once the sequences are ready, they are transferred to the command prioritization and command coupling algorithms, where those algorithms will be updated and sent. During run-time, the sequence of events is recorded, and every update is given to the sequence prediction algorithm through the communication system. This update triggers an event for both the command prioritization and the command coupling. For the command coupling, this will execute another update on the sequence, where the algorithm reiterates the sequences and checks for any new couples. If there are any, those will be sent to the UI. The event or command is added to any corresponding sequence for prioritization.

In the next step, the calculation and combination of the prediction for each model are performed. The commands are sorted according to their prediction and then sent to the UI. Overall, this process allows for the efficient and effective utilization of the recommender system, improving user experience and satisfaction. An overview of those processes can be viewed in figure 5.2

Overall, the implementation of the recommender system involves several distinct parts, each with its unique algorithms and processes. However, the system's ultimate goal is to provide users with an efficient and reliable recommendation system for their commands.

5.2 Recording History of User Data

As the initial stage of the recommendation system, this report will present the implementation of the logging system for the historical data. As the proposed system is expected to support robots independent of their reasoning system or working framework, the logging system should be implemented on top of the UI, a shared world representation, or directly on the communication system between the two. However, due to the unavailability of a shared world representation at the time of this thesis, the logging system was implemented directly on the UI. This gives the additional advantage of also having the sent data available as processed in the UI. As the UI is based on the Qt framework, the scripting language was C++ and qml.

In most frameworks, the logging rate is part of the logging definition. Logging all data in the published rate would result in much useless data. Therefore, the logging rate should be explicitly considered if it is set up on the communication system itself. This setup should provide most of the useful information to the user, or at least the fraction of the data that is helpful to train prediction models for maximum accuracy.

For the current setup, an event-logging style has been chosen, as it vastly reduces the amount of data logged. The developer can define a set of distinguishable events that provide the developer with crucial information about the ongoing changes in the robot's world. These events can be defined for each individual robot and in conjunction with all existing robots in the current environment. Then, on each occurrence of the corresponding event, it will be added to the logged sequence.

The log of events that is maintained includes comprehensive data about all incoming and outgoing chat messages, object data, as well as communication between the robots and the operator. Those specifically chosen events are logged for the purpose of covering most of the current information that the UI can access. This includes objects, their positions, and the available actions and movements within the environment. In addition to this, the integration of chat messages is enabling natural language processing, which will allow for the specification of possible missions and their commands in future work. This module could also be used for command prioritization model algorithms.

The log file is constantly updated with object-specific data, including the object definition itself, available commands, and positioning, which enables developers to define specifications about the environment within which the robots are operating. Furthermore, the commanding log provides information about how the robots interact with the world based on the sent commands and their response, allowing for evaluation and improvement of their performance.

Table 5.1 Sample Log File

Time	Participants	Source	Event	Type	Parameter	Message
20:03:39	104	Justin	Add Frame		[1 0 0 4.050000190734863 0 1 0	
20:03:39	101	Justin	Add Frame		[1 0 0 7.719353675842285 0 1 0	
20:03:39	2	rov1	Add Frame		[1 0 0 1.4186742305755615 0 1	
20:03:39	1	Justin	Add Frame		[0.7071067690849304 -0.70710	
20:03:39	100	Justin	Add Frame		[0.6131675839424133 -0.29828	
20:03:41	Operator UI	Justin	Add Object	[rollin_justin _robot]	id: 1	
20:03:41	Operator UI	Justin	Add Object	[floor]	id: 104	
20:03:41	Operator UI	Justin	Add Object	[robex_rodin _object]	id: 101	
20:03:41	Operator	OperatorUI	Current Robot Changed	Justin	1	
20:03:41	Operator UI	Justin	Robot Added	Justin		
20:03:41	Justin	Operator	Send Command	request_command Justin	inputs: [joystick sigma]	
20:03:41	[1]	Justin	Add Command	request_command Justin		
20:03:41	[1]	Justin	Add Command	move left		
20:03:41	[1]	Justin	Add Command	move right		
20:03:41	[1]	Justin	Add Command	Move XY	[\{p_default_value: 9 p_name: ta	
20:03:41	[1]	Justin	Add Command	Look around	[\{p_default_value: 0.5 p_max: 0	
20:03:41	[1]	Justin	Add Command	Deactivate		
20:03:41	Operator UI	rov1	Add Object	[rover _robot]	id: 2	
20:03:41	Operator UI	rov1	Add Robot	rov1		
20:03:41	[2]	rov1	Add Command	Disable		
20:03:41	[2]	rov1	Add Command	request_command rov1		
20:03:41	[2]	rov1	Add Command	Do Something		
20:03:43	Justin	Justin	Command Reply	request_command Justin	true	\{joystick: \{42: \{g
20:04:00	Justin	Operator	Send Command	move left		
20:04:00	Justin	Justin	Command Reply	move left	true	
20:04:01	1	Justin	Update Frame		[0.7071067690849304 -0.70710	
20:04:20	Interact	Operator	Send Command	request_command rov1	inputs: [joystick sigma]	
20:04:20	Operator	OperatorUI	Current Robot Changed	rov1	2	
20:04:20	rov1	rov1	Command Reply	request_command rov1	true	\{joystick: \{42: \{g
20:04:24	rov1	Operator	Send Command	Do Something		
20:04:24	rov1	rov1	Command Reply	Do Something	true	

Data is also stored about the occurrences when the robots are directly controlled via teleoperation, depending on the active mode. For each of these events, a basic structure is used to cover essential data, which includes the time of the event, participants (which can be logged by name or by object ID), the type and sub-type of the event, as well as specified parameters and a message in connection to the event.

By maintaining such a comprehensive log of events, the aim is to ensure that all the required information is available to optimize the performance of the recommender system and improve its overall functionality.

An event-based history logging system example is available in table 5.1. The log files are stored in csv-format, which is cost-effective in terms of storage and easy to read. Another advantage is that the file is human-readable, and information can be directly extracted from those files. Here is an overview of the events logged and their names. The events of the logged data E_l are formally defined as follows based on the concept:

- P = {"Justin," "Interact," "LAMA," "Bert," "Operator," "DIP," "Seismometer," "SPU1," "SPU2," "SPU3," "VSU," "Lever1", "Lever2", Lever3"}
  - This encompasses all the available robots and objects, with each object having an ID that can be used to identify it. Additionally, an operator variable has been included, primarily for UI interactions that do not involve robots or objects. Since not all objects are pertinent to this thesis, only those within the state machine described later will be explained in greater depth.
- S = {"Justin," "Interact," "LAMA," "Bert," "Operator"}
  - This shows where the event originates from. Hence, it only includes participants who can actively interact with the environment
- T = {"Remove Object," "Remove Command," "Remove Robot," "Add Frame," "Add Object," "Add Command," "Add Robot," "Send Command," "Command Reply," "Enable Teleop Mode," "Disable Teleop Mode," "Configure Gripper," "Current Robot Changed," "Update Frame," "Send Chat," Receive Chat"}
- T_s - This includes all the subcategories of event types, which cannot be fully listed here due to their length. The list also varies depending on the commands programmed in the robots, so that it may differ greatly for different systems.

Additionally, to understand the meaning of the logged events, here is a short overview of them. Some have been coupled together due to their similarity.

- **Add {Object, Command, Robot}** - Add the corresponding object to the active set
- **Remove {Object, Command, Robot}** - Remove the corresponding object from the active set
- **Add Frame** - Store the first transformation matrix of the specified object
- **Update Frame** - Update the transformation matrix of an object if the distance is more than 20cm. Rotation deviation is not yet considered
- **Send command** - Logged on execution request for a currently active command
- **Command Reply** - Logged on finished execution of a command with its execution message
- **Enable Teleop Mode** - Start one of the teleoperation modes implemented on the robots
- **Disable Teleop Mode** - End teleoperation
- **Current Robot Changed** - Change the robot in the robot select widget in the top right corner of the user interface
- **Send/Receive Chat** - Log the chat messages

5.2.1 Pre-processing

With the options for historical data now defined, this section aims to explain the process through which the data will be made useful for the rest of the system. As per the concept, the process begins with mapping the log data to make it processable by the mapping Pre. This mapping takes events from the log E_l and maps them to E_p. For the purpose of implementing this thesis, only the command data, including the "Add Command," "Remove Command," and "Send Command" events, have been processed for the models. The "Command Reply" has been omitted during this feasibility study but can be added at any given point. Thus, the mapping only includes those events where the exact mapping is:

```
(p in E_p)
for e in E_l:
    p(i) = size(E_p)
    p(t) = "Command"
    p(d) = e(t) + e(s)
    p(c) = classify(e(t))
```

In analyzing the mapping of events, it is evident that the ID of the processed event is always set equivalent to the previous number of already processed events, making it unique. The type of event is specified as "Command," which is the only type of processed event in the course of this thesis. The defining representation is a combination of the event type and the source of the event. This representation corresponds to the command name and executing robot in the log files. The classify function will be explained in detail in an upcoming paragraph.

During the parsing of the logged files, the mapping is executed when reading the data from the files, with the data loaded using the pandas library. The file is first searched for logged "Add command" events, which are then added to E_f in the given form. Through NLP classification, the command name is classified as one of seven options. As the data is already filtered for the command events, the need to process all data is skipped by not previously calculating E_p. The file is then filtered for "Send command" events, where the sequence $s \in S_f$ is extracted. This process is executed for each available file, resulting in a set of sequences defining D_f. Due to the classification of the command, there are already two data sets available for training the models, as the sequences of IDs and the sequences of classes can be used. The execution of the filtering as a first step is an advantage in the current project, as there are not many different models available, and those available are working on the same subset of events. Hence, setting the filter as the first step makes the process faster. This is mentioned as id deviates from the proposed concept.

It is worth noting that, as the commands cannot be processed the same way on run-time, the sequence will be recorded from the communication system. The data is sent via a service of the framework DDS under the topic name "surfa.execute_command." It works on the following message definition.

```
module ExecuteCommandService{
    struct RobotCommandRequest {
        // Request for command execution
        string<255>    robot_id;
        uint           command_id;
        uint           user_id;
        string<255>    parameters;
    };
    struct RobotCommandReply {
        //reply after command execution
        bool           success;
        string<255>    message;
    };
};
```

In order to retrieve the command name in this scenario, one must iterate through a list of active commands, as the command name is currently unavailable in the definition. These active commands can be accessed by parsing the information published on "world_representation.commands" using the message definition provided. The data from this topic is constantly updated within the recommendation system and is stored in an active command dictionary that references the corresponding robot and command name for each command_id. It is important to note that this command_id is different from the ID specified in the recommendation system. The robot name can be extracted from the robot_id, which is already provided as the robot's name. The command name can be found in the text field of the command. By utilizing this dictionary, the commands can be added to the currently recorded sequence of events on run-time.

```
struct Command{
    uint                    id;
    string<255>             text;
    sequence<uint, 100>     object_ids;
    string<255>             parameters;
};
struct Commands{
    string<255>             robot_id;
    sequence<Command,100>   commands;
};
```

In the upcoming study, the robot will operate in an error-free environment, making the sequence of commands issued sufficient for model training. However, if the system were to be applied to actual robots, it would be beneficial to include a "Command Reply" in the sequence. This would improve predictions as they would also consider the outcome of command execution. Since the models were trained on artificially generated data in a simulated environment, no errors were possible. This implementation can also assist in error handling, although this is beyond the scope of the thesis.

Classification The classification of commands has been based on a Natural Language Processing (NLP) model. To process natural language, transformers have been deemed highly effective in previous studies. Therefore, a pre-trained model has been used in this study. The NLP model has only been utilized as an embedding system, as insufficient data was available to fine-tune a model for the needs of this study. The model that was chosen for this study is the 'paraphrase-albert-small-v2' model, which is part of the sentence-transformers library and is licensed under the Apache 2.0 license [22][73]. This model is relatively small in size and has the ability to embed phrases of words. By using the cosine distance, the model can map semantically similar phrases close to each other. Each phrase is mapped to a multidimensional space with its embedding, which makes it easier to distinguish between different commands.

To classify each command, a k nearest neighbor classifier has been used. The scikit-learn implementation of the algorithm has been employed for this purpose [66]. In order to train the model, a 50-entry data-set has been used. This data set covers roughly seven to eight entries per class. Although this data-set needs to be bigger to enable a more robust classification of each possible command, it was sufficient for the limited amount of commands the system is currently working with. However, training the model on a more extensive data set is highly recommended for a less restrictive environment. Additionally, the classification quality still needs to be verified, as its performance has only been checked on several occasions. Due to the number of data-points, the classification is done on the two nearest neighbors. The commands are classified into one of seven distinct classes, which are presented in the following.

- **Prepare** - Prepare the environment for another action or command
- **Relocation** - Moves the robot to another position or pose
- **Visual Alignment** - Realigns the camera of the robot

- **Capture** - Acquiring an object, mostly by picking it up
- **Release** - Part with an object, put it down
- **Maintenance** - Keep the objects in the environment functional by cleaning or updating them
- **Finish** - Bring the robot or the environmental objects in a standardized configuration after finishing a task

The classes that were selected were based on the command classes that were defined in Roldan et al. [76]. These classes were slightly modified to better fit the project's requirements. In the current environment, they serve as an ideal introductory assignment for new commands, as their purpose is clearly captured through their class definitions. The chosen classes are, therefore, instrumental in ensuring that new commands are comprehensible and can be implemented effectively.

5.2.2 Command Prioritization

The Command Prioritization process is a critical component of the system, and it is important to understand how it works. The data is processed in a way that allows it to be passed to the different models. The prioritization was set up using a bootstrap aggregating approach, which means that multiple algorithms are implemented in this approach. A handler was created for each of these algorithms, which has functions for dealing with both the training and prediction phases.

During the training phase, the previously processed sequences are passed to the handlers for training, and the resulting models are saved within the handlers. In the prediction phase, the functions are passed to the currently recorded sequence of events, which is then given to the algorithms for the prediction calculation. As the different algorithms return results in different forms, the results are then processed to return a two-column pandas DataFrame with a list of the command IDs and the scores. The scores are also normalized to sum to one.

The handlers have a set of internal parameters for each of the predictors, which include the sequence type, such as "Command" or "Command Class" for the current system. However, this can be extended with many more types of sequences in future development. Additionally, most algorithms offer some parameterization, which can be set when adding the algorithm and will default to some predefined values if not. Currently, the CPT+ and Subseq algorithms are available, which have been chosen as they allow for predictions even based on a minimal training set. These algorithms have proven to be effective and efficient in the current system, but future development may incorporate additional algorithms to improve accuracy and performance.

CPT+ This algorithm has been implemented with the help of the Python wrapper developed for the spmf library by Professor Fournier-Viger of Shenzhen University [23]. The spmf library is a powerful data-mining tool that offers a wide range of algorithms for sequence mining and prediction, among other things. However, since CPT+ has not been implemented as part of the wrapper, the Java library has been adapted to execute it from there. This process involved parameterizing the algorithm with a set of parameters that were copied from the internal example usage script provided within the library. It is important to note that for more accurate prediction, these parameters should be further adapted.

The wrapper typically did not offer a separate training and predictions phase, so the adaptations in the library included two options. For the training phase, the library is presented with the training data and saves the resulting model in a binary file. This file is then referenced during prediction in combination with the current sequence of the given sequence type. To improve ease of use, the wrapper was modified slightly by adding another sequence type, which enables the user to input the pure sequence without putting each item into a single item item-set prior to prediction.

It is worth noting that after prediction, no further processing other than normalizing is required. In summary, the implementation of CPT+ using the spmf library, although not straightforward, has been made possible by adapting the Java library to execute it.

Subseq The algorithm used in this implementation is written in C++ with a Python wrapper, and it is based on the research paper titled "Succinct BWT-Based Sequence Prediction" by Ktistakis et al. (2019). The implementation process using the wrapper is simple and only requires one parameter to adjust. This parameter sets the number of similar queries for the sequence to confidently predict the next command. The default threshold is set to one, which is essential given limited training data. However, the algorithm does not return the calculated probabilities. Instead, it classifies the item based on the highest probability or returns a list of the top-k items. Since the top-k option fits better for the given purpose, it was chosen. K is set to the highest ID in the sequence, which might not be ideal for a large data-set but works here. If the predictions do not reach the threshold for each top-k item, the returned sequence will only contain the items above the threshold, which could be less than k items. Next, the item sequence gets rated from 1 to n, where n is the number of returned items. These predictions are then normalized to sum to one and returned. However, there are better solutions to this, as the actual probabilities rarely match a linear distribution. Therefore, in future versions, the wrapper will be adapted to find a new option.

Bagging To optimize the prediction accuracy of this system, various algorithms can be implemented and hence need to be combined afterward. Based on bibliographic research, the two most prevalent options for combining different classifications are priority selection and bagging. However, for this thesis, the bagging method was chosen as it is uncertain which algorithm would work better under which circumstances. Moreover, it would require a lot of data to train a model for extracting that knowledge. For the bagging method, a weighted option was chosen where the weights are updated on each iteration based on the overall performance of each model.

Initially, the implementation of weight had been conducted on each command individually. This had a harmful effect on the prediction of highly rated commands. Therefore, the model score for each model's complete set of prediction values was used as a weighting. For each item in the combined training and sequence data, algorithms place a prediction. The "Command" sequence type contains the IDs of the command events, while the "Command Class" contains the class identifying numbers.

As predictions usually only contain probabilities for a subset of available items, they are reordered in a vector based on those values for easier processing. This means that a command's distinct probability is mapped to its ID's index. This vector is called the forecast vector and is returned by each model. The outcome, meaning the actual following command, is mapped to the corresponding command index in a vector of the same size in a one-hot encoding. This vector is called the outcome vector. Using these vectors and their size N, the model score can be calculated with the Brier Score.

$$\frac{1}{N} \times \sum_{t=0}^{N-1} (f_t - o_t)^2$$

The Brier score measures the accuracy of a probability distribution by calculating the quadratic error between the actual outcome and the predicted outcome, which ranges from zero to one. A score close to zero indicates a highly accurate model, while a score close to one indicates poor performance. The system implements this calculation in vector form using the following equation.

$$\frac{1}{N} \times (f - 0)^T \times (f - o)$$

The Brier score of each model will be gathered in a vector called s. To prioritize the models that perform well, the scores will be inverted to create the weights w. However, the weights may not add up to one, so the vector needs to be normalized to ensure it still represents a probability distribution for the models' performances. This step is crucial when the output is supposed to be a probability distribution.

$$w_c = \frac{1 - s}{\sum_{i=0}^{N}(1 - s_i)}$$

The variable w_c represents the weight distribution for the current iteration. Since the models may not perform equally well with each update, the weights from the previous runs are combined with the current weights using the following equation. This results in the final weights used to combine the model predictions. The parameter "t" represents the number of updates done at the beginning of the update. The weights are initialized with an equal distribution before the first update.

$$w_{t+1} = \frac{(w_t \times t) + w_c}{t+1}$$

$$t_0 = 0$$

Using this implementation, the recommender system will improve its predictions by giving more weight to models with the most accurate predictions in the final combined prediction. After this, the only remaining step is post-processing the data to prepare it for sending to the UI.

5.2.3 Command Couples

Implementing the Command Couples is based on sequence mining, a part of data mining. Here, the aim is to extract high support sequences from a data-set. Due to that reason, it seems perfect to extract sub-sequences from the data to create a simpler way of interacting with the robots. As this field has many options for extracting those sub-sequences, multiple options have been added for the algorithm here. Other than for the Command Prioritization, one can choose only one option here. This is mainly because the different options serve a similar purpose, and there is probably not much help in using multiple models as no good measure of how to combine the outputs of the different algorithms is available. Here, the models are presented with a set of sequences containing the sequence of commands. The class sequences cannot produce meaningful data here. Hence, they are omitted. It is also to mention that a couple of "helper commands" exist within the commands. Those are not manually executed but mainly serve as a data request from the UI to the robots when they are chosen or parameterized differently. Those commands will also be filtered out, such that the sequences only contain data from manually executable commands. Presented this data, the chosen algorithm extracts the sub-sequences based on the parameters set in the model. The output then contains a list of the support and the corresponding sequence for each detected sub-sequence of the training data. Most of the models offer a top-k or support-based output. The implemented algorithms include NOSEP, QCSP, and PrefixSpan.

The implementation of Command Couples involves sequence mining, which is a component of data mining. The objective is to extract high support sequences from a given data-set. This is done to simplify interactions with robots by extracting sub-sequences from the data. There are multiple algorithms available for extracting these sub-sequences and the proposed system provides various options. However, for Command Prioritization, only one option can be selected as all options serve a similar purpose. Combining outputs from different algorithms is not feasible due to the absence of a good measure. The models are presented with a set of sequences containing the sequence of commands and only manually executable commands are included. Helper commands are filtered out as they serve as data requests from the UI to the robots and will be automatically executed anyways. The chosen algorithm extracts the sub-sequences based on the model parameters and outputs a list of the support and the corresponding sequence for each detected sub-sequence. Most models offer a top-k or support-based output and the implemented algorithms include NOSEP, QCSP, and PrefixSpan.

NOSEP This algorithm, known as Nonoverlapping Sequence Pattern Mining With Gap Constraints [102], is part of the spmf library, just like CPT+. Its title indicates that the main constraint for subsequence extraction is that sub-sequences should not overlap. It also incorporates a gap constraint, which allows for cohesive sub-sequences with a specific gap length, specified in the number of items. The minimum and maximum size of this gap can be set as parameters. Other parameters include the minimum and maximum length of the extracted subsequences and the minimum support. All sub-sequences will be presented with their support, and the algorithm can detect a subsequence multiple times in one training sequence. It does not offer a top-k variant. This algorithm can be useful for presenting the operator with a set of couples that only contain each task sequence once, as overlapping is not allowed. However, parameterization should be specific to the available training data, as the support number and gap constraints necessary for extracting the most valuable sequences may vary. Additionally, some sequences presented may contain impractical combinations of command sequences, due to the non-overlapping constraint. This incapacitates detecting the individual sequences, even if they might be useful.

QCSP The spmf library also includes the Quantile-based Cohesive Sequential Patterns algorithm. Hence, it has been implemented alike. Unlike NOSEP, this algorithm does not view overlapping sub-sequences as a constraint. However, it does prioritize cohesive sub-sequences by using the alpha threshold parameter. This parameter ensures that the containing sequence of a detected sub-sequence is not longer than alpha times the number of items in the detected sub-sequence. For an alpha value of one, the sequence is only counted if there are no gaps between the items. As a result, this algorithm excels in cohesiveness.

In contrast to other algorithms, this one detects all possible sub-sequences, including overlapping ones. It also offers parameters for maximum sub-sequence length, minimum support, and top-k sequence extraction. Hence, it operates on a top-k approach and even allows for multi-detection of a sub-sequence in a single training sequence.

PrefixSpan The PrefixSpan algorithm [67] is the only Python native library available here. It includes the standard implementation of the PrefixSpan algorithm, as well as the BIDE [99] and FEAT algorithm [24]. BIDE is capable of detecting closed sequential patterns, while FEAT defines generator sequential patterns. A closed sequential pattern is the longest pattern detected within the data at the same support. This implies that every detected pattern with the same support that is strictly included in another detected pattern will not be part of the output. In contrast, a generator sequential pattern is the exact opposite. Each pattern containing another will not be included in the output for each subsequence detected with the same support. The algorithm used can be defined by setting the parameters 'closed' or 'generator.' If 'closed' is set to true, BIDE will be executed. If 'generator' is set to true, FEAT will be executed. If both are false, PrefixSpan will be used. All of these algorithms work with the definition of the parameters support, minimum, and maximum length. Given these parameters, the sub-sequences will be extracted, and cohesion is not considered. Additionally, it can detect a pattern once per sequence in the training set. Therefore, it is mainly suitable for data-sets where each task is recorded independently.

5.2.4 Post-processing

After gathering the necessary data, the next step is to make it processable for the UI. This involves preparing it for sending by converting it to a Dynamic Data object using the Python dds-framework for rti [78]. The message definitions are detailed below. Due to the size of the message, the couples will be sent one after another if new data is available, while the prioritization values will be sent continuously in a single package. To ensure accuracy, the sequences' length ('len') was added to the given message definitions. This was necessary due to a faulty implementation of the reader in the rti library, which tends to overwrite packages without first deleting the previous message. As a result, new sequences may be created if the new message has a shorter sequence than the previous one since it appends the remainder of the previous message to the current one.

```
struct Recommendation{
    sequence<string<255>,100>   robot_id;
    sequence<short,100>         command_id;
    sequence<short,100>         placing;
    short                       len;
};
```

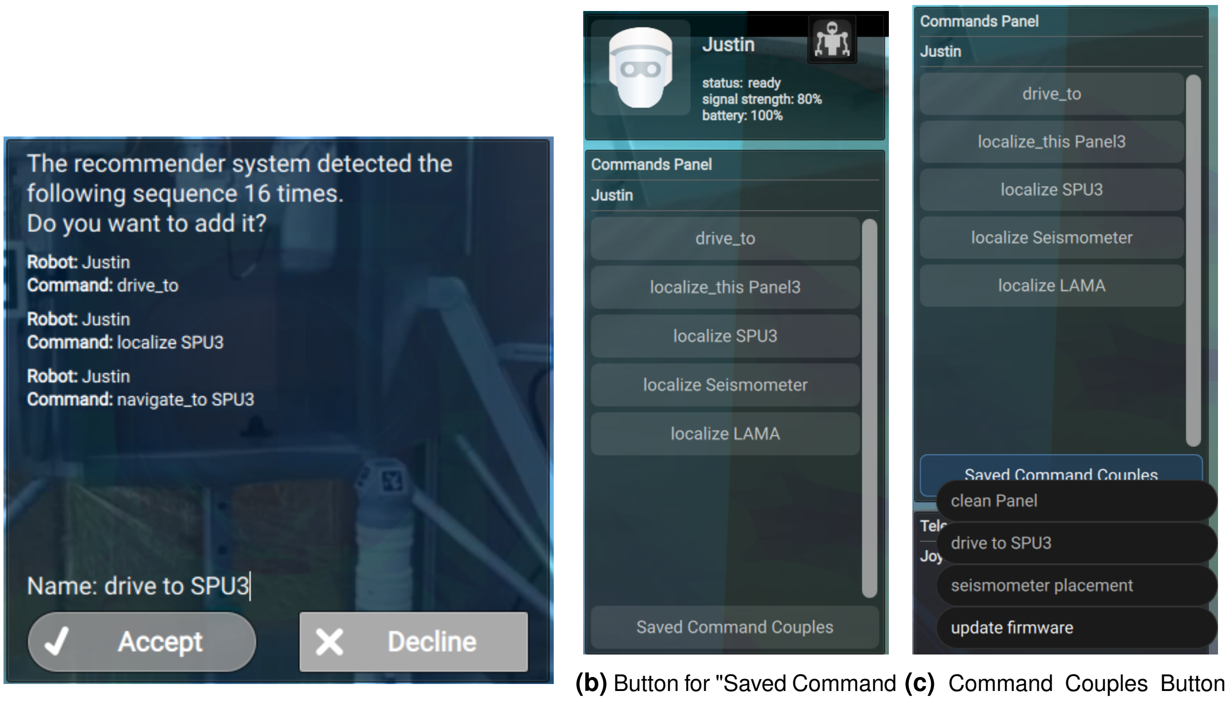
```
struct Coupled{
    short                       support;
    sequence<string<255>,100>   robot_id;
    sequence<string<255>,100>   command_id;
    short                       len;
};
```

5.3 Handling in UI

In order to effectively handle the data in the UI, certain functions have been implemented for both systems. These functions include message parsing for the new message types and functions that enable the usage of the newly acquired data. It is important to note that the parsing of messages will only occur if the corresponding data is sent, and the UI will still function as it did before without utilizing these new functions in this case. Overall, the addition of these functions will allow for a more streamlined and efficient handling of data within the UI.

5.3.1 Command Prioritization

In the current version, the Prioritization command has been added to the UI with the purpose of realigning the available commands based on their rating. This means that commands with the highest rating will now appear at the top of the list, while those with a lower rating will be placed towards the bottom. For any command without a rating value assigned, it will be added to the bottom of the list in alphabetical order. To facilitate this realignment, the already implemented sort function has been adapted accordingly. These changes were made with the aim of improving the user experience and making it easier to access the most important commands quickly.



The recommender system detected the following sequence 16 times.
Do you want to add it?

Robot: Justin
Command: drive_to

Robot: Justin
Command: localize SPU3

Robot: Justin
Command: navigate_to SPU3

Name: drive to SPU3

(a) Pop-up to add Command Couple Suggestions **(b)** Button for "Saved Command Couples" **(c)** Command Couples Button extended

Figure 5.3 Adding Command Couples and Executing them

5.3.2 Command Couples

The modifications made to facilitate the Command Couples have been more extensive, as it was necessary to enable the addition and execution of generated couples. When messages containing identified couples are received, a pop-up will be displayed on the UI, as depicted in figure 5.3a. This pop-up notifies the operator that a particular sequence of commands has been detected a certain number of times. Should the operator find the sequence useful, they can add it by giving it a name and accepting it. Conversely, if deemed unnecessary, they can decline it.

However, if the operator fails to specify a name, the sequence will be added under the name '...', which may become unmanageable for multiple cases with the same name. Therefore, a name management tool should be implemented to address this issue.

After the couples have been added, a newly added button at the bottom of the command list, as shown in figure 5.3b, provides access to them. Clicking on this panel displays the current list of executable couples, as seen in figure 5.3c. The list is filtered to show only the available sequences by checking for the availability of the first command of the sequence.

When executing a command couple, a status indicator, as shown in figure 5.4, will appear on the top left of the UI. The indicator will show the icon of the currently active robot, along with a progress bar located underneath. The progress bar will display a small bar for each command in the sequence. The bar corresponding to the currently executed command will appear in yellow and turn green once the command is finished. All other commands will be presented in a light grey color. To stop the execution of the current sequence, simply press the small x located in the top right corner of the indicator. This way, progress can be tracked and commands can be managed with ease.

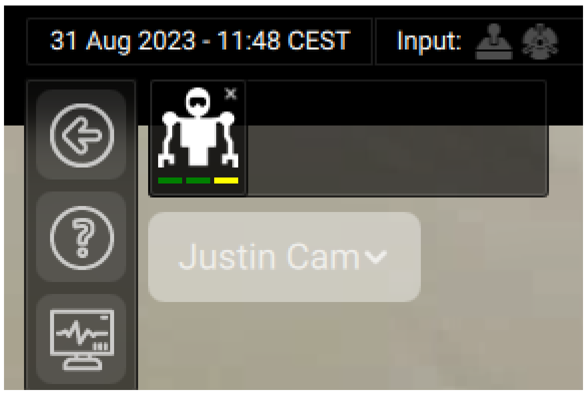
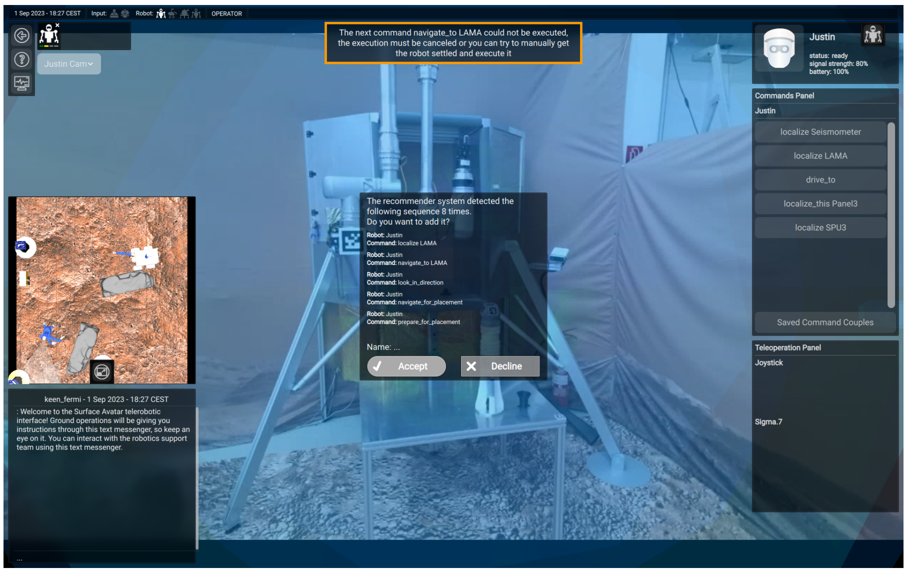


Figure 5.4 Progress Icon for Command Couple Handler

The command handler operating in the back of the system is capable of operating with multiple sequences simultaneously. However, this requires some adaptations to the current UI, as it can only handle the execution of one command at a time. The reason for this implementation is that the efficiency of multi-robot control would improve if the robots could execute commands simultaneously. To allow for the execution of multiple command sequences, an indicator is added to the right of the current one for each command couple added to execution. When the operator initiates the execution of a sequence, the first command begins immediately as soon as the handler is added. Once the first command has been executed, the robot will reply with the "Command Reply" message. At this point, the handler will check which currently active handler has been executing this command and will try to start the following command. If the command is unavailable at that time, the handler will try to start the command multiple times within the next 2.5 seconds. This adaptation is necessary in case of a bad connection on some occasions where the available commands are not updated on time. If the command remains unavailable within this time frame, the indicator will start blinking, indicating that an interaction with the handler is necessary. When the operator clicks on the indicator, the handler tries to send the command one last time and outputs an error message stating that the requested command is not available if it is not. This error message will be presented in the top middle of the screen, as shown in figure 5.5. The progress bar of the currently active command will also turn red to indicate the event. Upon receiving this message, the operator can choose to continue the current sequence by manually commanding the robot to a state where it can execute the following command and execute it. This manual adaptation might include more than one additional command or teleoperation mode. Alternatively, the operator can cancel the command couple by clicking the small x in the indicator.



The next command navigate_to LAMA could not be executed, the execution must be canceled or you can try to manually get the robot settled and execute it

status: ready
signal strength: 80%
battery: 100%

The recommender system detected the following sequence 8 times.
Do you want to add it?
Robot: Justin
Command: localize LAMA
Robot: Justin
Command: navigate_to LAMA
Robot: Justin
Command: look_in_direction
Robot: Justin
Command: navigate_for_placement
Robot: Justin
Command: prepare_for_placement

Name: ...

keen_fermi - 1 Sep 2023 - 18:27 CEST
: Welcome to the Surface Avatar telerobotic interface! Ground operations will be giving you instructions through this text messenger, so keep an eye on it. You can interact with the robotics support team using this text messenger.

Figure 5.5 Full overview of the implementation for Coupled Commands including the error message for a unavailable next command

In addition to detecting errors, user interaction is also required for executing a parameterized command. Currently, there are no automated methods for handling parameters, so the operator still needs to define the values that correspond to the current command. The system will indicate this requirement by a blinking indicator. When the operator clicks on the indicator, a window will open, allowing them to enter the necessary parameter values and proceed with the rest of the command sequence. If the first command in the sequence requires parameterization, the window will open automatically without any further input from the operator.

5.4 Test environment

In order to ensure a smooth and efficient testing process, a simplified simulation of the robot has been created for testing purposes as the actual robots may not always be readily available. This is where the Surface Avatar Virtual Test Robot (Surav) comes into play. It was initially developed as an example of the data that needs to be published and how to implement the DDS framework in collaboration with the European Space Agency (ESA).

However, at this stage, the framework was incapable of transmitting video data or sending data for multiple robots. Therefore, Surav was modified to send simulated data for each presented robot, including Justin, LAMA, the Rover, and Bert, with the latter serving as a placeholder until it becomes part of the team. To accomplish this, a parent robot class was created, which sets up a basic robot without much functionality, but with the capability to send all the necessary data to the UI.

Within the Justin class, which is designed to simulate a basic version of the actual robot, a state machine was added, enabling the simulated robot to perform three primary tasks and controls the information sent for each. This approach is based on a basic state machine that includes several actions, states, and a few world parameters. The robot can perform the following tasks: "Clean Panel3," "Seismometer Placement," and "Update Firmware."

Simulating the entire scenario in Unreal Engine would require a significant amount of work, primarily due to the multitude of parameters involved, including all 19 of Justin's joints. Therefore, the necessary actions were recorded during execution on the real robot using the RTI recording service.

As infinite action recording was not feasible, certain parameterized commands such as "drive_around" and "look_around" were restricted to two options each. This limitation meant that it was only possible to move to two fixed positions within the map and look either up or down with Justin.

Due to time constraints, creating a multi-robot state machine was impossible. Therefore, the testing was mainly done with Justin alone, but the framework has also been tested on some randomly created multi-robot data recorded during test runs.

To provide a basic overview of the implemented commands, the corresponding objects, and the parameters used within the state machine, they will be stated and explained in the following. For similar commands that have been independently implemented for different objects, those commands will not be stated individually. Instead, each variation will be stated in curly brackets. The same is done for similar commands like activate and deactivate, where the deviation is stated in brackets, as in the following example: (de)activate. The commands are designed so that the command's white spaces are replaced by '_' to differentiate the objects that correspond to the command.

Objects

- **Solar Panel Unit (SPU)** - Responsible for any computation and storage of solar panel energy management
- **DIP** - Data interface probe (interface to connect Justin with SPU)
- **LAMA** - Lander Mounted Arm (as described in section 4), also command-able as robot in full setup
- **Seismometer** - A device that detects and records ground noises and vibrations

Parameters

- **string location** - Current location of the Robot
- **string look_at** - All objects whose April Tag is currently visible, separated by space
- **string gaze_direction** - Indicates whether Justin is looking in the "Center" or "Down" direction currently
- **string localized** - Current object, which is localized with high accuracy
- **bool seis_picked** - Indicates whether Justin has the Seismometer in his hand
- **bool spu_active** - Indicates whether the SPU is active or shut down
- **bool dip_connected** - Indicates whether Justin is connected to the SPU currently
- **bool panel_turned** - Indicates whether the panel is in standard alignment or turned to the left
- **bool panel_locked** - Indicates whether panel is currently locked

Actions

- **localize {SPU3, LAMA, SPU3, Panel3, Seismometer}** - Tries to localize the specified object with respect to Justin. This task is successful if one of its April-Tags is recognizable in the camera.
- **navigate_to {SPU3, LAMA, placement_position}** - When the object is localized in sufficient accuracy, Justin navigates to the object, such that the standard interactions with the objects are within its reachability range, where the option placement_position is usually a drive_around action

- **drive_to** - Is the restricted form of drive_around, where one can choose a position and its rotation within the map and navigate to it. Here, it will enable the operator to pick from two options, including a pose where the SPU3 is visible and one where the LAMA is visible.
- **look_in_direction** - Is the restricted form of look_around, where the operator could define the angles to which Justin's head would tilt. Here, the operator can look down or center the view.
- **(dis)connect DIP** - (Dis)connects the DIP to the SPU
- **download_data** - Loads the stored data from the SPU to Justin
- **update_firmware** - Justin installs the newest firmware on the SPU
- **(de)active SPU** - Turns SPU on/(off)
- **(un)lock Panel** - (Enables)/disables rotation of the mounted solar panel
- **rotate panel** - Rotates the panel to the given rotation value (only supports 90° turn and back within the simulation, where the robot decides which way to turn, based on internal state)
- **clean Panel** - cleans the given solar panel
- **pick Seismometer** - The correctly localized Seismometer will be picked up
- **prepare_for_placement** - Justin bends down his upper body such that the picked Seismometer can be placed on the floor afterward

The state machine has been implemented to make the corresponding commands available for each state, as shown in figure 5.6. With this implementation, one can see that the state machine covers 14 distinct states, which enable the robot to navigate through the given environment. The state IDs are defined within the curly brackets for each state. To ensure ease of handling, states with the same set of commands have not been merged into a single state.

To begin, the simulation always starts on the start state 0. This state comes with an according parameter set stating an active SPU, an unturned and locked panel, and a pose at the position 'start0' (where LAMA is visible) with no connected or picked objects. From there, one can navigate through the scene with the corresponding commands for each state. Within the overview figure, only the state transition commands are given, meaning that the rest of the available commands can change the system's internal state, which is represented by the given parameters. On execution of one of those, the state machine's state remains the same.

As an additional transition, one can see that one circular object can be seen right below the start state, which has a star sign within. This sign is a simple representation for each state, which offers the command drive_to. On execution of this command, one can transition to the start state by executing this command, which will also move the robot to either the start position in front of LAMA or the SPU.

It is important to note that the same action can also lead to multiple states depending on the internal state. For each transition where that applies, there is a condition for the transition stated in brackets on the corresponding arrow. As all transitions depend on boolean parameters, the transitions where the parameter must be false are stated with an exclamation mark in front and without otherwise, as this is common practice in programming.

The state machine has three final states, which cover the end states of the previously mentioned tasks. At those, the machine has valid exit points that leave the system stable. Nevertheless, as the system is only a simulation in this case, there exists no risk if the system is left at any other state.

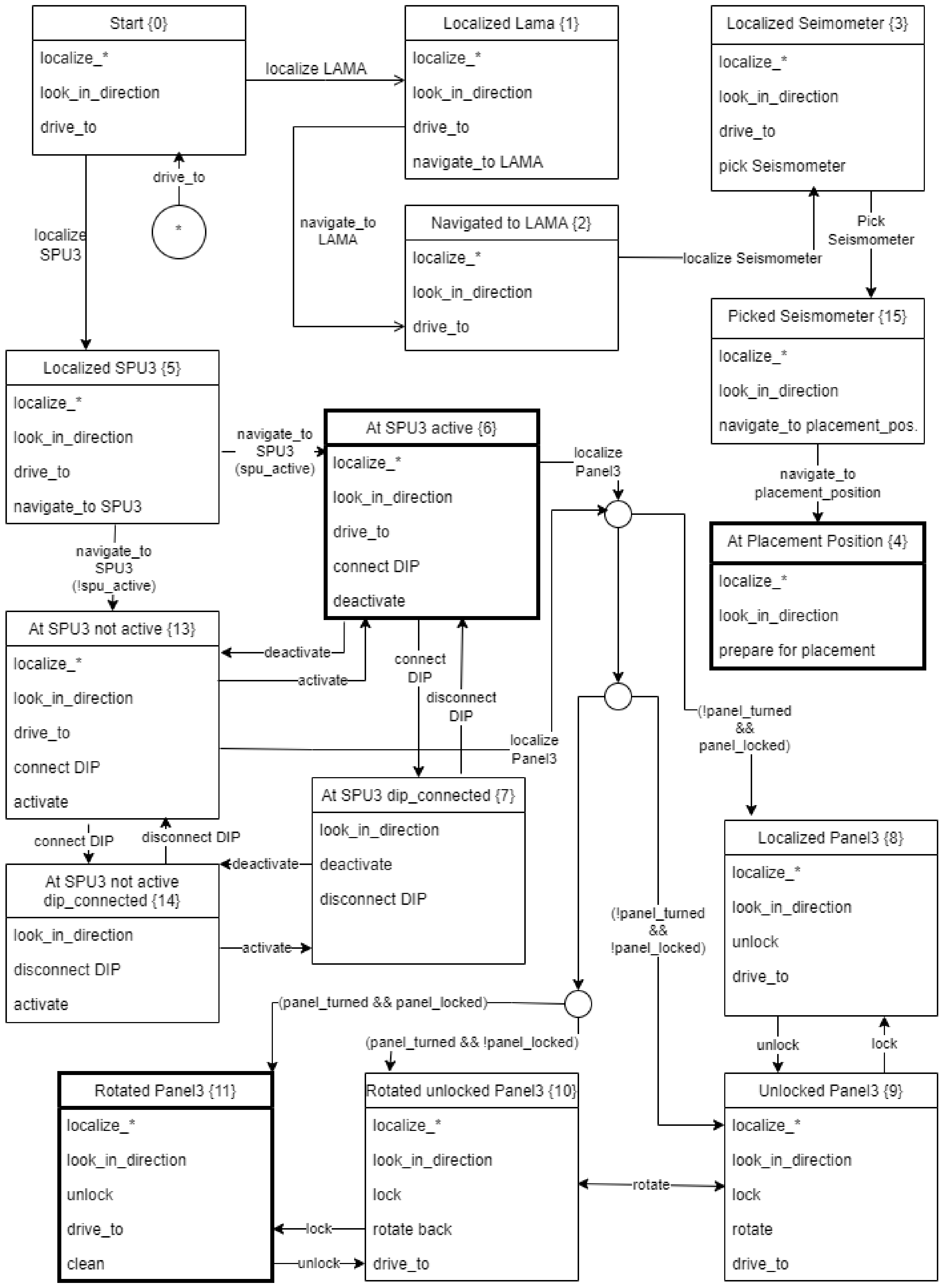


Figure 5.6 Basic overview of the implemented state-machine

6 Evaluation

Given the detailed description of the system and its various processes, it becomes imperative to evaluate the effectiveness of the proposed setup. In order to validate the efficacy of the system, it is necessary to conduct a comprehensive evaluation. As such, this part of the thesis delves into the setup and evaluation of a user study, wherein multiple participants were given the opportunity to work with the system and assess its usability based on standardized questionnaires such as NASA TLX [32] and PSSUQ [45]. These questionnaires measure the perceived task load and the quality of information displayed within the UI.

In addition to the standardized questionnaires, data was also collected on the usage speed and accuracy of the models during the execution time of each participant. This data will be evaluated and presented in this section as well. Furthermore, participants were given an additional questionnaire that included open-ended questions, allowing them to express their thoughts and opinions on various aspects of the system.

Overall, this evaluation is critical in determining the effectiveness of the proposed system and its ability to meet the intended goals. By analyzing the feedback received from the participants, it will be possible to identify areas of improvement and make necessary changes to enhance the system's functionality.

6.1 User Study

6.1.1 Setup

The participants in the user study were kindly requested to perform three tasks, namely "Clean Panel3," "Seismometer Placement," and "Update Firmware," which were implemented in Surav within the given simulation. For each task, they were provided with a written explanation of the task, as stated in the following.

Seismometer Placement: Localize LAMA and navigate to it, then localize the Seismometer Placement and pick it. As soon as you have the Seismometer, you can back off into the placement position. Finally, you prepare the robot for the Seismometer placement on the floor.

Update Firmware: Localize SPU3 and navigate to it, then connect the DIP to download the current data and update the firmware. To finalize the procedure, deactivate and then reactivate the SPU. After that, you can disconnect the DIP.

Clean Panel: Localize SPU3 and navigate to it. Unlock the panel and rotate it to have it accessible for cleaning. After re-locking it, you can let the robot clean the panel.

In the study, participants were asked to perform a series of tasks using the recommender system. The first task was completed without any additional help from the system to have a baseline measure. The second task was completed with the assistance of command prioritization, which reordered the commands based on their probability. Finally, participants attempted to complete the last task with the help of the system's coupled commands.

During this final task, participants were presented with the top-10 detected command couples within the recorded log data. They could either accept or decline these suggestions, choosing to add them to a saved list or discard them. Participants were encouraged to use the added command couples to complete the task, and the command prioritization continued to be active throughout.

The tasks were reordered for each run to prevent any false conclusions based on the difficulty of a particular task. However, the order of the given recommendations remained the same for each participant, as the learning effect was deemed marginal due to the changing tasks. Due to that reason, the improvement in speed performance is not solely attributable to the help of the command probability for that reason. It's worth noting that some participants were already familiar with the interface, as they were part of the development team. The speed improvement was similar in this group in comparison to other participants. Hence, one can infer the minimum learning effect to be proved.

Questionnaires In order to provide a better understanding of the user study results, we will present an overview of the questionnaires given to participants. For each task, participants completed the TLX and PSSUQ, which were structured in the following manner.

- **TLX** - rate on a scale from 1 to 20, where 1 is very low, and 20 is very high
  - Mental Demand - How mentally demanding was the task?
  - Physical Demand - How physically demanding was the task?
  - Temporal Demand - How hurried or rushed was the pace of the task?
  - Performance - How successful were you in accomplishing what you were asked to do?
  - Effort - How hard did you have to work to accomplish your level of performance?
  - Frustration - How insecure, discouraged, irritated, stressed and annoyed were you?
- **PSSUQ** - rate on a scale from 1 to 7 where 1 is equal to strong agreement and 7 to strong disagreement
  - 1. Overall, I am satisfied with how easy it is to use this system.
  - 2. It was simple to use this system.
  - 3. I was able to complete the tasks and scenarios quickly using this system.
  - 4. I felt comfortable using this system
  - 5. It was easy to learn to use this system.
  - 6. I believe I could become productive quickly using this system.
  - 7. The system gave error messages that clearly told me how to fix problems.
  - 8. Whenever I made a mistake using the system, I could recover easily and quickly.
  - 9. The information (such as online help, on-screen messages, and other documentation) provided with this system was clear.
  - 10. It was easy to find the information I needed.
  - 11. The information was effective in helping me complete the tasks and scenarios.
  - 12. The organization of information on the system screens was clear.
  - 13. The interface of this system was pleasant.
  - 14. I liked using the interface of this system.
  - 15. This system has all the functions and capabilities I expect it to have.
  - 16. Overall, I am satisfied with this system.

To assess the data in the PSSUQ, a shortened version is available, which involves averaging scores from four different sections of the questionnaire. These sections include evaluating overall performance (all questions), system usefulness (questions 1-6), information quality (questions 7-12), and interface quality (questions 13-16).

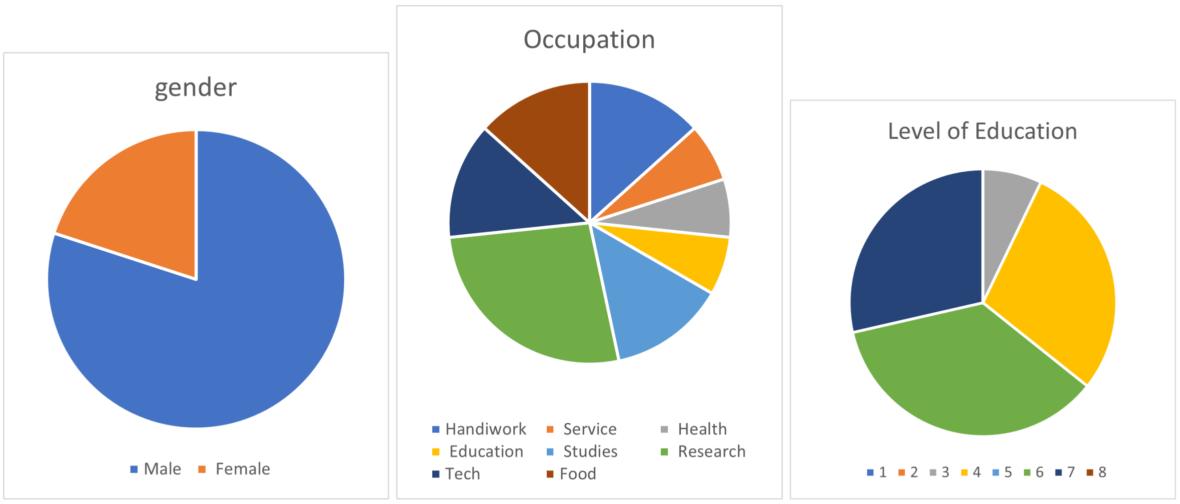
As part of the study, participants were asked to complete an additional questionnaire that utilized the same rating system as the PSSUQ. This questionnaire included questions about the general system and both recommendation system modules. Participants were also invited to share any additional feedback they had about each component of the system here.

- **General**
  - I would like to have more Help and Info Pages within the system in general.
  - I would like to have an Info-Page for each of the commands.
  - I would like to have more automatic recommendation systems.
  - I would like to have more manually executable recommendation systems.
- **Command Probabilities**
  - I think this system helps me achieve my task.
  - I find the reordering of the commands irritating.
  - I would rather like to have a High Lighting of the most probable next command.
- **Command Coupling**
  - I find the "Detected Command Couple" pop-up irritating.
  - I would prefer a user-controlled pop-up with an indicator (light bulb).
  - I would like to be able to create and add my own Couples.

6.1.2 Participant Overview

Fifteen people voluntarily participated in this study, with three being female and twelve being male (see Figure 6.1a). The age range of participants was between 19 and 36, with an average age of 27.7.

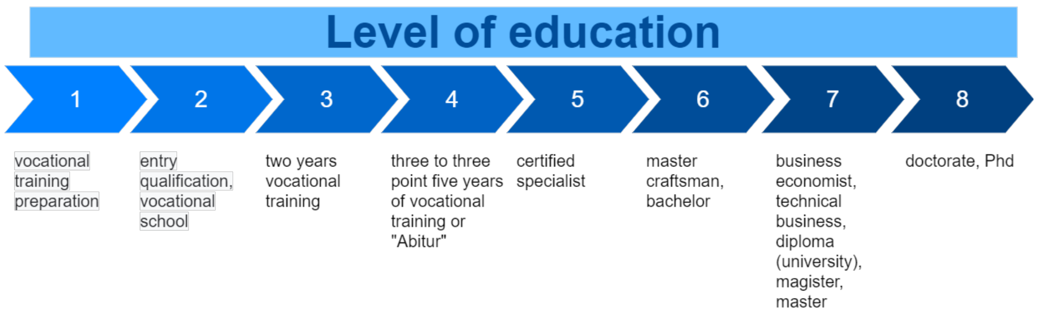
Figure 6.1c shows the corresponding levels of education, rated on an 8-level scale according to the DQR [50]. Bachelor's degrees and similar diplomas are on level six, while a master's degree is on level seven. No participant had achieved level eight, which is a finished PhD or doctorate. The majority of participants had a completed bachelor's degree, followed by an equal number of participants with a master's degree or Abitur. This also applies to other degrees of similar value for each statement.



(a) Gender **(b)** Occupation **(c)** Level of education

Figure 6.1 Overview of User Study Participants

The participants' occupations were varied across many fields (see Figure 6.1b), including six people in research or still studying in an research related area, and an additional two in a tech-related field. With another two in handiwork, the average expertise in operating and maintaining technical applications is high.



Level of education
1
2
3
4
5
6
7
8
vocational training preparation
entry qualification, vocational school
two years vocational training
three to three point five years of vocational training or "Abitur"
certified specialist
master craftsman, bachelor
business economist, technical business, diploma (university), magister, master
doctorate, Phd

Figure 6.2 Levels of education according to the DQR

After carefully analyzing the overview of the participants, it is evident that the selection of participants is diverse, covering a wide range of experience in the technical field. Many different occupation fields and levels of education are represented, providing a diverse representation of potential users. It is worth noting that the variation in age could use some more coverage, and it would be beneficial if the gender assignment were more equally distributed. Despite this, a general first impression of how well people respond to the proposed system should be feasible. Overall, the selection of participants provides a valuable insight into the technical field, and their experience and feedback will undoubtedly contribute to the success of the proposed system.

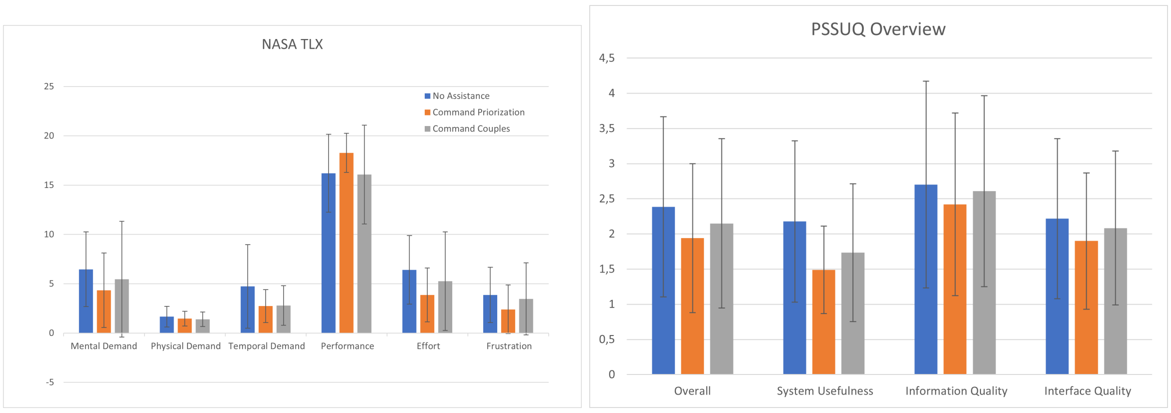
6.1.3 Results

In Figure 6.3, the diagrams show that both examples had a positive impact on the user. To clarify the information presented in the diagrams, each bar represents the average answer given by participants to the questions, with the standard error also visualized for each bar. The blue bar represents the first round of the experiment with no additional help, while the orange bar represents the second round with command prioritization in the form of reordering commands. The gray bars show the same answers for the coupled commands, with the rating including both adding the commands and their execution.

Moving on to the results in this chapter, we begin with a quick overview of the overall outcome. The TLX showed an average improvement of 1.73 points when comparing the command prioritization to the no-help model, which equals an improvement of 8.67 percent on the 20-point scale. The PSSUQ also showed an average improvement of 0.44 points, which equals 6.32 percent on the seven-point scale. However, the command coupling did not perform quite as well, with an average improvement of 0.77 points for the TLX and 0.23 points for the PSSUQ, equaling 3.83 percent and 3.36 percent improvement, respectively.

With this first overview of the results, the outcome can be discussed in more detail. Firstly, the outcome of the TLX (figure 6.3a) will be covered, mainly focusing on how high the task load was. The most significant improvement for the prioritization is available in mental demand (2.13 p.) and the effort (2.53 p.) needed to execute the task. This improvement shows that prioritization is a well-responded measure to reduce mental demand.

The most significant improvement for the coupling focused on the temporal demand, with a 1.93-point improvement. This is mostly due to the fact that the execution of the coupled command is comparatively fast and easy. On the other hand, many people had problems adding the proposed couples at the beginning of the third round. The adding process can be very demanding when the operator has not yet worked with the system, as the suggestions can be overwhelming if the routines are unknown. However, many participants claimed that they would feel much more comfortable using this system if they were familiar with the UI and the robots. Due to that, it can also be seen that, on average, the participants rated their performance improvement as the lowest here, with only a 0.13-point difference. One can also note here



(a) Results NASA Task Load Index **(b)** Short Results of PSSUQ

Figure 6.3 User Study Result Charts

that the participants who were part of the team could interact with the coupled commands very easily and rated the improvement higher than participants who were not. This divergence in the perceived ease of use can also be seen in the data, as the variance for the coupled commands is significantly higher than for the other two experiments.

Next, the PSSUQ will be discussed, which mainly measures the information availability on display and its overall ease of use. The outcome can be seen in figure 6.4 for a summarized overview and figure 6.3b for an in-detail overview, including each available question as answered. In the combined version, the system usefulness did increase the most, with 0.69 points for prioritization and 0.44 points for coupling. This minor increase is probably a result of the interface not changing much in both cases. Hence, the additional functionalities are mainly for easier use of the UI. The information interface quality did only improve slightly for the coupling and had a minor but slightly more significant effect on the prioritization experiments. A reason for the higher improvement in information quality for prioritization can be the fact that the learning effort for the coupled commands is higher than for the prioritization system. In contrast, prioritization makes finding the most valuable and helpful commands easier. Regarding the specific questions, the improvement in command prioritization was highest on the questions about satisfaction with the system and the ability to learn to use the system, making it a good addition for new users. For the couples, it was also observed that the most considerable improvement lies in satisfaction with the system. In contrast, the learning of it was not as easy. Hence, this is better added as a system that comes into play after some experience with the system.

Based on the results of the additional questions that were provided at the end of the study, it can be inferred that the participants had varying opinions and were somewhat undecided on most of the questions

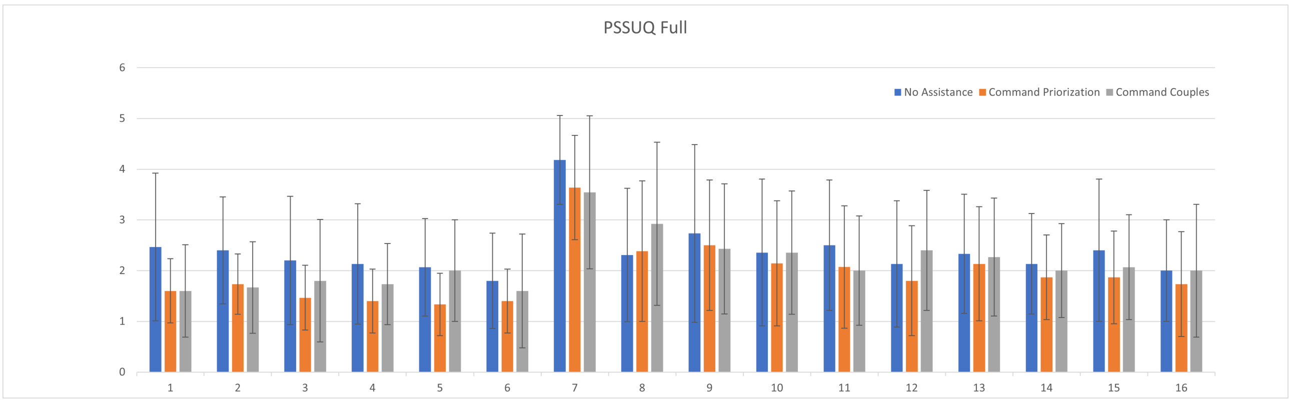


Figure 6.4 Full results of PSSUQ

(as shown in figure 6.5). In general, the participants gave an average rating of around three to four points for these questions.

However, it is interesting to note that the highest approval rating with an average of 1.73 points was obtained for the question on whether the command prioritization in the form of reordering is helpful in achieving the given task. Similarly, the participants stated that the reordering did not cause any irritation with an average rating of 5.8 points.

On the other hand, the participants showed a slight disagreement (ranging from 4 to 5 points) with regards to the need for manually executable recommendation systems and the manual call for the Coupled Command pop-ups. This suggests that the participants may have slightly favored autonomous systems compared to manually executable ones.

Lastly, there was a slightly more pronounced disagreement among the participants regarding whether the pop-up was irritating, with an average rating of 5.2 points. However, overall it can be concluded that the pop-up was well-perceived by the participants.

During the study, participants provided valuable feedback on the general system as well as the two support systems. With regards to the general system, many participants expressed a desire for additional information on the currently executed command, particularly for those without visual feedback on the screen. Participants also requested a history system for notifications to provide a better overview of the current state. On the other hand, some participants preferred user preference options, such as the ability to disable or enable each support system. This would allow for greater customization and control over the system, which could lead to a more efficient and personalized experience.

Regarding command prioritization, participants had mixed feelings about highlighting the most probable next command within the predefined questions. While some preferred a combination of highlighting and reordering the commands, others found the highlighting distracting and felt it could lead to potential decision-making errors. Also, one participant suggested that displaying probabilities next to the commands would make decision-making easier and more accurate.

Most of the feedback concerned command coupling, with participants requesting more help to distinguish which sequences might be useful. Some participants suggested an overview of the sequences rather than one request after another, while others requested a "Start again" button placed on the screen to reiterate the given examples. Additionally, a "Decline all" button was suggested to provide an option for participants to reject all suggested couples. Participants also requested a quality check before the couples were sent, which would provide an extra layer of assurance and accuracy.

One team member suggested an option to switch the levels of autonomy for the commands, putting additional development requirements on the robots' side. This would allow for greater flexibility and customization in the execution of tasks for the requirements of each robot, as the internal planning can be used for the creation of higher level commands.

Despite the various areas for improvement, participants were generally pleased with the ease of executing tasks with the given command couples and were inclined to work with such a system. With continued feedback and improvements, the system has the potential to become even more effective and user-friendly.

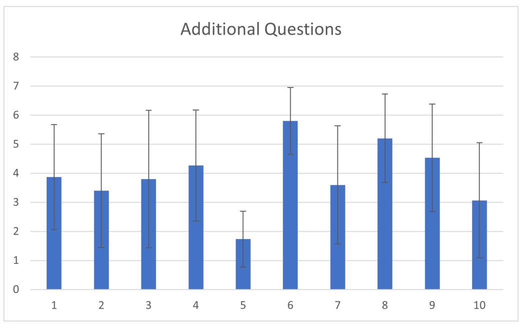


Figure 6.5 Results of additional questions

6.2 Effectiveness and Accuracy

During the User Study, in addition to the questionnaires, a system was implemented to record the user data. This system comprised of various parameters such as time-frames between events, model accuracy represented in the form of quadratic error, and the assignment of whether the chosen command fell within the first three, five, or ten calculated commands. It's important to note that this assignment to the top-k commands applies to the unfiltered state where each command was valid as the following command. Additionally, each option's average value was calculated on every run, including the cases with no assistance, with the command prioritization, and with the couples. Furthermore, an additional calculation was done for the combined average. To ensure that the runs between different participants were comparable, the state of the prediction and coupling models were saved on the first participant, and those saved models were used for all other participants. Specifically, the prefix-span algorithm was used for the command coupling, where the detected couples were stored. The models for sequence prediction were defined as follows during the User Study:

- 1. CPT+ - Command Sequence
- 2. CPT+ - Command Class Sequence
- 3. Subseq - Command Sequence
- 4. Subseq - Command Class Sequence

The results of the command-sequence-based algorithms showed a promising outcome with a low squared error of 0.023 for CPT+ and 0.022 for subseq. However, the accuracy for the classified sequences was much lower, with an error of 0.22 for both algorithms. Despite this, they still managed to predict the correct class most of the time. Overall, the Subseq models had a slightly higher accuracy than CPT+.

From the recordings of whether the executed command belonged to the top-k commands, it was found that CPT+ had better accuracy. However, this could be due to a higher variance over the commands for CPT+ than subseq, or a computation error, and requires further research to confirm.

In future user studies, the measure of the top-k command class should be adapted as the current implementation lists all class assignments for commands. This means that the top-k depends on the number of commands assigned to a particular class, and the fraction of commands assigned to this class listed on top of the list can vary. As such, the measure does not provide valuable information in its current form. Unfortunately, this error was only detected after the execution of this user study, and therefore, no valid conclusions could be drawn from it.

	CPT - Command	CPT - Command Class	Subseq - Command	Subseq - Command Class
Top 3	0.76580688	0.06396605	0.72835097	0.27574956
Top 5	0.80364859	0.24697972	0.75229277	0.32783289
Top 10	0.84276896	0.25045194	0.78118386	0.43982584

Table 6.1 Top-k accuracy for different models

The time between the commands drops from 47 seconds to 38.1 sec for the prioritization and 28.2 sec for the couples. This equals an 18.9% improvement for the prioritization and a 40% improvement for the coupled commands. Here, the quite pronounced reduction for the couples is mainly due to the automatic execution of the following command on availability. Here it also is to note that the time to the first command increased by an average of 3:26 minutes in comparison to the prioritization round. The time to the first command for the first round has been omitted here due to the time the participants needed to get familiar with the interface. This significant difference is due to the previous task of adding the command couples.

Hence, adding the couples enhances productivity and efficiency if at least eleven commands are executed based on this framework after adding. However, this number will probably reduce once the users are familiar with the system. Frequently, there were questions about the system during the study run.

Regarding model performance, the current speed of the models is sufficient, with an average prediction time of 1.41 seconds, including updates for both recommendations. It is sufficient due to an execution time of at least five seconds for any command implemented in the surveyed context. The time has been measured on a contemporary Linux PC. It shows room for improvement, in any case. One major part would be the exchange of the spmf-library used for the CPT+, NOSEP, and QCSP algorithms, as the plugin is not developed for a speedy performance. A native library or a wrapper, considering time constraints, would improve the system here. The slowness of those algorithms can also be seen in comparing the two currently implemented ones. The library for the Subseq algorithm has an average execution time of 1.83 milliseconds (ms), whereas the CPT+ algorithm takes about 553.9 ms.

6.3 Summary

The study demonstrated that both support systems implemented are effective in assisting operators with their tasks. While each system has a positive impact on the operator's performance, they are beneficial in different ways. The command prioritization system reduces cognitive load, while the coupling system improves the execution speed of commands. Feedback received during the study highlighted areas for improvement, such as providing a better overview of recommended options and added sequences for command couples.

There is room for improvement in terms of system speed, as external libraries used in the models are slowing it down. This issue can be resolved by writing new libraries or finding native ones. The models performed well in terms of accuracy, based on squared error, but require further investigation regarding the top-k evaluation. Evaluating the performance differences between implemented algorithms within the coupled commands was not possible due to time constraints and should be part of future research.

Overall, the prioritization and coupling functionalities have the potential to improve system usability and reduce task load. However, some participants experienced difficulty with the coupling system, indicating a need for additional training or familiarization. The study provides valuable insights into the system's usability and can be used to enhance its functionality and user experience. Participants' opinions and preferences were gathered through additional questions at the study's end, revealing positive feedback regarding the reordering feature and Coupled Command pop-ups.

7 Discussion

Throughout this thesis, the current status of controlling a team of heterogeneous robots and how to make this process easier has been elaborated on in great detail. The topic has been covered from various angles, including different UI layouts, physical input devices, and knowledge representations. The goal was to explore the different modalities in which the available data in those systems can aid the operator in achieving their tasks. To achieve this objective, a range of standard approaches, mostly part of the data mining research field, and a few neural network-based approaches were discussed.

In the concept, an approach for the creation of a support system trained on historical user data was presented. The first part was implementing a logging system. This was followed by processing steps to create the desired information. The information was then translated into a code description in the implementation part, which was based on the previously discussed robotic system architecture and its limitations. However, some functions will only be available in future adaptations of the User Interface.

The User Study conducted showed that both implemented modules had a positive effect on the user. The current progress status of the framework was surveyed, and it provided points to improve on and possible future development ideas.

It was stated that there are many possibilities for reducing the operator's workload. However, the workload is not the only parameter one should optimize, as the level of immersion and the operator's awareness of the surroundings also play an essential role. It was specifically noticed during the testing of the Coupled Commands that participants lost interest in the provided camera stream and overall interface due to longer command sequences. This loss of interest resulted in a lower level of immersion. Also, the cognitive load does not always decrease when systems operate autonomously. A survey about self-driving cars found that the perceived workload decreased during automated driving, but monitoring caused a higher cognitive load than manual driving [89]. Therefore, a frequently received suggestion of the user study participants was that the support systems should be manually activatable.

In conclusion, significant progress in cognitive load reduction can be achieved for operating robotic systems of one or more robots by setting the following goals:

- Increasing the operator's immersion in the environment
- Increasing the operator's situational awareness
- Improving intuitiveness of the interface
- Deflecting information on the system to haptic feedback or similar input devices
- Creating support systems for decision-making in the control process

By reducing the cognitive load one needs, those approaches can also increase the usability of the given system. It is evident that the changes and developments made during the course of this thesis fall under the fourth point. As previously mentioned, the primary obstacles in creating a support system are selecting relevant data to gather and processing it in a way that can benefit the user. In this field, numerous processing options are available, each with several applications. Therefore, the following section provides a brief overview of the most significant options.

- Natural Language Processing
    - based on written input
    - based on verbal input
- Task classification for prediction
- Creating a Knowledge database
    - User Behaviour Knowledge:
        - Sequence Prediction
        - Extraction of frequent item-sets or sequences within the system
    - System/Robot Knowledge:
        - World representation
        - Task ontologies

The proposed system is designed to create a comprehensive knowledge database of the user's behavior within the processing options, including the inference from the acquired data to support the user. For instance, the "Command Prioritization" can be seen as an example of sequence prediction, while the "Command Coupling" extracts frequent sequences of commands from the user's behavior. The system has been built on a small training data set, as it has been tested on a data set of approximately 25 recorded UI runs in the given simulated environment performed by experienced users. The accuracy rate of the system for placing in the top 3 predicted commands is 77

During the evaluation phase, mostly positive feedback was retrieved with the given setup, including this small data set. Most users part of the user study would like to work with the given system. A positive effect was shown on both the workload and the user interface rating of the users for both systems. The use of the TLX and the PSSUQ questionnaires has retrieved this information.

However, the coupling of the commands could have been more well-perceived, as the information given on-screen still needs to be improved here. The users were asking for more options to display the gained knowledge and a way to iterate back and forth between the extracted sequences. It was also stated that the sequences of commands should be visualized again before execution, such that the operator is better aware of what the robots will do.

In order to create a more robust approach, it is necessary to increase the size of the training data. Additionally, it was suggested that instead of just coupling commands in sequences, it would be beneficial to offer a range of different levels of autonomy for the available commands. For example, one setting could present the commands as they are in this thesis, while another could offer complete tasks from the user study as individual commands, such as "Clean panel." However, this approach would require more from the operating robots, as they would need to be able to provide commands within the corresponding level of autonomy. As the approach for this thesis aims to be primarily independent of the robots' capabilities, this option was not chosen in this scope.

The reduced immersion in the execution of sequences also resulted in a loss of situational awareness, particularly in cases where execution errors occurred. In these instances, the operator needed time to understand the robot's current state and identify the problem causing the non-availability of the following command. To address this issue, a potential solution could be autonomous error management, which would allow the robot to recover from errors and continue executing the given sequence autonomously. To achieve this, the robot would need to be aware of the complete sequence of commands from the beginning of execution and capable of processing this new input.

7.1 Possible improvements on present system

Based on the previous section, there are several areas in which this system can be improved and extended. One of the most prominent areas that should be improved is the handling of Command Couples within the UI. The testers have requested more options for dealing with additional data, so adding more information and options to the interface is a viable solution. To start with, an overview of the given suggestions from the recommender system should be added so that the user can switch back and forth between the proposed options. This overview enables the user to compare the options and then make an informed decision. It is better than the current method of adding the couples by guessing whether the current one is the most ideal.

Furthermore, an info window on execution start that displays the commands to be executed in that sequence should be added. This feature will give the users a clear idea of what commands will be executed and when. Finally, more options should be added to the handler, including error handling capabilities. For example, a system like the one proposed in Filthaut's thesis [21] could be added. She suggested several options for implementing error handling, including a system that informs the user about possible solutions for solving the error state on occasion.

Another great addition would be an overview of the command names by clicking the indicator. In the current version, it can be challenging to infer which progress bar corresponds to which command. Given these additional adaptations, the system will likely enhance the current setup even more. On the algorithm side, the detection of "Command Couples" could be combined with a knowledge ontology for all the connections of actions with objects and more. This addition would enable a first check of the feasibility of the send couples, such that the operator can be more sure about his decisions. The given examples are already tested prior to sending, so it makes it easier for the operator to make an informed decision.

The ontology could also be used to infer a possible parameterization of any parameterizable command. Another option for parameterization, at least in cases where the parameters available contain options to choose form, would be creating a sequence including this information. This enables the additional prediction to infer on which parameter was most likely used in this case. For cases where the parameters are floating point values or strings, this process might be a bit more complex. An example here would be to store the command sequence must in conjunction with their parameters, where one can then extract all options in the prediction tree and infer on the parameter by, for example extracting the most prominent cluster centroid from the data for floating point values. For strings, one can check for similarity and propose an option where many similar items are available in the data.

The couples could also be adapted so that the sequence includes a call to teleoperate a robot. This addition is beneficial for actions that still need to be implemented and are usually conducted via teleoperation. It also makes it easier to create long sequences if the sequence informs the operator as soon as the teleoperation is necessary and proposes the kind of teleoperation needed.

With regards to improving the command prioritization process, another critical advancement would involve creating a more extensive database of user data. This database would enable neural network-based approaches for time series forecasting, which could be used independently or as an addition to the currently implemented forecasting algorithms. Additionally, an exciting opportunity here would lie in an ontology-based reasoning system, which could predict the following command based on a set of (probabilistic) rules defined in this ontology.

To adapt the current implementation, it would also be beneficial to extend the training database for command classification and test the classification for accuracy, given a couple of text classification options. In this regard, it can be useful to test some different options for text classification, some of which have already been stated in the bibliographic research.

Since the modular approach permits new modules to be added, one option here would include adding prompt-based support systems. For example, a conversational system proposed in Zheng et al. [105] could be implemented, allowing users to interact with the robot conversationally. This addition would extend the current status of the recommender system to be a framework that enables more natural communication between the human operator and the robotic system. Unlike the current approach, where the recommender system operates independently of the robot's systems, this would entail direct communication with the robots. Also, in some cases, it might be a better approach to implement such a system directly on the robot. However, it is worth noting that this is only one possible option, as many more options have already been stated in the bibliographic research.

Regarding these systems, it is important to note that enabling them should always be voluntary. Many operators prefer a system that is consistent in the offered interaction possibilities. As these recommendation systems often adapt the interface of the running system, they should not be forced on the operators if they are already familiar with the current system and prefer to keep the status quo.

7.2 Outlook on Recommendation or Support Systems in Multi-Robot Scenarios

As highlighted in the preceding sections, the realm of recommendation and support systems offers a plethora of options. However, most of these systems are currently unable to handle more than one robot at a time. To make many of the proposed systems usable for multi-robot scenarios, one must be able to adapt the current system. Nonetheless, the most significant challenge is testing. The development of the robotic team and the support system typically occurs simultaneously. Therefore, it is crucial to work on a standard for test beds that allows developers to test their proposed frameworks before applying them to an actual robotic team. Michael, Fink, and Kumar conducted a survey for different applications, providing this opportunity [58]. They also proposed a framework developed primarily for large robotic teams that can efficiently conduct studies on how to improve in the field of multi-robot systems. On the robots' planning sides, there are also many options for future work. Blumenkamp et al.'s graph-neural-network-based navigation for multiple robots would be an excellent addition for any robotic team supporting many mobile ones [9].

Moreover, the proposed system can handle the commands of multiple robots. However, during the master thesis, both modules were only tested in a single-robot environment. Additionally, testing each new function and algorithm added to the current framework was not possible. Therefore, for future development, the current framework must be tested for multiple parameterizations of the algorithms and during the support in different tasks. A study conducted on experienced users would also help determine whether the modules become more effective as a later addition to the system. Overall, it is evident that the development of multi-robot support systems and testing frameworks will significantly contribute to the advancement of robotics in the future.

7.3 Summary

Upon conducting a comprehensive analysis of the thesis, it has become apparent that there are several areas that could benefit from further refinement. Nevertheless, we are pleased to report that the proposed system was well-received by the majority of participants in the user study. As a result, we firmly believe that both modules have the potential to make a significant contribution to the current state of the user interface. This thesis serves as an adequate starting point for the implementation of decision support systems within the context of the Surface Avatar project. Thus, incorporating such systems can confidently be deemed a valuable addition to the already installed haptic feedback teleoperation systems. Overall, we are optimistic about the potential impact of this thesis and its contribution to the field.

A Appendix A - User Study data

Table A.1 Participant Data

id	date	time	age	occupation	lvl of ed.	Gender	teleoperation	team member
7	25.08.2023	16:40	24	handiwork	6	M	N	N
6	25.08.2023	15:55	26	Service	4	M	N	N
5	25.08.2023	15:15	25	Health	6	M	N	N
2	24.08.2023	17:30	36	Education	7	M	N	N
1	23.08.2023	12:30	19	Study	4	M	L	Y
4	25.08.2023	13:06	24	Study	6	F	N	N
16	01.09.2023	16:34	32	handiwork	3	M	N	N
14	30.08.2023	18:47	27	Tech	4	M	N	N
13	30.08.2023	18:00	27	Tech	6	F	N	N
12	30.08.2023	16:17	32	Food		M	L	N
11	30.08.2023	11:03	27	Research	6	M	N	N
8	29.08.2023	13:09	32	Food	4	M	N	N
10	28.08.2023	13:00	29	Research	7	M	Y	Y
3	28.08.2023	09:30	30	Research	7	M	Y	Y

Table A.2 TLX - No support

Task	Mental	Physical	Temporal	Performance	Effort	Frustration
1	1	1	1	17	1	1
1	2	1	3	19	4	1
2	7	1	2	19	6	7
2	5	1	3	16	4	1
1	7	3	15	5	8	4
3	8	3	2	19	6	2
3	4	4	4	11	5	3
2	5	1	10	20	3	4
1	11	2	6	15	12	10
1	12	1	1	18	3	1
2	5	1	4	16	8	3
2	1	1	1	16	5	2
3	13	3	8	18	13	7
3	6	1	1	20	11	5

Table A.3 PSSUQ - No support

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	2	2	1	1	1		1	1	1	1	1	2	2	1	1
1	1	1	1	1	1	1	4	2	2	1	1	1	1	1	1	1
2	1	2	1	2	2	2	5	1	4	2	2	1	3	2	1	1
2	2	2	1	2	2	1		1	2			2	2	2	1	2
1	5	4	3	3	3	4	6	5	6	5	5	5	6	5	5	4
3	1	1	1	1	2	1	4	2	1	1	1	2	2	1	2	1
3	2	2	3	1	2	1	4	2	1	2	2	1	2	2	2	2
2	2	2	2	1	1	1			1	1	3	1	2	1	3	2
1	5	4	4	4	3	3	3	3	4	3	3	2	3	3	3	3
1	1	1	1	1	1	1	4	1	1	1	1	1	1	2	1	1
2	2	3	2	2	3	2	5	2	3	2	3	3	2	2	3	2
2	3	2	1	4	1	2	4	4	4	4	4	4	2	2	4	2
3	5	4	5	4	4	3	4	4	6	5	4	3	3	3	5	4
3	3	3	3	3	3	2	3	2	2	2	2	2	2	2	2	2

Table A.4 TLX - Prioritization

Task	Mental	Physical	Temporal	Performance	Effort	Frustration
2	1	1	1	20	1	1
3	4	1	1	20	3	3
1	2	1	2	20	2	2
3	2	1	3	20	3	2
2	7	3	3	18	4	2
2	10	2	1	20	8	2
2	2	2	2	16	5	2
3	3	1	4	20	4	1
2	7	2	6	14	11	11
3	14	1	1	16	5	2
1	2	1	3	18	2	2
3	1	1	1	16	1	1
2	6	3	5	18	4	3
1	2	1	3	20	4	1

Table A.5 PSSUQ - Priorization

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	2	2	2	1	1	1		1	1	1	1	1	2	2	1	1
3	1	1	1	1	1	1	4	2	2	2	1	1	1	1	1	1
1	1	2	1	1	1	1	5	1	5	3	2	2	1	1	2	1
3	2	2	1	1	1	1		1				2	2	2	2	2
2	2	2	1	2	2	2	4	4	4	3	4	2	2	2	1	1
2	2	2	1	1	1	1	4	4	2	2	1	1	2	1	2	1
2	1	2	2	2	1	2	4	2	2	2	2	2	2	2	1	1
3	1	1	1	1	1	1			1	1	1	1	2	1	3	2
2	3	3	3	3	3	3	4	4	4	5	4	4	4	4	3	4

Table A.5 PSSUQ - Priorization

3	2	1	1	2	1	1	1	1	1	1	1	1	1	2	1	1
1	1	1	2	1	2	1	4	1	3	1	2	1	2	2	1	2
3	1	1	1	1	1	1	4	4	4	4	4	4	5	2	4	4
2	2	2	2	2	1	2	3	4	2	2	3	3	3	3	2	2
1	2	2	1	1	1	1	3	2	2	1	1	1	1	1	2	2

Table A.6 TLX - Couples

Task	Mental	Physical	Temporal	Performance	Effort	Frustration
3	1	1	1	20	1	1
2	2	1	2	20	2	3
3	15	1	4	10	16	11
1	13	1	6	5	5	3
3	1	1	2	17	3	1
1	14	3	2	20	11	2
1	3	3	3	10	3	2
1	13	1	3	20	11	3
3	12	2	8	11	12	13
2	1	1	1	20	1	1
3	2	1	2	19	2	2
1	1	1	1	12	8	5
1	2	2	4	19	2	2
2	1	1	2	20	1	1

Table A.7 PSSUQ - Couples

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
3	1	1	1	1	1	1		1	1	1	1	2	2	2	2	2
2	1	1	1	1	1	1	5	2	2	2	2	2	1	1	1	2
3	4	4	5	3	5	5	6	3	4	4		5	3	3	4	6
1	2	2	3	2	2	1		6				2	2	2	3	1
3	1	1	1	1	2	2	4	4	4	3	3	3	3	2	1	1
1	1	1	2	2	2	1	2	4	2	2	2	2	1	1	2	1
1	1	1	2	2	1	2	3	2	2	2	2	1	2	2	2	2
1	2	2	1	2	2	1			1	4	1	4	2	2	1	2
3	3	3	3	3	3	3	4	3	5	4	4	3	4	4	3	3
2	1	1	1	1	2	1	1	1	1	1	1	1	1	2	1	1
3	1	2	1	1	2	1	4	2	2	1	2	1	2	1	2	2
1	2	1	3	3	2	2	5	5	4	4	4	4	5	3	4	3
1	2	2	1	2	2	1	3	4	2	2	1	2	2	1	2	1
2	1	1	1	1	1	1	2	1	2	1	1	2	1	1	1	1

Table A.8 General Questions

more help	cmd info	auto rec.	exec rec.	helps	irritating	highlight		controlled	create
7	7	7	7	1	7	7		7	7
5	6	6	6	2	6	3		6	5
3	3	6	3	3	4	2		3	4
5	2	2	2	2	6	2		4	1
1	1	1	4	1	5	2		4	2
6	5	5	5	2	3	5		6	2
3	2	2	1	1	6	2		6	7
6	3	6	4	1	7	3		2	1
3	2	3	3	3	5	2		3	3
3	4	7	7	1	6	7		7	1
3	2	1	6	1	6	2		2	2
3	5	6	6	4	6	5		6	2
2	1	2	3	2	6	2		2	2
2	2	1	5	1	7	3		6	3

Additional Remarks:

- **General:**
  - I would like to have information of the command that is being executed, or it's status.
  - Support Systems dis-enable function in user pref.
  - hover highlight for the commands, add history to shown notifications
- **Prioritization:**
  - combining reordering and highlighting
  - I think having a highlighting of the most probable command could make the user make mistakes, since they could be making decisions driven by visual stimuli.
  - show command probabilities on UI
- **Couples:**
  - Start again button to reiterate the couples, with experience it would be easier to evaluate the quality of the suggested couples
  - have a list of the couples other than seeing them one after another
  - command overview for added couples, as in which commands will be executed
  - going through suggestions was a little cumbersome, but it was easy afterwards, other than coupling commands switching between levels of autonomy would be nice
  - add a decline all button for the couples
  - getting an overview over all sequences first, instead of showing one after the other, would be nice
  - additional button to navigate to previously shown command coupling, would be nice
  - It would be great to have the feasibility of the coupled tasks as well as what the sequence achieves/goal

Recorded Data

```
Model accuracy
[0.02417346 0.23044763 0.02395105 0.23632046]
[0.02283204 0.22366086 0.02168186 0.22848287]
[0.02253351 0.21541931 0.01959171 0.2102533 ]

[0.02302233 0.22219291 0.02143135 0.22364165]

Top 3
[0.62946429 0.0875     0.57113095 0.11230159]
[0.74851852 0.0587037  0.69806878 0.32097884]
[0.86965812 0.05555556 0.86004274 0.3241453 ]

[0.76580688 0.06396605 0.72835097 0.27574956]

Top 5
[0.7139881  0.22619048 0.64761905 0.13313492]
[0.78685185 0.24304233 0.71473545 0.37097884]
[0.87820513 0.26431624 0.86004274 0.39786325]

[0.80364859 0.24697972 0.75229277 0.32783289]

Top 10
[0.73482143 0.22619048 0.72728175 0.29077381]
[0.83074074 0.25137566 0.7415873  0.47291005]
[0.92307692 0.26431624 0.86004274 0.49337607]

[0.84276896 0.25045194 0.78118386 0.43982584]

Time to first command
0:02:54.961911
0:01:32.284854
0:04:58.167440

0:03:00.879170

Time between commands
0:00:47.020309
0:00:39.131454
0:00:28.198252

0:00:37.187896

Model computation times
[datetime.timedelta(microseconds=554917)
 datetime.timedelta(microseconds=558469)
 datetime.timedelta(microseconds=1756)
 datetime.timedelta(microseconds=2008)]
[datetime.timedelta(microseconds=566015)
 datetime.timedelta(microseconds=561436)
 datetime.timedelta(microseconds=1937)
 datetime.timedelta(microseconds=1609)]
[datetime.timedelta(microseconds=539903)
 datetime.timedelta(microseconds=543118)
 datetime.timedelta(microseconds=1689)
 datetime.timedelta(microseconds=2071)]

[datetime.timedelta(microseconds=553866)
 datetime.timedelta(microseconds=554103)
 datetime.timedelta(microseconds=1803)
 datetime.timedelta(microseconds=1874)]

Full Prediction Time
0:00:01.397940
0:00:01.426325
0:00:01.400882

0:00:01.410244

Prediction to command
0:00:39.695265
0:00:37.667307
0:00:26.781195

0:00:34.245613
```

B Appendix B - User Study Questionaires

User Study Recommendation System

Introduction:

You will take part in a research study conducted by Nicole Grabner for a master thesis covering user interface enhancements using data mining technology. The use of this study is to evaluate the effect of these enhancements and create data about the computation and reaction time of the procedures.

Procedures:

You will undertake three different tasks on the UI each with none, and both created recommendations implemented on it, which is for one a command prioritization algorithm and a command coupling algorithm.

Risks and Benefits:

There is a low chance for any risks in this study as it only involves operating a simulated robot and it will enable the investigator to evaluate the work done.

Confidentiality:

Your identity will be kept confidential to the extent provided by law. Your data will be assigned to a unique identifier and all data will be stored in a secure environment. Only the research team will have access to the data.

Voluntary Participation:

Participation in this study is entirely voluntary and you have the right to withdraw from the study at any time.

Consent:

You are okay with recording usage data during the experiment. You also stated that the gained data can be used anonymously to evaluate the user experience of the given system. You have understood all the above information and agree with it.

General:

Date: ____

Time: ____

Proband:

Name: ____

Age: ____

Occupation: ____

Lvl of Education: ____

Gender: ____

Familiar with Teleoperation Systems in General:

____

Team Member: ____

Identifying No: ____

Study Supervision:

Investigator: Nicole Grabner

Institution: Modex Lab, DLR Robotics and Mechatronics Center

Contact Information: nicole.grabner@dlr.de

Participant's Signature: ________________________Date: _________________

NASA Task Load Index

Hart and Staveland's NASA Task Load Index (TLX) method assesses workload on five 7-point scales. Increments of high, medium and low estimates for each point result in 21 gradations on the scales.

Task:	
Recommendations:	

Mental Demand: How mentally demanding was the task?

Very Low Very High

Physical Demand: How physically demanding was the task?

Very Low Very High

Temporal Demand: How hurried or rushed was the pace of the task?

Very Low Very High

Performance: How successful were you in accomplishing what you were asked to do?

Very Low Very High

Effort: How hard did you have to work to accomplish your level of performance?

Very Low Very High

Frustration: How insecure, discouraged, irritated, stressed and annoyed were you?

Very Low Very High

Post Study System Usability Questionaire (PSSUQ)

On a scale between Strongly Agree to Strongly Disagree, please rate the following statements:

Questions 1 to 16: Overall Questions 1 to 6: System Usefulness (SYSUSE) Questions 7 to 12: Information Quality (INFOQUAL) Questions 13 to 16: Interface Quality (INTERQUAL)	Task	Recommendations

1. Overall, I am satisfied with how easy it is to use this system.

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

2. It was simple to use this system.

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

3. I was able to complete the tasks and scenarios quickly using this system.

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

4. I felt comfortable using this system.

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

5. It was easy to learn to use this system.

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

6. I believe I could become productive quickly using this system.

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

7. The system gave error messages that clearly told me how to fix problems.

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

8. Whenever I made a mistake using the system, I could recover easily and quickly.

Strongly Agree | Neither Agree Nor Disagree | Strongly Disagree

9. The information (such as online help, on-screen messages, and other documentation) provided with this system was clear.

Strongly Agree | Neither Agree Nor Disagree | Strongly Disagree

10. It was easy to find the information I needed.

Strongly Agree | Neither Agree Nor Disagree | Strongly Disagree

11. The information was effective in helping me complete the tasks and scenarios.

Strongly Agree | Neither Agree Nor Disagree | Strongly Disagree

12. The organization of information on the system screens was clear.

Strongly Agree | Neither Agree Nor Disagree | Strongly Disagree

13. The interface of this system was pleasant.

Strongly Agree | Neither Agree Nor Disagree | Strongly Disagree

14. I liked using the interface of this system.

Strongly Agree | Neither Agree Nor Disagree | Strongly Disagree

15. This system has all the functions and capabilities I expect it to have.

Strongly Agree | Neither Agree Nor Disagree | Strongly Disagree

16. Overall, I am satisfied with this system.

Strongly Agree | Neither Agree Nor Disagree | Strongly Disagree

General:

I would like to have more Help and Info Pages within the system in general.

Strongly Agree — Neither Agree Nor Disagree — Strongly Disagree

I would like to have an Info-Page for each of the commands.

Strongly Agree — Neither Agree Nor Disagree — Strongly Disagree

I would like to have more automatic recommendation systems.

Strongly Agree — Neither Agree Nor Disagree — Strongly Disagree

I would like to have more manually executable recommendation systems.

Strongly Agree — Neither Agree Nor Disagree — Strongly Disagree

Any additional remarks?

Command probabilities:

I think this system helps me achieve my task.

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

I find the reordering of the commands irritating.

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

I would rather like to have a High Lighting of the most probable next command.

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

Any additional remarks?

Command Coupling:

I find the “Detected Command Couple” pop-up irritating.

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

I would prefer a User controlled pop-up with a indicator (light bulb).

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

I would like to be able to create and add my own Couples.

Strongly Agree Neither Agree Nor Disagree Strongly Disagree

Any additional remarks?

Task description

Seismometer Placement:

Localize LAMA and navigate to it, then localize the Seismometer Placement and pick it. As soon as you have the Seismometer you can back off into the placement position. Finally, you prepare the robot for the Seismometer placement on the floor.

Update Firmware:

Localize SPU3 and navigate to it, then connect the DIP to download the current data and update the firmware. To finalize the procedure, deactivate and then reactivate the SPU. After that, you can disconnect the DIP.

Clean Panel:

Localize SPU3 and navigate to it. Unlock the panel and rotate it to have it accessible for cleaning. After relocking it you can let the robot clean the panel.

Bibliography

[1] Samarth Aggarwal et al. "Goal-driven Command Recommendations for Analysts". In: Association for Computing Machinery, Inc, June 2020, pp. 160–169. ISBN: 9781450375832. DOI: 10.1145/3383313.3412255.

[2] Michael Ahn et al. *Do As I Can, Not As I Say: Grounding Language in Robotic Affordances*. 2022. arXiv: 2204.01691 [cs.RO].

[3] William Albert and E Dixon. "Is this what you expected? The use of expectation measures in usability testing". In: Jan. 2003.

[4] John Bateman et al. "Heterogeneous Ontologies and Hybrid Reasoning for Service Robotics: The EASE Framework". In: *ROBOT 2017: Third Iberian Robotics Conference*. Ed. by Anibal Ollero et al. Cham: Springer International Publishing, 2018, pp. 417–428. ISBN: 978-3-319-70833-1.

[5] Leonie Becker, Bernhard Weber, and Nicolai Bechtel. "Haptic Guidance for Teleoperation: Optimizing Performance and User Experience". In: *Haptics: Science, Technology, Applications*. Ed. by Hasti Seifi et al. Cham: Springer International Publishing, 2022, pp. 129–137. ISBN: 978-3-031-06249-0.

[6] Julian Besag. "An Introduction to Markov Chain Monte Carlo Methods". In: *Mathematical Foundations of Speech and Language Processing*. Ed. by Mark Johnson et al. New York, NY: Springer New York, 2004, pp. 247–270. ISBN: 978-1-4419-9017-4.

[7] Daniel Beßler. "Ontological Representation of Activity Context for Flexible Robot Task Execution". University of Bremen, Sept. 2022.

[8] Daniel Beßler et al. "A Formal Model of Affordances for Flexible Robotic Task Execution". In: Sept. 2020. DOI: 10.3233/FAIA200374.

[9] Jan Blumenkamp et al. "A Framework for Real-World Multi-Robot Systems Running Decentralized GNN-Based Policies". In: Institute of Electrical and Electronics Engineers Inc., 2022, pp. 8772–8778. ISBN: 9781728196817. DOI: 10.1109/ICRA46639.2022.9811744.

[10] Leo Breiman. "Bagging predictors". In: 1996, pp. 123–140. DOI: 10.1007/BF00058655.

[11] John Brooke. "SUS: A quick and dirty usability scale". In: *Usability Eval. Ind.* 189 (Nov. 1995).

[12] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[13] Jonathan H. Clark et al. "scpCanine/scp: Pre-training an Efficient Tokenization-Free Encoder for Language Representation". In: *Transactions of the Association for Computational Linguistics* 10 (2022), pp. 73–91. DOI: 10.1162/tacl_a_00448. URL: https://doi.org/10.1162%2Ftacl_a_00448.

[14] Fred J. Damerau. "A Technique for Computer Detection and Correction of Spelling Errors". In: *Commun. ACM* 7.3 (Mar. 1964), pp. 171–176. ISSN: 0001-0782. DOI: 10.1145/363958.363994. URL: https://doi.org/10.1145/363958.363994.

[15] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[16] ESA. *Analog-1 Interact rover*. Nov. 18, 2019. URL: https://www.esa.int/ESA_Multimedia/Images/2019/11/Analog-1_Interact_rover (visited on 09/06/2023).

[17] ESA. *ExPeRT Exploration Preparation Research and Technology*. 2023. URL: https://www.esa.int/About_Us/Business_with_ESA/Business_Opportunities/ExPeRT_Exploration_Preparation_Research_and_Technology%5C%20 (visited on 09/06/2023).

[18] ESA. *Orbiting astronaut oversees robot team on Earth*. July 25, 2023. URL: https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Orbiting_astronaut_oversees_robot_team_on_Earth (visited on 09/06/2023).

[19] ESA. *Rover plus astronaut complete Mount Etna challenge*. July 6, 2022. URL: https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Rover_plus_astronaut_complete_Mount_Etna_challenge (visited on 09/06/2023).

[20] Len Feremans, Boris Cule, and Bart Goethals. "Mining Top-k Quantile-based Cohesive Sequential Patterns". In: *SDM*. 2018. URL: https://api.semanticscholar.org/CorpusID:21664612.

[21] Liliane Filthaut. "Error Handling in Space". MA thesis. University of applied Sciences Munich, 2023.

[22] The Apache Software Foundation. *Apache License*. 2004. URL: https://www.apache.org/licenses/LICENSE-2.0.txt (visited on 09/08/2023).

[23] P. Fournier-Viger et al. "The SPMF Open-Source Data Mining Library Version 2." In: *Proc. 19th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2016) Part III*. Springer LNCS 9853, 2016, pp. 36–40.

[24] Chuancong Gao et al. "Efficient Mining of Frequent Sequence Generators". In: *Proceedings of the 17th International Conference on World Wide Web*. WWW '08. Beijing, China: Association for Computing Machinery, 2008, pp. 1051–1052. ISBN: 9781605580852. DOI: 10.1145/1367497.1367651. URL: https://doi.org/10.1145/1367497.1367651.

[25] Marko Gasparic and Francesco Ricci. "IDE Interaction Support with Command Recommender Systems". In: *IEEE Access* 8 (June 2020), pp. 19256–19270. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2967840.

[26] Malik Ghallab et al. "PDDL - The Planning Domain Definition Language". In: (Aug. 1998).

[27] Katherine Driggs-Campbell Girish Chowdhary Chinmay Soman. *Levels of Autonomy for Field Robots*. 2020. URL: https://www.earthsense.co/news/2020/7/24/levels-of-autonomy-for-field-robots (visited on 08/23/2023).

[28] Ted Gueniche, Philippe Fournier-Viger, and Vincent S. Tseng. "Compact Prediction Tree: A Lossless Model for Accurate Sequence Prediction". In: *Advanced Data Mining and Applications*. Ed. by Hiroshi Motoda et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 177–188. ISBN: 978-3-642-53917-6.

[29] Ted Gueniche et al. "CPT+: Decreasing the time/space complexity of the Compact Prediction Tree". In: ed. by Tru Cao et al. Springer International Publishing, 2015, pp. 625–636. ISBN: 978-3-319-18032-8.

[30] Darren Guinness, Daniel Szafir, and Shaun K. Kane. "GUI Robots: Using Off-the-Shelf Robots as Tangible Input and Output Devices for Unmodified GUI Applications". In: *Proceedings of the 2017 Conference on Designing Interactive Systems*. DIS '17. Edinburgh, United Kingdom: Association for Computing Machinery, 2017, pp. 767–778. ISBN: 9781450349222. DOI: 10.1145/3064663.3064706. URL: https://doi.org/10.1145/3064663.3064706.

[31] Akshit Gupta et al. "To Trust or Not To Trust: How a Conversational Interface Affects Trust in a Decision Support System". In: *Proceedings of the ACM Web Conference 2022*. WWW '22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 3531–3540. ISBN: 9781450390965. DOI: 10.1145/3485447.3512248. URL: https://doi.org/10.1145/3485447.3512248.

[32] Sandra G. Hart and Lowell E. Staveland. “Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research”. In: *Human Mental Workload*. Ed. by Peter A. Hancock and Najmedin Meshkati. Vol. 52. Advances in Psychology. North-Holland, 1988, pp. 139–183. DOI: https://doi.org/10.1016/S0166-4115(08)62386-9. URL: https://www.sciencedirect.com/science/article/pii/S0166411508623869.

[33] Pengcheng He et al. *DeBERTa: Decoding-enhanced BERT with Disentangled Attention*. 2021. arXiv: 2006.03654 [cs.CL].

[34] Wenlong Huang et al. *Inner Monologue: Embodied Reasoning through Planning with Language Models*. 2022. arXiv: 2207.05608 [cs.RO].

[35] Nor Farzana Jeffri and Dayang Rambli. “A review of augmented reality systems and their effects on mental workload and task performance”. In: *Heliyon* 7 (Mar. 2021), e06277. DOI: 10.1016/j.heliyon.2021.e06277.

[36] Marcel Kaufmann et al. “Copiloting Autonomous Multi-Robot Missions: A Game-inspired Supervisory Control Interface”. In: *arXiv preprint arXiv:2204.06647* (June 2022). URL: http://arxiv.org/abs/2204.06647.

[37] Jurek Kirakowski and Mary Corbett. “SUMI: the Software Usability Measurement Inventory”. In: *British Journal of Educational Technology* 24 (Oct. 2006), pp. 210–212. DOI: 10.1111/j.1467-8535.1993.tb00076.x.

[38] Elsa A Kirchner et al. “An intelligent man-machine interface-multi-robot control adapted for task engagement based on single-trial detectability of P300”. In: *Frontiers in Human Neuroscience* 10 (June 2016). ISSN: 16625161. DOI: 10.3389/fnhum.2016.00291.

[39] Thomas Krueger et al. “Designing and Testing a Robotic Avatar for Space-to-Ground Teleoperation: the Developers’ Insights”. In: Oct. 2020.

[40] Rafael Ktistakis et al. “Succinct BWT-Based Sequence Prediction”. In: Aug. 2019, pp. 91–101. ISBN: 978-3-030-27617-1. DOI: 10.1007/978-3-030-27618-8_7.

[41] Michaela Kümpel et al. “Robotic Shopping Assistance for Everyone: Dynamic Query Generation on a Semantic Digital Twin as a Basis for Autonomous Shopping Assistance”. In: *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom*. 2023, pp. 2523–2525. URL: https://www.southampton.ac.uk/~eg/AAMAS2023/pdfs/p2523.pdf.

[42] Zhenzhong Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2019. arXiv: 1909.11942 [cs.CL].

[43] Daniel Leidner, Christoph Borst, and Gerd Hirzinger. “Things Are Made for What They Are: Solving Manipulation Tasks by Using Functional Object Classes”. In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)* (2012), pp. 429–435.

[44] Daniel Sebastian Leidner. “Cognitive Reasoning for Compliant Robot Manipulation”. University of Bremen, Oct. 2017.

[45] James Lewis. *Using the PSSUQ and CSUQ in User Experience Research and Practice*. Aug. 2019. ISBN: 1733339205.

[46] James Lewis and James R. “IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use”. In: *International Journal of Human-Computer Interaction* 7 (Feb. 1995), pp. 57–. DOI: 10.1080/10447319509526110.

[47] Wei Li et al. “Design and evaluation of a command recommendation system for software applications”. In: *ACM Transactions on Computer-Human Interaction* 18 (2 June 2011). ISSN: 10730516. DOI: 10.1145/1970378.1970380.

[48] Neal Y Lii et al. "Introduction to Surface Avatar: the First Heterogeneous Robotic Team to be Commanded with Scalable Autonomy from the ISS". In: International Astronautical Federation, IAF, Sept. 2022. URL: https://iafastro.directory/iac/paper/id/73187/summary/.

[49] Bryan Lim and Stefan Zohren. "Time-series forecasting with deep learning: a survey". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 379.2194 (Feb. 2021), p. 20200209. DOI: 10.1098/rsta.2020.0209. URL: https://doi.org/10.1098%2Frsta.2020.0209.

[50] "Liste der zugeordneten Qualifikationen". In: Bund-Länder-Koordinierungsstellefür den Deutschen Qualifikationsrahmen für lebenslanges Lernen (DQR), 2017. URL: https://www.dqr.de/dqr/shareddocs/downloads/media/content/liste-der-zugeordneten-qualifikationen_01082017.pdf?__blob=publicationFile%5C&v=2.

[51] Evan Lutins. *Ensemble Methods in Machine Learning: What are They and Why Use Them?* 2017. URL: https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f (visited on 09/13/2023).

[52] Rute Luz et al. "On the Use of Haptic Tablets for UGV Teleoperation in Unstructured Environments: System Design and Evaluation". In: *IEEE Access* 7 (June 2019), pp. 95443–95454. DOI: 10.1109/access.2019.2928981.

[53] Rute Luz et al. "Traction Awareness Through Haptic Feedback for the Teleoperation of UGVs". In: 2018, pp. 313–319. ISBN: 9781538679807. DOI: 10.1109/ROMAN.2018.8525740. URL: https://youtu.be/szHA2nIhjAs.

[54] Maximilian Maier et al. "TINA: The Modular Torque Controlled Robotic Arm - A Study for Mars Sample Return". In: *2021 IEEE Aerospace Conference (50100)*. 2021, pp. 1–10. DOI: 10.1109/AERO50100.2021.9438222.

[55] Sumaira Manzoor et al. "Ontology-Based Knowledge Representation in Robotic Systems: A Survey Oriented toward Applications". In: *Applied Sciences* 11.10 (2021). ISSN: 2076-3417. DOI: 10.3390/app11104324. URL: https://www.mdpi.com/2076-3417/11/10/4324.

[56] Alice Martin et al. *The Monte Carlo Transformer: a stochastic self-attention model for sequence prediction*. 2020. arXiv: 2007.08620 [cs.LG].

[57] Ricardo P. Masini, Marcelo C. Medeiros, and Eduardo F. Mendes. *Machine Learning Advances for Time Series Forecasting*. 2020. arXiv: 2012.12802 [econ.EM].

[58] Nathan Michael, Jonathan Fink, and Vijay Kumar. "Experimental Testbed for Large Multirobot Teams: Verification and Validation". In: *IEEE Robotics and Automation Magazine* 15 (1 2008), pp. 53–61. ISSN: 10709932. DOI: 10.1109/M-RA.2007.914924.

[59] Tomas Mikolov et al. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. arXiv: 1310.4546 [cs.CL].

[60] Yuichiro Mori et al. "Multi-robot Coordination Based on Ontologies and Semantic Web Service". In: *Knowledge Management and Acquisition for Smart Systems and Services*. Ed. by Yang Sok Kim, Byeong Ho Kang, and Deborah Richards. Cham: Springer International Publishing, 2014, pp. 150–164. ISBN: 978-3-319-13332-4.

[61] Swati Nagori and Hemant Kumar Soni. "Issues and research challenges in sequential pattern mining". In: Institute of Electrical and Electronics Engineers Inc., June 2020. ISBN: 9781728192512. DOI: 10.1109/ICADEE51157.2020.9368943.

[62] Alexander Nussbaumer, Andrew Pope, and Karen Neville. "A framework for applying ethics-by-design to decision support systems for emergency management". In: *Information Systems Journal* 33.1 (2023), pp. 34–55. DOI: https://doi.org/10.1111/isj.12350. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/isj.12350. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/isj.12350.

[63] Alberto Olivares-Alarcos et al. "A review and comparison of ontology-based approaches to robot autonomy". In: *Knowledge Engineering Review* 34 (2019). ISSN: 14698005. DOI: 10.1017/S0269888919000237.

[64] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[65] Eleftheria Maria Pechlivani et al. "Towards Sustainable Farming: A Robust Decision Support System's Architecture for Agriculture 4.0". In: *2023 24th International Conference on Digital Signal Processing (DSP)*. 2023, pp. 1–5. DOI: 10.1109/DSP58604.2023.10167922.

[66] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[67] Jian Pei et al. "PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth". In: *Proceedings of the 17th International Conference on Data Engineering*. USA: IEEE Computer Society, 2001, pp. 215–224. ISBN: 0769510019.

[68] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[69] Sampo Pyysalo et al. *WikiBERT models: deep transfer learning for many languages*. 2020. arXiv: 2006.01538 [cs.CL].

[70] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).

[71] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2018). URL: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

[72] Antonin Raffin et al. "Learning to Exploit Elastic Actuators for Quadruped Locomotion". In: (Sept. 2022). DOI: 10.48550/arXiv.2209.07171.

[73] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: http://arxiv.org/abs/1908.10084.

[74] Baptiste Rocca. *Introduction to recommender systems*. 2019. URL: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada (visited on 08/31/2023).

[75] Jesús Juan Roldán, Jaime Del Cerro, and Antonio Barrientos. "Multiple robots, single operator: considerations about information and commanding". In: June 2016. URL: https://www.researchgate.net/publication/327208105.

[76] Juan Jesús Roldán et al. "Multi-robot interfaces and operator situational awareness: Study of the impact of immersion and prediction". In: *Sensors (Switzerland)* 17 (8 June 2017). ISSN: 14248220. DOI: 10.3390/s17081720.

[77] Riccardo Rosati et al. "From knowledge-based to big data analytic model: a novel IoT and machine learning based decision support system for predictive maintenance in Industry 4.0". In: *Journal of Intelligent Manufacturing* 34 (1 Jan. 1, 2023), pp. 107–121. DOI: 10.1007/s10845-022-01960-x.

[78] *RTI Connext DDS Getting Started*. 2020. URL: https://community.rti.com/static/documentation/connext-dds/6.0.1/doc/manuals/connext_dds/getting_started/index.html (visited on 09/06/2023).

[79] Walid Saba. "Machine Learning Won't Solve Natural Language Understanding". In: *The Gradient* (2021).

[80] Mikel Sagardia et al. “A Platform for Bimanual Virtual Assembly Training with Haptic Feedback in Large Multi-Object Environments”. In: *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*. VRST ’16. Munich, Germany: Association for Computing Machinery, 2016, pp. 153–162. ISBN: 9781450344913. DOI: 10.1145/2993369.2993386. URL: https://doi.org/10.1145/2993369.2993386.

[81] Ryo Sakagami. “Cooperative Operation Framework for Heterogeneous Robotic Teams in Planetary Exploration”. MA thesis. University of Tokyo, June 2020.

[82] Ryo Sakagami et al. “ROSMC: A High-Level Mission Operation Framework for Heterogeneous Robotic Teams”. In: IEEE, June 2023. URL: https://github.com/DLR-RM/rosmc.

[83] Jeff Sauro and Joseph S. Dumas. “Comparison of Three One-Question, Post-Task Usability Questionnaires”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’09. Boston, MA, USA: Association for Computing Machinery, 2009, pp. 1599–1608. ISBN: 9781605582467. DOI: 10.1145/1518701.1518946. URL: https://doi.org/10.1145/1518701.1518946.

[84] Jeff Sauro and James Lewis. “Correlations among prototypical usability metrics: Evidence for the construct of usability”. In: Apr. 2009, pp. 1609–1618. DOI: 10.1145/1518701.1518947.

[85] Philipp Matthias Schäfer et al. “Flexible Robotic Assembly Based on Ontological Representation of Tasks, Skills, and Resources”. In: *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*. Nov. 2021, pp. 702–706. DOI: 10.24963/kr.2021/73. URL: https://doi.org/10.24963/kr.2021/73.

[86] Peter Schmaus et al. “Extending the Knowledge Driven Approach for Scalable Autonomy Teleoperation of a Robotic Avatar”. In: 2023, pp. 1–10. ISBN: 9781665490320. DOI: 10.1109/AERO55745.2023.10115960.

[87] Ishika Singh et al. “ProgPrompt: Generating Situated Robot Task Plans using Large Language Models”. In: (June 2022). URL: http://arxiv.org/abs/2209.11302.

[88] Roland U Sonsalla et al. “Field testing of a Cooperative Multi-Robot Sample Return Mission in Mars Analogue Environment”. In: 2017.

[89] Jork Stapel, Freddy Antony Mullakkal-Babu, and Riender Happee. “Automated driving reduces perceived workload, but monitoring causes higher cognitive load than manual driving”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 60 (2019), pp. 590–605. ISSN: 1369-8478. DOI: https://doi.org/10.1016/j.trf.2018.11.006. URL: https://www.sciencedirect.com/science/article/pii/S1369847818301335.

[90] Mohammad Mustafa Taye. *Understanding Semantic Web and Ontologies: Theory and Applications*. 2010. arXiv: 1006.4567 [cs.AI].

[91] Moritz Tenorth and Michael Beetz. “KNOWROB: Knowledge Processing for Autonomous Personal Robots”. In: *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IROS’09. St. Louis, MO, USA: IEEE Press, 2009, pp. 4261–4266. ISBN: 9781424438037.

[92] Govind Thattai and Qiaozi Gao. *Amazon releases code, datasets for developing embodied AI agents*. Mar. 2023. URL: https://www.amazon.science/blog/amazon-releases-code-datasets-for-developing-embodied-ai-agents (visited on 08/30/2023).

[93] Andreas Tobergte et al. “The sigma.7 haptic interface for MiroSurge: A new bi-manual surgical console”. In: Sept. 2011. DOI: 10.1109/IROS.2011.6094433.

[94] Surya Tokdar and Robert Kass. “Importance sampling: A review”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2 (Jan. 2010), pp. 54–60. DOI: 10.1002/wics.56.

[95] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].

[96] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[97] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[98] Jörn Vogel et al. "An Ecosystem for Heterogeneous Robotic Assistants in Caregiving: Core Functionalities and Use Cases". In: *IEEE Robotics & Automation Magazine* PP (Dec. 2020). DOI: 10.1109/MRA.2020.3032142.

[99] Jianyong Wang and Jiawei Han. "BIDE: Efficient Mining of Frequent Closed Sequences". In: *Proceedings of the 20th International Conference on Data Engineering*. ICDE '04. USA: IEEE Computer Society, 2004, p. 79. ISBN: 0769520650.

[100] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020. URL: https://github.com/huggingface/transformers.

[101] Kjetil Wormnes et al. "ANALOG-1 ISS – The first part of an analogue mission to guide ESA's robotic moon exploration efforts". In: *Open Astronomy* 31 (Jan. 2022), pp. 5–14. DOI: 10.1515/astro-2022-0002.

[102] Youxi Wu et al. "NOSEP: Nonoverlapping Sequence Pattern Mining With Gap Constraints." In: *IEEE transactions on cybernetics* 48 10 (2018), pp. 2809–2822. URL: https://api.semanticscholar.org/CorpusID:195667536.

[103] Ikuya Yamada et al. *LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention*. 2020. arXiv: 2010.01057 [cs.CL].

[104] Peter T. Yamak, Li Yujian, and Pius K. Gadosey. "A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting". In: *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*. ACAI '19. Sanya, China: Association for Computing Machinery, 2020, pp. 49–55. ISBN: 9781450372619. DOI: 10.1145/3377713.3377722. URL: https://doi.org/10.1145/3377713.3377722.

[105] Kaizhi Zheng et al. *JARVIS: A Neuro-Symbolic Commonsense Reasoning Framework for Conversational Embodied Agents*. 2022. arXiv: 2208.13266 [cs.AI].