Heriot-Watt University

Masters Thesis

The Design and Development of an Elderly Telecare Application Using a Cloud Telepresence Robot-Agnostic Development Framework

Author:
Md Rakin Sarder

Supervisor:
Dr. Mauro Dragone

A thesis submitted in fulfilment of the requirements for the degree of MSc.

in the

School of Engineering and Physical Sciences

July 2021

Declaration of Authorship

I, Md Rakin Sarder(Matriculation no: H00331973), declare that this thesis titled, 'The Design and Development of an Elderly Telecare Application Using a Cloud Telepresence Robot-Agnostic Development Framework' is submitted for the SSI+ Master Thesis Project.

I declare that the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. I have not copied other material verbatim except in explicit quotes. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: [illegible]

Date: 20/07/2021

“With beyond measure is a man’s greatest treasure.”

J. K. Rowling

Abstract

With the rise of the elder population around the world, the demand for active elder care and support is increasing. As the ratio of the number of care professionals with respect to the elderlies continue to decrease, people are looking for alternative solutions for assisting and supporting the elderlies to live longer, healthier and happier lives. This influx of the elderlies vs the caregivers is responsible for the growing interest in care telepresence robotics, which can offer the flexibility to offload some burden from the caregivers. With the help of telepresence robots, caregivers can reach out to the elderlies for providing remote care, have enhanced social interactions with them and monitor their wellbeing from a distance.

A remote operator can move, interact and engage with a telepresence robot in a remote environment without the need for them to be physically present there. With cloud integration, telepresence robots become manageable and accessible from anywhere around the world. Over the years, several cloud robotic frameworks have been proposed to develop robotic applications and services, however very few of them have been specifically designed to address the specifications and considerations of telepresence robotic application development. In this thesis, a microservice-based rapid prototyping framework for early-stage cloud telepresence robotic application development have been presented. The proposed framework is suitable for remote development and deployment over the cloud without the need for physical accessibility to a telepresence robot. Experimental tests to evaluate the framework's I/O communication channels was conducted to benchmark its performance. It was observed that the framework was compatible with a range of web communication protocols, while maintaining persistent, reliable and low-latency connection with a telepresence robot.

Using the proposed framework, a telepresence robotic application for elder care based on a social telepresence robot have been developed and deployed for caregivers to remotely connect to the elderlies. The proposed telecare robotic application was tailored to assist the caregivers to checkup on their patients and conduct remote consultation sessions using Pepper as the telepresence robot. A user-centric design approach was followed to design the application, and a Human Robot Interaction (HRI) study was conducted with professionals working in some elder care institutions to evaluate the developed prototype of robotic telecare application. Although the user evaluation study was conducted with a limited number of participants, they showed generally positive attitudes towards the application. The participants also provided feedback on some issues they observed during their evaluation sessions, and shared some ideas to improve the proposed telecare application.

Acknowledgements

Conducting this thesis was a challenging feat for me, especially due to being forced to conduct this thesis remotely. However, I have been able to handle the challenges successfully with the mentoring and inspiration of my supervisor, Dr. Mauro Dragone. He has supported tremendously throughout the course of my thesis, not just with concepts, advises and guidance, but also with hands-on advices on various parts of my thesis and also by helping me to conduct the laboratory sessions remotely during the lockdown. Usually it goes the opposite way, but I was really grateful to have Dr. Mauro as my supervisor.

I would like to specially express my gratitude to Scott Macleod and Moujahid Meriam for their relentless support for aiding my remote laboratory works, managing the laboratory equipment physically on behalf of me, despite being busy with their own projects. Without their help, it would have been much more difficult to complete this project within the deadline.

I am also very grateful to join and become a part of the CARE (Cognitive Assistive Robotic Environments) research group of Heriot-Watt University during this project. I have learned a lot from the group's weekly sessions, and received effective feedbacks on my project from the other members of the group.

Additionally, I would like to thank for inputs of Ronnie Smith, Dr. Mary Stewart and the participants of the Robotics+CARE mashup event to provide valuable feedback on my project. Their inputs contributed significantly towards the pathway of successful completion of this thesis.

Last, but not the least, I would like to express my warm gratitude to my parents and my sister, who supported me from far away by their love, encouragement and motivation.

Contents

List of Figures

List of Tables

Abbreviations

HRI	**H**uman **R**obot **I**nteraction
AAL	**A**mbient **A**ssisted **L**iving
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
VPS	**V**irual **P**rivate **S**erver
RetraDev	**RE**mote **T**elepresence **R**obotic **A**pplication **DEV**lopment framework
VM	**V**irual **M**achine
SLR	**S**ystematic **L**iterature **R**eview
ROS	**R**obot **O**perating **S**ystem
RSR	**R**obot **S**ervice **R**unner
UI	**U**ser **I**nterface
API	**A**pplication **P**rogramming **I**nterface
SSL/TLS	**S**ecured **S**ocket **L**ayer / **T**ransport **L**ayer **S**ecurity
URI	**U**niversal **R**esource **I**dentifier
URL	**U**niversal **R**esource **L**ocator
P2P	**P**eer to **P**eer
DNS	**D**omain **N**ame **S**ystem
JSON	**J**avaScript **O**bject **N**otation
RTABMAP	**R**eal-**T**ime **A**ppearance-**B**ased **M**apping
VPN	**V**irtual **P**rivate **N**etwork
RTT	**R**ound **T**rip **T**ime
WoZ	**W**izard **o**f **O**z

To my charming and lovely partner Nini, who has provided the most important contribution to my life, by staying at my side.

Chapter 1

Introduction

Due to the advancement of robotic technologies, robots are gradually becoming more and more incorporated with our daily lives. In the early days of robotics, robots were primarily used in manufacturing and production, however in today's world robots are being equally deployed outside the factories as well [1]. The increasing demand and participation of robots in our lives is raising the need for advancement in the domain of Human Robot Interaction (HRI). Researches are identifying new problem areas where robots can be included, with the focus on human values, necessity, perception, acceptance and adaptation.

One of the prospective fields where robots are making significant impact is elder care. According to "World Population Prospects 2020" by UN DESA, the current share of world population aged 65 and above is about 9.3%, and is projected to almost double by 2050 [2]. The UK and Europe has the second largest share (17.5%) of ageing population in the world, which is expected to increase even further up to 25% by 2050 primarily due to their low birthrate [3]. The inevitable rise of the ageing population lays both cultural and socio-economic challenges, and this phenomenon is eventually raising the demand for health and social care services for the elderlies.

Subsequently, the major challenge to the healthcare sector is to provide sufficient elderly healthcare professionals[4]. However, there is an increasing scarcity of care professionals compared to the elderlies in medicals facilities, rehabilitation centers and residential care housings. This shortage is eventually causing increased workloads on the care professionals, and reduces the quality of care [5]. Several studies have reported burnout among the care professionals, as they often suffer from exhaustion, frustration and lack of motivation [6, 7]. The mental and physical wellbeing of the care professionals is equally important in the healthcare ecosystem, as they impact on the quality of care [7].

Robotics and Ambient Assisted Living (AAL) technologies are addressing a number ageing-associated challenges as they reduce the dependency of older adults over others in their daily lives. Telepresence robotics, for instance, are enabling social inclusion, remote caring, health monitoring and various forms of assistance to the older adults [4, 8]. In a care ecosystem, the inclusion of telepresence robots can benefit both the older adult as well as the remote care professionals in the long run. Telepresence robot allows both the family members and care personnels to remotely contact the elderlies living alone, while also enabling them to move around and interact with the remote environment [9]. Hospitals and assisted living facilities are employing telepresence robots for remote companionship and assistance. The inclusion of social aspect on telepresence robotics can additionally provide a range of other care and support services, such as autonomous interactions, understanding emotional and mental state, and engagement with the elderlies during the absence of care personnel [10].

The prospect of social telepresence robots in elderly care opens a major challenge from the perspective of HRI: care professionals will need to be able to adapt and accept the inclusion of new telerobotic care systems. While all the focus for care delivery goes to the older adults, the acceptance and experience of the caregivers using telepresence robots for various care related activities are quite significant. Telepresence robotic systems usually equip remote users with a user interface with which they can move around with their robot on a remote setting, see through the robot's camera, consult their patients via teleconferencing and perform any additional operation supported by the system. The design quality of this user interface is important for a positive user experience of a remote caregiver [11].

Additionally, due to the occurrence of a global pandemic that started in 2020, remote care is not just an option anymore, it has become a necessity for many cases. Distance caregiving facilities for the care professionals is now more significant than ever; for their patient's safety from the pandemic and due to the social distancing rules imposed by the policy makers. Telepresence robot can play a crucial role in order to support the care professionals in this case. Several researches and projects have launched focusing on the inclusion of telerobotic care and nursing services to support patients and older adults during the pandemic [12].

Telepresence robots must enable the care professionals to access their patients remotely without any external help, which means they must be able to access and manage the robots (which are serving the older adults) remotely over a cloud network [13]. By the nature of their operation, telepresence robots rely on a network. The two core features of telepresence robots are: teleconferencing and teleoperation, both of which requires internet, through which data is transmitted from the robot's end to the remote user's end. Integration of cloud computing with telepresence robotics can bring several benefits, such as:

- Hosting telepresence robot operating application on the cloud as a service, which can be accessed by any remote user across any platform without installation.
- Send teleoperation commands over the internet
- Sharing workloads with the robot, especially tasks that may be computationally expensive for the robot (such as vision recognition, speech processing, SLAM)
- Host web communication channels for data streaming (A/V, sensors)

1.1 Research Problem

Different projects and researches had taken initiatives regarding the feasibility and execution of remote robotic system development in the past. Several frameworks have been proposed to address the issue of remote experimentation with robots, such as [14]. However, the significance of remote robotic development has risen significantly due to the ongoing global pandemic. The prospect of remote robotic development is not just valuable for researchers who cannot be physically present with the robots due to restrictions, but also for care-centric applications, where a designated robotic expert may not be available within the robot's vicinity. However, very few of these proposed frameworks address the development of care-based cloud telepresence robotic solutions. A cloud-based remote telepresence robotic development framework addressing the need and challenges of elderly care can be beneficial for researchers and professionals to rapidly prototype, test, develop and maintain care telepresence robotic applications.

It is expected that the inclusion rate of telepresence robotics in elder care will increase over the next few years [15]. Remote caregivers and relatives are expected to make more interactions with older adults through these telepresence systems. Most of these interactions through these robots are to be made without any expert interventions over the cloud. As a result, researchers and developers need to opt for user-centric designs [16]. The perception, choice, viewpoints and acceptability of remote users in this context (caregivers, relatives, clinicians) must be reflected upon a complete care robotic solution. Care telepresence applications or services running on the cloud might exhibit network-related or operational errors. Additionally, the telepresence user interface might be too complex for the remote users to use, and the organization of different elements on the user interface might intimidate some users, especially users with technical self-efficacy. The issues may not just be limited only within the aforementioned context. Therefore, it is important to learn from the experience and evaluation of the remote users of care telepresence applications through a standard user evaluation model.

A standard telepresence robot usually requires manual operation from a remote user. By their designs, they are often coined as a "tablet on a wheel" [17]. Communication through telepresence robots have been proven effective over its handheld video conferencing counterpart. However, as more users started to use telepresence robots, focus on more useful features such as the social aspects of telerobotic communication are becoming more significant [18]. In addition to the manual teleoperation, researches are exploring and identifying scopes for autonomy within these robot. Internal robotic behaviors, such as autonomous navigation, person recognition, gaze tracking can augment and supplement the already existing telepresence systems, and can be beneficial for elder care robotics [19]. Instead of the conventional "tablet on a wheel" telepresence robots, a humanoid robot with a tablet screen for teleconferencing has the scope to provide autonomous functionalities and an enhanced social participation for the remote users [20]. However, we need to identify and understand what social or autonomous features of a humanoid telepresence robot are actually plausible to be designed and prototyped using a remote telepresence development framework, where a physical access to the telepresence robot is not possible.

1.1.1 Research questions

Considering all the facts and problems identified in this section, the following research questions have been identified and investigated in this study:

RQ-1: What is the state of the art in care-based telepresence robotics?

RQ-2: What can be an ideal cloud-communication framework for remote programming, testing and application prototype development for telepresence robotics?

RQ-3: Besides teleoperation and teleconferencing, what additional telehealth features needs to be added in a telepresence robotic application to aid remote caregiving tasks?

RQ-4: How can we measure the quality of experience of caregivers when they use a robotic care telepresence User Interface (UI) to access their patients or relatives?

1.1.2 Extended research questions

As this project evolved, the scopes of the primary research questions were expanded to explore the connotations generated from the conducted experiments and literature study.

The extended research questions are as follows:

ERQ-1: Does the usage of a social robot for telepresence bring additional benefits for elderly care applications?

ERQ-2: What are the autonomous features of a humanoid telepresence robot which can contribute to elder care?

ERQ-3: What are the possible constraints robotic engineers have to face when they work with a remotely located telepresence robot?

1.2 Proposed Methodology

The core focus of this thesis was to design a cloud-robotic framework for telepresence application development, and to use it for developing a telepresence robotic application with a user interface tailored for elderly care. A user-centric design approach was adopted in this research in order to learn from the experience of care professionals, who were given a UI for experiencing. For the telepresence robot, Pepper v2.5 was used. Pepper is an intelligent humanoid robot developed by Softbank Robotics, with a tablet screen attached to its chest. Considering these points and the research questions, the following works have been done:

1. **Literature review:** A literature review was conducted partially following the guideline of a systematic review approach. The idea for following a systematic approach was to properly construct the review methodologies and organize the whole review process for future reference. At the beginning, literature on the background concepts on elder care, ambient assisted living and telepresence robotics were studied. This was followed by reviewing the current state-of-the-art of telepresence robotics used in elder care, discussions on some frameworks for cloud-based robotic development, and various communication protocols available for remote data transmission.

2. **Investigating the suitability of communication protocols for remote programming and testing with Pepper:** As mentioned in the extended research

question, the major challenge that was addressed was the remote execution of this whole study. Due to the COVID-19 pandemic, laboratory facility was not accessible due to the lockdown imposed by the Scottish government. As a result, the robot was not physically accessible. Various communication protocols and architectures have been tested and analyzed whether they could support establishing bi-directional communication channels with Pepper.

3. **Designing a cloud telepresence robotic framework:** The next phase of the project involved designing a cloud telepresence robotic framework, which was named as the **RetraDev** framework. This phase involved designing the architecture of the proposed framework, identifying its internal sub-structures, components, data flow and processes.

4. **Developing a care telepresence application using the proposed framework and communication protocols:** Through the literature study and peer suggestions, several issues and requirements for remote care were identified. In this phase, these findings were used to design a care telepresence application "HWU Telecare". The proposed framework in the previous point was used to develop this application.

5. **User evaluation:** Following the completion of HWU Telecare, a pilot study was conducted with professionals caregivers where they participated into testing a proposed UI of the telecare application. Their feedback, acceptance and quality of experience were analyzed and reported in this thesis.

To identify the user type of a telepresence robot, the following conventions has been used throughout this dissertation:

- **Remote user (operator):** The user who is controlling and using a telepresence robot from a distance (not physically present with the robot). For example: care professionals, relatives, teachers, physician etc.

- **Local user:** The user who is physically present with a telepresence robot. For example: elderlies, patients, children etc.

These two conventions will be very significant to understand the different concepts, definitions and constructs presented in this dissertation.

1.3 Thesis outline

The dissertation is organized as follows: Chapter 2 presents the literature review associated with this study. Chapter 3 presents the proposed cloud telepresence robotic framework **RetraDev** for remote programming and development. This includes a detailed objective of the proposed framework **RetraDev**, followed by its architecture, requirements and methodology. The section concludes with an experimental evaluation of the framework.

Chapter 4 presents the design and architecture of ”HWU Telecare”. Details regarding its structure, components and development process have been presented with technical reference drawn from Chapter 3.

Chapter 5 presents the design of the user evaluation experiment for HWU Telecare. It includes the goals, criteria and the model constructed for the evaluation of the proposed care telepresence robotic system. The content also includes information about the participants, and the tasks they were assigned during their experience sessions. Lastly, this section presents the findings and outcomes from the pilot user evaluation study, and confers the positive experience and acceptance among the participants. Some written feedback provided by the participants for improving HWU Telecare have also been presented.

Chapter 6 concludes this study by summarizing the whole dissertation and addresses the research questions. It also discusses some limitations of the current research and the future direction of this project.

Chapter 2

Literature Review

2.1 Introduction

Smart technologies combined with robotics are being developed to provide care and service to the elder population over the last decades. As people grow older, they start loosing their physical and cognitive abilities, and they gradually find performing their daily tasks increasingly difficult. Old age often brings functional dependencies of the elderlies over others, which can bring significant challenges over their caregivers.

Elder care is becoming a prospective field for robotics and smart home applications. Some of the major challenges that fall in this field are:

- Health and wellness monitoring
- Assistance to household/personal chores
- Rehabilitation
- Ambient Assisted Living (AAL)
- Remote social interaction

2.2 Systematic review approach

The literature review was conducted in a systematic way in order to review the literature regarding cloud-connected telepresence robots and their suitability in elderly care. The main goals to fulfill the review process are:

- Study and construct the base of knowledge on robotic telepresence systems, cloud computing in robotics and Ambient Assisted Living (AAL) in care-giving applications.
- Review the current state-of-the-art in this domain based on the research questions.
- Combine the base knowledge and the current state-of-the-art to validate the defined project scope and derive the suitable methodologies.

A systematic literature review (SLR) is a detailed research process of derivation, selection and analysis of literature in order to model solutions to some research questions or topics.

However, in this project a systematic methodology is being followed in order to structure the review process for this project, rather than fully emphasizing only on an SLR research. This will allow to clearly distinguish the knowledge base and current trends, and to build up a solid resource base for this project.

2.2.1 Review Methodology

As already discussed, the project aims to focus around mobile elderly services using assistive technologies. Evaluating the existing domestic robotic technologies and their state-of-the-art is crucial to build up a knowledge base to address this aim and to find solutions to the research questions. Another major focus for the review process was to determine the lifecycle and future directions of this project.

The review stage followed the standard procedure of a systematic review process set by the PRISMA-P (Preferred Reporting Items for Systematic reviews and Meta-Analyses for Protocols) guideline [21]. The PRISMA-P guideline was prepared with the goal of making the compilation and publishing of systematic review methods through a defined protocol. It contains a 17-item checklist of recommendations to include in a plan for systematic review. In this literature study, some of items from the PRISMA-P checklist was followed. Focusing primarily on the the current state-of-the-art, study selection was conducted in two separate stages. They are:

- **Publications from 2018-2021:** Recent technologies and evaluation of the current state-of-the-art.
- **Publications before 2018:** Study and construct the base of knowledge on robotic telepresence systems, cloud robotics and robotics in care-giving applications.

2.2.1.1 Criteria for study selection

Given the objectives of the literature study, a multi-stage approach was used for the selection of the publications. The first stage of the process involves defining the key topics of search. Analyzing the research questions, a set of primary topics for research selection were extracted as shown in Table 2.1. Each topic has been coded with a topic number for further references throughout this paper:

Topic Code	**Topic name**
TP1	Service-based robotic applications
TP2	Gerontechnology
TP3	User Interface (UI) for telepresence robotics
TP4	Remote robotic assistance & monitoring
TP5	Cloud connected robotic technology
TP6	Ambient Assisted Living (AAL)
TP7	Emergency response in residential care
TP8	Cloud framework for robotics
TP9	Communication protocols and robotics

Table 2.1: Search criteria topics

The main purpose of deriving the primary topics was to find the relevant search terms and phrases in different scientific databases. The next step of the selection criteria involves fusing these set of topics together to form search terms and look for the returned relevant results. The selection criteria was squeezed down from a larger set to a smaller subset after fine tuning through several trials.

Different combinations of keywords were performed in order to derive the suitable search terms. The following databases were chosen to conduct the literature study (Table 2.2):

Database name	**Access type**
Google Scholar	Open access
ProQuest	HWU Institutional Access
IEEE Xplore	HWU Institutional Access
ScienceDirect	HWU Institutional Access
HWU Discovery	HWU Institutional Access

Table 2.2: Databases chosen for the literature study

The databases were chosen primarily based on the indexed publications related to telepresence robotics, cloud systems and their different applications. Also, publication indexes and Heriot-Watt University's institutional accessibility were significant in their selection process.

After performing some several preliminary search trials, the following search phrases were derived (Table 2.3):

Search terms and phrases	**Topics covered**
Care robotics	TP1, TP2, TP4
Robotic (adoption during) covid-19 pandemic	TP4, TP7, TP6
Elderly adults service robots	TP1, TP2
Care telepresence human robot interaction	TP1, TP2, TP7
Cloud based service robotics elderly	TP1, TP2, TP5
Elderly care telepresence HRI	TP3, TP6
Remote control and monitor of telepresence robot	TP4, TP5
Cloud infrastructure home service robots	TP1, TP5
Communication protocol robotics	TP4, TP5, TP9
Cloud robotics framework	TP8
Telepresence robotic development frameworks	TP8, TP9
Telepresence robot user interface/ UI for teleoperation	TP3

Table 2.3: Search keywords and phrases used for the literature review based on the topics selected. The keywords and phrases were mixed and combined to form different search patterns

The resultant search phrases cover all the criteria topics defined earlier. As we can see from the table above that emphasis was given on the keywords ”cloud”, ”elderly care”, ”telepresence” and ”framework”. However, effective search phrases were formed with the combination of the keywords in order to narrow down the context. Search trials using vaguely broad contextual search phrases such as ”robotics”, ”cloud service” were not conducted.

After narrowing down the search phrases, the search results were accumulated. ”Jabref” and ”Publish or Perish” applications were used to search and export the bulk result in .bib format [22, 23]. These two tools covered all the databases except HWU Discovery. Since the literature review loosely followed a systematic approach, a limited number of studies have been reviewed, which had high relevance to this project. This have been achieved using the aforementioned softwares. These softwares helped to extract refined results from the databases, and grouped them to be exported altogether.

2.2.1.2 Exclusion criteria

The following criteria were considered for excluding a study:

- Articles not written in English language
- Vaguely broad discussion about a certain topic, such as ”robotics”, ”cloud service”
- Telepresence or care robots which are entirely theoretical (Design/testing phase is not complete)

- Articles which discusses works or technical solutions not relevant to the research topics defined earlier
- Survey reports which focuses primarily on the context of social science rather than the defined topics

2.2.1.3 Study screening

The screening process was performed into steps. Each step screens out studies which do not meet the defined criteria. The first step was to de-duplicate the publications [24]. The second and the third step involve evaluating the title/abstract and the full content of the publications. The final step was to assess their eligibility based on the research questions and criteria for study selection. A summary of this approach step by step with the result is given in Figure 2.1.

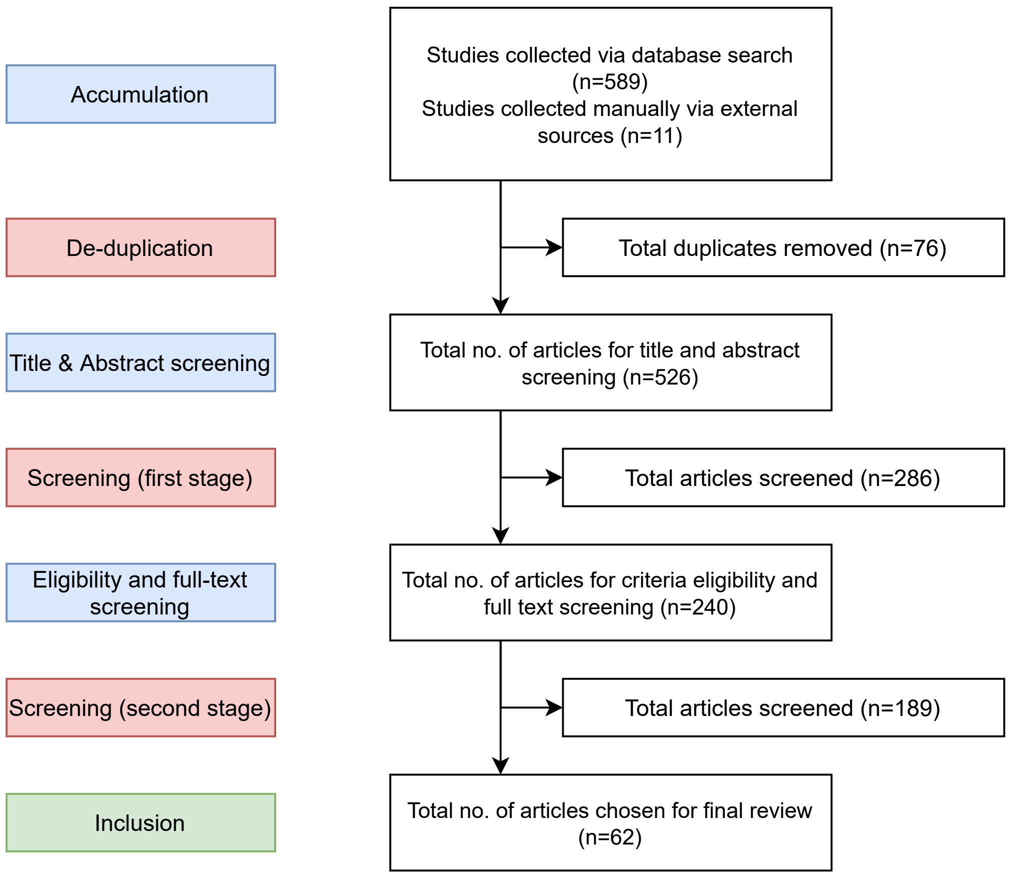

Accumulation
Studies collected via database search (n=589)
Studies collected manually via external sources (n=11)
De-duplication
Total duplicates removed (n=76)
Title & Abstract screening
Total no. of articles for title and abstract screening (n=526)
Screening (first stage)
Total articles screened (n=286)
Eligibility and full-text screening
Total no. of articles for criteria eligibility and full text screening (n=240)
Screening (second stage)
Total articles screened (n=189)
Inclusion
Total no. of articles chosen for final review (n=62)

FIGURE 2.1: Flowchart of the screening process following the PRISMA-P guidelines.

2.3 Background review

2.3.1 Ambient Assisted Living (AAL)

Gerontechnology is an emerging field of research and practice that comes from the terms gerontology (scientific study of aging) and technology [25]. It deals with the technological interventions in the environment and activities of older adults. The primary goal of gerontechnology is to produce a sustainable and quality lifestyle for the elderlies with the help of technology so that they can live an independent life and can experience easier social participation [26].

The design focuses of gerontechnological innovations and solutions can be characterized in two ways: (1) inclusive (target-specific) design, and (2) assisted technological design [16]. Ambient Assisted Living (AAL) technology describes the latter focus of gerontechnology. The AAL concept was founded in 2008 by the AAL Programme in Europe [27]. It is derived from Ambient Intelligence, a model of Information and Communication Technology (ICT) that enhances the capabilities of people through the inclusion of context-aware and adaptive digital environments [28, 29]. AAL solutions may vary from user to user depending on their unique scenario, however the primary goals for any AAL based solution are [29, 30]:

- Enable the older adults to increase their social engagement and reduce isolation
- Allow older adults to reduce dependency over others so that they can live independently
- Empower the older adults to preserve their health
- Increase their quality of living

Besides the primary goals, there are some other goals which have been identified. They are [29, 31]:

- Creation of support networks around the older adults
- Enhance security, provide faster emergency response service
- Consolidate caregivers, healthcare professionals and family members into the care framework
- Enhancing user interfaces of different AAL components to make them easier and intuitive to the older adults

- Analyze the mental wellbeing of the older adults and provide sufficient mental health support

Any AAL technological solution consists of several components which work in conjunction to each other. These can be categorized as follows:

- **Smart home (Domotics):** Smart homes collect and analyze rich context information to automate various operations and provide services in a home for the residents [28]. On a basic level, it comprises of different types of sensors inside a house to collect user and local context information. In case of elder care, smart homes focuses on monitoring the residents, assess their health, welfare and provide assistance in their Activities of Daily Living (ADL) [31].
- **Mobile and wearable sensors:** Advancement in MEMS technology and smart phones have allowed researchers and professionals to design context aware applications to monitor end-users [28]. Different communication systems, such as ZigBee, Bluetooth LE, NFC, WiMedia can be integrated with user's body area network (BAN) to collect sensor data and transmit them outside the body (beyond BAN) to doctors and professionals via middleware systems [32, 33]. Smartphones these days are packed with multiple sensors which can be used to accumulate data, as well as use the smartphone's computational capability to analyze them [34]. Several sensors of smart phones, such as accelerometer, gyroscope, GPS, proximity can be used to monitor the mobility of an older adult and track their activities. Wearable sensors such as ECG, glucose sensor, blood sensor monitors heart activity and acts during heart attack, keep diabetes level under control and check blood coagulation respectively [35]. E-textiles or fabric sensors can noninvasively collect physiological data and monitor health condition [28].
- **Assistive robotics:** Assistive robots actively help the older adults to overcome their physical and cognitive constraints that come with their age via assisting with their day to day activities [16]. Assistive robots can assist with ADLs, home equipments and with personal and social activities. For example, assistive robots can fetch objects, such as medicine or glass of water for the elders [36], pick up dropped object by the elders, can allow the older adults to telecommunicate with their families and friends, assist the elders into engaging into different activities such as gardening, crafting, home organizing [37].

There exists several ethical and acceptability challenges while deploying an AAL system. Researchers and engineers should enquire about the feasibility, preferences and specific issues from the older adults and the caregivers before deploying an AAL technology, such as smart home. Moreover these criterion vary from individual to individual, the same solution may not be suitable for everyone. For example, Deepika et. al reported that some older adults were reluctant to reside in an AAL setting because they would become less active physically [31]. In another study, Heek et. al pointed out that technical self-efficacy of the caregivers have direct impact on the perception and attitude towards AAL technology [38]. Privacy concern, data security and fear of surveillance also create restraint among the caregivers [29].

Table 2.4 gives a brief overview about some of the concept, challenges and requirements of Ambient Assisted Living (AAL) technologies from the older adults point of view [30, 29, 28, 31, 39].

Concept	**Challenges**	**Requirements**
Using ICT for improving the quality of living of the elderlies	Older adult's age, gender, physical and cognitive capabilities demand simplicity, ease of use and non-stressful design	User-centric design
Reduce dependency on others	Concern about becoming less physically active	Intuitive age-friendly user interfaces
Assist with the ADLs	Technical self-efficacy and skepticism towards using new technologies	Designed solution must not disrupt or intervene user's activities negatively
Track the health and well-being using context-aware tools	Concern about privacy and personal security	Maintain data privacy as well as individual's privacy
Enhance security, communication and reduce social isolation	Reluctant to use complicated solutions	Solution must be equipped with adequate communication facilities

Table 2.4: Overview of AAL: concept, challenges and design requirements from older adult's perspective

Since the caregivers and health professionals often fall under a standard care model for the elderlies, issues and challenges concerning them are quite significant as well. Table 2.5 gives a brief overview about some of the concept, challenges and requirements of Ambient Assisted Living (AAL) technologies from caregivers' point of view [36, 40, 41, 38, 30].

Concept	**Challenges**	**Requirements**
Using ICT for reducing workloads from caregivers in a elder/disabled care setting	Caregivers feel some technical solutions interrupt with their care work	Designed solution must not overlap on crucial care tasks
Allow caregivers to monitor and analyze physical and mental health state of the older adults	Technical self-efficacy (TSE)	Intuitive user interfaces suitable to be used by people with low TSE
Assist caregivers to organize and schedule various care-centric operations	Privacy concern and reluctant to be under surveillance	Caregiver's privacy, digital as well as physical system security of AAL should be considered

Table 2.5: Overview of AAL: concept, challenges and design requirements from caregiver's perspective

2.3.2 Care robotics

As the ratio of the ageing population to the healthcare and well-being service provider continues to increase, the demand for elderly welfare technologies to address this challenge become more significant [42]. This rapidly growing demographic challenge is addressed by welfare technologies, which refers to the technologies that ameliorate the life and the quality of living for the elderly and handicapped population. Welfare technologies are deployed primarily into their private accommodations, and allow the users to live an independent life with reduced difficulty and risk by offering various micro and macro services, social stimuli and entertainment [43, 44].

Care robotics is a subset of service robotics that falls under the domain of welfare technology [45]. They are used either as a personal robot or as a family caretaker robot that may be operated autonomously or semi-autonomously by a user to provide different assistive services and support which are not limited within the medical domain [46, 47].

One of the earliest care robot prototype designed for elder and disabled care was the Health Care Robot (HCR) by NASA Jet Propulsion Laboratory (JPL) and Real World Interface, Inc. (RWI) [48]. Since then, care robotics has been developing over time in order to add more autonomy, suitability, stability and adaptability. Care robots are gaining significant popularity in elder care and their use has been seen to benefit the caregivers and the family members as well as the elderlies [49]. In the past, more focus was given to the controller mechanism, the user interface, communication and navigation of the robots. At present, due to the advancement of smart systems and robotic technologies, large range of use cases for care robots are explored and analyzed. According to [6, 50, 51], some of the major use cases of care robotics for elder care are:

- Cognitive training
- Facilitating communication
- Remote monitoring of health and daily activities
- Companionship
- Entertainment
- Guidance
- Emergency support

When designing a deployable care robot, some primary technical requirements should be considered. The robots must be adaptable to the deployed environment [52]. The physical structure of a robot must be accessible and acceptable by the elderlies, and they must maintain safety when operating near the user's vicinity. In case of teleoperation, suitable fail-safe mechanism must be present [53]. The robots must maintain privacy and security during remote communication and teleoperation [54].

Besides the technical challenges, robotic developers also need to consider various health, social and ethical challenges [55, 56, 57]. Johansson et al. reported complicated challenges robots face in terms of user attitude and perception towards them [42]. Hudson et al. has observed users have often found difficult to accept robotic care systems, especially the aged people, as some of them tend to be critical to approve new technologies in their daily lives [58]. Some studies have also found that caregivers and health professionals sometimes find it difficult to accept the integration of care robots in their care institutions [42, 59].

Table 2.6 gives a summary about the concept, challenges and requirements of care robotics in elder care [6, 50, 55, 56, 57, 54].

Concept	**Challenges**	**Requirements**
Special type of service robotics used for patient and elderly care	Cost and workload	Chosen/designed robot must be adaptable to the environment
Can be autonomous/semi-autonomous	Stigma and ethical challenges	Affable physical design
Provides assistance, social stimuli, helps user to be more independent	Fear of high maintenance	The robot must be easy to be reached and to be interacted with
Can be used for responding to emergency situations	Ergonomic risk with HRI (Human Robot Interaction)	Fail-safe mechanism
Monitor the health and activities of the elderlies	Operator inefficiency may impact the service provided by care robots	Privacy and security

Table 2.6: Overview of care robotics concept, challenges and design requirements

2.3.3 Telepresence robotics

Telepresence robots allow users to create an ambient presence of themselves from their current location to a remote location as if they are present there. At their core, robots adopting telepresence technology can provide remote connection to a user located at a distant location, allowing them to interact with the remote environment, navigate freely within the robot's vicinity and communicate with the users located there without the need of their assistance [17]. Telepresence falls under the broad category of telerobotics, which also includes teleoperation [60]. The concept of telepresence was first proposed by Marvin Minski of MIT on 1980 [61]. Telepresence robots started manufacturing commercially from the 2000s, while primarily focusing on the medical application field [62]. As the microelectronics, computing and communication technologies developed over the years, telepresence robotics sector continued to develop rapidly, with focusing on a wider range of applications [63].

In an ideal telepresence robot architecture, there are four components, as shown in Figure 2.2 [17]. At the top level, there is Mobile Remote Presence system (MRP). It contains the whole architecture of the telepresence system where the robot communicates with the remote operator (pilot user) over a bi-directional communication channel. The robot is equipped with with a video conferencing system which is typically powered by a tablet and optionally with external cameras. Different sensors and actuators are added to the robot for navigational, operational and monitoring purposes [64]. The pilot user control and interact through the robot via a user interface, which primarily consists of the video conferencing channel, navigation controls of the base and the head, maps

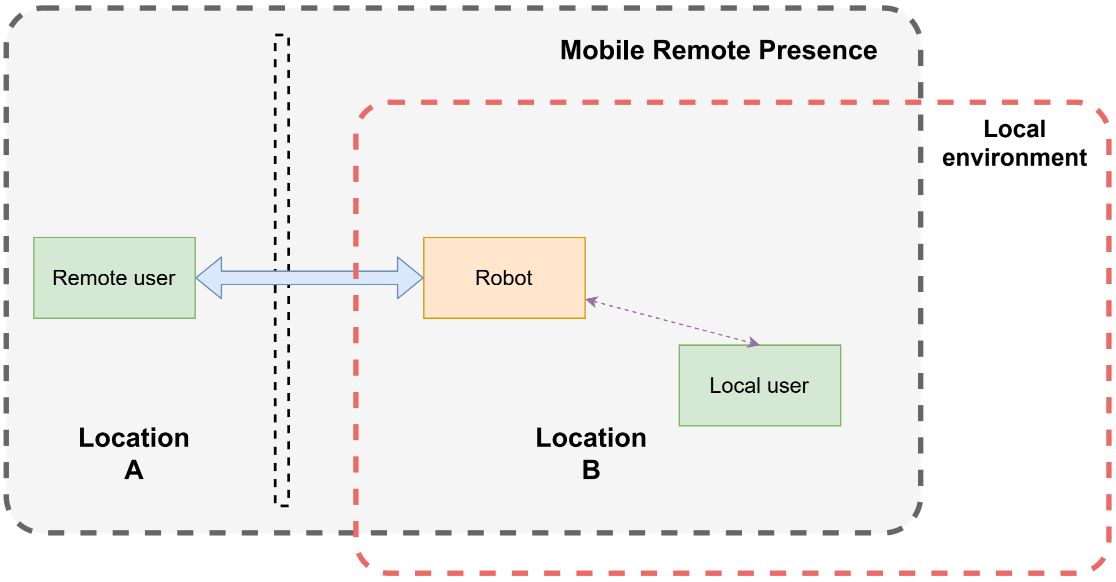


FIGURE 2.2: Top-level view of the architecture of a telepresence robotic system

and readings from the remote sensors on the robot. Most of the telepresence robots are constructed in such a way that their heights approximately matches the height of a regular human (starting from 118.1 cm) [65]. This physical construct of a telepresence robot creates an physical ambience of the remote operator, thus making it appealing and easier to be interacted with by the local users. The top mount of a telepresence robot is capable of rotating along its axes, which allows the pilot user to look around the robot's surroundings, capturing a wider area of video feed.

Telepresence robots come with a variety of designs and functions, which are mostly determined by the desired uses and implementations. Based on various studies on the design considerations and user requirements, the key functionalities and features that a telepresence robot needs to have are [62, 8]:

- Robust maneuverability
- Controllable
- Wide field of view
- Smooth navigation with or without operator intervention
- Interactive A/V control and transmission
- Video conferencing mount should be set on at least 0.6m ground height
- Electric autonomy
- Compact and appealing physical structure

- Connectivity fail-safe mechanism

Beside the technical challenges, telepresence robots need to consider an ethical guideline for development. Marketta et. al pointed out that local users to a telepresence robot such as older adults and caregivers are skeptical about the fact that the robot might be eavesdropping in the background [66]. They also pointed out privacy concerns among the residents and the care professionals. In another study, docking operation was found out to be a challenging factor for the residents [9]. Older adults sometimes feel anxious around the robot, because of the fear of scaring movement of the robot in their proximity [9].

Table 2.7 gives a brief overview about some of the concepts, challenges and requirements of telepresence robotics. While developing this overview focus was given on the context of elder care [17, 8, 66, 9].

Concept	**Challenges**	**Requirements**
Creates a maneuverable physical persona of a user located at a remote location	Technical self-efficacy of any care worker may create burden for them	Efficient maneuverability and internal collision avoidance system
Establishes a video conferencing system with a local-to-robot user	Limited access and control for the local-to-robot users	Low latency and reliable connection for video conferencing and controls
Navigation can be controlled manually or semi-autonomously	Fear of eavesdropping	The robot needs to have efficient power supply, docking and charging mechanism
Allows remote user to interact with a remote locality	Privacy concern	Robot's view angle should be high for effective operation
Can collect remote context data for monitoring and analysis	Scaring movements of the robot make local-to-robot users agitated	Physical construct must not have any harmful edges
Can collect remote context data for monitoring and analysis	Fully network dependent, which makes it vulnerable to malfunctions due to network connectivity errors	Fail-safe mechanism

Table 2.7: Overview of telepresence robotics concept, challenges and design requirements

2.3.4 Cloud computing in robotics

Cloud robotics is one of the advancing robotic technologies that is being empowered with the rapid development of internet and communication technologies. To define it broadly, cloud robotics is a special branch of robotics where robots are not limited within their internal memory, sensing, controlling and computational capability, rather they utilize different micro and macro remote services deployed on the cloud to perform different operations [67]. This implies that cloud-connected robots are able to perform resource-intensive services and accomplish tasks with much higher efficiency without worrying on the limitations of its own hardware system. This feature allows cloud-connected robots to perform large-scale ubiquitous operations. Since cloud connected robots can utilize memory stacks of a cloud infrastructure, they can record and store large amount of data which can also be analyzed in the cloud, making information much more accessible to the users [68]. Cloud-connected robotic networking operates on the basis of Rapid Provisioning and Releasing (RPaR), and the resources (e.g., storage, GPS positioning, vision recognition services) can be accessed on-demand by the robots themselves [69]. Generally, a cloud robotic architecture consists of a cloud infrastructure on the top-level and the connected robotic features at the low-level (figure 2.3). This cloud infrastructure contains servers, memory stacks, rendering engines, database services and other components. Essentially, incorporation of cloud services to a robot breaks down its single-module architecture which reduces the computational load on it by distributing various operations to the cloud.

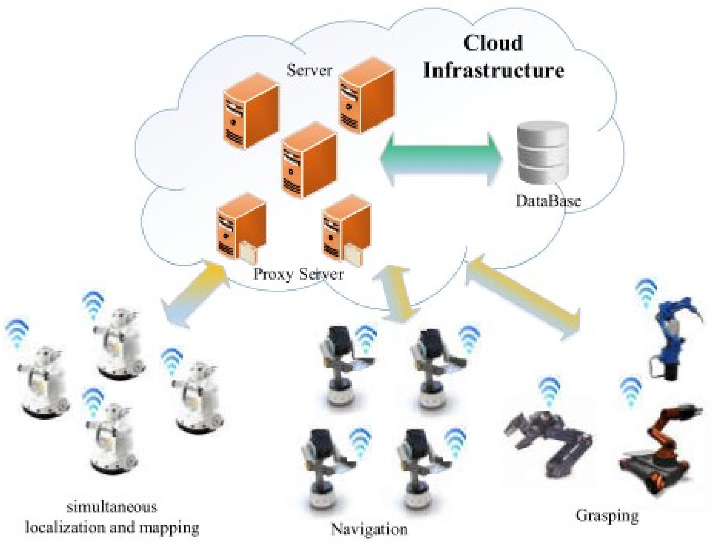


Figure 2.3: System architecture of cloud robotics [68]

The concept of cloud robotics first came into the industry on 1994 [67]. In its early years, cloud robotic solutions focused on teleoperation and local communication. As the internet and global communication technologies continued to develop, cloud systems started to become more available to developers and end users, leading to the further advancements of cloud robotics. Curated cloud architectures specifically for robotics started to develop. One of the global scale projects was Roboearth, which launched in 2009 [70]. At present, there exists several cloud robotics frameworks, PaaS and SaaS in today's world, such as RoboBrain, Rapyuta, ROS, Noos and C2RO [67, 70, 71, 72]. Globally available cloud services such as AWS, Google cloud, IBM Watson provides different robot-specific SaaS solutions which can be utilized by robots to perform operations such as SLAM, motion planning, speech recognition, task planning, vision recognition, scheduling and monitoring. Since cloud services are accessible to anyone, developers can build their own cloud robotic frameworks and applications using open-source softwares such as ROS (Robot Operating System) [72]. The benefit of using such softwares is that developers do no need to reinvent the wheel of architectural development of robotics, rather they can focus on the application and service level for a robot.

Cloud-connected robots, however, has some limitations. The major limitation of cloud robotics is that it can only perform operations that are non-real time [68]. This means that tasks such as motion control, collision avoidance and homeostasis cannot be performed using cloud services, as they heavily relies on real-time feedbacks and controls. Also, robots relying heaving on cloud services are heavily affected by latency and connectivity issues, thus impacting their overall quality of services [73]. Any mission-critical operation of a robot, therefore, needs to be processed internally. The key functionalities and considerations that a cloud robotic system needs to have are [67, 73, 74]:

- Resource allocation and scheduling of operations
- Cloud security
- Precise structure or schema of the data used during robot-cloud interaction must be defined
- Communication process must be optimized to achieve near real-time operation (load balancing)
- Implementation of service quality assurance mechanism

Table 2.8 gives a brief overview about some of the concepts, challenges and requirements of cloud robotics [73, 68, 72, 70].

Concept	**Challenges**	**Requirements**
Robots can utilize the power of HPC (High Performance Computing) over cloud to perform resource-intensive operations	Trust and security issues	Effective security measures, such as data encryption, hacking attack prevention, strong firewall must be implemented
Real-time tasks can be performed within the robot while non-real time operations can be performed in the cloud	Limited to mostly non-real time tasks	Communication system between a robot and a cloud platform must be optimized
Reduces overload on a connected robot by distributing tasks across remote cloud services	Connectivity and latency issues may cause data corruption, robotic malfunction	Robot-to-cloud or cloud-to-robot communications must be made as close to real time as possible
Robots can access and store big data over the cloud	Inconsistent structure of data received/sent during robot-cloud interaction cumulatively may cause major disruption	Effective load balancing
Allows user to monitor a robot's state and activities	Technical issue on the cloud end directly affect a robot's quality of operation	A good practice is to utilize open source and open access platforms, and contributing to the collective learning on cloud robotics

Table 2.8: Overview of cloud robotics concept, challenges and design requirements

2.4 Recent works and discussions

In this section the state of the art of AAL, incorporation of robotics with AAL, cloud connected robotics and telepresence robotics will be discussed. The core focus in this state-of-the-art study was given on elderly care and care services. The exclusion criteria mentioned in section 2.2.1.2 was followed, such as the omitting telepresence robots not designed physically yet. Contents are sectioned according to the significance and relevance to this project.

2.4.1 Advancement of robotic telepresence in elderly care

At present, different types of telepresence robots have been developed and utilized in care applications. These robots can be classified in three broad categories based on their development stage. They are:

1. Commercially available robot
2. Robot under development stage/recently launched for commercial/application level deployment
3. Conceptual robots

Some of the examples of commercial telepresence robots are Ava, Double, PadBot, Giraff, VGO, Pepper and Ohmni (figure 2.4. The main benefit of using a commercial robot is that engineers and software developers can directly jump on the application level of development, instead of focusing on designing the robot's mechanical structure, internal hardware system and the software framework to operate it. As a result, these robots have been found to be used in different care-centric (including clinical and non-health elderly care) studies and applications. A comparative study of the features of these robots have been presented in table 2.9.

At present, there are several robots have been recently launched and some are being planned to be launched commercially. For example, focusing on the ongoing COVID-19 pandemic situation Lio of F&P robotics (Poland) has been designed and launched [12]. It is a personal assistant robot which can also work on disinfecting nursing homes [12]. The robot runs on ROS (Robot Operating System), allowing researchers and developers to access its sensor streams, actuator controls and build custom solutions easily [75]. A DIY robot creation framework called LHF Connect was launched in 2020 focusing on the pandemic scenario [76]. LHF Connect, specifically designed for remote communication and teleportation, can be used by researchers and developers to build their own robot using its libraries and other open-source robot software frameworks. This modular DIY robot has been piloted and tested in a medical facility for collecting user surveys and responses [77]. GoBe robot has been announced to be launched by Blue ocean robotics (Denmark) with the focus on social inclusion and reducing carbon emission by remote travelling. The robot consists of a 21.5" screen to reproduce pilot user's face, with a set of sensors on-board for obstacle avoidance, people detection and navigation. [78].

Several research works have been working on designing application-centric telepresence robots. Gabriella et. al proposed ROBIN, a telepresence robot as a part of the GIRAFF-PLUS project, which focuses on monitoring an supporting the older adults [79]. The robot is integrated into the GIRAFFPLUS ecosystem, which is essentially an integrated sensor network based smart-home. The robot claimed enhancement in its communication module, which includes sending and receiving text message to provide health and wellbeing advice to the older adults through collecting context data from them and their surroundings. The robot also proposed a feature called "Shared space", which allows caregivers or pilot users to see physiological and personal data of the older adults on their screen beside the video stream coming from the robot. This enables the care professionals to engage into a consultation with the older adult regarding their health and their activities. ROBIN robot also proposed enhancement in the controls for the secondary users (older adults), via gesture and speech command recognition services. Brennan et. al proposed VROOM, a telepresence robot framework where the main focus was given on the representation of the user [80]. Local user is equipped with an AR interface, which overlays the telepresence robot with an avatar of the pilot user. The pilot user is equipped with a VR interface to get an immersive presence in the local environment and also to record the pilot user's head and hand movement to be mimicked to the avatar being seen by the local user.

Telepresence robot for care and social inclusion application can also benefit older adults as the primary users, as demonstrated by ChiCaro by Abe et. al [81]. ChiCaro robot project focuses on increasing remote communication and interaction with toddlers or babies and their grandparents. Chicaro's height is 350 mm, matching the height range of babies and crouching toddlers. It is equipped with miniature hands capable of waving children and receiving small objects. Telepresence robots have also been used considering the mental health and emotional benefit. One such example is Pudu robot, which has been designed using a rapid prototyping method and tested with the focus of mitigating social isolation for quarantined, socially isolated patients in hospitals and clinics [82]. Built on ROS, the robot is equipped with mounted tablet and Raspberry pi 4 as its CPU.

Figure 2.4 shows all the robots that have been reviewed and compared for this study.

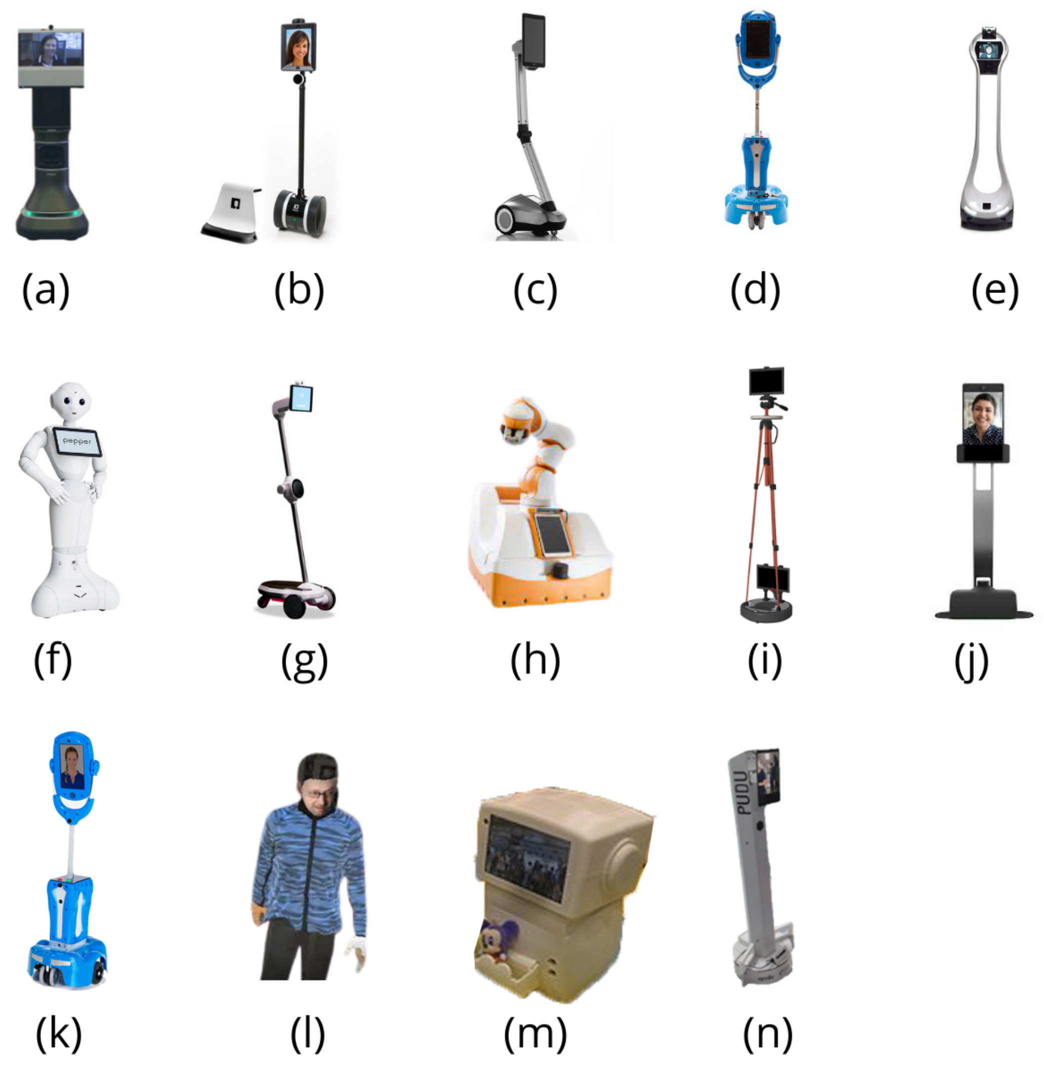


FIGURE 2.4: Telepresence robots reviewed in this study: (a) Ava 500 [83] (b) Double 2 [84] (c) Padbot P2 [85] (d) Giraff [10] (e)Vgo [17] (f)Pepper [86] (g)Ohmni [87] (h) Lio [75](i) LHF Connect [76] (j) GoBe [78] (k) ROBIN [79] (l)VROOM [80] (m) ChiCaro [81] (n) Pudu [82]

Robot	**Production**	**Unique features**	**Applications**	**Limitations**	**Selected reference**
Ava 500	Commercial	Autonomous p2p navigation, 3d mapping, auto docking	Office teleconferencing	Too expensive, not suitable for mass level care application	[83, 88]
Double 2	Commercial	Adjustable height, self-driving, navigation over video	Remote communication, remote health monitoring and consultation	No end effectors, Lacks compelling physical representation	[84]
PadBot P1, P2	Commercial	Adjustable height, tilt-able, auto docking, anti-falling	Patient/elder escort, remote monitoring, remote communication	No end effectors, Requires external iOS or android tablet, Lacks compelling physical representation	[89, 85]
Giraff	Commercial	Vertically rotating head mount, intuitive user control interface	Remote companionship, remote communication, health monitoring and consultation	No end effectors, on-board computer is limited, No internal obstacle avoidance	[10, 90]
Vgo	Commercial	Vertically tiltable camera, cliff sensor	Elder monitoring and escort, remote family visiting, health monitoring and consultation	No end effectors, fixed height, No internal obstacle avoidance	[17, 91]
Pepper	Commercial	Humanoid, emotion perception, on-board AI, integrated chest tablet, autonomous life	Remote companionship, personal assistant, robotic therapy, cognitive training	Scaring arm movement, poor navigation on rough surface	[92, 93, 86, 94]
Ohmni	Commercial	Horizontal and vertical tilting camera, adjustable height, additional navigational camera	Secured remote visits, telemedicine support, interactive remote conversations with gestures	Becomes completely non-operative during hibernation mode	[87, 95]
Lio	Recently launched	Multifunctional arm, Autonomous item pick and transport, autonomous navigation, on-board AI	Telemedicine service, nursing assistant, health monitoring and consultation	Low-placed tablet makes it challenging for remote telepresence, bulky	[75, 12]

LHF Connect	Recently launched	DIY robotic kit, cheap, navigational camera, open-source	Remote medical service, health monitoring and consultation, remote visit	No end effectors, all setups (including cloud) needs to be done by a professional	[76, 77]
GoBe	Recently launched	Robust physical structure, navigation camera, autonomous navigation	Health monitoring and consultation, remote visit	No end effectors, Fixed height	[78]
ROBIN	Conceptual	Enhanced gesture and speech recognition, ”Shared space” real-time ata visualization over remote communication	Older adult monitoring in private homes, social inclusion	No end effectors, can only provide virtual assistive services	[79]
VROOM	Conceptual	Immersive telepresence using VR and AR interface, virtual avatar overlay	Remote communication with enriched represenatation, augmented gesture and posture transmission	No case studies found for care application	[80]
ChiCaro	Conceptual	Small height, miniature fixed hands	Elder to children interaction, remote visiting	Only suitable for babies and toddlers as the local user	[81]
Pudu	Conceptual	Vertically tiltable head mount, built using open-source	Mental and emotional health monitoring, remote consultation	No end effectors, fixed height, only manual transmission	[82]

Table 2.9: Comparison of some telepresence robots identified in this study focusing on different aspects related to telepresence robotics and older care

A summary of some telepresence robots identified in this study has been presented in Table 2.9. Applications and other aspects of the robotic system that does not go with telepresence or elder care was not included. We can see that in the case of most of the robots, there is a lack of interaction and interactive communication capability due to the absence of robotic arms. Generally, a human use their arms and other body parts to make perceptive communication with other. Several case studies related to older adults have mentioned the difficulty into perceiving telepresent remote user’s speech, since in case of telepresence the remote user focuses on their facial display to the local user [92, 96]. Also, the lack of an end-effector in a robot also implies that the robot itself is not suitable for various physical operative tasks. Alongside this issue, several other case studies involving older adults and caregivers have also identified the lack of stair climbing ability as a major drawback to these telerobotic systems[97]. Due to the lack of stair climbing ability of these robots, they are limited to operate only on its current floor, unless someone manually carries them to another floor. This issue is also challenging for any user who carries them, as some robots may not be easy to be carried upstairs in a residence.

From the table, we can also see that robotic representation can play a significant role in the Human Robot Interaction (HRI) between an older adult and a telepresence robot. In order to make such telepresence robots acceptable and adaptable to the older adults, robot designers should focus on making the robot physically appealing, robust, safe and self-manageable.

2.4.2 Communication frameworks for network-based robotic communication

One of the key challenges that researchers and professionals have to address while designing robotics researchers and professionals use various software communication frameworks for robotic application developments. These technologies define the communication models and patterns among the different entities of an architecture, such as publisher-subscriber model, client-server model, request-reply model and pipelines. Some of these communication frameworks are discussed below:

1. **ROS:** ROS (Robot Operating System) is robotics suite that acts as a middleware among the hardware and software components of a robotic architecture. It is the de-facto standard for robotic application development. The ROS framework uses a graph-based computational model for structuring the various entities and services of a robotic system. It uses a publisher-subscriber model for its ROS Topics, and client-server architectures for its ROS Services and Actionlib.

   However, ROS is not a native cloud robotic communication system, it functions more as a central processing unit inside a robotic architecture. To incorporate ROS into a cloud robotic framework, an external communication transport mechanism is required. One of the common communication transport option for ROS is rosbridge.

2. **Rosbridge:** Rosbridge or "rosbridge_suite" is a ros package available in the ROS repository. It is a JSON-based API service to allow bi-directional communication between ROS and non-ROS programs over websocket protocol [98]. A Standard ROS architecture with Rosbridge running is given in figure 2.5. Rosbridge_suite contains the package to run the websocket server on this ROS side, and on the client side it provides developers roslib libraries, which can be used to develop client-side applications. Using roslib, client side applications can publish/subscribe to various rostopics, access/create ros services and also create action clients or action servers.

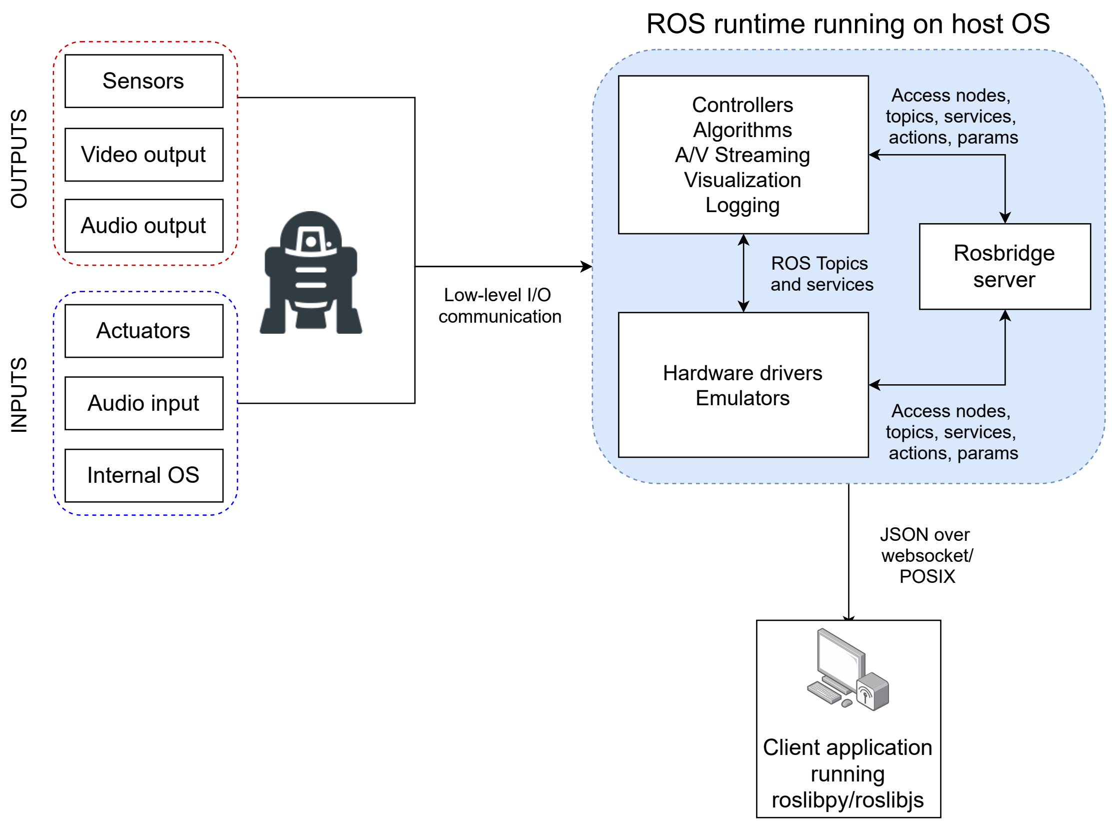

ROS runtime running on host OS
OUTPUTS
Sensors
Video output
Audio output
INPUTS
Actuators
Audio input
Internal OS
Low-level I/O communication
Controllers
Algorithms
A/V Streaming
Visualization
Logging
Access nodes, topics, services, actions, params
ROS Topics and services
Rosbridge server
Hardware drivers
Emulators
Access nodes, topics, services, actions, params
JSON over websocket/ POSIX
Client application running roslibpy/roslibjs

Figure 2.5: Standard ROS architecture with Rosbridge running

Rosbridge is actively used by researchers and professionals to develop network-connected robotic applications [98]. Since ROS is a de-facto standard for robotics research and development, rosbridge is more convenient to be paired with ROS for network-based communication. Robotic solutions often require to integrate user interface with the robot. For instance, Wilson et. al used rosbridge to integrate web user interface and an enhanced Turtlebot robot running ROS to provide elder residents of a smart home to have enhanced interactions with their environment [99]. Pavón-Pulido et. al proposed an exoskeleton robot based tele-rehabilitation system for older adults [100]. They used rosbridge over internet to integrate the user interface of a therapist with ROS running on the exoskeleton robot equipped by their patient. A Kinect equipped on the therapist's side could capture RGB-D stream of the therapist's lower limb motion, which is transmitted through rosbridge to convert into joint commands of the robot running on ROS. Transmitting teleoperation control in telepresence robots is a common usage of rosbrige, as mentioned in [101].

3. **ZeroMQ**: ZeroMQ is an open-source queue-based socket-based networking library focusing on performance, concurrency and atomicity [102]. It is a message-based middleware similar to ROS, however ZeroMQ does not have any platform dependency like ROS (official ROS supports Ubuntu and Debian only), as it is easily embeddable and it supports a wide-range of programming languages and platforms. This allows developers to use ZeroMQ for building custom middleware solutions for different applications, such as game engines, financial services, robotics and IoT [102]. ZeroMQ socket runs on ZMTP (ZeroMQ Message Transport Protocol) for peer-to-peer message transportation [103].

   Various studies and works have been reported to use ZeroMQ for robotic applications. ZeroMQ can be paired with a ROS project to significantly enhance the performance of remote communication. ROS topics are primarily based on a single publisher/subscriber model archetype over thr TCPROS transport layer protocol. ZeroMQ has several messaging patterns with multiple socket types available for remote communication, such as pub/sub, request-reply, pipeline, exclusive pair, client-server and radio-dish. It can go stackless (inproc:// or ipc://) and also run distributed grid computing processes (tipc://). Coronado et. al proposed an open-source, distributed component-based robot-programming library NEP (Node Primitives) for enhanced Human Robot Interaction (HRI), which is based on ZeroMQ and nanomsg [104]. Danter et. al proposed a lightweight middleware system as an alternative to ROS, which is suitable for very low-latency robotic applications [105]. They use the ZMTP and the UDT protocol for designing their networking middleware.

4. **Socket.io**: Socket.io is a Javascript based event-driven, bi-directional web communication library. It runs upon Engine.io, and it uses a collection of transport protocols, such as long-polling and websockets to establish reliable connections [106]. Aside from connection reliability, socket.io also provides built-in message broadcasting, detecting disconnection and automated reconnection, multiplexing support and an API service for developers to build socket.io based applications [107]. Unlike ZeroMQ or ROS, Socket.io is not a middleware technology. Apart from Javascript, socket.io also supports some other programming languages, such as Python, Java, C++, Dart and others. As a result, socket.io is not limited only to web-centric solutions, but also a wide-range of cross-platform solutions.

Socket.io is actively used for designing network-based robotic applications. It has been reported to design standalone communication libraries for ROS without using rosbridge, such as Rowma ROS [108]. It can be used to establish ”handshaking” in WebRTC services, which can provide fault-tolerant and stable relays of A/V streams in a telepresence robotic system [14]. Jin et. al from Lenovo proposed a desktop robot OYaYa combined with a web dashboard for interactive emotion management through recognition and imitation [109]. They implemented a client-server model for its software which uses socket.io for its internal communication due to socket.io’s event-driven architecture.

5. **Animus**: Animus is a non-ROS, cloud-based universal robot programming framework and connectivity service developed by Cyberselves [110]. Animus provides cross-platform client SDKs (in Python and Unity) for robot-agnostic coding, which enable developers to write portable codes entirely for the client side [110]. Animus drivers are installed inside the robot’s internal OS, which pairs up with an Animus client running on a user’s machine via the WebRTC protocol.

   Since the Animus coding framework is robot-agnostic, same code written for one robot of one vendor may be used for another robot of another vendor. Animus divides the sensors and actuator channels of a robot into different sets of *modalities*. This adds a high-level abstraction over the robot’s internal drivers and channels, which makes the coding process simpler for remote development. Animus maintains a very low latency connection so that any tasks or commands send to the robot execute efficiently without much delay. For their data interchange format, Animus uses Google’s Protocol Buffer (protobuff), which is an open source library for serializing structured data [111]. Protobuff is a binary transport format, which makes it significantly smaller in size and lighter than JSON, XML or YAML, as a result the bandwidth and memory requirement is significantly optimized.

   Animus *modality* may have either a *get_modality* or *set_modality* function. For an output channel, Animus provides a *set_modality* function to send commands with data, while for an input channel it provides *get_modality* to receive context information collected using a robot’s sensors.

   The current development of Animus supports three commercial social robots (Pepper, NAO and Miro-E) and cross-platform client SDKs. The project is currently under development, with limited documentation and examples available. Since the framework is proprietary, developers need to additionally learn the proprietary coding structure of the library to implement robotic algorithms with it. The current iteration of the Animus driver does not support all of the component I/O channels mentioned in their documentation for some robots(for example, *proprioception* and *homeostatis* modalities mentioned in Animus documentation is not currently available for Pepper). Features are being constantly added in the new updates of Animus.

2.4.3 Cloud robotic platforms

There are several cloud robotic platforms existing in today's world. These platforms often uses known robotic frameworks such as ROS (Robot Operating System) at their core. Based on accessibility of their architecture and source types, these platforms can be either open-source, or commercial. Some of the active cloud robotic platforms who operates on the basis of PaaS (Platform-as-a-Service), SaaS (Software-as-a-Service) or both reported to be used in care robotics are discussed below:

1. **RAPP:** Funded by the European Commission as a research project, RAPP is an open source cloud platform for developing and delivering care robotic solutions [112]. The core development of RAPP was based on targeting people who are at risk of exclusion, especially the older adults. RAPP targets solutions for robotic software developers and healthcare professionals who intend to design and employ robots with healthcare-centric applications [113]. Beside application services, they also provide infrastructure for developing and deploying cloud robotic applications focusing on the needs of elderly people. The web-interface of RAPP is called the RAPP store, which allows users to register their robots to the platform and access the platform services [112]. The core of the platform runs on ROS, which powers various services, such as computer vision, ontology integration, cognitive exercises, path planing and others. Rosbridge websocket server establishes the primary communication bridge between the robot platform and the cloud platform web server, which hosts the API endpoints for the web services. RAPP also employs API services to handle low-level robotic operations. In addition to this, RAPP provides Docker-based containerization service for developers for creating and testing programs, process executions and services which are not part of RAPP Improvement Center (RIC) [112].

2. **Rapyuta**: Rapyuta is a secured and distributed cloud computing platform and framework based on elastic computing model [71]. It has three layers of communications, a set of functions and a group of commands that the user may use to control the system. Since the basis of the service calls is regular HTTP requests, they can be used by non robotic solutions as well. Calling services are similar to invoking basic functions, making it much easier for users to use. Finally, utilities can be accessed via the framework's python or C++ API client libraries.

Rapyuta is open-source, and provides open-access to its repositories. Similar to RAPP, Rapyuta also utilizes ROS for its own framework operations and middleware system [71]. However, non-ROS clients can also access Rapyuta service over rosbrige-websocket protocol. In addition to that, Rapyuta platform provides infrastructure services through its Rapyuta Console.

3. **Robot web tools**: Robot web tools is a software framework which consists of a set of modules and components for developing web-based robot applications based completely on ROS [114]. Unlike Rapyuta or RAPP, robot web tools does not provide a platform as a part of its service, rather it provides the libraries, server-side ROS packages, application stacks and widgets for visualizations and navigation. Using these modules and components, users can build their own cloud robotic platform using their preferred cloud infrastructure. Although it is an open-source project, the libraries and other components are actively maintained by the community. Robot web tools also provides support for non-ROS programs to connect to its ROS-based ecosystem using the rosbridge library [98].

   Various studies have used robot web tools for designing socially assistive robots. Short et. al proposed SPRITE, an interactive platform robot for tabletop engagement focusing on the health and wellness of children and adults via social engagement without contact [115]. Lil'Flo is a telerehabilitation system using a social assistive hybrid robot constructed from NAO and Vgo [116]. They used the robot web tools and ROS project as the backend core to develop the user interface, WebRTC ROS streaming channel, 3D visualization and also to acquire robot transformations. Phillip et. al proposed an assistive teleoperable manipulator for people with motor impairments which uses robot web tools for establishing communication between the client application and the robot, as well as stream video from the robot to the client application [117].

Apart from the aforementioned cloud robotic platforms and frameworks, there are some other active platforms as well, such as C2RO Cloud Robotics, AWS Robomaker, Microsoft Robotics Studio (MSRS) and The Construct [118, 119]. C2RO is a commercial solution and their proprietary source is not open-accessible. C2RO focuses primarily on industrial automation and smart technologies driven by AI-powered data analytics. MSRS, another non-ROS based solution, is based on the .NET framework. They provide a Service-Oriented Architecture (SOA) for robots, however they are not being actively maintained by Microsoft anymore due to Microsoft's new focus on ROS. The Construct provides a complete cloud ROS solution, providing pre-configured server infrastructure to build and ROS projects, including RViz visualization services, Gazebo simulation engine, gtest, SLAM and others [120]. However, The Construct is not open-accessible and is focused primarily towards robotics education based on ROS.

2.4.4 Design considerations of User Interfaces (UI) for telepresence robots

UI design is a key factor to deliver an optimal QoE (Quality of Experience) for the users of any telepresence robot [121]. The user interface must be easily operable by the remote user to deliver them an effective ambience presence. Researchers and robot application developers focus on designing platform-independent UI using web technologies, as it reduces software compatibility or permission-related issues [14], although some commercial telepresence robot developers design their own companion apps targeting specific platforms, such as iOS or Android [122]. Several studies have been conducted over the years regarding the development of the remote user's interface. Telepresence UI components can be classified based on their significance of usage, as shown in figure 2.6 [121, 123, 11].

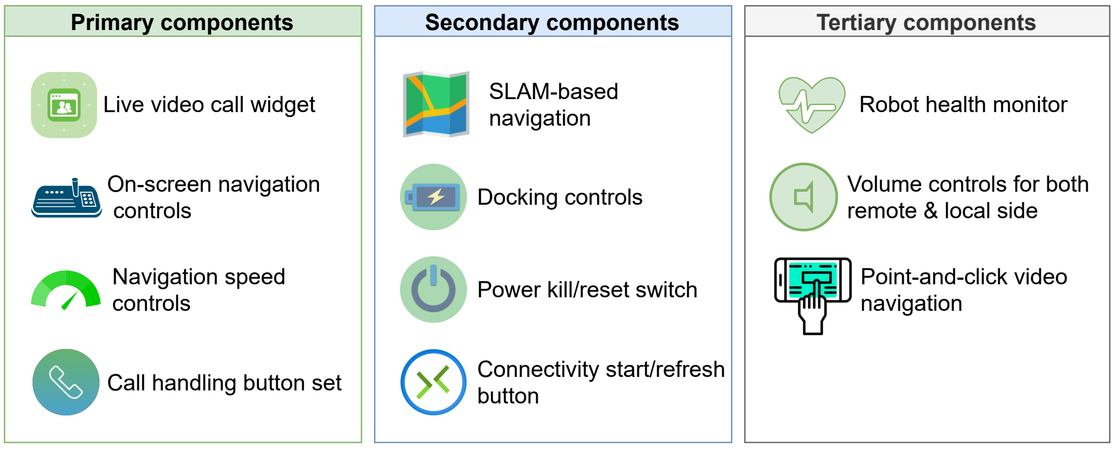


FIGURE 2.6: Standard UI components of the remote operator of a telepresence robot classified according to their significance of usage

The primary components provides the basic standard of telepresence experience to the remote user. Although maps can be significant for effective localization and navigation, many telepresence robot developers focuses on full-screen video streaming, video resolution and the quality of the A/V stream, and opt out of including map-based navigation. An alternative to map-based navigation can be point-and-click video navigation, as mentioned in [121]. Point-and-click video navigation can support the autonomous navigation capabilities for a telepresence robot. However, with point-and-click remote users can only navigate within the viewing angle of the robot. Also, precise positioning and controlling is not possible during point-and-click mode, unless the chosen navigating point by the user is far away from the robot's localized position. A hybrid navigation control using on or off screen controls and point-and-click video navigation can give a better experience to the user [121]. Although some designs opts for on-screen navigation controls (which is essential for touch-based devices), off-screen navigation controls using keyboard or joystick have been reported to be quite convenient to various groups of remote users as well [124].

Apart from navigation and A/V components, robot developers also focus on displaying a robot's state to the remote user [13]. Information regarding a robot's health, such as battery percentage, internal temperature and sensory activities can be helpful for the remote user to make decisions prior to execute any task in the local environment [13]. Another useful component can be volume control to remotely manage the audio volume at the robot side as well as the remote side. Additionally, some developers add UI options for localized robots for commanding them to commute to their docking station or their home position [19].

UI design consideration must be made in case of the local users as well, who interact with the remote telepresent user via the tablet or screen attached to the robot [123, 79]. In most cases, these tablets can be running client applications for teleconferencing and remote control. Local users may need to allow/end remote teleconferencing calls or call remote users by themselves, control the active state of the tablet, reconfigure the tablet's network connectivity and do other operations. The tablets can also be used to show important information and contents to the user. Telepresence robot paired with certain telemedicine equipments can measure and monitor patient's vital signs and health data, and this can be displayed to a local user through the robot's tablet as mentioned in [13]. If a social robot is paired with a telepresence system, then local users can request the robot to provide various information, such as weather status and daily news which the robot can display to the user through its tablet [125]. This allows an enhanced HCI (Human-Robot Interaction) for the local user.

Chapter 3

RetraDev (REmote Telepresence Robotic Application DEVelopment) Framework

This chapter discusses the proposed RetraDev framework for developing cloud-based telerobotic services and applications. Continuing up with the discussion from Chapter 1, one of the key goals of this study was to find a suitable way for working with telepresence robots without having any physical or local network access to them. The idea was to develop a software and a networking infrastructure that can provide rapid prototyping capability to abstract over any telepresence robotic hardware for programming real-time and non-real-time operations remotely, allow interfacing and application development.

3.1 Objectives

The current state-of-the-art and the limitations of some of the robotic frameworks available for remote robotic development have been discussed in Chapter 2. RetraDev was designed to provide a scalable, flexible and easy-to-use development framework specifically for telepresence robotic applications. The main design goals and features of the RetraDev framework are:

3.1.1 Telepresence robot-agnostic development

RetraDev framework was designed to configure a telepresence robotic development infrastructure on the remote developer's side, without having any dependencies on the telepresence robot's side. This have been achieved by implementing a set of API end-points, event-driven data transmission protocols and suitable communication models.

3.1.2 Locally hosted cloud application development

RetraDev will allow developers to run various robotic operations as services on their own computers, instead of requiring to develop, configure and deploy services and applications on a cloud server on a continuous basis, using its LocalCloud module. By using RetraDev LocalCloud, users will be able to securely expose their in-progress telepresence robotic applications to their peers for usage and testing, establish I/O communication channels with their robots over the internet, all from their own computers. This will reduce the time, complexity and resources required for DevOps operations, and essentially allowing developers to focus more on developing services for their telepresence robots.

3.1.3 Modularity

RetraDev framework breaks down each of its primary components into separate modules called *microservices*. Microservicing allows the RetraDev framework to distribute the computation and tasks concurrently, since each microservices run independently from each other. RetraDev provides the communication mechanism among the microservices using both API (client-server) and websockets (publisher-subscriber).

3.1.4 Open-source and Lightweight

RetraDev framework is lightweight, with minimum dependencies. RetraDev is an open-source framework, all of its source code are available on GitHub (see Appendix for the Github repository links). At its core, it runs on NodeJs and Python 3. Robotic developers with NodeJs and Python 3.x installed can immediately start designing telepresence robotic applications and services using the RetraDev framework. The installation and configuration step of the RetraDev framework is provided in Appendices A and B.

3.2 Framework Architecture

A top-level view of the RetraDev framework is given in Figure 3.1. As discussed already, the RetraDev framework was designed focusing on rapid prototyping and deployment of telepresence robotic applications from the localhost of a developer's computer. The core cloud components of the RetraDev framework are: LocalCloud and Bastion Server.

LocalCloud is the backend server which runs a set of microservices corresponding to various telepresence robotic operations. It also handles the low-level operations of the architecture (communications, database, non-RT algorithms). RetraDev LocalCloud is exposed over the internet to be accessed by a user via Bastion tunneling [126], or SSH port forwarding. In the case of RetraDev framework, the Bastion server serves the purpose of DNS (Domain Name Server) resolving, caching, SSL/TLS host and CDN (Content Delivery Network). Through the assignment of static URIs, each microservices running on LocalCloud are given a fixed access URIs, which makes other services of this cloud architecture to access each other. Applications deployed using the RetraDev framework are served to the users and subscribers over the HTTPS protocol. The core communication protocol of RetraDev is Socket.io [107].

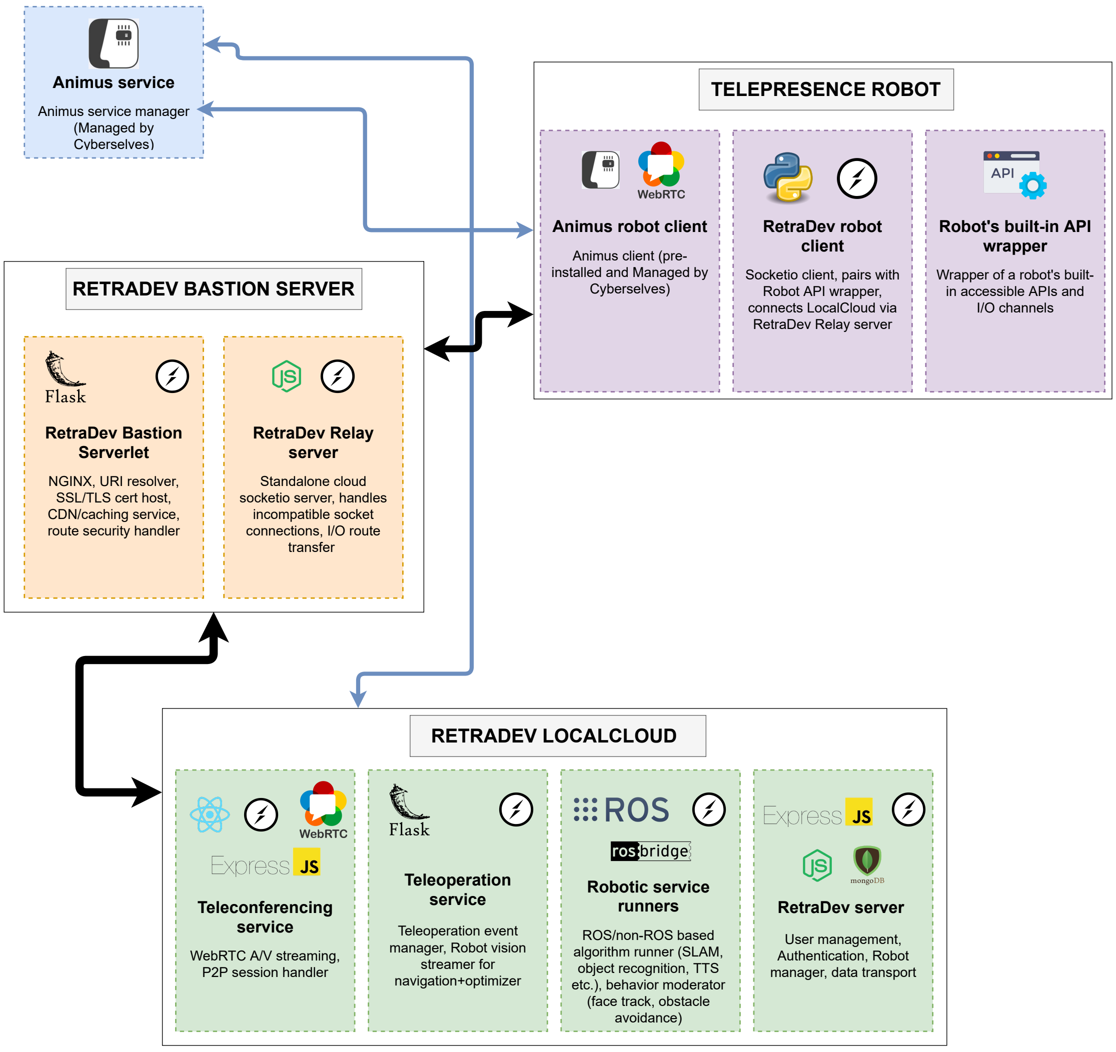

Animus service
Animus service manager (Managed by Cyberselves)
TELEPRESENCE ROBOT
WebRTC
Animus robot client
Animus client (pre-installed and Managed by Cyberselves)
RetraDev robot client
Socketio client, pairs with Robot API wrapper, connects LocalCloud via RetraDev Relay server
API
Robot's built-in API wrapper
Wrapper of a robot's built-in accessible APIs and I/O channels
RETRADEV BASTION SERVER
Flask
RetraDev Bastion Serverlet
NGINX, URI resolver, SSL/TLS cert host, CDN/caching service, route security handler
RetraDev Relay server
Standalone cloud socketio server, handles incompatible socket connections, I/O route transfer
RETRADEV LOCALCLOUD
WebRTC
Express JS
Teleconferencing service
WebRTC A/V streaming, P2P session handler
Flask
Teleoperation service
Teleoperation event manager, Robot vision streamer for navigation+optimizer
ROS
rosbridge
Robotic service runners
ROS/non-ROS based algorithm runner (SLAM, object recognition, TTS etc.), behavior moderator (face track, obstacle avoidance)
Express JS
mongoDB
RetraDev server
User management, Authentication, Robot manager, data transport

FIGURE 3.1: Top-level view of the RetraDev architecture

3.3 Top-level components

3.3.1 LocalCloud

LocalCloud sits at the core of the RetraDev framework. As already discussed, LocalCloud runs a set of microservices to operate various telepresence tasks. Microservices are a collection of small, self-contained services which runs in parallel to each other in a loosely-coupled fashion (have limited dependencies over the other services). Each microservice runs on their designated communication protocols. The concept of microservice-based framework designs have been adopted in several studies and works. For example, Panchea et. al proposed OpenTera, a robotic microservice framework for rapid prototyping and development of robotic application with telehealth capabilities, targetting the COVID-19 pandemic [13]. In OpenTera, microservices communicate with each other using REST API and WebRTC. Ercolano et. al proposed a a set of social robotic behaviors as a set of robotic microservices aimed for the care and engagement with the patients of Dementia [127].

At the current design version of RetraDev LocalCloud, there are four microservices in its architecture (figure 3.1). They are discussed below:

3.3.1.1 RetraDev server

RetraDev server is the central unit of the whole RetraDev architecture. Although defined as a microservice, RetraDev server acts as the central node to all other microservices, serverlets, communication clients and relay server. The internal architecture of the RetraDev server is shown in Figure 3.2. All the elements of the RetraDev architecture either directly or indirectly report and receive messages from the RetraDev server. It is built upon NodeJs, ExpressJS and MongoDB.

The core features of the RetraDev server are discussed below:

1. **Data transport**: RetraDev server runs the central websocket server over the Socket.io protocol that provides the websocket endpoints (socket.io events), and also the REST API endpoints to handle JSON based HTTP/HTTPS requests. The websocket endpoints work upon a *publisher-subscriber* model, while the API endpoints work upon a *client-server* model. The API endpoints are useful for developers to design front-ends for their telepresence applications, while the websocket endpoints allows them to streamline real-time communication on the front-end with other components of the RetraDev architecture.

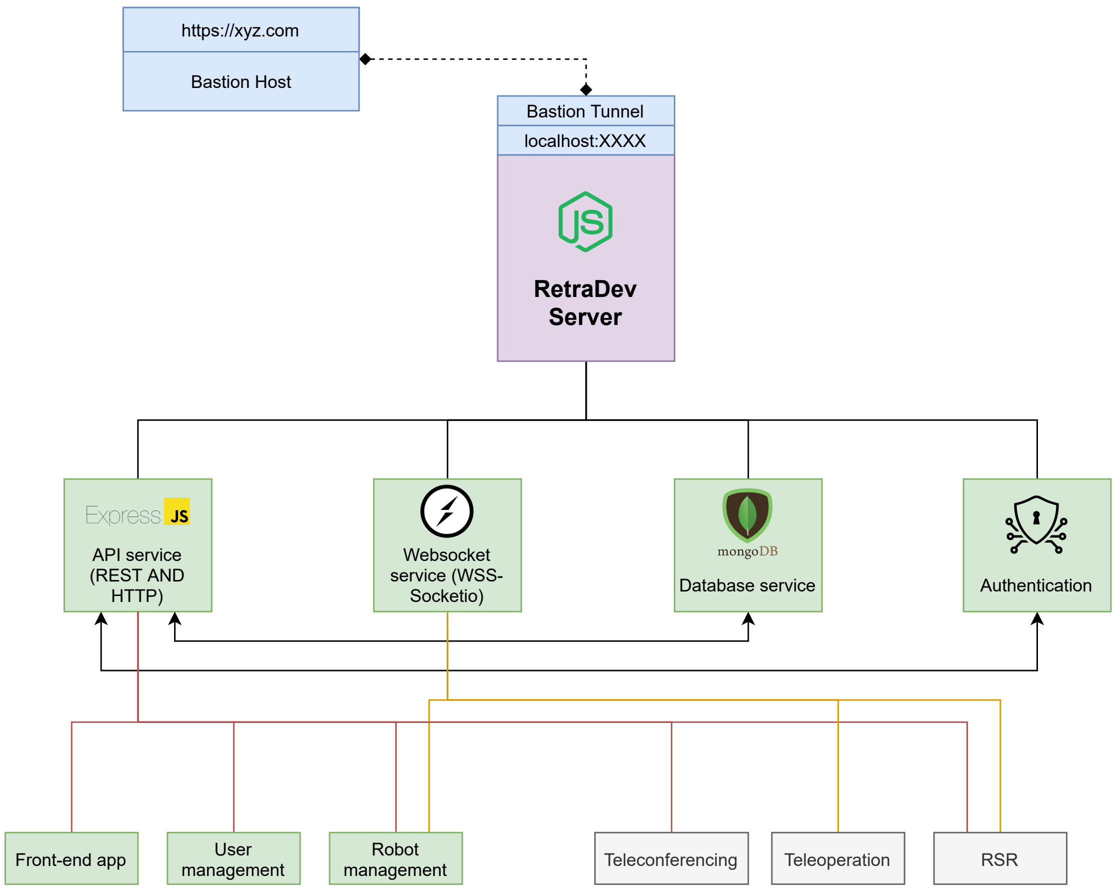


FIGURE 3.2: Internal architecture of the RetraDev server. *(The arrows indicate the direction of data flow)*

2. **Data storage**: As mentioned already, RetraDev uses MongoDB as its central database system. The reason of choosing MongoDB is because it is an open-source database, which runs on noSQL. Unlike SQL, Robotic developers do not have to learn any pre-defined database schema for programming their telepresence robotic application using this noSQL based approach. The usage of a noSQL database is also beneficial for the fact that it can handle unstructured data with dynamic schemas, which can be beneficial for various robotic tasks, such as vision recognition, data acquisition, fault analysis, path planning, localization and others.

3. **Authentication**: The server provides secured authentication services to robotic application front-ends and other microservices. For authentication, RetraDev server implements a session-based authentication, with a session lifetime. It uses *jsonwebtoken*[1] to authenticate users, which stores the session access token on the client side rather than the system side. It also uses *bcrypt*[2] for password hashing.

[1]https://jwt.io/
[2]https://www.npmjs.com/package/bcrypt

Each authentication requests are handled over a secured protocol, either HTTPS or WSS-socketio.

4. **User management**: The server provides user management service for developers to store application user information, and to control accesses of users over different services and operations. User data are stored securely in MongoDB. Currently, it provides the following API endpoints for user registration, login and management.

   (a) */api/users/register - POST*: This endpoint is used to register a user. It requires a username, user email and password for registration. Developers can additionally add other entities in the schema to collect and store additional user data. Username's are stored as the unique keys to identify in the database

   (b) */api/users/login - POST*: This endpoint is used for allowing a user access to a RetraDev application. It uses the Authentication service to generate a secured session access for the user, using a user's username and password.

   (c) */api/users/get_user - GET*: Retrieves a particular user with a username. It requires an auth-token from the user to allow access to this endpoint.

5. **Robot management**: The server manages all the robots of a user in his/her/their account. The robot manager performs several operations, such as error handling, event logging, managing robot's uptime and signal forwarding from one microservice to another. Each robot is designated by a unique ID. RetraDev users and RetraDev powered telepresence robots have a many-to-many relationship in the system, meaning one user can have multiple robots, while one robot can have multiple users. Access and session controls of the robots are handled by the robot manager.

   (a) */api/robots/register - POST*: This endpoint is used to register a robot in a user's account. It requires the telepresence robot's details (name, IP, location, I/O channels) and the username. Developers can additionally add other entities in the schema to collect and store additional user data. Upon registration, a unique ID is generated for the robot, and are stored as the unique keys to identify in the database.

   (b) */api/robots/get_all_user_robots - GET*: This endpoint is used for allowing users to retrieve all robots in their accounts.

   (c) */api/robots/get_details - POST*: Retrieves a particular robot for a user when a *robot ID* and a *username* is passed.

   (d) */api/robots/update - POST*: Updates the existing details of a robot in the database.

3.3.1.2 Teleconferencing service

Teleconferencing is one of the core components of a telepresence system. For teleconferencing, RetraDev framework provides a Peer-To-Peer (P2P) video conferencing service, which runs upon the WebRTC[3] and Socket.io. The core architecture of the teleconferencing service is given in Figure 3.3. The teleconferencing service has been designed

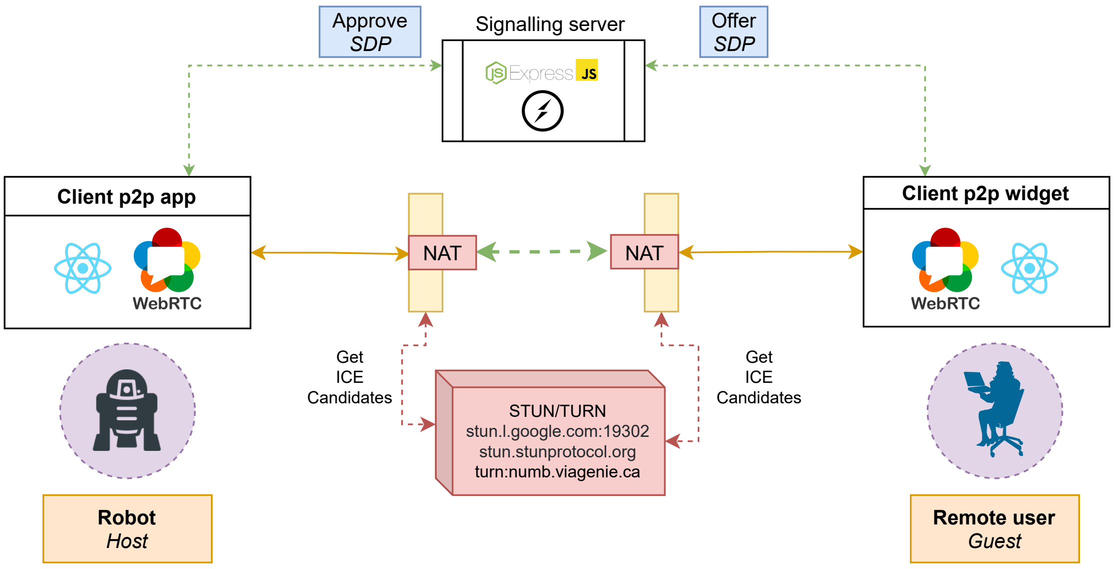


FIGURE 3.3: Peer-to-Peer "handshaking" of the teleconferencing service

in such a way that it can be used as a standalone web application, and also can be embedded as an *iframe* or *embed* widget in a custom web application. When it is embedded as a widget, it provides developers to add or customize Vanilla javascript *(event listeners)* and *(event emitters)*. These allows developers to interact and transmit data or commands with their embedded teleconferencing widget. Currently, RetraDev teleconferencing widget provides a set of event listeners and event emitters by default mentioned in Table 3.1). Developers have the flexibility to add their own listeners and emitters as per their design requirements.

The P2P "handshaking" is administered by Socket.io, which is served by an ExpressJS backend, and a ReactJs[4] frontend with WebRTC running for the A/V streaming between the peers. Socket.io also manages the signaling between the peers (Figure 3.3). In this architecture, the telepresence robot act as the "host" of the teleconference. A "guest" (remote user) can send a connection "offer" via SDP (Session Description Protocol, using WebRTC *RTCSessionDescription()*) to the robot's attached tablet/screen to join its teleconference room. If the "host" robot is available, it sends back a response session

[3]https://webrtc.org/
[4]https://reactjs.org/

Event types	**Event name**	**Description**
Listeners	STARTCALL	To automate the joining process from outside the teleconferencing widget
	HIDEJOINBUTTON	To hide the *Join* button in case the developer wants to customize a *Join* button from outside the widget
	CHECKROOMSIZE	To check whether the "host" robot side has started the session or not
	DONTSTARTCALL	To restrict a remote user from starting a call. Useful when a developer opts for running other operations (such as system calibration) before teleconferencing starts
Emitters	roomsize	Sends the current room size to the parent container when the widgets starts. The parent container app can then decide whether to allow starting a session or not

Table 3.1: Default available Javascript listeners and responders of RetraDev teleconferencing

description via SDP. Once both the "host" and the "guest" configures remote and their own local descriptions, then a WebRTC peer-to-peer connection is initialized using the *RTCPeerConnection* function, which uses *TURN or STUN* server by Google to set the ICE (Interactive Connectivity Establishment) candidates and exchange connectivity information, such as external NATs, port restrictions, IP addresses etc (Figure 3.3).

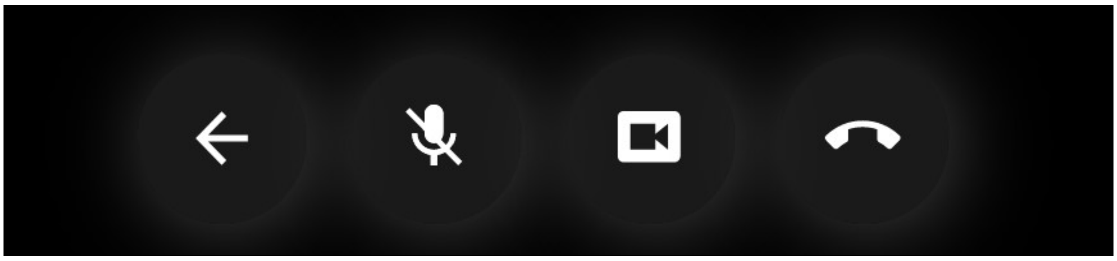


Figure 3.4: Standard teleconferencing call options provided for RetraDev P2P Teleconferencing microservice

Once a call session starts between the robot "host" and the guest, users have access to standard teleconferencing call options, such as "hang up", "mute", and "turn-off video" (Figure 3.4). Each WebRTC session is encrypted end-to-end via SSL/TLS certificate. Developers can upload their custom session certificates during the deployment of this microservice.

3.3.1.3 Teleoperation service

The teleoperation microservice provides access to the sensor and actuator I/O channels of a telepresence robot which are required for teleoperating the robot. It runs on a "publisher-subscriber" model. RetraDev teleoperation microservice currently provides the following I/O channels by default:

1. Robot drive motor (output)
2. Robot camera sensor stream (one or multiple, input)
3. Laser data stream (input)
4. Sonar data stream (input)

Apart from the above channels, developers can expand this microservice as required. The teleoperation microservice works on the basis of *Supervisory control* [121]. The architecture of the teleoperation service is shown in Figure 3.5. It runs on a Flask server which hosts the RetraDev Socket.io client. At the core of operation, real-time communication is handled using websockets over the Socket.io protocol. Currently, at the application layer, the teleoperation microservice uses Animus to receive video streams from a robot and to transmit motor commands to a robot. The central RetraDev server, which runs the core Socket.io server, receives navigation commands from a user. Upon receiving the command, it is relayed to the Teleoperation server. The navigation commands are sent to the robot for execution via Animus.

The Flask server provides several API endpoints for developers. Figure 3.6 shows the breakdown of the Flask application along with the *RetraDevRobot* class, which wraps the *animus_client* and *animus_utils* classes. Some of the important endpoints and functions are briefly discussed below:

1. */get_robots*: It is a POST endpoint. It takes Animus user email and password as inputs, and returns all the robots in a user's account.

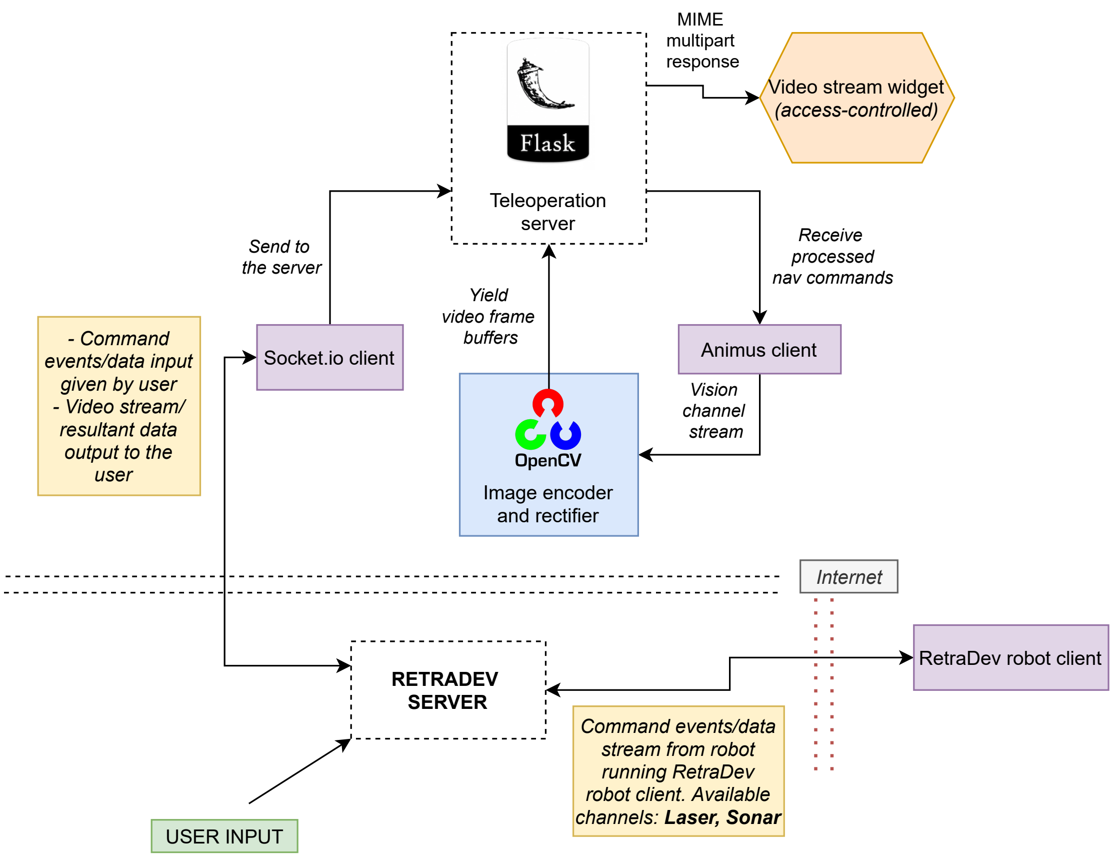
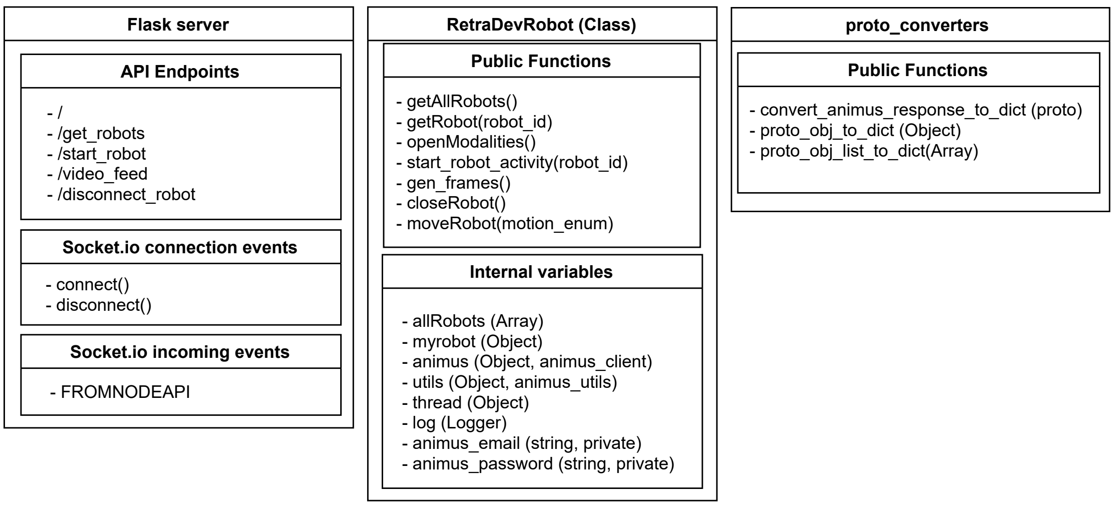

MIME multipart response
Video stream widget (access-controlled)
Flask
Teleoperation server
Send to the server
Receive processed nav commands
Yield video frame buffers
- Command events/data input given by user - Video stream/ resultant data output to the user
Socket.io client
Animus client
Vision channel stream
OpenCV
Image encoder and rectifier
Internet
RETRADEV SERVER
RetraDev robot client
Command events/data stream from robot running RetraDev robot client. Available channels: Laser, Sonar
USER INPUT

FIGURE 3.5: Teleoperation microservice architecture *(the direction of arrows indicate the flow of incoming/outgoing data)*



Flask server

API Endpoints
- /
- /get_robots
- /start_robot
- /video_feed
- /disconnect_robot

Socket.io connection events
- connect()
- disconnect()

Socket.io incoming events
- FROMNODEAPI

RetraDevRobot (Class)

Public Functions
- getAllRobots()
- getRobot(robot_id)
- openModalities()
- start_robot_activity(robot_id)
- gen_frames()
- closeRobot()
- moveRobot(motion_enum)

Internal variables
- allRobots (Array)
- myrobot (Object)
- animus (Object, animus_client)
- utils (Object, animus_utils)
- thread (Object)
- log (Logger)
- animus_email (string, private)
- animus_password (string, private)

proto_converters

Public Functions
- convert_animus_response_to_dict (proto)
- proto_obj_to_dict (Object)
- proto_obj_list_to_dict(Array)

FIGURE 3.6: Breakdown of the teleoperation server class (with the classes, endpoints, methods and variables)

2. */start_robot*: It is a POST endpoint. It takes a unique *robot_id* as input, which identifies a specific robot in an Animus user's account. If the unique ID matches, then it starts up an Animus connection with the robot, opening up all of its available *modalities.*

3. */video_feed*: It is a GET endpoint, which provides access to the video stream. The response format is a MIME/Multipart message that contains stream image buffers.

4. *Socketio.on("FROMNODEAPI")*: This is the subscription event handler of the teleoperation server which receives motion commands emitted from the RetraDev server. The input motion command is a string which is interpreted as an enumerator. It wraps around the *moveRobot(motion_enum)* function of the AnimusRobot class on the basis of some control logics. The enumerator set consists of 10 elements, 4 of which corresponds to robot's head movement, 5 for robot's navigational movement, and 1 for stopping all motions. The list of available enumerators is given in Table 3.2.

Enum	**Description**
head_up	Move robot's head up by *head_angle_incrementer*
head_down	Move robot's head down by *head_angle_incrementer*
head_left	Move robot's head left by *head_angle_incrementer*
head_right	Move robot's head right by *head_angle_incrementer*
rotate_left	Rotate robot's body left by *body_rotation_speed*
rotate_right	Rotate robot's body right by *body_rotation_speed*
forward	Move robot forward at *base_speed* in m/s
back	Move robot backward at *base_speed* in m/s
left	Move robot left sideways at *base_speed* in m/s
right	Move robot right sideways at *base_speed* in m/s
nullmotion	Stop all ongoing base movements

Table 3.2: Enum operators defined for controlling robot motions

The velocity values of each motions can be pre-defined by the developer during development, or can be changed later on. The value *head_angle_incrementer* corresponds for only head movements, while the *body_rotation_speed* corresponds to a robot's angular base movements, and *base_speed* corresponds to a robot's linear movements. Of course, the enumerator list can be expanded to support more motion commands. The current list was designed focusing on Pepper, as it was the robot that was used for running experiments.

5. *gen_frames()*: This is a member function of the RetraDevRobot class. When a robot connection is started and its I/O channels are opened by *start_robot_activity* and *openModalities*, the *gen_frames()* runs an infinite while loop to captures image streams on a continuous fashion, encode each images into jpeg, convert them into buffers and generate a multipart image message encoded into string buffers.

```
 while True:
     try:
         image_list, err = self.myrobot.get_modality("vision",
True)
     except:
         continue
     if err.success:
         ret, buffer = cv2.imencode('.jpg', image_list[0].image)
         frame = buffer.tobytes()
         # concatenate frame one by one and show result
         yield (b'--frame\r\n'
             b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n
')

```

For manual teleoperation, some additional channel access are provided by the LocalCloud. These channels are default event channels for aiding teleoperation, robot monitoring, streaming and transmitting signal. Some useful channels to aid teleoperation are:

- Sonar: For obstacle avoidance
- Laser: For obstacle avoidance and localization

A RetraDev robot client acts as a handler for these channels from the robot's side. Details about RetraDev robot client has been discussed in section 3.3.3.2.

3.3.1.4 RSR - Robotic Service Runner

Robotic Service Runner, or RSR is a microservice in the RetraDev framework that allows developers to run non-real-time robotic algorithms on the cloud for their telepresence robotic operations. RSR does not provide any specific algorithm library for developers, rather it provides the necessary sensor data streaming, visualization, plotting, data processing and communication transport mechanisms for running robotic algorithms and services on the LocalCloud. It uses a telepresence robot's API or ROS to run its operation. RSR will be discussed in details in section 3.4.

3.3.2 Bastion Server

3.3.2.1 Objective

One of the biggest challenges for any cloud-based robotic project development is to test and run the project over the internet. Robotic projects often require peer validation or external user tests. Services or applications running on "localhost" can only be tested by users who are physically present with the computer. Therefore, connected or network-dependent applications/services should be tested over the internet, running on a cloud server. However, there might be several resource requirements for a cloud or a network-based application, such as vision libraries, robotic algorithm runners, streaming, GPU, extensive RAM usage, CPU cores etc. Setting up a cloud server for robotics project development can be challenging. Also, resource intensive robotic web application/services requires a commercially expensive, high-end cloud tier server to host and run, especially for early stage robotic research projects [128].

A good solution to test and expose application/service on development over the internet can be SSH tunneling [129]. Through SSH tunneling, a user can share their locally hosted applications over the internet by making them accessible through URIs, essentially allowing a remote user located anywhere in the world to access locally deployed service.

3.3.2.2 SSH Tunneling services for robotics

There are some freeware and premium SSH tunneling services which can be found online [130]. Such as:

- Ngrok
- Localtunnel
- Pagekite
- Serveo
- Teleconsole

Among these services, Ngrok[5] is the most popular and widely used for local application hosting over the internet. Ngrok is an independent application, which is not dependent

[5]https://www.ngrok.com/

on any programming framework to run (unlike Localtunnel, which requires NodeJs to run). Ngrok can serve any web application over multiple protocols, such as HTTP, HTTPS and SSH [131]. It is available for all operating systems. Ngrok have been used for various cloud robotics and Human Robot Interaction (HRI) studies. Chu et. al proposed SBOT, a social media-based indoor object tracking robot, that used SSH tunneling via Ngrok to expose their application server which hosted the database and web crawler for parsing user messages from LINE chat [132]. Deuerlein et. al proposed a cloud-based natural speech processing system for controlling telerobots over the cloud, where they used ngrok to expose a microservice to receive voice commands over the cloud and convert it to teleoperation commands [133].

While ngrok has a lot of advantages of hosting HTTP/HTTPS based applications, it has the following limitations:

- **Dynamic access URL assignment:** Ngrok is a freemium service. The free version of ngrok assigns publicly available ngrok subdomains each time the application is tunnel via ngrok. The subdomain names change every time the free ngrok service is run. This makes application development processes complicated in some cases, as the remote URL changes every time. While the premium version can be an option where a user can set the domain name, it can be a bit pricey if a project requires to host distributed modular services over multiple URLs as the number of concurrent tunnels are limited. Static URL assignment is crucial in case of robotic prototyping using microservices, such as the RetraDev framework.
- **Does not support various COM protocols:** Tunneling using ngrok does not provide support for various transport protocols, such as polling, websocket RFC 6455 and socket-io. There are documentation that shows the compatibility of rosbridge with ngrok [134], but some connectivity trials using rosbridge and ngrok conducted in this project did not produce satisfactory results, especially over SSL/TLS.
- **No control over SSL/TLS certificates:** While ngrok provides both HTTPS and HTTP support, the SSL/TLS certificates are served by ngrok, and the user does not have any control over its management.

Localtunnel is a popular alternative to ngrok, which is completely free [135]. It supports both HTTP and HTTPS protocols. However, it has the similar issues with ngrok, added with its nodejs requirement. Users cannot set custom URLs in Localtunnel. Although it does offer to set custom subdomain name, but that is subject to availability. Localtunnel also brings some security bottlenecks to the tunneling process.

3.3.2.3 Design goal

A great alternative to the whole localhost application and service hosting can be SSH port forwarding through Bastion Hosts (or Jump servers). In this process, a locally hosted application can be exposed to the internet by forwarding the local port to the remote port of a cloud server (jump server) over SSH. The cloud server acts as the Bastion Host [126]. Since the cloud server is exposed to the internet, any request made to a particular URL which is pointed to the cloud server will essentially forwarded securely through the tunnel to the locally hosted application over SSH.

3.3.2.4 Working process

A basic system diagram of RetraDev Bastion host is given in Figure 3.7. At first, the local developer forwards the ports of locally hosted applications/services to the Bastion Host by starting a port forwarding SSH session. Now when a robot client makes a remote request to a URL over a transport protocol (HTTP/HTTPS/TCP/WS/WSS), the requests traverses following standard internet routing methods, and the DNS of the URL (A record or CNAME record) resolves to the Bastion Host, or the cloud server. The cloud server upon receiving the request may check for its validity and authenticity (required for websocket transports), and then relays the request to the local application through the forwarded port over SSH. The local application receives the request, and then sends back the requested data or contents, through the same Bastion tunnel to the robot client. In this way a bi-directional channel is established between the local developer and a remote robot client. A step-by-step process of setting up the RetraDev Bastion Host has been presented in Appendix B.

3.3.2.5 Relay server

The RetraDev Bastion host is different from standard Bastion hosts, because it also hosts a **Relay server**. Relay server is a socket.io relay service that relays socket.io client events from the robot side to the central RetraDev server. While designing the LocalCloud component based on Pepper NaoQi v2.5, it was found that Pepper was unable to connect to the LocalCloud websocket service because of the tunnel routing. Since LocalCloud runs on a developer's local machine, the LocalCloud was exposed via Bastion tunneling only during exposing the project for usage and testing. As a result, the robot client installed inside Pepper was unable to persist a socket.io connection beyond the lifetime of a LocalCloud exposed session. Also, due to Pepper NaoQI's old SSL certificates, it was difficult to establish an encrypted websocket channel over the WSS

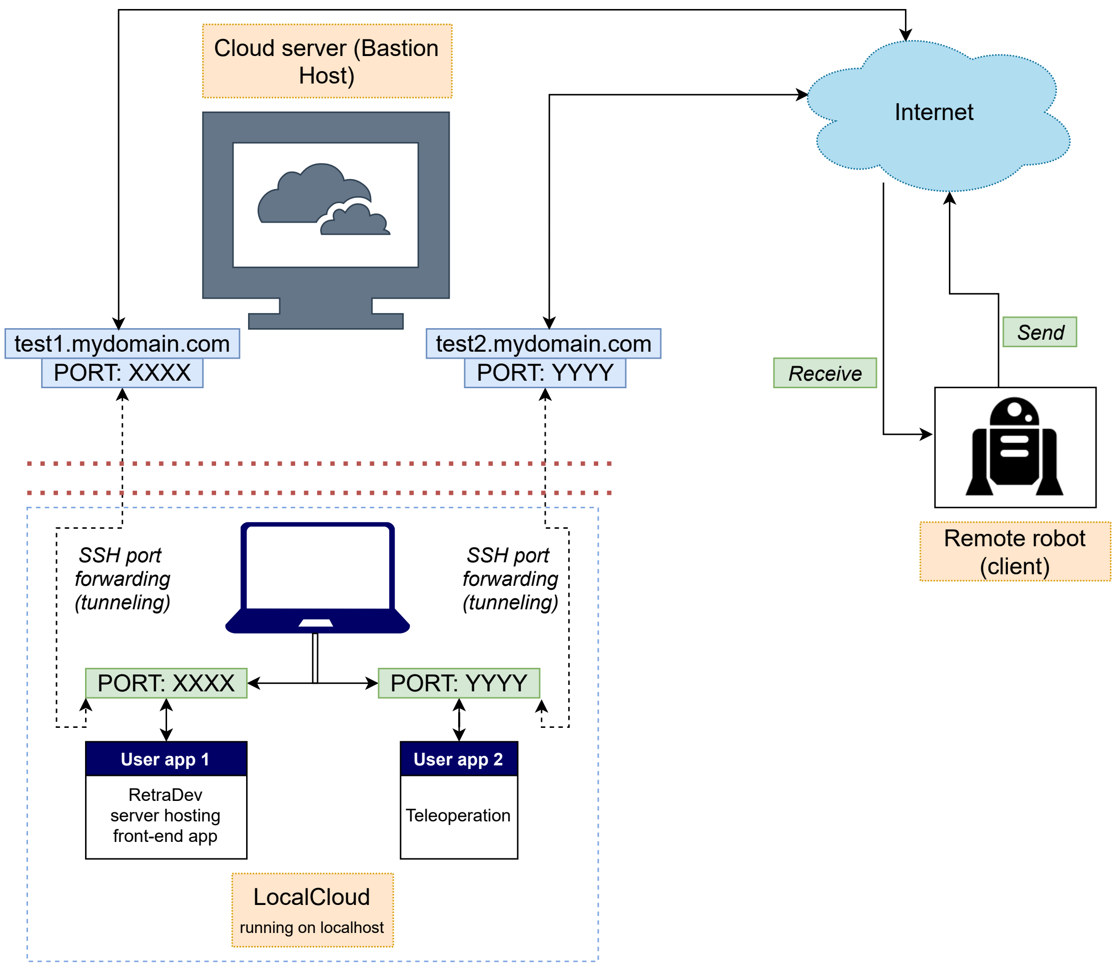


FIGURE 3.7: RetraDev Bastion Hosting architecture and its working process

protocol. These two issues brought up the necessity of developing the Relay server. The usage of a cloud VPS for tunneling allows a developer to host their own socket.io relay servers, which will establish a persistent connection with the RetraDev robot clients "on behalf of" the LocalCloud.

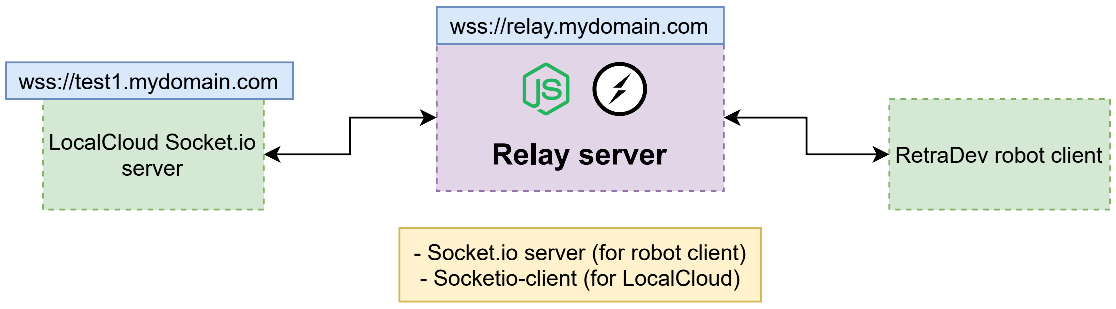

wss://relay.mydomain.com
wss://test1.mydomain.com
LocalCloud Socket.io server
Relay server
RetraDev robot client
- Socket.io server (for robot client)
- Socketio-client (for LocalCloud)

FIGURE 3.8: Relay server residing inside a RetraDev Bastion host

Figure 3.8 shows a simplified working diagram of websocket communication using a

Relay server. The relay server is hosted inside the Bastion host cloud VPS, which runs on NodeJs. It is assigned to a URI, whose DNS is resolved by the Nginx service of the Bastion Host. Using *certbot*[6], the URI can be easily SSL/TLS encrypted, allowing WSS protocol access over the port 443. The robot client can establish a persistent bi-directional connection with the relay server, which can receive socket.io broadcasts from a RetraDev robot client in real-time. When LocalCloud is started and exposed over the internet through Bastion tunneling, it can start receiving the forwarded data stream broadcasts from the relay server's socket.io-client. This way the robot client does not need to wait for the LocalCloud's availability, and continue streaming data to the relay server. Also, when a LocalCloud microservice wants to send data back to the robot, then it can follow the opposite websocket route, that is from LocalCloud to the relay server, and from the relay server to the robot. The only potential disadvantage of a relay server is that it may be prone to latency issues. An analysis of the performance of a relay server in the RetraDev framework have been presented later in this chapter.

3.3.2.6 Server configuration

To run RetraDev Bastion Host, a developer needs to have the following resources (Table 3.3).

Requirement	**Description**
FQDN	With a fully qualified domain name (FQDN), a developer can host one or multiple number of RetraDev microservices using sub-domains, which will be their static URIs.
Linux VPS server with SSH daemon installed	A cloud VPS server running on Linux is needed as the RetraDev Bastion host. The minimum required configuration are as follows: - RAM: 1GB - CPU: 1 core - Network In: 40GB - Network Out: 1000GB - Bandwidth: 1TB
Firewall daemon	Access to the server's firewall is required to open the required tunneling ports. The port 80 and 443 needs to be open to expose LocalCloud application and services to remote users and robot clients.

Table 3.3: Minimum system requirements to run the RetraDev Bastion host

Figure 3.9 shows the internal components of a cloud server running as a Bastion Host. An ideal cloud server is protected by a firewall, controlling the incoming and outgoing data streams of the server. For each locally hosted applications, a corresponding server runner needs to be created, using Nginx.

[6]https://certbot.eff.org/

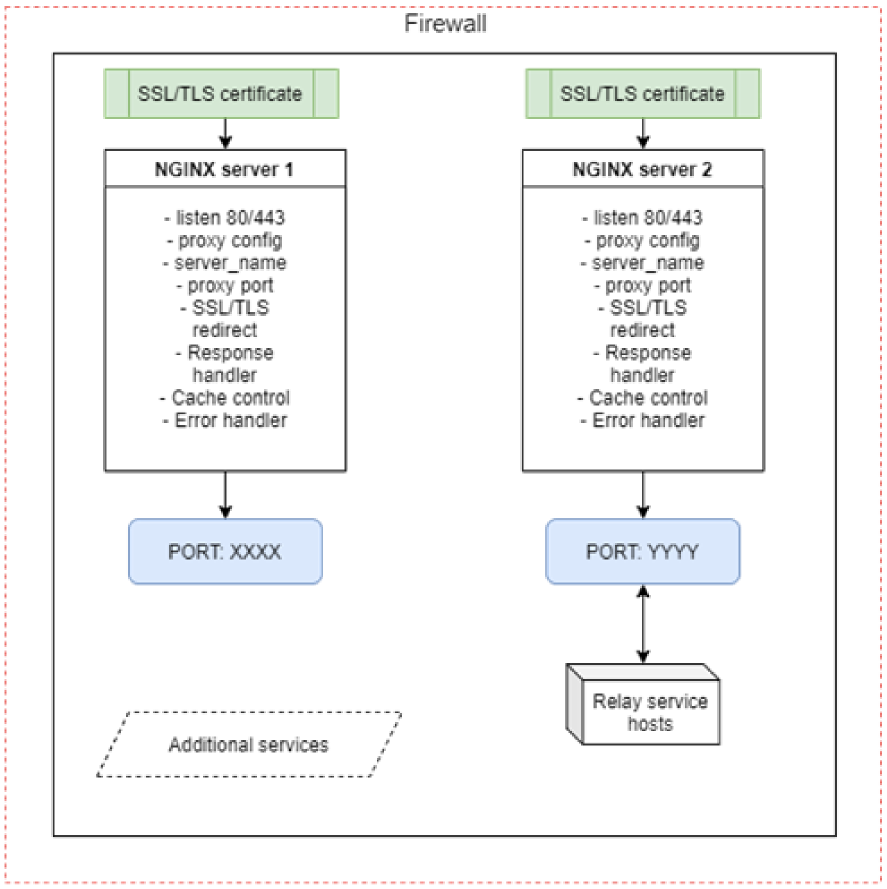


FIGURE 3.9: Internal components of cloud server running Bastion host with Nginx as the server runners

In this architecture, we can see Nginx has been used as the server runners. The Nginx server runner assigns a name (the resolvable URI) corresponding to a local proxy port. Additionally, the runner also contains configuration regarding the proxy, error handlers, SSL/TLS certificates which will be served along with the application and other options. For establishing the SSH tunnel, the corresponding ports need to be opened for I/O access. Additionally, the Bastion Host also connects the relay server (server 2 in Figure 3.9), which can be necessary for some specific applications.

3.3.2.7 Potential benefits

There are several benefits of using SSH tunneling through Bastion Host. They are:

- **Control over security:** Users have full control over the security of their telepresence robotic microservices since they have access to the server.

- **Easy local monitoring support:** SSH tunneling through Bastion Host can be easily monitored on the user's PC, if the microservices are run by a process manager such as "pm2"[7].

- **Reducing cloud setup complexity:** Cloud setup and configuration for robotic project deployment can be tricky, especially when the project has a number of dependencies. While developers can configure their local environment suitable for their project development, the same configuration can be challenging for developers on the cloud side. When a robotic application is fully ready for deployment, it can be deployed on a cloud server by configuring the environment one time. However, before the final completion of a development process, RetraDev Bastion hosting makes it easier for developers by cutting out the complexities of DevOps operations, rather just focus on the application development, share them with peers and test them over the network.

- **Permanent configuration:** Once a SSH tunnel is fully configured, it can be paired with "pm2" process manager, which can be used as a permanent configuration for the LocalCloud microservices to be used over the internet using their own PC as the server infrastructure. This allows post-production level testing and allow a user to fully serve a complete app over the internet as a cloud hosted application.

- **Static URLs:** Through this process, users set their own custom domain names for each of their LocalCloud microservices. These names are static, which allows convenience for programming any scripts or services by not requiring to query or change the request URLs every time.

- **Unlimited RetraDev microservice hosting:** A user with a Fully Qualified Domain Name (FQDN) can host unlimited number of RetraDev microservices on their localhost through Bastion tunneling, by setting subdomains for each of the applications. Since the cloud VPS is only used for tunneling and relaying, developers do not need high-end servers requirements for their applications and services [136]. This essentially removes the limitation of Ngrok for tunneling a very limited number of concurrent applications.

- **No limitation over available protocols:** Since the user has access to the cloud server, any RetraDev project that may require protocols apart from HTTP/HTTPS (such as UDP, RTSP, Polling, WS and WSS) can be deployed through a Bastion Host by writing and running transport protocol scripts on the server, essentially configuring a user-defined protocol for communication. Some protocols which have been tested and verified through this method are:

[7]https://pm2.keymetrics.io/

- HTTP/HTTPS
- WS/WSS
- WS-socketio/WSS-socketio
- OMQ
- Rosbridge Autobahn
- Polling
- UDP

3.3.2.8 Limitations

Despite some major advantages, RetraDev Bastion Host has the following drawbacks. Developers need to consider security aspects of their local machines when they expose their local port through SSH tunneling. Since SSH already establishes a secured connection between the local machine and the cloud server, security features need to be implemented on the server side, so that the developer's machine is protected from any unwanted attacks. It is always advisable not to serve local applications over self-signed SSL/TLS certificates. Developers can also limit control access over the SSH channel, so that only the required microservices are exposed and nothing else.

3.3.3 Telepresence robot client drivers

In the RetraDev framework, telepresence robots communicate with external microservices using robot client drivers. Client drivers, or simply client, are either an Animus robot client (which are installed by Cyberselves along with their driver software), or a Socket.io client which connects to the LocalCloud via the relay server. Robot clients establishes a bi-directional communication channel between the telepresence robots with LocalCloud microservices. They can receive commands, such as teleoperation. They can process each commands and execute the corresponding task by using a telepresence robot's API. Clients can also stream data to the LocalCloud services, the data type may be camera feed stream, audio recording, real-time context data (sonar, laser, infrared etc.), homeostasis and others.

3.3.3.1 Animus robot client

Telepresence robots configured with Animus can handle incoming teleoperation commands via Animus robot client, while it uses the client to stream vision, audio, proprioception and homeostasis data. Animus provides access to 9 modalities, however not all robot supports these 9 modalities. For instance, the Animus client of the Pepper NaoQi robot currently supports only 5 modalities. The number of supported modalities vary from robots to robots. Animus clients communicate with Animus server over WebRTC, which is managed by Cyberselves. Cyberselves partners with robotic manufacturers for deploying their proprietary Animus service, as a result manufacturers provide full and high-speed accesses to the different I/O channels of their robots to Animus. Animus was adopted as a temporary component in the application layer of RetraDev for its current development. In the future, the whole process of running a RetraDev-powered robot will be conducted by its own client engine.

3.3.3.2 RetraDev robot client

RetraDev robot client provides access to all the I/O channels of a telepresence robot. RetraDev robot client runs a socketio-client to incorporate a robot to the RetraDev architecture. The client have been designed using Python. It can be installed as a runtime service for a telepresence robot, or can be started manually by the developer. It comes with a wrapper library over the built-in APIs of a robot. This wrapper library allows developers to write robot-agnostic codes for the robot clients. The approach is much similar to a middleware system, such as ROS. The RetraDev robot client can be functioned alongside with ROS, however it is particularly useful when ROS is unavailable (some telepresence robots does not support ROS) or difficult to be packaged and installed without a physical access to the robot.

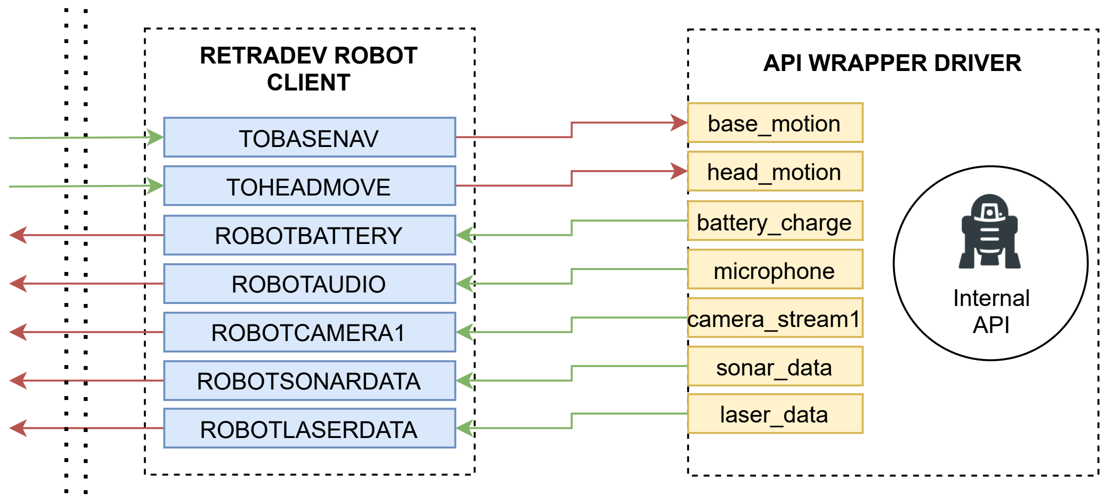


Figure 3.10: Working diagram of RetraDev robot client and an API wrapper driver over robot's internal API

The working process of the RetraDev robot client is shown in Figure 3.10. The direction of the arrows identify whether its an incoming (right arrow) or outgoing (left) signal.

The required input (actuators and robot's CPU) and output (sensors) APIs are wrapped using user-developed wrapper functions or abstractions. Each wrapper function corresponds to a socket.io event, which is a part of a set of "RetraDev-defined events". Events are launched and handled by the Socket.io server running on the LocalCloud. This set of RetraDev events are robot-agnostic websocket channels for writing robotic programs, which remains the same for all robots. Currently, the following set of robot-agnostic socket.io events are available for client-side development (Table 3.4).

Client event	**Equivalent LocalCloud event channels**	**Description**
ROBOTSONARDATA	SONARDATA	Sonar data collected from different sensors (JSON object, sample response: {*"front":x,"back":y*})
ROBOTLASERDATA	LASERDATA	Laser data collected from different lasers (JSON object, sample response: {*"front_shovel_1":x, "front_shovel_2":y, "front_vertical_1":z ...* })
ROBOTCAMERA1	CAMERA1	Image data collected from 1 camera of a robot (base64 string, sample response: *data:image/jpeg;base64, AX3JDEuujkkaaa...*)
ROBOTBATTERY	BATTERYDATA	Battery charge available (in percentage) of a robot (integer, sample response: *56*)
ROBOTAUDIO	AUDIODATA	Microphone recorded audio sampled at 16kHz-single channel or at 48kHz-4 channels (base64 string, sample response: *data:audio/wav;base64, UklGRhwMAAB...*)
TOBASENAV	BASENAV	Robot's base navigation commands (JSON Object, sample response: {*"forward":x,"backward":0, "rotate_left":0,"rotate_right":0, "left":0,"right":0,*})
TOHEADMOVE	HEADMOVE	Robot's head movement commands (JSON Object, sample response: {*"head_up":x,"head_down":0, "head_left":0,"head_right":0*})

Table 3.4: Robot agnostic Socket.io events available for client-side development using RetraDev robot client

From the table, we can see that each RetraDev robot client event corresponds to a LocalCloud event, relayed by the Relay server. Events contain payload/response either in integer, string, bytes or JSON format. Sensor events, such as sonar and laser can take dynamic JSON object as input, corresponding to the number of units that are integrated with the robot's body. For example, a robot may have a sonar at the front, and one at the back. In that case, the corresponding API wrapper would transmit JSON payload to the *ROBOTSONARDATA* event as:

```
    {
        "front": "val1",
        "back": "val2
    }

```

The process of attaching payloads to each outgoing (sensor-captured) events is the same for other sensor modules as well. In the table, we can also see that the camera stream event name is called "ROBOTCAMERA1", which captures camera sensor data for one camera of a roobt. Considering a telepresence robot might have multiple camera sensors, for each camera a corresponding LocalCloud event channel can be defined. For example, the Pepper robot has two 2D cameras on its head. To capture vision streams from both of them, a developer needs to map image streams from *AL::kTopCamera* against "ROBOT-CAMERA1", and map image streams from *AL::kBottomCamera* against "ROBOT-CAMERA2".

3.4 RetraDev RSR - Robot Service Runner

RetraDev Robotic Service Runner is an additional component in this proposed RetraDev framework which can allow developers to employ non-real-time, autonomous or intelligent behaviors to their telepresence robots, ideally using ROS. RSR essentially establishes a connection between ROS and the RetraDev LocalCloud using rosbridge. Since both ROS and LocalCloud resides on the same machine, rosbridge routing occurs through the localhost, meaning no specific routing is required to expose the ROS nodes, topics or services over the internet. RetraDev takes care of exposing these ROS components over the internet using LocalCloud and Bastion host. By exposing RSR services, external telepresence robots can still be connected to ROS over the internet. In this way, developers do not have to spend on expensive cloud infrastructure to run ROS on the cloud during the early stage of research and development of their telepresence robotic applications.

3.4.1 Objectives

RSR was primarily designed focusing on the following points:

- **Rapid prototyping**: RetraDev RSR approach reduces the time and effort required to make ROS components available over the internet. While rosbridge is an excellent service to integrate and communicate with non-ROS components with ROS, developers would still need to host the rosbridge_server and ROS on a cloud infrastructure for accessing and establishing a bridge. This task requires DNS management, SSL/TLS management, server security considerations and server monitoring. RSR reduces the additional complexity of deploying a rosbridge_server on a cloud host.

- **Adding autonomy to telepresence robots**: Referring to section 2.4.1 regarding the state-of-the-art study of robotic telepresence, we can see that many commercial and proposed telepresence robots are employing autonomous functionalities on-board alongside manual operations. The autonomous functionalities are added to a telepresence robot to aid the existing tasks and purposes of a robot, such as navigation, gesture response and gaze tracking. RSR was provisioned into the RetraDev framework with the vision to add these functionalities to the telepresence robots under development.

3.4.2 How RSR works inside the RetraDev framework

The architecture of the RSR microservice inside the RetraDev framework is given in Figure 3.11. Both ROS and the LocalCloud runs on the same machine. The RSR microservices run inside the LocalCloud, and they are assigned unique URIs by the Bastion Host. Each RSR microservice can represent different operations, such as robotic mapping, localization, vision processing and recognition. These can be facilitated by several ROS web-based libraries and widgets, such as rvizweb, ros2djs, mpegcanvasjs etc. The RetraDev server can additionally provide several other functionalities to the RSR services, such as database operations, serving static RSR front-ends and establish a connection with the teleoperation service. By being exposed over the internet via RetraDev LocalCloud and Bastion host, RSR services aids ROS-based remote development and accessibility using local infrastructures.

Each RSR microservices contain the following components for external communication:

- Served on a specific Bastion-tunneled local port assigned by the RetraDev server.

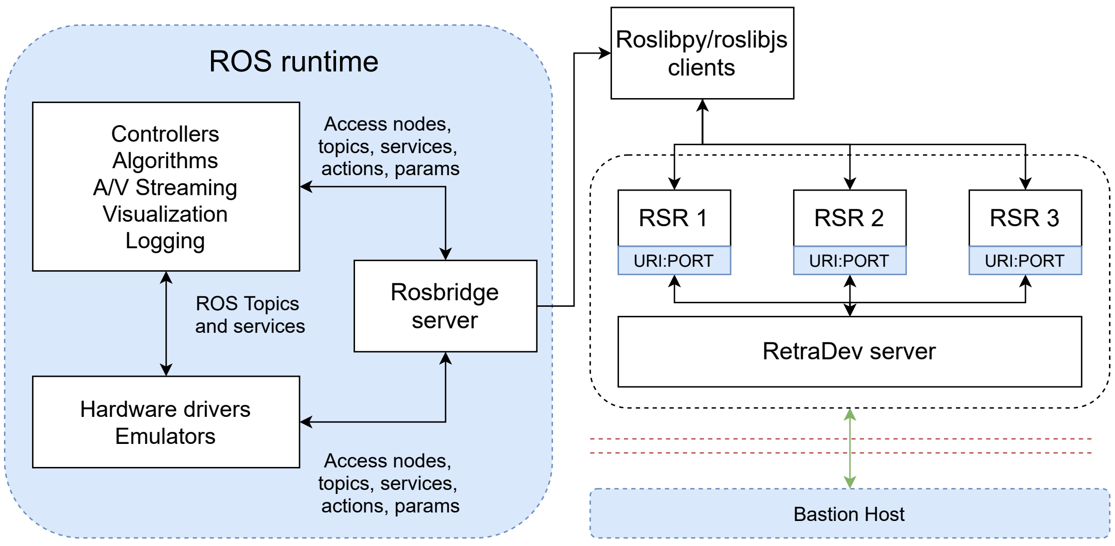


FIGURE 3.11: RSR working process with multiple RSR running in parallel with the LocalCloud

- Establish and persist a socketio connection with the RetraDev server (*socketio.connect("localhost:XXXX")*)
- Establish and persist a rosbridge connection with the rosbridge server (*socketio.connect("localhost:YYYY")*)

3.4.3 Use case exploration: Cloud Mapping and Localization using local infrastructure

One of the most sought features in indoor telepresence robotics these days is localization and semi-autonomous navigation (Refer to the literature discussion in section 2.4.1). Indoor localization can aid teleoperator users to identify at what position on the indoor map they are located, thus allowing them to make smoother navigation decisions. Semi-autonomous navigation can aid remote users to easily navigate indoors to any specific room or position within the indoor localized environment. This provides easement for navigation, and at the same time allow the robot to make sharp turns around any corners or navigate through paths which would have been complex for the remote user to teleoperate [19]. Semi-autonomous navigation can also be paired with voice-based command or gesture recognition, so that local users can also interact and command the robots to navigate to a particular point inside their apartments [79]. Many commercial and conceptual telepresence robots are adopting autonomous navigation functionality alongside manual teleoperation for enhancing user experiences.

To navigate autonomously, a robot needs to localize it on a known mapped environment. An initial use case for RSR was explored during this project, that is to map an unknown environment remotely on the cloud using the Pepper robot, and to localize the robot inside the map. The mapping RSR and localization RSR are deployed locally and visualized using Rvizweb, a browser based robot visualization library powered by ros-rviz, roswww, rosbridge_server and tf2_web_republisher [137]. This experiment was conducted using a simulated Pepper running on Gazebo and ROS. Gazebo is a robotic simulator, which is used for rendering robotic models, emulating robotic behaviors and simulate robotic environments [138].

To implement the mapping and localization RSRs, RTABMAP was used. RTABMAP (Real-Time Appearance Based Mapping) is a 3D appearance-based SLAM algorithm proposed by Labbe et. al in 2011[139]. RTABMAP can be used as a standalone application, or as a ROS package (rtabmap_ros). To map using RTABMAP, a RGB-D camera, stereo camera or a Lidar can be used [140]. The general working procedure of the RTABMAP algorithm using a robot and a RGB-D camera is as follows:

1. Caputre real-time RGB+depth images continuously and store in a database along with the visual odometry and pose estimation data
2. Match new captured scenes with the stored scenes in the database. If a probabilistic match is found, a loop closure occurs, identifying the likelihood of the new scene coming from a previously captured scene or from a new scene.
3. When the loop closure is confirmed, the graph of the map is updated instantly with a new constraint (robot's position and orientation).
4. A graph optimization algorithm is implemented to correct odometry errors and optimize the graph. Additionally, to process loop closures and graph optimization in near real-time, a memory management algorithm is employed to minimize the size of the map. This memory management feature of RTABMAP is a crucial component for cloud-based mapping.

Pepper has two RGB camera sensors (*AL::kTopCamera* and *AL::kBottomCamera*) and one depth camera (*AL::kDepthCamera*), which makes it suitable to employ the RTABMAP algorithm for mapping and localization.

3.4.3.1 Experimentation setup

To setup an environment for the RTABMAP simulation using ROS, the following ROS packages were used [141]:

- pepper_virtual
- pepper_description
- pepper_meshes
- pepper_roobt
- naoqi_driver
- Gazebo_ros

For the simulation, a Gazebo World model (*test_zone.world*) was used from [142]. This Gazebo World was designed by Fetch Robotics, and it contains an indoor of an apartment, with a hallway, two rooms, and three corridors (Figure 3.12). The experiment was conducted on a machine with the following system configurations: (a) Intel Core i7-7th gen, (b) 16GB DDR4 RAM, (c) Nvidia GTX 1060 GPU, (d) 756GB SSD. The experiment was conducted using ROS Melodic running on Ubuntu 18.04.

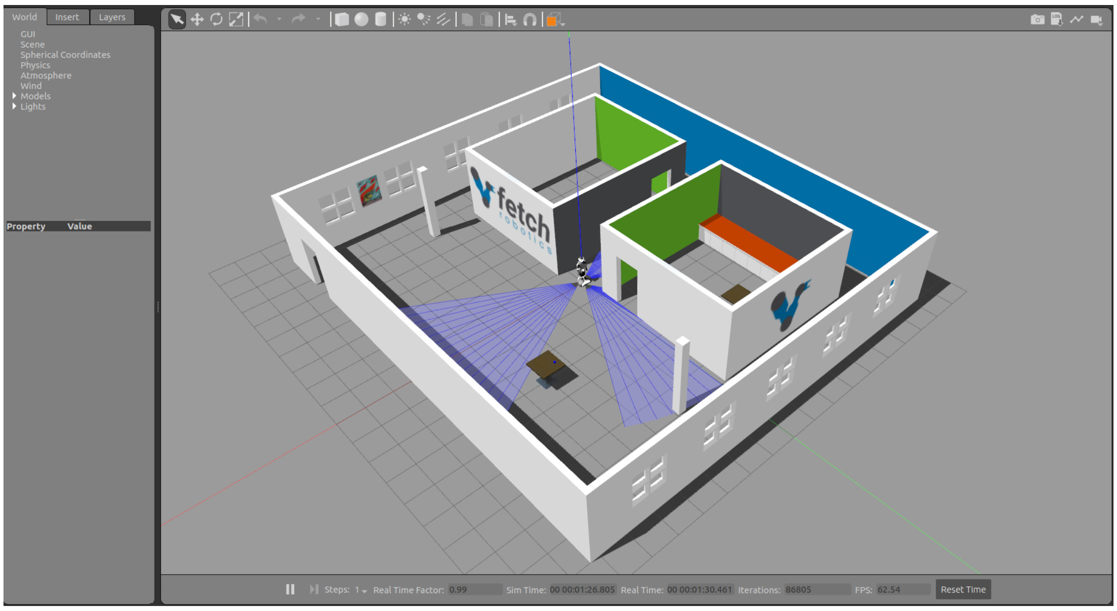


FIGURE 3.12: Virtual Pepper inside a Gazebo World

The mapping session was conducted by teleoperating the robot using the *teleop_twist _keyboard* ROS package. To run the RTABMAP algorithm for the Pepper robot, a roslaunch file was created for the the *pepper_virtual* package. The launch file launches rtabmap node with its argument and parameters mapped with Pepper's rostopics. The arguments of the launch file is as follows:

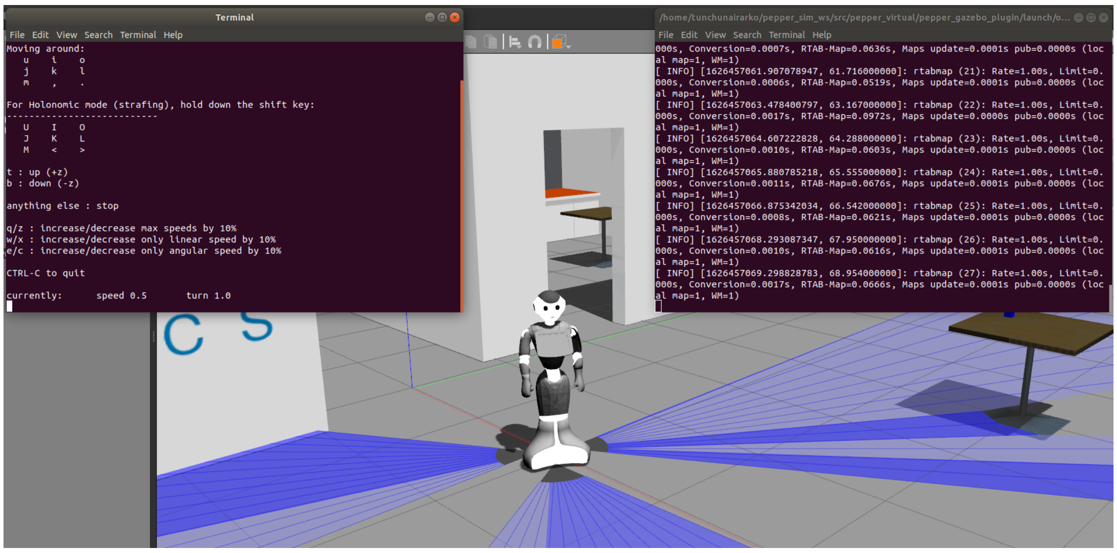
```
<!-- RTAB-Map arguments -->
<arg name="database_path"     default="rtabmap.db"/>
```

```
<arg name="rgb_topic"   default="/pepper/camera/front/image_raw"/>
<arg name="depth_topic" default="/pepper/camera/depth/image_raw"/>
<arg name="camera_info_topic" default="/pepper/camera/front/camera_info"/
  >
```

The input parameters of the *rtabmap* rosnode are as follows:

```
<!-- RTAB-Map Inputs -->
<remap from="odom" to="/pepper/odom"/>
<remap from="scan" to="/pepper/laser_2"/>
<remap from="rgb/image" to="/pepper/camera/front/image_raw"/>
<remap from="depth/image" to="/pepper/camera/depth/image_raw"/>
<remap from="rgb/camera_info" to="/pepper/camera/front/camera_info"/>
```

A screenshot of running the RTABMAP algorithm on Pepper inside the Gazebo world is shown in Figure 3.13 and a graph view of the rosnodes and rostopics generated using *rqt_graph* is shown in Figure 3.14.


Terminal
File Edit View Search Terminal Help
/home/tunchunairarko/pepper_sim_ws/src/pepper_virtual/pepper_gazebo_plugin/launch/o...
File Edit View Search Terminal Help

```
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
```

```
000s, Conversion=0.0007s, RTAB-Map=0.0636s, Maps update=0.0001s pub=0.0000s (loc
al map=1, WM=1)
[ INFO] [1626457061.907078947, 61.716000000]: rtabmap (21): Rate=1.00s, Limit=0.
000s, Conversion=0.0006s, RTAB-Map=0.0519s, Maps update=0.0001s pub=0.0000s (loc
al map=1, WM=1)
[ INFO] [1626457063.478400797, 63.167000000]: rtabmap (22): Rate=1.00s, Limit=0.
000s, Conversion=0.0017s, RTAB-Map=0.0972s, Maps update=0.0000s pub=0.0000s (loc
al map=1, WM=1)
[ INFO] [1626457064.607222828, 64.288000000]: rtabmap (23): Rate=1.00s, Limit=0.
000s, Conversion=0.0010s, RTAB-Map=0.0603s, Maps update=0.0001s pub=0.0000s (loc
al map=1, WM=1)
[ INFO] [1626457065.880785218, 65.555000000]: rtabmap (24): Rate=1.00s, Limit=0.
000s, Conversion=0.0011s, RTAB-Map=0.0676s, Maps update=0.0001s pub=0.0000s (loc
al map=1, WM=1)
[ INFO] [1626457066.875342034, 66.542000000]: rtabmap (25): Rate=1.00s, Limit=0.
000s, Conversion=0.0008s, RTAB-Map=0.0621s, Maps update=0.0001s pub=0.0000s (loc
al map=1, WM=1)
[ INFO] [1626457068.293087347, 67.950000000]: rtabmap (26): Rate=1.00s, Limit=0.
000s, Conversion=0.0010s, RTAB-Map=0.0616s, Maps update=0.0001s pub=0.0000s (loc
al map=1, WM=1)
[ INFO] [1626457069.298828783, 68.954000000]: rtabmap (27): Rate=1.00s, Limit=0.
000s, Conversion=0.0017s, RTAB-Map=0.0666s, Maps update=0.0001s pub=0.0000s (loc
al map=1, WM=1)
```

FIGURE 3.13: RTABMAP running inside the Gazebo simulation with Pepper being teleoperated

After launching RTABMAP_ros for pepper, rviz was launched. A custom rviz configuration file with Pepper and RTABMAP topics was created following [141]. Following the launch of rviz, the rvizweb package was launched. Rvizweb launches a NodeJs web server to run the RSR service for visualizing the grip occupancy map at port 12432 (port number was available from the RetraDev configuration). Once both rviz and rvizweb were launched, Bastion tunnel on port 12432 was started to make the map visualizing RSR microservice accessible securely over the web at *https://robotapi.isensetune.com* (Figure 3.15).

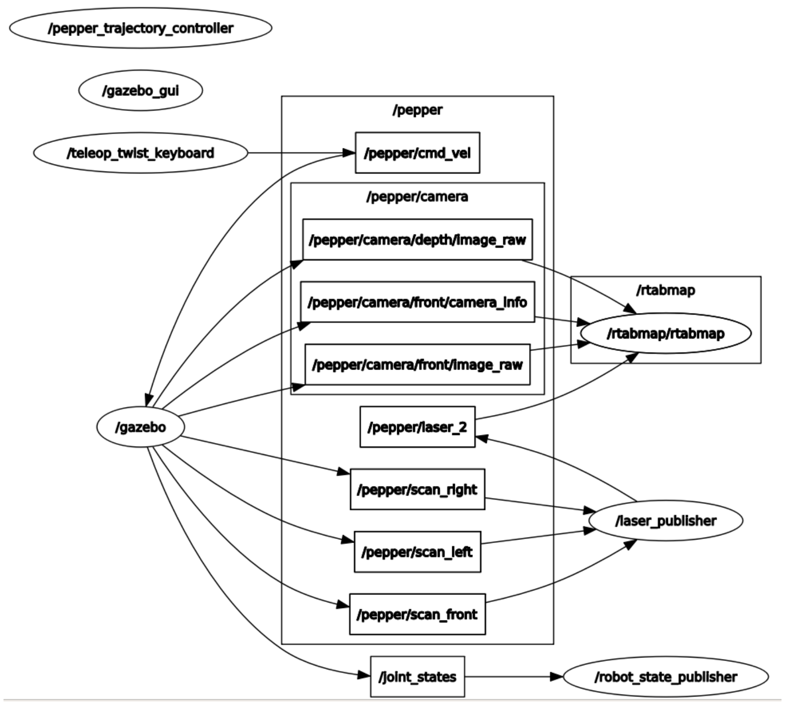


FIGURE 3.14: RQT graph with Gazebo, Pepper and RTABMAP nodes and topics

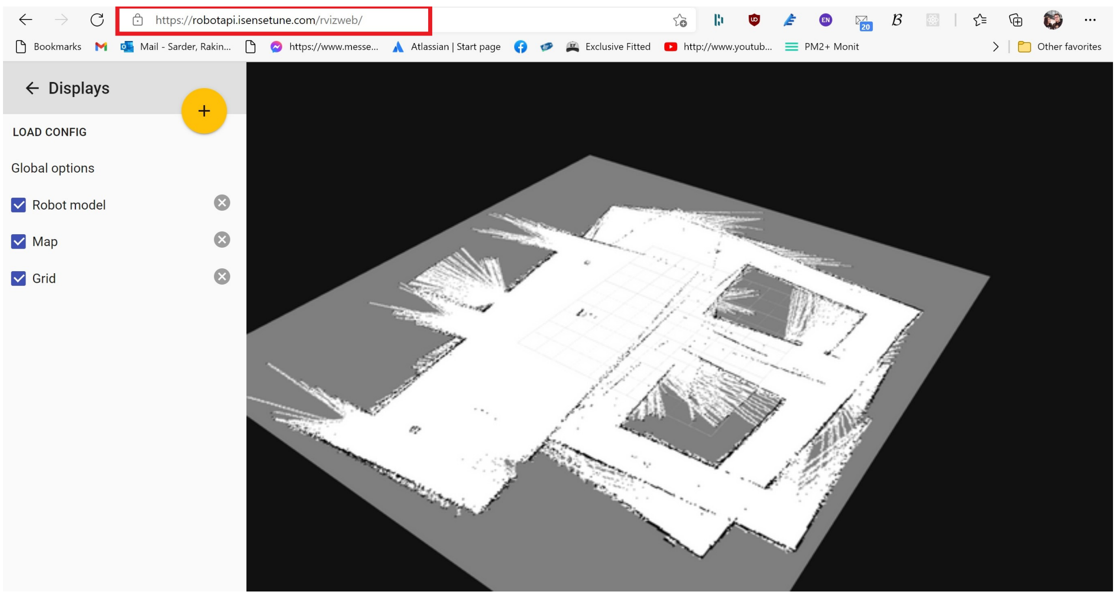


FIGURE 3.15: Visualizing the Occupancy grid map as an RSR service by deploying over the web

The starting point view of the Gazebo world visualized by both rviz and rvizweb is shown in figure 3.16.

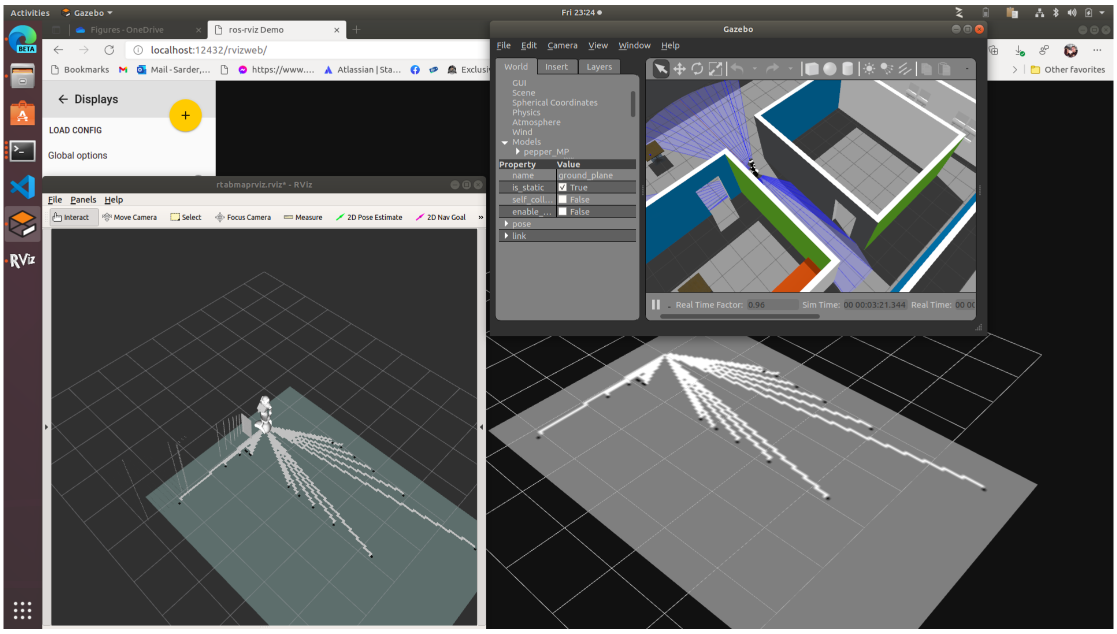


FIGURE 3.16: Starting point view of mapping process in Gazebo, Rviz and Rvizweb

3.4.3.2 Results

For generating the map, the robot was teleoperated across the full apartment. The initial target was to have at least 5 loop closure detections, as a result the robot was navigated across to acquire more images. The RTABMAP update rate was set at 5Hz, with the time limit for processing was set at 5ms. The robot travelled a distance of 258m across the entire apartment. A total of 1914 images were acquired and saved, with 56 global loop closures detected. The images with the odometry data were stored into the rtabmap.db database, and the final size of it was 311 MB. The final output 2d grid map exported from RTABMAP_rviz and rvizweb respectively is shown in figure 3.17. We can see that rvizweb rendered the grid occupancy map similar to the manual exported one.

Loop closures were detected at various situations during the mapping phase. Figure 3.18 shows a local match occuring prior to detect a loop closure nearby.

Figure 3.19 shows a dense point cloud map corresponding to the grid occupancy map generated along with robot's trajectory estimation. In order to test the RSR visualization service over the internet, a separate PC was used, globally connected to two different networking routes. Figure 3.20 shows the two PCs with which the remote access part of the experiment was tested. In the figure, PC B was running the RetraDev framework

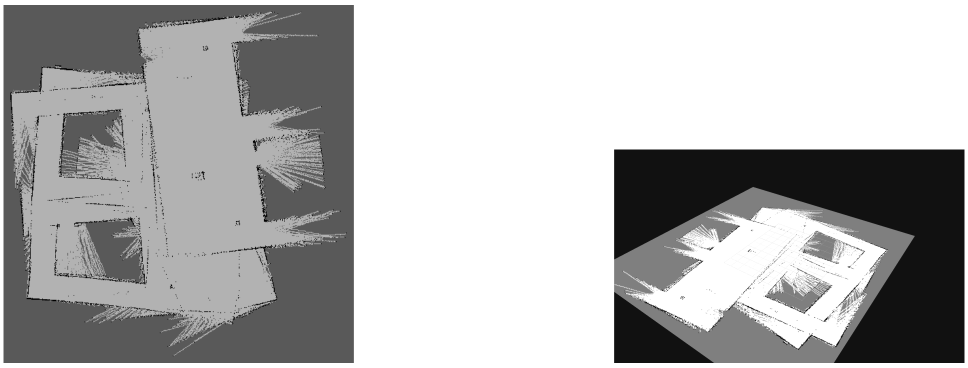


Figure 3.17: Final generated occupancy grid map generated, left one was taken from rviz, and the right one from rvizweb

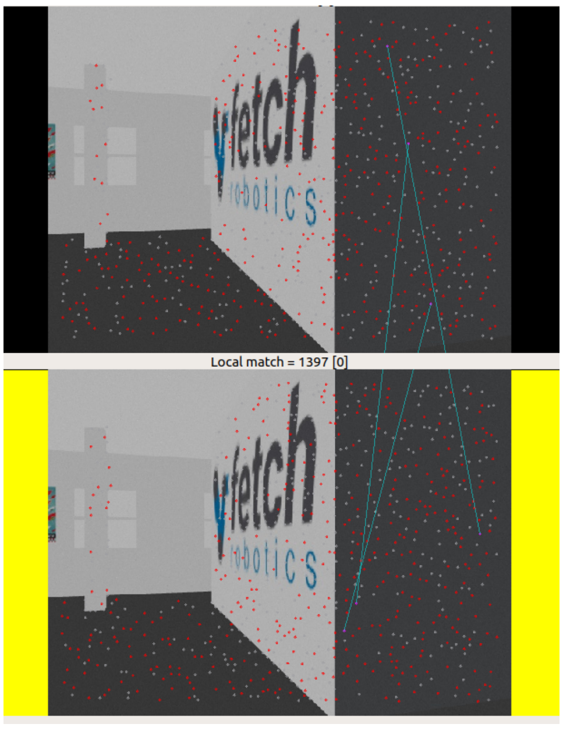


Figure 3.18: Detection of local match in Rviz

along with ROS, while PC A was used as a "remote user pc". A Obfuscated VPN service was used for proxying PC A's location to a UK server, so that all internet traffic of PC A would routed through the UK server, while PC B's traffic was routing through the ISP provided server from Bergen, Norway. The rvizweb service which was tunneled to *https://robotapi.isensetune.com*, was successfully accessible through this URL, including the underlying rosbridge websocket connection. The occupancy grid map was successfully being generated to PC A during the operation. The final RTABMAP generated map in 3d point cloud and grid occupancy map along with trajectory is shown in figure 3.21.

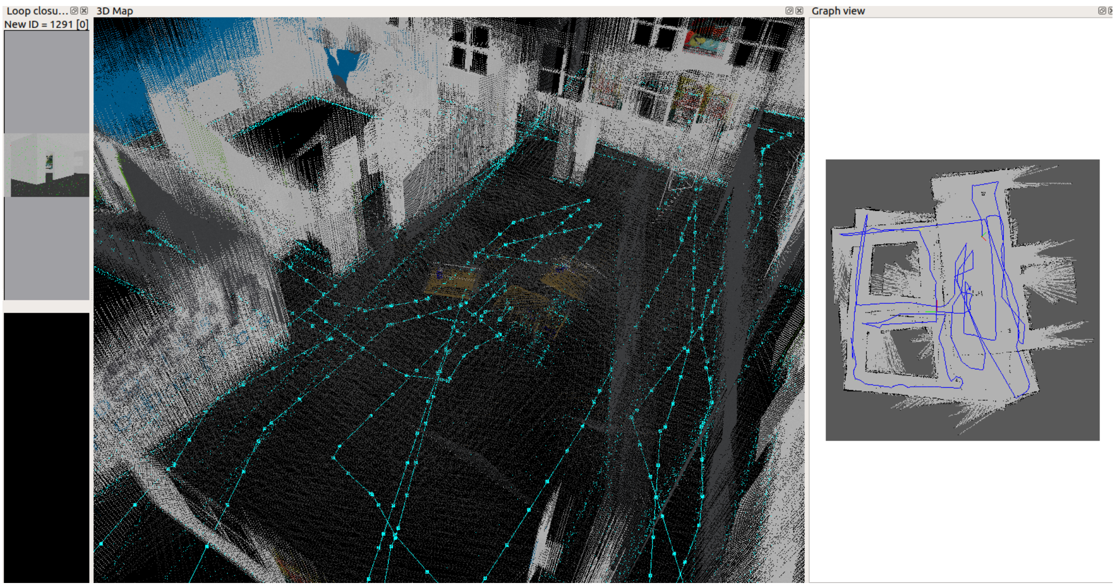


FIGURE 3.19: Robot's estimated trajectory and 3d dense Point cloud map

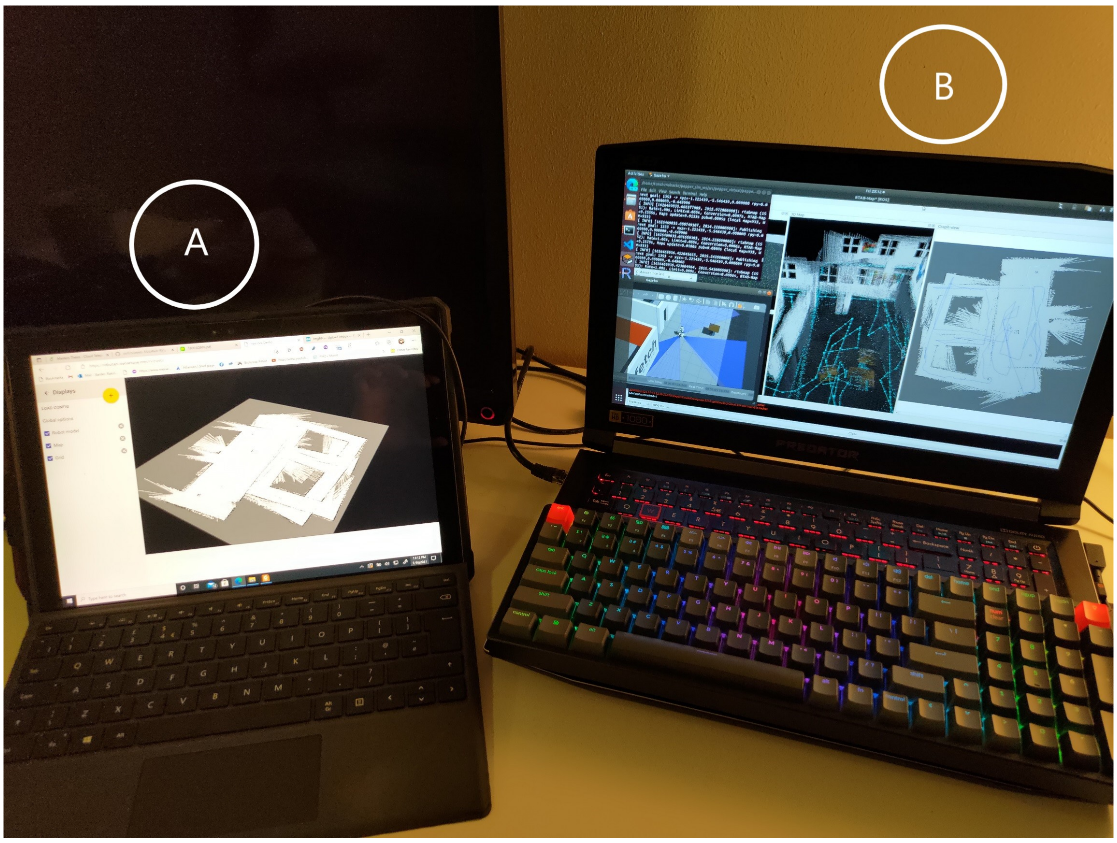
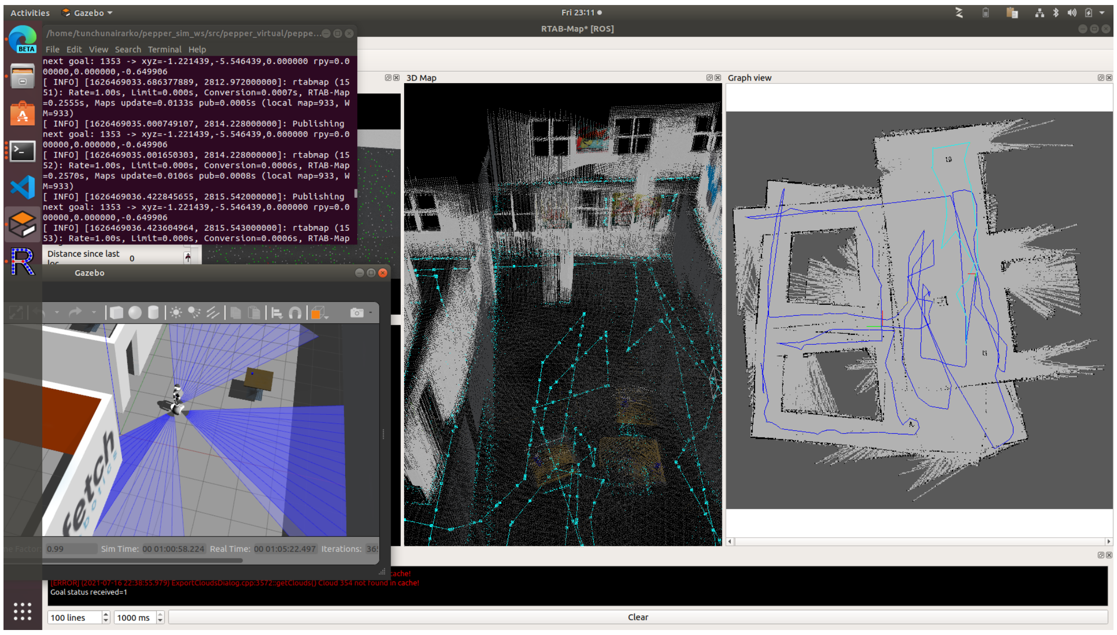


FIGURE 3.20: Live demonstration of testing the mapping process over the web. PC-B was running rtabmap-ros, along with rvizweb and RetraDev LocalCloud, while PC-A was used to browse the access URL (robotcon.isensetune.com/rvizweb) of the map visualizer from the UK, which was routed through a VPN service


Activities
Gazebo
/home/tunchunairarko/pepper_sim_ws/src/pepper_virtual/peppe...
File Edit View Search Terminal Help
RTAB-Map* [ROS]
3D Map
Graph view
Distance since last loc 0
Gazebo
Sim Time: 00 01:00:58.224
Real Time: 00 01:05:22.497
Iterations:
100 lines
1000 ms
Clear

```
next goal: 1353 -> xyz=-1.221439,-5.546439,0.000000 rpy=0.0
00000,0.000000,-0.649906
[ INFO] [1626469033.686377889, 2812.972000000]: rtabmap (15
51): Rate=1.00s, Limit=0.000s, Conversion=0.0007s, RTAB-Map
=0.2555s, Maps update=0.0133s pub=0.0005s (local map=933, W
M=933)
[ INFO] [1626469035.000749107, 2814.228000000]: Publishing
next goal: 1353 -> xyz=-1.221439,-5.546439,0.000000 rpy=0.0
00000,0.000000,-0.649906
[ INFO] [1626469035.001650303, 2814.228000000]: rtabmap (15
52): Rate=1.00s, Limit=0.000s, Conversion=0.0006s, RTAB-Map
=0.2570s, Maps update=0.0106s pub=0.0008s (local map=933, W
M=933)
[ INFO] [1626469036.422845655, 2815.542000000]: Publishing
next goal: 1353 -> xyz=-1.221439,-5.546439,0.000000 rpy=0.0
00000,0.000000,-0.649906
[ INFO] [1626469036.423604964, 2815.543000000]: rtabmap (15
53): Rate=1.00s, Limit=0.000s, Conversion=0.0006s, RTAB-Map
```

```
[ERROR] (2021-07-16 22:38:55.979) ExportCloudsDialog.cpp:3572::getClouds() Cloud 354 not found in cache!
Goal status received=1
```

Figure 3.21: Final maps generated as Occupancy grid and dense point cloud. The cyan color on the Occupancy grid map shows an attempt made to plan a path between two waypoints

3.4.3.3 Discussion on the result

The goal of this use case experiment was to map and localize Pepper in an unknown environment inside the Gazebo world, and visualize with rvizweb as a RSR microservice in real-time over the internet, while being hosted inside the LocalCloud. The performance and the quality of the map generation was, however, not satisfactory. This maybe because there was not enough illumination in the Gazebo world, which was especially observed around some corner of the three corridors inside the apartment. The RTABMAP frequency might have also played a role, as the map update rate was chosen at 1Hz, while the teleoperation was occuring at 1m/s inside the apartment. The issue was identified later in the middle of the experiment, and was update rate was changed to 5Hz. During the initial phase, the teleoperation was also conducted much faster, which might have caused errors into capture images around the outside walls. There were several collisions that occured while teleoperating the robot, which might have also caused issue with the mapping. Another issue which was observed in the web visualization, that is it did not render Pepper properly on the map. It was observed that while map update was starting, Pepper was rendered correctly at the beginning, however it disappeared once the update continued. This could have be an issue with the rvizweb plugin failing to render pepper's URDF and meshes properly. Although Pepper was not visible on rvizweb, the occupancy grid generation was being conducted successfully in real-time over the web, and the visualization service running as a successful RSR component by being exposed over the internet securely by the RetraDev framework was observed.

3.5 Performance evaluation of RetraDev

For any telerobotic framework, one of the biggest challenges is to tackle the communication delay in mobile communication. Since the central component of the RetraDev framework resides on a private network hosted by a laptop or a PC, its performance can be significantly impacted by the quality of the network and the data transmission through the Bastion tunnel.

An initial benchmarking of the RetraDev framework was conducted to evaluate its performance. The benchmarking was conducted primarily on the basis of its communication performance to evaluate the quality of the exchange of information between the different microservices. The performance of RetraDev LocalCloud for hosting cloud telepresence robotic services was also analyzed.

The following criteria were analyzed:

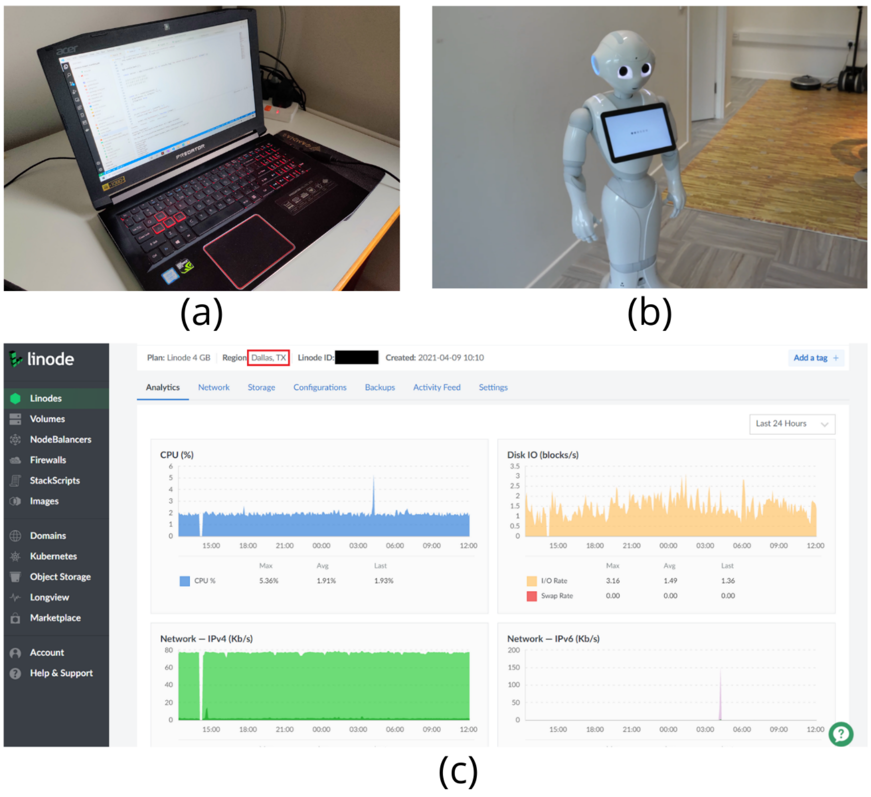
- RTT (Round-Trip Time)
- Latency of the I/O channels
  - **Input channels:** Sonar, battery charge, gaze tracker status
  - **Output channel:** Teleoperation

3.5.1 Experimentation setup

For the experimentation and benchmarking, the laptop of the author was used. The laptop was an Acer Predator Helios 300 running on Windows 10 Pro. The RetraDev LocalCloud was setup and configured to run following its installation process (See Appendix for details regarding the installation). A Linode Virtual Machine (Virtual Private Server) running on Ubuntu 20.04 was chosen for running the Bastion Host along with the RetraDev Relay server. The experimentation was conducted with the robot Pepper, where the robot clients were already installed and configured. Table 3.5 shows the used machines/services for the benchmarking, their technical configurations and physical locations of each components. Each microservices in the LocalCloud were assigned a PORT and a URI for their operation. Figure 3.22 shows the experimentation setup containing the components at three different locations globally.

RetraDev components	**Geographical location**	**Used machines/services**	**System configuration**
LocalCloud	Bergen, Norway	Acer Predator Helios 300	Intel Core i7, Nvidia GTX 1060, 16GB RAM, Windows 10 Pro
		Zyxel Emg3525-T50b Dual band router	2x2 802.11ac WLAN, AC1200 WiFi, NAT configured
		Telia Norge As	Download speed: 52.11 Mbps, Upload speed: 48.90 Mbps
Bastion Host+Relay server	Dallas, Texas, USA	Linode VM with Linode Firewall (with root access)	1 core CPU, 1 GB RAM, Ubuntu 20.04, Nginx v1.14.0, npm 7.9.0, Certbot 1.16.0
Telepresence robot clients (Animus and RetraDev)	Currie, Edinburgh	Pepper v1.8a	NaoQi OS v2.5, python2, Atom E3845 Quad core CPU, 4GB DDR3 RAM

TABLE 3.5: Technical specifications of the setup for benchmarking the RetraDev framework



Plan: Linode 4 GB | Region: Dallas, TX | Linode ID: [redacted] | Created: 2021-04-09 10:10

	Max	Avg	Last
CPU %	5.36%	1.91%	1.93%

	Max	Avg	Last
I/O Rate	3.16	1.49	1.36
Swap Rate	0.00	0.00	0.00

FIGURE 3.22: Experimental setup (a) PC running the RetraDev LocalCloud (b) Pepper located at the RALT lab, HWU (c) RetraDev Bastion Host powered by Linode VM

3.5.2 Results

Round-Trip request delays

To measure the Round-Trip Time (RTT), a request-reply model was employed in the form of ping-pong scheme [143]. The robot client inside Pepper sent the ping requests, while in response the RetraDev server sent back pong responses. The client calculated the RTT value by measuring the time window between the request and the response (Figure 3.23).

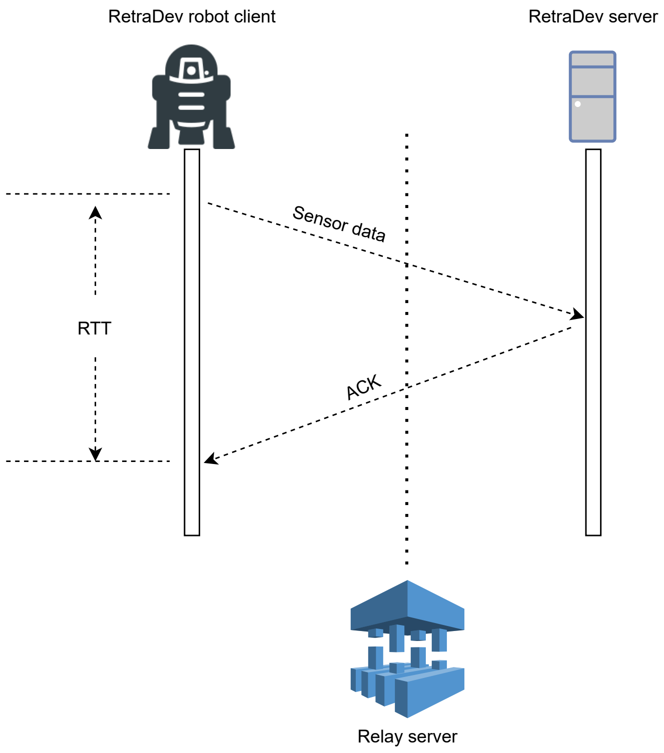


Figure 3.23: Round trip diagram between the RetraDev robot client and the server

Figure 3.24 shows the round-trip time delays in several cases. In the first case, a ping-pong message with a data packet 60b was sent to the server and received back. In the second case, a ping-pong message with a data packet 60Kb (a base64 encoded jpeg image) was sent to the server and received back. During both cases, the teleoperation microservice was actively controlling the robot. The startup RTT values were excluded from the calculation. From the figure, we can see that the RTT delay with 60kb data was slightly higher than 60b at some points (worst case RTT for 60kbps reached at 714.23ms), however both the RTs had similar delay patterns (60b averaging at 61.04ms and 60kb averaging at 105.52ms). The variance decreased for both of them as the time progressed, showing the connection reliability. The mean RTTs were much lower than the highest acceptable threshold standard for RTT by the International Telecommunication Union, which is 500ms [144, 145].

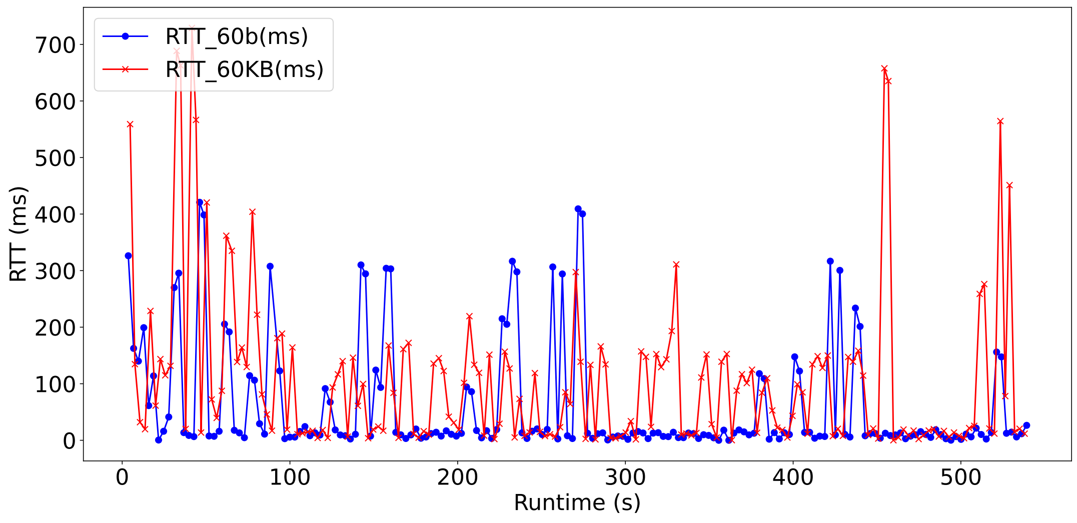


FIGURE 3.24: RTT delays captured for round-trip events during two test session

For the third case of testing with round-trips, a simulated jitter was added for inter-packet arrival through the websocket channel to a data size of 60b (figure 3.25[8]. The jitter was added by the relay server through the Npm *jitter-time*, which generates random time intervals (in milliseconds) between a maximum and a minimum thresholds on the given time values. We can see that the performance dropped significantly compared to the previous two cases, however the RTT mean value was still below 500ms (323ms). Even during the jitter, the connection maintained its reliability with low variance (var(X)=0.0323). Table 3.6 gives the summary and the quantity of interests resulted from the experiments.

Case	**Packet size**	**No of events**	**RTT_mean (ms)**	**RTT _variance**	**Execution time (s)**
RTT_60b	60b	178	61.04	0.0105	539.225
RTT_60Kb	60Kb	178	105.52	0.0208	538.022
RTT_jitter	60b	164	323.96	0.0323	538.9374

TABLE 3.6: Round-Trip Time statistical summary of the three conducted sessions

Single-trip latency performance (input channels)

Figure 3.26 shows the single-trip latency graph for three separate input channels of the RetraDev framework: sonar, batter and gaze track status in a single telepresence session. The frequencies of transmission of each input signals were different, however the graph how the latency events were recorded across the runtime of the session. The

[8]https://www.npmjs.com/package/jitter-time

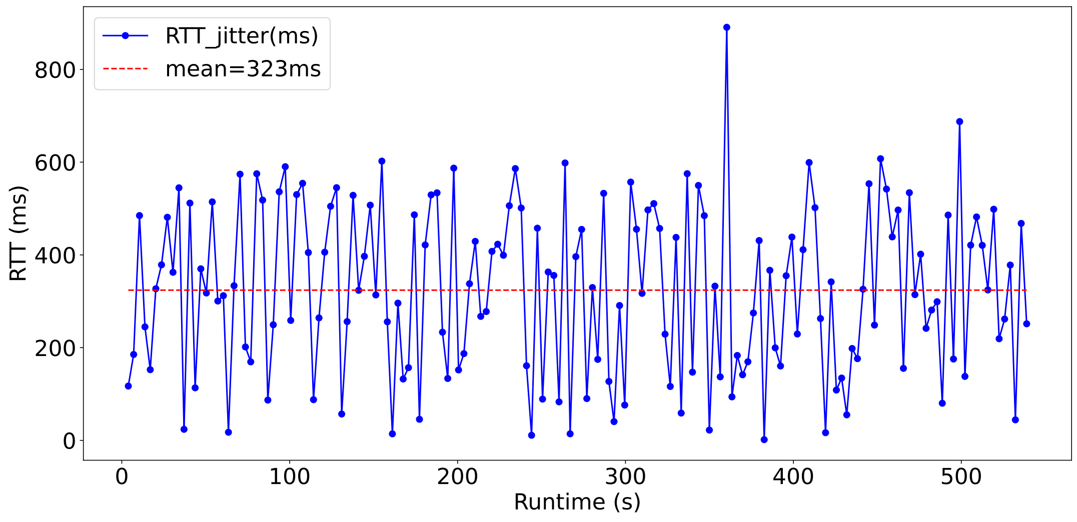


FIGURE 3.25: RTT delays captured for round-trip events during a simulated jitter session

orange color on the bars shows the time taken for a socket event to travel from Pepper to the RetraDev relay server, while the blue color on the bars show the time taken for the socket event to travel from the relay server to the RetraDev LocalCloud. From the figure and table 3.7 we can see that the average delays between the Pepper-Relay were smaller compared to the Relay-LocalCloud pairs. This can be due to the fact that the trip from the Relay server to the LocalCloud through the Bastion tunnel is longer than the trip from Pepper to the Relay server. This displays an overhead cost of using a Relay server in the RetraDev architecture.

Input channel	**Data size**	**No of events**	**Pepper-Relay latency _mean (ms)**	**Relay-LocalCloud latency _mean(ms)**	**Total Latency _mean (ms)**
Sonar	136B	69	76.016	172.958	248.974
Battery	122B	35	24.600	91.972	116.573
Gaze track	123B	46	60.325	139.217	199.541

TABLE 3.7: Single-Trip Latency statistical summary of the three conducted sessions

Single-trip latency performance (teleoperation)

Figure 3.27 shows the latency performance of teleoperation across three different sessions conducted on three different dates (May 25, June 08 and June 24). These three sessions were low-load sessions, meaning the teleoperation was being conducted at a low frequency only for testing purpose. The average latencies across the three sessions were 135.196 ms, 202.506 ms and 82.375 ms respectively. We can see that the latencies for

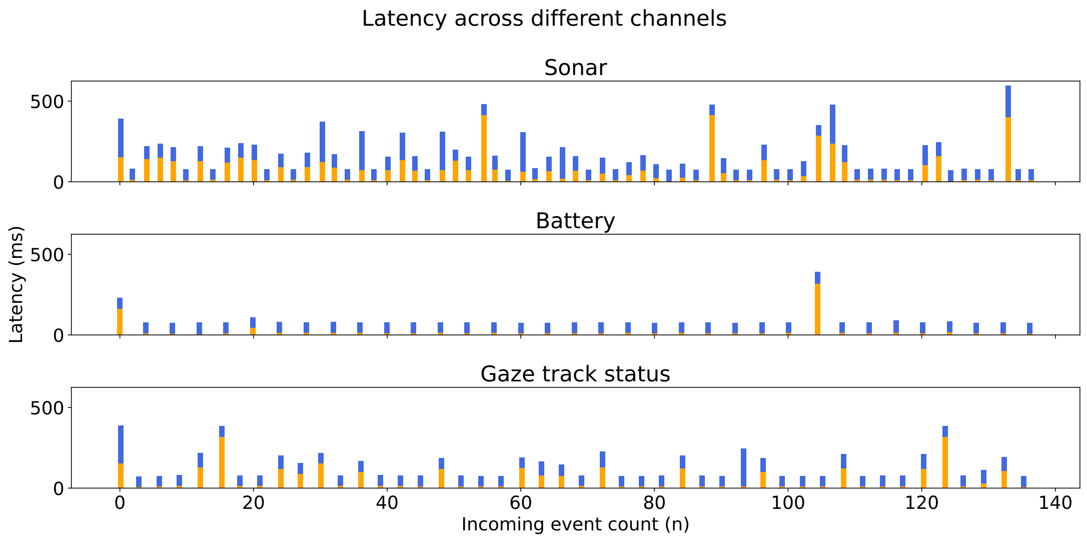


FIGURE 3.26: Latency across different incoming channels of RetraDev LocalCloud. Orange colors on the bar shows how much time a socket.io event took to reach from Pepper to the relay server

teleoperation in each of the session were minimal, except for the session on June 08, when some major delays were experienced at the beginning of the session. However, the connection become approximately stable afterwards.

Figure 3.28 shows the latency performance of teleoperation across two different sessions conducted on two different dates (June 15 and July 13). These two session were heavy-load sessions, meaning the teleoperation was being conducted in parallel to teleconferencing, including sonar, battery and gaze tracker streaming to the RetraDev LocalCloud for a longer time. The session of July 13 was collected from a user evaluation session conducted in this study (Chapter 5). We can see that even under heavy load, the teleoperation channel maintained transmission and reception stability, executing operations in near RT. However, we can see that there were some significant delays observed at some points during the session of June 15, which impacted the remote teleoperation performance by queueing up the signals for some time, and releasing them all afterwards, resulting an unstable motion execution on Pepper.

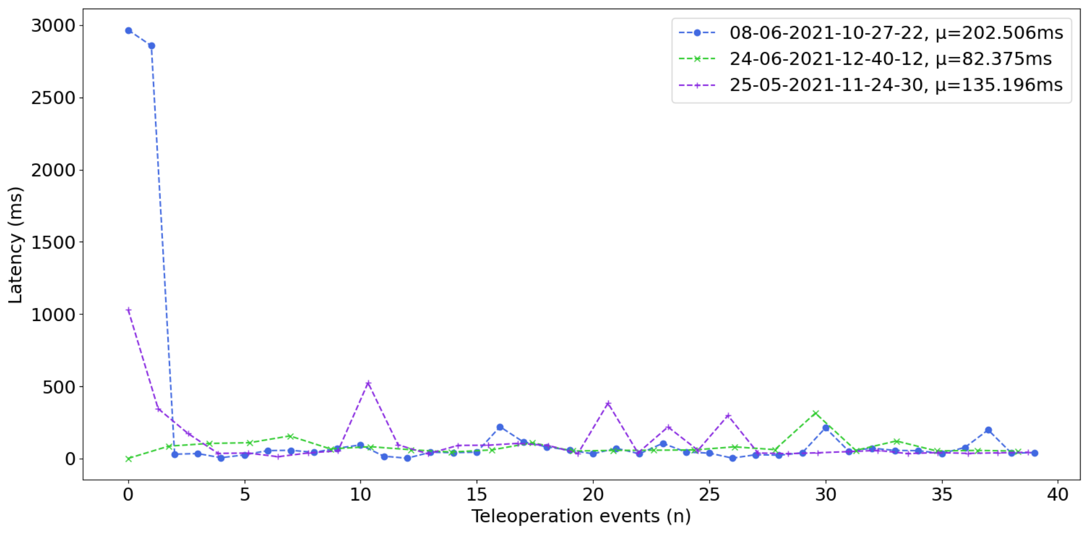


FIGURE 3.27: Teleoperation latency under low load (High time duration between each teleoperation command sent, depicting low transmission frequency)

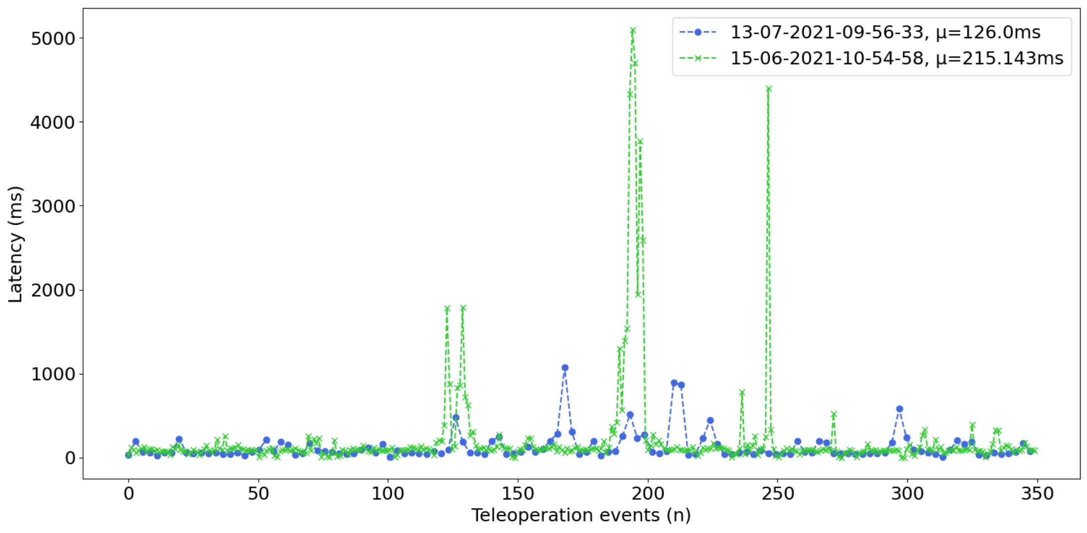


FIGURE 3.28: Teleoperation latency under high load (Low time duration between each teleoperation command sent, depicting low transmission frequency) with other I/O channels of Pepper active

Chapter 4

A Cloud Based Telepresence Robotic Application for Elder Care

In this chapter, a cloud-based elderly telecare application using telepresence robots named **HWU Telecare** will be presented. The proposed application was developed using the **RetraDev** framework. This chapter will point out some requirements for elderly remote caregiving, drawing reference from the literature study, and how the proposed HWU Telecare addresses them. This will be followed by its architecture, implementation and the proposed user interface for remote caregivers.

4.1 Design goals

When it comes to elder care, a remote caregiving opportunity for caregivers should involve the following abilities:

- Non-physical consultation (teleconsultation)
- To be virtually present with the elders and look after their needs
- Routine checkup facility
- Patient monitoring
- Responding to emergency situations

HWU Telecare is a cloud-based telecare solution for elderly care that uses Pepper as a telepresence robot to act as a medium to provide assistanceship by the caregivers. At its current iteration of development, its software has been designed to provide the following features:

- A modern, web-based telepresence user interface for caregivers to conduct robotic telecare operations
- Remote face tracking ability of a robot to track face during teleconferencing
- Secured teleconferencing capability for the caregivers to reach out to their patients quickly from anywhere
- Real-time and safe robotic teleoperation with obstacle avoidance capability
- User access control and management
- Care telepresence robot monitoring and management
- Elder patient health monitoring and data acquisition

The underlying cloud and backend architecture of HWU Telecare uses the RetraDev framework. The aforementioned features have been developed and tested with a physical Pepper v1.8a robot.

4.2 Software architecture

The software architecture of HWU Telecare using the Pepper robot is shown in Figure 4.1. The front-end of the application was developed using ReactJs, which is a Javascript based front-end UI development framework. Alongside React, Bootstrap 4 and CSS3 has been used for front-end layout construction, template-based UI component usage and to stylize the interface. The two main communication mediums of HWU Telecare with the backend is REST API and SocketIo. The backend of the application is served directly by the RetraDev server itself. In terms of development, HWU Telecare application is essentially a MERN (Mongo, Express, React, Node) stack application.

As we can see in figure 4.1, the two robotic clients, Animus and RetraDev, each wraps and maps a set of Pepper's NaoQi API endpoints. Animus client maps the base motors (WheelB, WheelFR and WheelFL) and the head motors (HeadYaw, HeadPitch) with its *motor* modality, while it maps the top camera of Pepper's head (kTopCamera) with its *vision* modality. The RetraDev robot client maps the sonars (front_sensor and

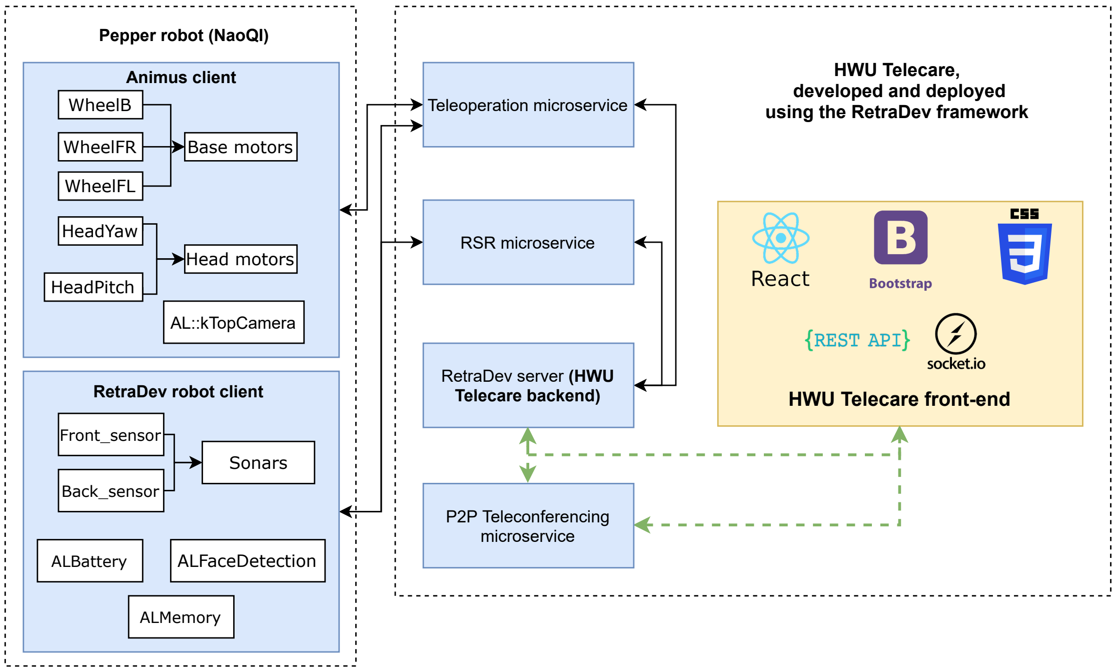
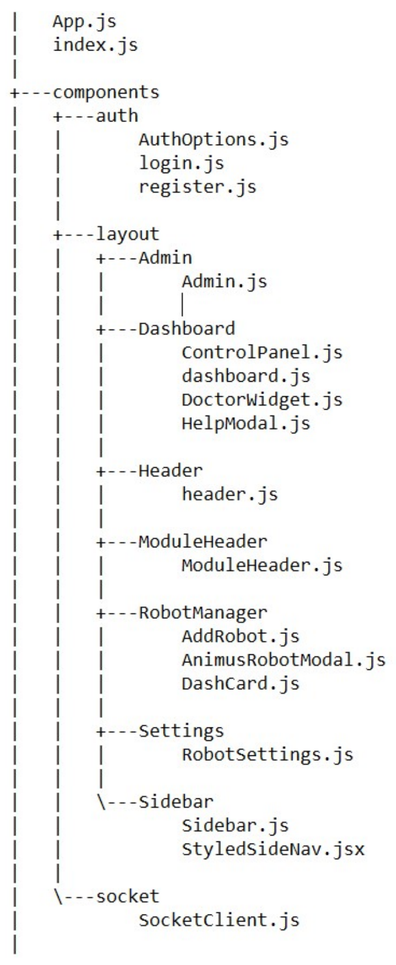


Figure 4.1: Architecture of HWU Telecare using Pepper

back_sensor) with its *ROBOTSONARDATA* channel and the battery (Charge sensor, Device/SubDeviceList/Battery/Charge/Sensor/Value) with its *ROBOTBATTERYDATA* channel. Another custom channel was added to the RetraDev robot client for Pepper, which is *ROBOTFACETRACKER*. *ROBOTFACETRACKER* maps the *ALFaceDetection* API of NaoQi, and allows front-end users to enable or disable face tracking ability for the robot remotely. This was implemented to give a remote user an ability to always point Pepper's camera towards the face of a local user while having a teleconference call.

4.3 Core UI components

Several React components have been developed to complete the UI structure. The complete UI structure inside the React front-end app is shown in Figure 4.2. The components have been classified based on their types (UI pages, headers, sidebars, section elements and communication clients), and their working operation. The "auth" parent component organizes all UI components related to user authentication, while the "layout" parent component organizes everything related to the telepresence dashboard.



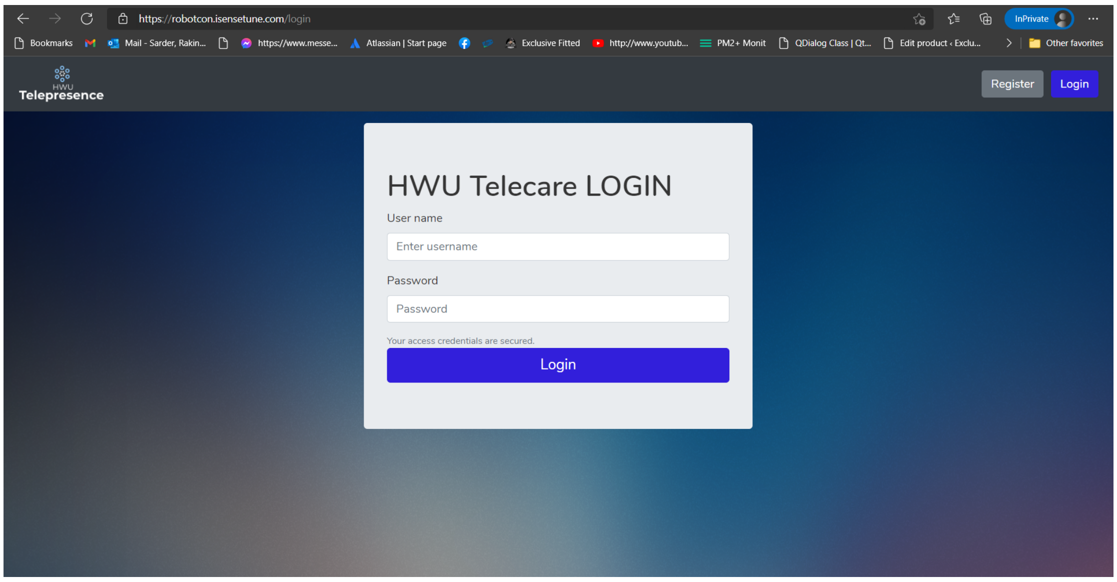
```
|   App.js
|   index.js
|
+---components
|   +---auth
|   |       AuthOptions.js
|   |       login.js
|   |       register.js
|   |
|   +---layout
|   |   +---Admin
|   |   |       Admin.js
|   |   |       |
|   |   +---Dashboard
|   |   |       ControlPanel.js
|   |   |       dashboard.js
|   |   |       DoctorWidget.js
|   |   |       HelpModal.js
|   |   |
|   |   +---Header
|   |   |       header.js
|   |   |
|   |   +---ModuleHeader
|   |   |       ModuleHeader.js
|   |   |
|   |   +---RobotManager
|   |   |       AddRobot.js
|   |   |       AnimusRobotModal.js
|   |   |       DashCard.js
|   |   |
|   |   +---Settings
|   |   |       RobotSettings.js
|   |   |
|   |   \---Sidebar
|   |           Sidebar.js
|   |           StyledSideNav.jsx
|   |
|   \---socket
|           SocketClient.js
|
```

Figure 4.2: Structure of the UI components

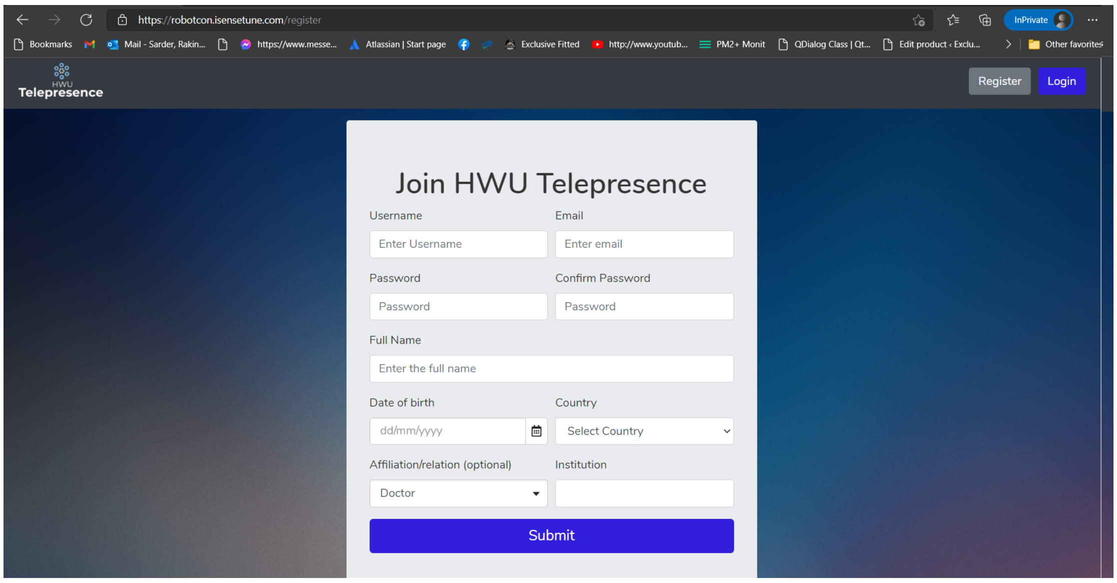
4.3.1 Authentication

Authentication is the landing component of the UI of HWU Telecare. The authentication component groups the *Login* and *Register* components (figure 4.3). When a user enters the HWU Telecare application, they are initially routed to the *Login* component. User can log in to the system using their username and password. While typing the password, users can click on "Show" to display the password. When the user submit the login form, an HTTP POST request is made to the RetraDev server with the username and password in the request body via Axios. The server then searches the database to look for the existence of the user. If the user exists and the password is valid, the backend generates a session JWT with a validity period of 10 days, signed by the *JWT_SECRET* of the RetraDev server. Upon receiving the session token *auth_token*, the central *App* component of the front-end app sets a *UserContext* data with the token and the user data, which is a context variable and it is stored in the browser's session for the current session. The *auth_token* is stored in the Local Storage of the user's browser, which is used for validating the user later so that they do not have to log in every time they enter the web application. The returned response codes are as follows:

200 (success)	**500 (error/fail)**	**400 (bad request)**
user: id,displayName, username, email, token: session_token	error: err.message (Username not found/internal server error/wrong password)	error: err.message (bad request)



(A) Login



(B) Register

Figure 4.3: Authentication component

The *Register* component provides a registration form for users to register into the system. Upon routing to the *Register* component, a user needs to fill up some personal details for their profile and their chosen authentication details (username and password). When the user submit the registration form, an HTTP POST request is made to the RetraDev server with the user and authentication details in the request body via Axios. The server backend at first checks whether the user already exists or not. If the user does not exist, then it checks up the password stregth, followed by the verification that the user submitted all the form details. If everything goes well, then the user is registered into the database, and the backend generates a session JWT with a validity period of 10 days, signed by the *JWT_SECRET* of the RetraDev server. Upon receiving the session token *auth_token*, the central *App* component of the front-end app sets a *UserContext* data with the token and the user data, which is a context variable and it is stored in the browser's session for the current session. The *auth_token* is stored in the Local Storage of the user's browser, which is used for validating the user later so that they do not have to log in every time they enter the web application.

200 (success)	**500 (error/fail)**	**400 (bad request)**
user: id,displayName, username, email, token: session_token	error: err.message (Username already exists/internal server error/weak password)	error: err.message (bad request)

4.3.2 Telecare Dashboard v1.0

The **Telecare dashboard v1.0** gives the user to access their registered telepresence robots, conduct video conferencing sessions, teleoperate their robot using navigational controls and perform telecare operations. In this project, two user dashboards have been proposed, v1.0 and v2.0. Dashboard 2.0 is currently under development, and it is being designed and modeled to incorporate a lot of other functionalities from the RetraDev framework, including some RSR microservices.

A screenshot of the Dashboard 1.0 is shown in Figure 4.4. As we can see that there are several components existing in the dashboard, that corresponds to the different services that HWU Telecare offers. A user can access the Dashboard upon authentication. At first the dashboard component mounts by checking whether the user has a valid *auth_token* (session token) to access the dashboard. Next, the dashboard checks whether Pepper is switched on or not. If Pepper is not online/not accessible, then the dashboard disables all teleoperation buttons and teleconference "Join" button until the robot is live. When the robot is live, then the *header* starts displaying the "Connectivity strength" of the telerobotic session. The user can then click "Join" to send a "Join" request to the local user with the robot, who can then accept the request to allow the remote user access to Pepper's video feed. Figure 4.5 shows how it looks when the remote user is successfully connected with the robot and the local user.

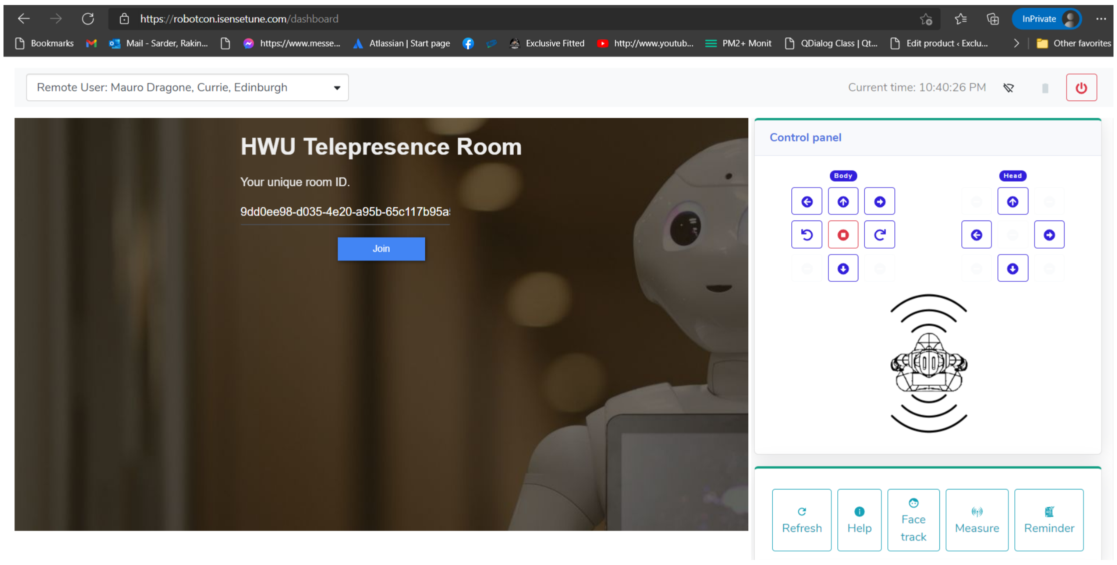

https://robotcon.isensetune.com/dashboard
Remote User: Mauro Dragone, Currie, Edinburgh
Current time: 10:40:26 PM
HWU Telepresence Room
Your unique room ID.
9dd0ee98-d035-4e20-a95b-65c117b95a
Join
Control panel
Body
Head
Refresh
Help
Face track
Measure
Reminder

FIGURE 4.4: Dashboard v1.0 User Interface

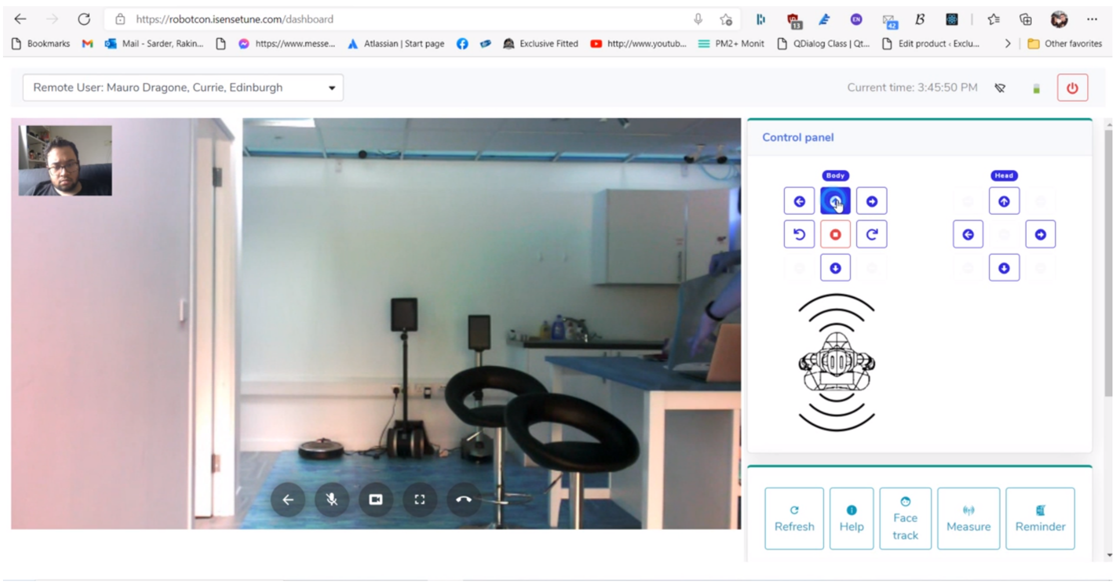

https://robotcon.isensetune.com/dashboard
Remote User: Mauro Dragone, Currie, Edinburgh
Current time: 3:45:50 PM
Control panel
Body
Head
Refresh
Help
Face track
Measure
Reminder

FIGURE 4.5: Dashboard with Pepper on and teleconference session started

4.3.2.1 Header

The *Header* component of the Dashboard contains the local user information, the battery state of the robot, the RetraDev connectivity strength between HWU Telecare and Pepper, and the disconnect button (Figure 4.6).

The connectivity strength is measured using the time of flight between the signal sent from the robot client to the RetraDev server. If the robot is offline, the connectivity

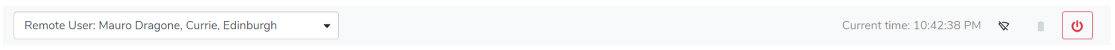

Remote User: Mauro Dragone, Currie, Edinburgh
Current time: 10:42:38 PM

FIGURE 4.6: Dashboard header

strength displays ”0” bars. The python code for measuring the connectivity strength in the backend is:

```python
 while(True):
     #get current time
     curtimeStamp = time.time()

     #get timestamp acquired on the robot side
     clientSignal = retraDev_client.getStream()

     time_diff=round(curtimeStamp-clientSignal.timeStamp,2)
     #inverting the TOF jsut to interpolate within the percentage
range
     inv_time=1/time_diff

     inv_time_cap=numpy.clip(inv_time,low_inv_time,high_inv_time)
     signal_percentage=sig_interpolate(inv_time_cap,low_inv_time,
     high_inv_time,0,100)
     #emit the signal strength to the dashboard
     socketio.emit("SIGNALSTRENGTH",signal_percentage)
```

4.3.2.2 Control Panel

The Control Panel child component contains the teleoperation controls and the proposed telecare tools for routine checkups. Currently, it contains two tools (i) To measure vital signs remotely using robot’s integrated sensors, and (ii) To set up dynamic reminders for the robot to remind the elderlies unattended. Both of these have been implemented in a Wizard-of-Oz (WoZ) style, as of now. They have been developed to simulate the behaviors of actual vital sign measurement and reminder setting using the robot, which was useful for the user evaluation of this application (Chapter 5). It was not possible to implement them with the physical robot because of not having physical access to the robot during the entire working phase of this project, and also it falls outside the scope of this project. The vital sign measuring WoZ tool contains three types of measurement options, they are: (i) blood pressure, (ii) pulse and (iii) body temperature (Figure 4.7). Each of these measurement tools have been modeled as no-contact measurement systems, based on the studies of [146, 147, 148]. When a user clicks on one of these options, it generates a pseudo random vital sign data, and saves it into the database for the respective elder. For setting up reminder (Figure 4.8), a user have the option to dynamically set up custom reminders, aside from the default 4 options provided in the dropdown (Medicine, Therapy, Diet, General advice). Users can also enter a message they would want the robot to say.

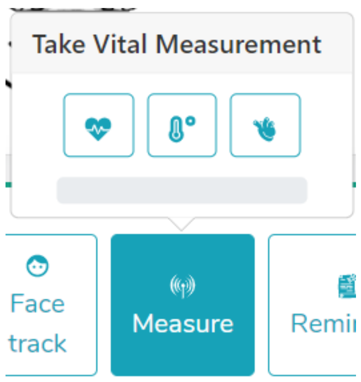


FIGURE 4.7: WoZ Vital sign measuring toolkit

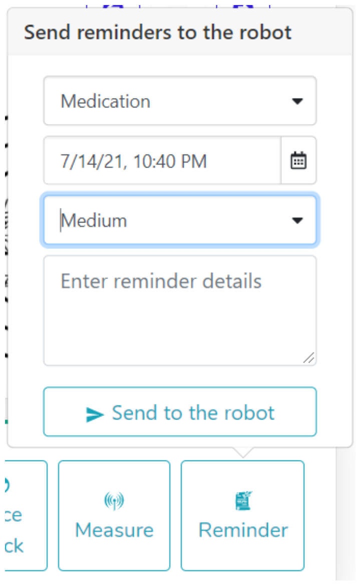


FIGURE 4.8: WoZ robotic reminder setting toolkit

FIGURE 4.9: Proposed telecare tools to conduct remote routine checkups

There are two sets of on-screen teleoperation controls on the dashboard, one for Pepper's *base* navigation, and one for it's *head* rotation (figure 4.10). The *base* navigation buttons work on continuous press, as long as the user keeps pressing a base navigation button, it will transmit the corresponding navigation command to the robot via the teleoperation microservice.

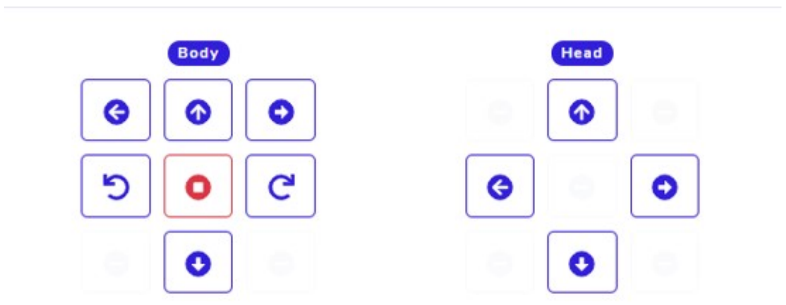


FIGURE 4.10: Teleoperation controls

The teleoperation sends linear velocity or angular velocity (for left and right rotation) values to Pepper, however it does not send a navigation termination when the user stops pressing a teleoperating button. This can result in continuous motion in Pepper, and it would continue to navigate without stopping. This was a critical issue that was observed in Animus library of the teleoperation microservice, which was resolved by sending an additional *nullmotion* command just after the release of each navigation buttons. This ensured that the robot does not continue to drift away without control.

```
  useEffect(() => { //React useEffect to check for clickVal state
 change
      if (clickVal === 0) { //0 means key is released
          socket.emit("frontenddata", "nullmotion")
      }
  }, [clickVal])
```

When the *head* rotation buttons transmit commands to the robot, Animus converts each input as the pitch and/or yaw angular positions. Animus does not transmit angular velocity to the head, as a result they resultant head rotation is a fixed output, unlike the base one. Table 4.1 shows the parametric values that were chosen for motion transmission purpose. The values are fixed for each input teleoperation command in the backend, and these values were derived after conducting numerous teleoperation trials. There were a few reasons for which the speed or threshold values were chosen to be fixed. One of the reasons was that adding controls for base velocity and angular increment may increase the complexity for the end user, observed from the study of [123]. The idea of developing teleoperation for Dashboard v1.0 was to make it as less challenging as possible for the caregivers, to address technical self-efficacy.

Base velocity (m/s)	**Base angular velocity (s^{-1})**	**Head rotation (θ)**
1.0	0.2618	3

Table 4.1: Fixed value parameters chosen to move Pepper

Apart from the teleoperation controller, the Control Panel child component also contains an obstacle visualization for remote operators to observe whether their is any obstacle in front or back of the Pepper robot. Figure 4.11 shows how different obstacle detection scenario may look like. In case if the robot's sonars send a value of 0.5m or less to HWU Telecare, the state of the obstacle detector visualizer changes immediately.

The base navigation buttons will stop transmitting navigation commands to the robot, until the received sonar data start getting back over *0.5m*. Of course, cloud-based obstacle avoidance is not a safe feature for any robot, since collision avoidance is a real-time necessity. That's why a runtime obstacle avoidance runner of Pepper was added to be triggered on every boot up. The threshold for built-in obstacle avoidance was set to *0.35m*.

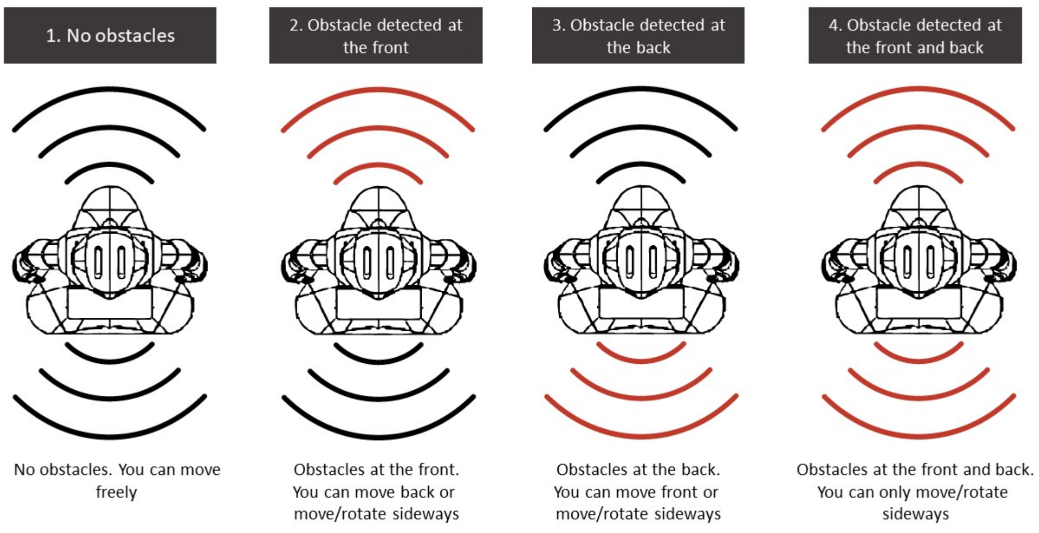

1. No obstacles
2. Obstacle detected at the front
3. Obstacle detected at the back
4. Obstacle detected at the front and back
No obstacles. You can move freely
Obstacles at the front. You can move back or move/rotate sideways
Obstacles at the back. You can move front or move/rotate sideways
Obstacles at the front and back. You can only move/rotate sideways

FIGURE 4.11: Visualizing obstacles in front, back or both of Pepper in real-time during teleoperation

4.3.2.3 Keyboard Teleoperation

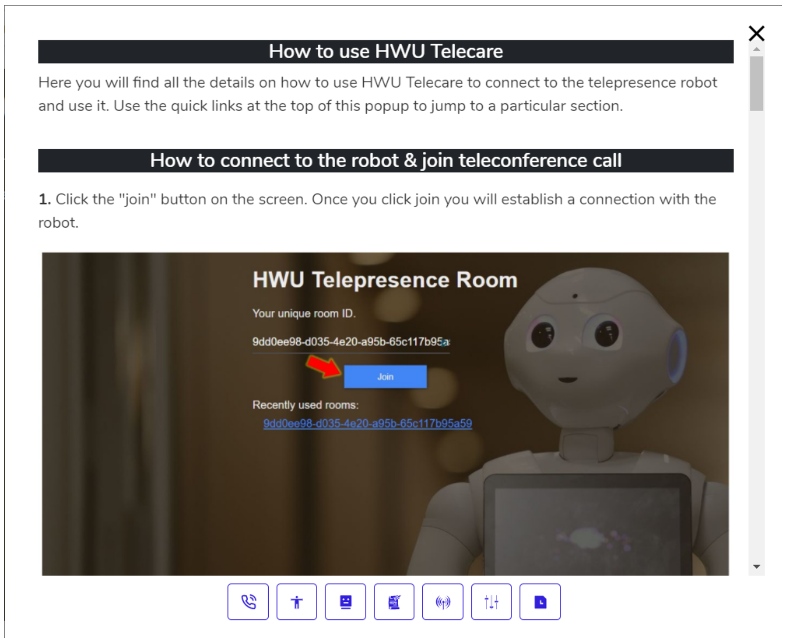
Along with the on-screen robot teleoperator, a keyboard-based teleoperation component was also added to the dashboard. Table 4.2 shows the button mapping to each directions. All the internal working process for keyboard-based teleoperation is similar to on-screen controls, that is the base navigation buttons work on continuous press, and sends a ”nullmotion” command upon release.

Keyboard button	**Description**	**Transmitted nav enum**
w	Move forward	*forward*
a	Move left	*left*
s	Move back	*back*
d	Move right	*right*
q	Rotate left	*rotate_left*
e	Rotate right	*rotate_right*
i	Increase head pitch	*head_up*
k	Decrease head pitch	*head_down*
j	Decrease head yaw	*head_left*
l	Increase head yaw	*head_right*

TABLE 4.2: Keyboard navigation map

4.3.2.4 Help Modal

The Help modal is an additional child component which was added to the dashboard to provide detailed instructions to the user. It contains detailed explanation of each elements of the dashboard, and how to use them (Figure 4.12).



How to use HWU Telecare

Here you will find all the details on how to use HWU Telecare to connect to the telepresence robot and use it. Use the quick links at the top of this popup to jump to a particular section.

How to connect to the robot & join teleconference call

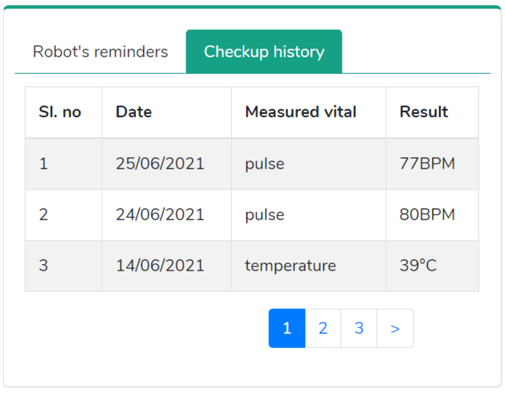
1. Click the "join" button on the screen. Once you click join you will establish a connection with the robot.

HWU Telepresence Room
Your unique room ID.
9dd0ee98-d035-4e20-a95b-65c117b95a
Join
Recently used rooms:
9dd0ee98-d035-4e20-a95b-65c117b95a59

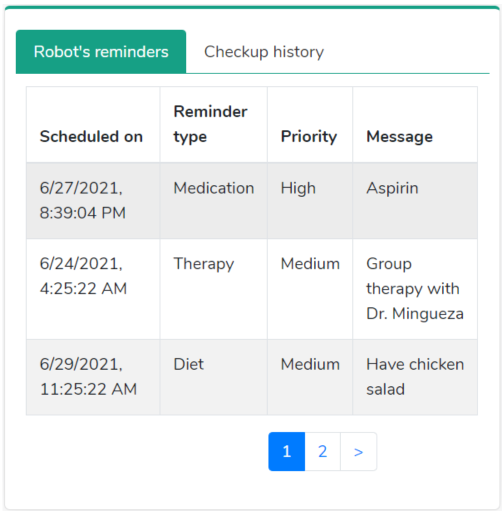
FIGURE 4.12: Detailed help and instruction popup for a telepresence user to use HWU Telecare

4.3.2.5 Local User Records

The Local User Records (LUR) component on the dashboard displays the simulated reminders sent to the robot by a user, and the lifetime vital measurement record using the vital sign measurement toolkit. For every new records, the LUR components updates automatically (Figure 4.13).



Sl. no	Date	Measured vital	Result
1	25/06/2021	pulse	77BPM
2	24/06/2021	pulse	80BPM
3	14/06/2021	temperature	39°C



Scheduled on	Reminder type	Priority	Message
6/27/2021, 8:39:04 PM	Medication	High	Aspirin
6/24/2021, 4:25:22 AM	Therapy	Medium	Group therapy with Dr. Mingueza
6/29/2021, 11:25:22 AM	Diet	Medium	Have chicken salad

Figure 4.13: Local User Records of HWU Telecare accessible from the dashboard

Chapter 5

HRI Evaluation of a Care Telepresence Remote User Interface

Use experience is crucial when it comes to designing welfare technologies for elder care. As [16] has mentioned, designing technological solutions for elder care should be inclusive. A Human Robot Interaction (HRI) pilot study was conducted to evaluate the quality and acceptability of a user interface designed for HWU Telecare, with Pepper as the telepresence robot employed for elderly care. The objective of this study was to understand how caregivers envision the proposed care telepresence solution and learn from their experience to improve the solution in the future. More specifically, the objectives were:

1. To evaluate the core functionalities of HWU Telecare along telepresence robot
2. Evaluate the Dashboard v1.0 of HWU Telecare, which was completed during the course of this project
3. Evaluate two utility tools to perform remote checkup using a telepresence robot (in a Wizard-of-Oz fashion)
4. Understand and learn from the experiences and feedbacks of the participants as to how the proposed HWU Telecare could be intended to be used in potential care-centric applications, and also to improved further

In addition to this, an additional goal of this study was to design a comprehensive user acceptance and evaluation model for care telepresence robotic users which can be used for conducting future user evaluation studies with professional or non-professional caregivers. The designed user evaluation method were formed basing on the fact that our proposed telecare solution will be cloud-based, providing a remote caregiving option to the caregivers employed in care facilities, nursing homes, hospitals or at home. That is why this study was held online remotely, which was divided into multiple sessions for each participants. Another reason to conduct the experiment remotely was due to the COVID-19 restrictions, as it was not possible or safe to physically meet and engage with the participants during the pandemic.

5.1 Experiment setup

The robot that was used to conduct this study was Pepper NaoQI v1.8. Pepper comes with an attached tablet on its chest, however in the NaoQi version of Pepper the chest tablet has limited capability to stream A/V content. To test the performance of the built-in chest tablet of Pepper several mp4 A/V clips were attempted to play, however Pepper's chest tablet was not capable to play those streams. As a result, we attached an additional Android tablet (Samsung Galaxy) on top of the existing Pepper's tablet for this experiment, which acted as the teleconferencing medium for the robot. Since the HWU Telecare application was designed and deployed using the RetraDev framework, the client p2p web app of the RetraDev teleconferencing service (see section 3.3.1.2) was opened up on the tablet as the teleconference "host". Figure 5.1 shows the setup of Pepper with the local user of the experiment at the Robotic Assisted Living Testbed (RALT), Heriot-Watt University.

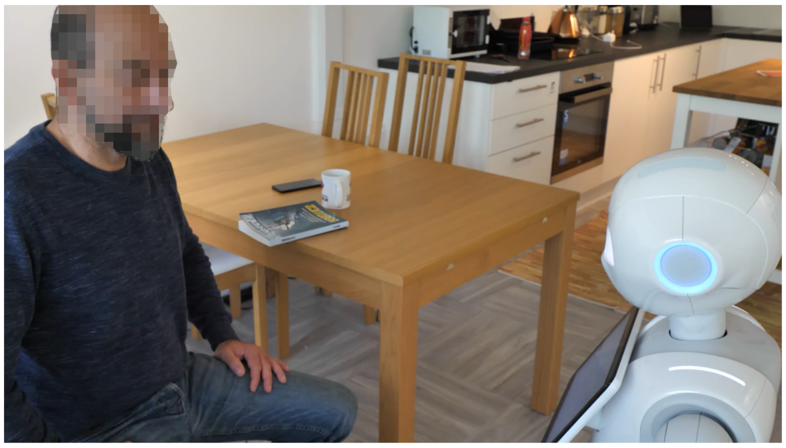
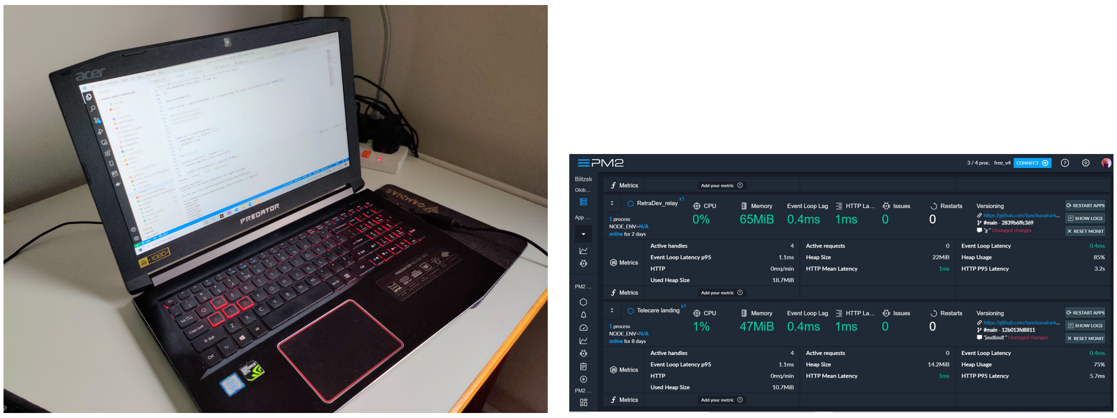


Figure 5.1: The setup at the lab: Pepper with the local user of the experiment at the Robotic Assisted Living Testbed (RALT), HWU

Participants conducted the experiments using their own laptops or PCs, from their preferred locations. The HWU Telecare application along with the RetraDev microservices were deployed and served from the LocalCloud, which was hosted on the researcher's PC. The main application was assigned a public URL (https://robotcon.isensetune.com) using the RetraDev Bastion Host. Figure 5.2 shows the setup of the RetraDev LocalCloud server along with the RetraDev Bastion host running the Relay server.



```
RetraDev_relay x1        CPU   Memory   Event Loop Lag   HTTP La...   Issues   Restarts
1 process                0%    65MiB    0.4ms            1ms          0        0
NODE_ENV=N/A
online for 2 days
Versioning
https://github.com/tunchunairark...
#main - 2839b6ffc369
"g" Unstaged changes

Metrics
Active handles             4          Active requests         0       Event Loop Latency   0.4ms
Event Loop Latency p95     1.1ms      Heap Size               22MiB   Heap Usage           85%
HTTP                       0req/min   HTTP Mean Latency       1ms     HTTP P95 Latency     3.2s
Used Heap Size             18.7MiB

Telecare landing x1      CPU   Memory   Event Loop Lag   HTTP La...   Issues   Restarts
1 process                1%    47MiB    0.4ms            1ms          0        0
NODE_ENV=N/A
online for 8 days
Versioning
https://github.com/tunchunairark...
#main - 12b013fd8811
"jasdfasdf" Unstaged changes

Metrics
Active handles             4          Active requests         0       Event Loop Latency   0.4ms
Event Loop Latency p95     1.1ms      Heap Size               14.2MiB Heap Usage           75%
HTTP                       0req/min   HTTP Mean Latency       1ms     HTTP P95 Latency     5.7ms
Used Heap Size             10.7MiB
```

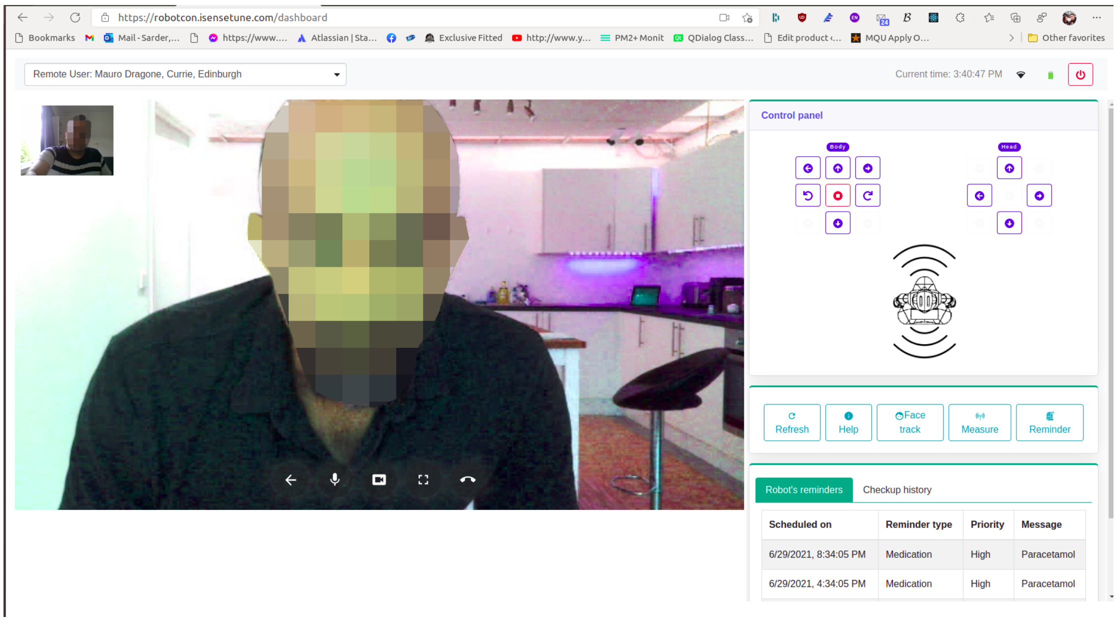
FIGURE 5.2: Remote setup of the RetraDev LocalCloud-hoste HWU Telecare application and Relay server running on the Bastion Host

No specific system requirements were asked from the participants of the experiments, as the telerobotic application was web-based. The participants were only asked to have a stable internet connection, and to participate in the test session using a PC with minimum screen resolution of 1024px. Participants were also asked not to participate using their phones, as the Telecare web application was not designed for smartphones. Figure 5.3 shows a teleconference session from the participant's side during an evaluation session.

5.2 Participants

5 participants participated in this pilot study with individual sessions for each of them. The participants were approached and chosen based on the one of the following criteria:

- Active or past working experience in a elder care organization, such as care homes, nursing homes and rehabilitation centers (as a care professional or as a worker)
- Adult person who have some experience in supporting elderly relatives, friends or peers
- Research, professional or academic experience in the field of psychology, gerontology, robotics and AAL



Remote User: Mauro Dragone, Currie, Edinburgh

Current time: 3:40:47 PM

Scheduled on	Reminder type	Priority	Message
6/29/2021, 8:34:05 PM	Medication	High	Paracetamol
6/29/2021, 4:34:05 PM	Medication	High	Paracetamol

FIGURE 5.3: A screenshot of the UI during a teleconference session from the participant's side during an evaluation session

Among the 5 participants, 4 participants were active professionals in two elder care organizations in Scotland, and 1 was a academician in the field of psychology. The recruitment had been conducted through informal invitation via email and via posting a "Call for participation" notice on the website of CARE research group of Heriot-Watt University. Participation in this pilot study was on a voluntary basis. Except gender and age group, no other personal information was collected and processed to preserve the anonymity of the participants. The participants also had the rights to their submitted data, and they were also free to withdraw from this pilot study at any time.

This pilot study was conducted after the Research Ethics committee of Heriot-Watt University approved the project. All participants were sent a participant information sheet, data management policy and a consent form prior to their participation in this study. All the evaluation sessions were conducted in accordance to the proposed methodology.

Although it was intended to involve more participants in this study, there were some complications to invite more participants in this study. The timeline of this project was 6 and a half months, and considering the development completion timelines of the RetraDev framework and HWU Telecare, there was limited time available to conduct the pilot study on a larger scale. The whole process of the pilot study was completed within the last two months of this dissertation submission, including the study design, evaluation model construction, experiment setup and invitation. The Research Ethics Committee usually take 2-3 weeks to approve a project, as a result there were even limited time to gather more participants.

5.3 Experiment design

The pilot study was conducted in two weeks. A total of 5 sessions were conducted as per the number of participants. Each testing sessions were of 30 40 minutes of length, excluding the questionnaires.

The participants were given a link to a website that contained details of the experiment. It contained the (i) technical requirements, (ii) a short video clip of the user interface, (iii) explanation of the experiment tasks and (iv) an instruction manual on how to use HWU Telecare. Each session started with a brief introduction of the researcher with the participant of that session. After that, the researcher would briefly explain each parts of the experiment to the participants, and how they can find help on the user interface during their sessions. After that the participants would start their experience sessions by navigating to the main HWU Telecare application provided on their user interface and start with their task list. Once they completed their sessions, they would find the link to the questionnaire to be filled up voluntarily to conclude their sessions.

5.3.1 Task list

Since this experiment involved only the potential remote users of HWU Telecare, an abstract *elder* persona was created, who would be communicating with the participants and taking their consultation. The project supervisor volunteered to be the actor of this invented persona. A short background story of the abstract *elder* persona was written, and was provided to the participants through the pre-experiment information website.

The primary task of the participants was to teleconsult the *elder* using HWU Telecare. According to the designed story, the *elder* lived in his residence accompanied by the telepresence robot, which they would be testing. The participants would also conduct a routine checkup with the utility tools available their user dashboards. As already mentioned, these checkup tools were designed in a ”Wizard-of-Oz” style to gather insights on their usefulness and acceptance, in accordance to this project.

The tasks were divided into two sections. They are as follows: **Stage one**

1. Start the Telecare application
2. Register a new account at HWU Telecare
3. Go through the help option on the dashboard to introduce with the components of the user interface

4. Start a teleconference call, send a call join request to the *elder*
5. Look around the surroundings use the robot's camera feed, head control and body navigation
6. Find the mirror and look through the eyes of the robot
7. Test and practice navigation using on-screen controls
8. Practice navigation, using the keyboard
9. Locate the following 3 objects in the apartment: couch, TV and coffee maker
10. Disconnect the call
11. Logout/disconnect from the dashboard

Stage two

1. Wait for a few minutes (3 5) after stage one, then log back in
2. Start a teleconference call once again with the *elder*.
3. Start a conversation with the *elder*. The *elder* would be sitting on the couch in the apartment.
4. Follow the *elder* when he moves around the apartment while he make a cup of coffee.
5. Follow the *elder* back when he finishes making the cup of coffee and moves back to the couch.
6. Use the vital measurement toolbox on the dashboard toolbox to measure the remote user's vitals.
7. Set a reminder/save notes for the robot to remind the *elder* later.
8. Disconnect and logout off the interface when the conversation ends to conclude the testing session

At the end of the task sessions, the participants were given a questionnaire which took approximately 10-15 minutes to complete. A thank you note was sent to each participants via email when they submitted their feedbacks.

5.4 Evaluation method

For designing the evaluation model, some constructs were formed for the purpose of this study, while some were adopted from standard user evaluation theoretical models. Table 5.1 shows the list of all the constructs that were used to formulate the questionnaire. The definitions of each of the constructs were tailored according to the proposed system. Since the proposed system is designed around telepresence robot, the term "telepresence robot" was used in place of "robot" to reduce ambiguity. The table also mentions the full name of each constructs along with their abbreviations, and the abbreviated terms were used to describe the construct interrelations. The questionnaire used for this study has been added in Appendix D.

The RTQ and PB constructs were formed based on the objectives and the design of this experiment. Except TLX (which was adopted fully from NASA-TLX) [149], the rest of the constructs were inspired from established user acceptance models. They are:

- **UTAUT:** Unified Theory of Acceptance and Use of Technology [150, 151]
- **TAM:** Technology Acceptance Model [150]
- **HANCON:** A hybrid of UTAUT and PAM model for evaluating attitude towards telepresence robots [152]
- **PSSUQ**: Post-Study System Usability Questionnaire [153]
- **ECT:** Expectation-Confirmation Theory [154]

Perceived Usefulness (PU) and Perceived Ease of Use (PEOU) are the primary constructs of TAM [150], however since the UTAUT model merges TAM in its architecture, and HANCON adopts its constructs from the UTAUT model, TAM has been studied and added as the inspiration sources [152]. Both PU and PEOU construct questionnaires have been partially modified to fulfill this study goals. Since this study primarily focuses on the remote web UI, the theoretical models Expectation-Confirmation Theory (ECT) and Post-Study System Usability Questionnaire have been studied and partially adopted to define several constructs related to the UI [153, 154]. We can also see that the most adopted model in this study was the HANCON model, proposed by Han et. al. HANCON model was initially proposed as a hybrid form of UTAUT and the Post Acceptance Model (PAM) by the authors [152]. Although the HANCON model focused on the acceptance and social context of telepresence robots in academic institutions, the constructs defined in that model were found to have broader significance beyond the academic usage of social telepresence robots.

Construct	**Definition**	**Inspired from**	**Reference**
Robotic Telepresence Quality (RTQ)	A user's perception on his/her/their success in operating the proposed telepresence robotic care system	*Proposed*	[9, 155]
Perceived Usefulness (PU)	A user's perception that using the proposed telepresence robotic care system will enhance elder caregiving works	TAM, UTAUT, HANCON	[150, 152, 151]
Perceived Ease of Use (PEOU)	A user's perception that using the proposed telepresence robotic care system will be free of effort	TAM, UTAUT, HANCON	[150, 152, 151]
Perceived Barrier (PB)	A user's concerns regarding the issues of using the proposed telepresence robotic care system	*Proposed*	[66, 10, 9, 8]
Task Load Index (TLX)	Calculating and assessing subjective Mental Workload (MWL) on a user while participating in an experiment	NASA-TLX	[149]
Information Quality(INFQ)	A user's opinion regarding the information and their organization on the User Interface	PSSUQ	[153]
Satisfaction (S)	A user's perceived satisfaction on using the proposed telepresence robotic care system	HANCON, ECT, PSSUQ	[152, 154, 153]
Expectations (E)	Addressing a user's adaptive expectations on the robotic care system	HANCON, ECT, PSSUQ	[152, 154, 153]
Perceived Enjoyment (PE)	A user's perception that he/she/they would enjoy providing elder care using the proposed telepresence robotic care system	HANCON	[152]
Perceived Sociability (PS)	A user's perception that the proposed telepresence robotic care system will allow to perform social behaviors	HANCON	[152]
Social Influence (SI)	A user's perception on how using the proposed telepresence robotic care system would improve his/her/their impression to his/her/their peers and colleagues	UTAUT, HANCON	[152, 151]
Intention To Use (ITU)	A users' intention to continue using the proposed telepresence robotic care system	HANCON	[152]

TABLE 5.1: Hypothetical construct interrelations prepared for the evaluation model

5.4.1 Construct interrelations

From table 5.1 we can see that 13 constructs have been defined for this study. The questionnaire which was provided to the participants contained 8 separate sections. All of these constructs were distributed in the questionnaire into 7 different sections, while the 1st section was designed to collect some basic non-identifiable demographic information from the participants, they are: gender and age range. The second section contained primarily the RTQ construct, the **first** set of questions. The RTQ construct primarily contained tailored questions related to the proposed telepresence robotic application. It contains three close-ended questions regarding the main task completions, and two sets of multi-item scale questions where each sets consisted of 7-point agreement-based Likert scales. The first multi-item set questions asked the participants how they perceived their achievements when they completed specific milestones of each tasks. The first set of questions also contained two open-ended questions regarding features they liked and the issues they might have faced.

The **second** set of multi-item scale questions asked the participants regarding how they observed the different entities of the application (A/V quality, contents, UI layout, options, navigation button lag and latency).

The **third** part of the questionnaire contained a set of 5-point Likert scale questions. The questions were related to the ambient persona the telepresence robot created for the participants and how they enjoyed the experience while engaging in a social activity using the robot. The constructs included were PE, PS and SI.

The **fourth** part of the questionnaire measured the perceived workload on the participants using the TLX construct. The TLX construct was fully adopted from the NASA-TLX questionnaire, which assessed the following subjectives: (i) Mental demand, (ii) Physical demand, (iii) Temporal demand, (iv) Performance, (v) Effort and (v) Frustration [156].

The **fifth** section of the questionnaire contained the PEOU construct. It contained 5 questions on a 7-point ”Likely” based Likert scale. The PEOU construct assessed how the participants perceived their efforts and comforts when they used the application, and how they perceived the expertise they would be needing if they intend to use it in the future.

The **sixth** section of the questionnaire contained 6 questions on a 7-point ”Likely” based Likert scale, 4 from the PU construct, 1 from the SI construct and 1 from the PE construct. The questions were tailored primarily about how the users (primariliy caregivers) perceived the usefulness of this telecare robotic application, and how it might assist them in their elder care-centric jobs. It also contained 1 open-ended question which asked the participants whether there were any particular feature they found the most useful from their perspective.

The **seventh** section of the questionnaire contained the PB construct. The PB construct was formed for the purpose of this study. During the literature review phase, it was found that in several user studies on elder care and telepresence robotics, the participants commonly express concerns regarding the following 6 potential issues (refer to section 2.3.3). They are:

1. Privacy
2. Confidentiality
3. Data protection
4. Technical self-efficacy
5. High learning curve
6. Network connectivity errors

However, most of these concerns came through open-ended questionnaires, rather than from a solid user evaluation model. Only a few studies have worked on designing a defined barrier evaluation model for robotic ambient assisted technology users, such as this one. That is why in this study, a new construct had been proposed that evaluates how remote telepresence users perceived the potential barriers to a telepresence robotic application employed for elder care. This construct contains 5 questions addressing all the aforementioned barriers, and asks the participants how they perceive each of the barriers on a 7-point ”Agreement” based Likert scale. In addition to this, the participants were also asked if there were any additional barriers they observed outside the 6 potential barriers.

Finally, the **eight** section contained the INFQ, E, ITU and S constructs. The questions of these sections were initially tailored basing on the PSSUQ questionnaire, which was later changed and modularized to fulfill the objectives of these study. This section primarily asked the participants to summarize their overall experience. There were one set of 7-point ”Agreemenet” based Likert scale questions, and two open-ended questions asking the participants for any additional feedback.

In the proposed model, an additional ”USE BEHAVIOR” or USE construct have been employed following [151]. The USE construct is an abstract construct in this model, which does not define any specific set of questionnaire, but rather identifies whether the evaluation satisfied the targetted use behavior of the proposed system. Six constructs have been theorized to contribute as direct determinants of ITU (Intention To Use). They are: PEOU, PB, PE, PU, E and SI. The constructs RTQ, INFQ and PS have been theorized as the indirect determinants of ITU. Furthermore, S (Satisfaction) is directly determined by ITU, PEOU and PB. Finally, S determines the USE construct.

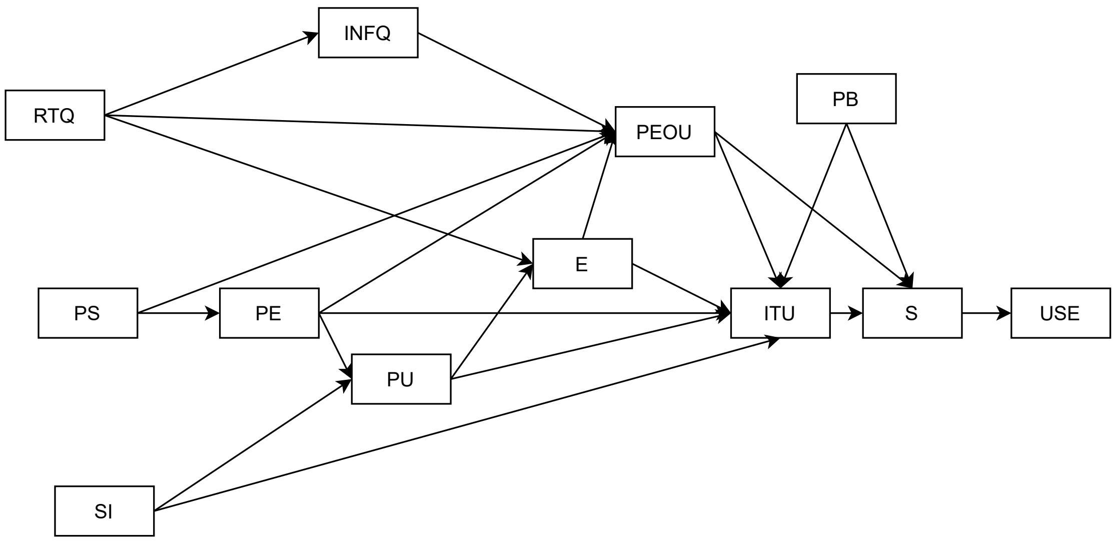


Figure 5.4: Construct interrelations diagram of the model proposed for this study

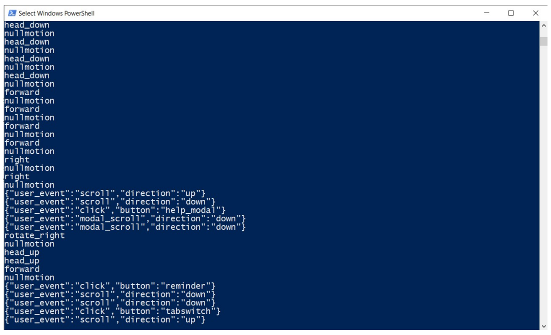
The following hypotheses were considered (here ”determined” means direct contribution, while ”influenced” mean partial contribution) (figure 5.4):

- **H1 -** ITU (Intention To Use) is determined by PEOU, PB, PE, PU, E and SI
- **H2 -** PEOU is determined by RTQ, INFQ and influenced by PS.
- **H3 -** INFQ is influenced by RTQ.
- **H4 -** PE is determined by PS.
- **H5 -** PU is influenced by PE and influenced by SI.
- **H6 -** E is directly influenced by PU and RTQ
- **H7 -** S is directly determined by PEOU, ITU and influenced by PB
- **H8 -** USE is directly determined by S

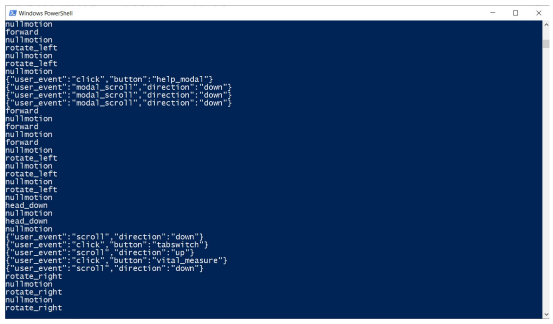
5.5 Results

Among the 5 participants, 3 were successful into running the HWU Telecare application in their PCs. There were some technical issues, which 4 of the 5 participants faced while conducting the experiment. During the experiment, user events were tracked and logged with the consent of the participants to see how the participants interacted with the user interface. The user event logs were collected from two of the three participants, as per their consent. Two example sets of events tracked from two participants are shown in figure 5.5. From the event logs, it was seen that in both the cases, the participants attempted to follow various steps of the tasks. One participant however, started with the navigating immediately after connecting with the robot, and followed back to the task list afterward. Another participant opted to navigate in front of a mirror inside the apartment to look through the eyes of the robot and enjoy the robotic ambience of the participant.

In both the cases, however, participants faced some latency issues. One participant did not report any latency issue during the experiment, but reported in the questionnaire, while the other participant reported immediately to the researcher. One reason why the second participant might have faced some latency issues was because of attempting to move too fast. By its design, teleoperation commands transmitting over the internet to a physical robot is prone to lags and latency, and when the transmission frequency increases, the lags increase as well (refer to section 3.5.2 for the latency observed in teleoperation). In addition to transmission, the second participant also reported that the screen resolution of the application did not support the participant's PC in the beginning. The issue had been reported and was fixed by the conducting researcher immediately, and allowed the participant to join once again.



```
head_down
nullmotion
head_down
nullmotion
head_down
nullmotion
head_down
nullmotion
forward
nullmotion
forward
nullmotion
forward
nullmotion
forward
nullmotion
right
nullmotion
right
nullmotion
{"user_event":"scroll","direction":"up"}
{"user_event":"scroll","direction":"down"}
{"user_event":"click","button":"help_modal"}
{"user_event":"modal_scroll","direction":"down"}
{"user_event":"modal_scroll","direction":"down"}
rotate_right
nullmotion
head_up
head_up
forward
nullmotion
{"user_event":"click","button":"reminder"}
{"user_event":"scroll","direction":"down"}
{"user_event":"scroll","direction":"down"}
{"user_event":"click","button":"tabswitch"}
{"user_event":"scroll","direction":"up"}
```



```
nullmotion
forward
nullmotion
rotate_left
nullmotion
rotate_left
nullmotion
{"user_event":"click","button":"help_modal"}
{"user_event":"modal_scroll","direction":"down"}
{"user_event":"modal_scroll","direction":"down"}
{"user_event":"modal_scroll","direction":"down"}
forward
nullmotion
forward
nullmotion
forward
nullmotion
rotate_left
nullmotion
rotate_left
nullmotion
rotate_left
nullmotion
head_down
nullmotion
head_down
nullmotion
{"user_event":"scroll","direction":"down"}
{"user_event":"click","button":"tabswitch"}
{"user_event":"scroll","direction":"up"}
{"user_event":"click","button":"vital_measure"}
{"user_event":"scroll","direction":"down"}
rotate_right
nullmotion
rotate_right
nullmotion
rotate_right
```

FIGURE 5.5: Logs from user event tracking collected from two sessions

Only one participant fully completed the experiment and the questionnaire, while the other two participants joined, used the application and opted to provide open-ended feedbacks on the usability without the questionnaire. As a result, in this the user evaluation, the core focus was given to the qualitative analysis on the user provided evaluations. Table 1 presents the responses provided by participant 3 who completed the questionnaire. We can see that the participant's experience and perception according to most of the constructs were positive. However, the mean response of TLX 0.0 depicts that design of the experiment may have been challenging for the participant to complete. Also, we can see that the PB construct response is 0.0, because the participant had some concerns regarding the connectivity and latency of the application. The participant also identified some issues during the telepresence session, as they were expressed through the 2nd part of RTQ. The social potential of the telepresence robotic application was highly observed by the participant (PS-mean=2.0, SI-mean=2.25, PE-mean=2.5). The participant found the solution useful (PU-mean=2.75) and easy to use (PEOU-mean=1.4). Overall, the solution was met with satisfaction to the participant (S-mean=2.0), and the participant found potential to use the solution in caregiving works (ITU-mean=2.0).

Construct	**Scale**	**Mean response**
RTQ-1	+3,-3	1.57
RTQ-2	+3,-3	0.44
PE	+3,-3	2.5
PS	+3,-3	2.0
SI	+3,-3	2.25
TLX	+3,-3	0.0
PEOU	+3,-3	1.4
PU	+3,-3	2.75
PB	+3,-3	0.0
INFQ	+3,-3	1.33
E	+3,-3	0.75
ITU	+3,-3	2.0
S	+3,-3	2.0

TABLE 5.2: User evaluation result from a participant

The responses from the open-ended questionnaire and feedbacks from the participants revealed that the participants liked the idea of deploying Pepper along the telepresence robotic application in elder care facilities. Regarding the most interesting feature and usefulness of the platform, participants liked their telerobotic ability to roam inside an elder's house and follow the person in care:

"The ability to move around the room letting you explore and follow the person in care."

"The most interesting and useful regarding my work experience is the ability to move around and check on the person in case of any emergency"

All the participants reported on the teleoperation latency during their sessions. One participant experienced significant delay in teleoperation, which impacted the participant's quality of experience. The other two participants did not experience the same level of latency like this participant, but they also expressed their concerns regarding the latency and driving the robot. One participant, however, perceived the latency issue as an adaptation issue, since it was the first time the participant used this telerobotic system:

"Driving was challenging because of the latency however if you are careful enough to slowly press controls can be done. Controlling the head of the robot was much easier. I am sure that a bit of practice would make things easier."

In addition to the teleoperation latency, participants also expressed concerns regarding the teleconference video and sound quality. According to their feedback, the video resolution at the remote user's end was low, also there were some synchronization issues between the video and the sound which the users reported. The low video quality quality can be drawn back to Pepper's camera resolution, as it could capture frames using its native colorspace YUV422 at 30fps in 640x480 (VGA) resolution. The synchronization delay between the sound and video can be drawn to issues with the WebRTC connection. One participant addressed the teleconferencing issue through the questionnaire feedback:

"Image quality was blurry in different moments and our conversation was difficult to follow in some point as sound quality was not the best."

As additional feedback to the system during the evaluation, two participants proposed some improvements for the platform. They proposed to have an improvement of the teleoperation connection for the system. One participant additionally suggested some improvements in the user interface regarding the placement and task execution through the telehealth buttons. Another participants suggested to use a camera with a wider camera lens for better navigation around an elder's apartment, and a night vision camera for night time checkup on the elders.

"Maybe a wider camera lense which could provide a wider angle of vision. I found difficult to move the robot controlling his head to check any objects on the floor. Also a night vision mode it might be useful when you have to provide welfare checks during the night."

One participant provided a feedback of how he would use the proposed system for his daily caregiving job. The participant likes the idea of video conferencing-on-a-wheel concept of the system. He also pointed out nighttime usage of the system for elder welfare and unsupervised checkups on the elders in a care facility. He also mentioned that with the proposed system, he will be able to move closer to the elder residents himself to hear properly what the residents say, thus enhancing the quality of the conversation between them.

"The care service, I am part of, delivers support via video calls. It would be beneficial and it would make things much easier for us to have the capacity to move in the customer´s room/flat. A good example of our daily tasks are welfare checks for customers who are tetraplegic mostly during the night. "

"A high quality video image is a great tool as they can suffer of vomiting being a life threatening issue. In many cases, our device has been moved not letting us have a whole image of our customer."

" Also, being able to get closer to them it will increase sound quality which is another common issue as they are not always able to be loud enough to have a conversation being apart. "

Chapter 6

Discussion and Conclusion

Telepresence robots have great potential to positively contribute to the elder welfare sector. The more the demand of elder care services increase, the more people are going to incorporate AAL technologies like telepresence robots into the care ecosystem. In order to fulfill the rising demand, a rapid prototyping framework specifically tailored for telepresence robotic application development can be beneficial for the early stages of care telepresence application development. In this study, a cloud framework was proposed focusing on remote programming, testing and application prototype development for telepresence robotics. It was also demonstrated how this framework can be used to develop and deploy a telecare application that would give remote caregivers a tool to easily connect, engage and checkup on elderlies under their care without being physically present. A standard user acceptance model was designed subjecting remote caregivers to understand their perception, attitude and interaction with care telepresence robotic applications, and it was later used to conduct a HRI evaluation study on the proposed **HWU Telecare** application. This chapter summarizes this dissertation and draws conclusion to regarding the answers to the proposed research questions.

6.1 Project Summary

6.1.1 Addressing the primary research questions

The following research questions were formulated and investigated in this study through literature review and the scope of development:

RQ-1: **What is the state of the art in care-based telepresence robotics?**

In chapter 2, a literature study was conducted following a systematic review process to present a meticulous summary on the existing research and development of care telepresence robotics and elder care. A review methodology was formulated by dividing the study in two stages. In the background study, the concept of Ambient Assisted Living (AAL), care telepresence robotics and cloud robotics were described, along with their potential challenges and requirements. In the state-of-the-art study, a comparative analysis was conducted on various commercial and conceptual robotic telepresence systems and how they are addressing the need and requirements of elder care. In addition to that, some popular communication frameworks reported to be used for robotics were discussed, followed by the discussions on some cloud robotic platforms and services were conducted, and how various elder care robotic researches have adopted them were presented. Finally, a comprehensive analysis on the design considerations for developing UI for telepresence robotic applications based on several HCI studies was presented.

RQ-2: **What can be an ideal cloud-communication framework for remote programming, testing and application prototype development for telepresence robotics?**

Chapter 3 presented the model of a cloud-communication framework for remote programming, testing and application prototype development for telepresence robotics which was named as the **RetraDev framework** (REmote Telepresence Robotic Application DEVelopment framework). The framework followed up on the need and limitations of remotely developing cloud telepresence application development, specially for elder care projects. The proposed framework addressed 4 key objectives: (i) telepresence robot-agnostic development (ii) cloud application development and deployment using own infrastructure for rapid prototyping (iii) achieving modularity using microservices (iv) lightweight and open source. The architecture of the framework was discussed from top to bottom, including how each components of the architecture addresses the core objectives of the framework. The primary component of the architecture, RetraDev LocalCloud was broken down by explaining its internal microservices, such as the RetraDev server, teleoperation service, RSR (Robotic Service Runner) and teleconferencing service. While the server, teleoperation and telepresence microservices are the core microservices to run telepresence robots over the cloud, RSR was proposed as an additional component to address the need for running additional robotic services for RetraDev-connected telepresence robots using ROS, such as SLAM (Simultaneous Localization and Mapping) for autonomous navigation. Detailed processes were presented on how a ROS runtime service can be run on the same local machine of LocalCloud and how the RetraDev framework abstracts it as an RSR microservice, integrates it and deploys it securely over the cloud. To demonstrate the functionality of RSR, a use case for mapping and localization was presented using a simulated Pepper, ROS and the RetraDev framework. By design, RetraDev is essentially a local development framework, but by combining it with the RetraDev Bastion Host the framework establishes its integration with the global cloud network. The internal architecture and the working process of the RetraDev Bastion Host was presented, along with its necessity, significance and the core differences from a regular Bastion Host. The robot-side clients of the RetraDev framework were discussed along with their working processes, how they abstract the robot's internal API with a robot-agnostic event-based API, and how the abstracted API integrates with the microservices of the RetraDev framework. Finally, a performance analysis of the framework was presented by benchmarking the various I/O communication of different microservices.

RQ-3: **How can we measure the quality of experience of caregivers when they use a robotic care telepresence User Interface (UI) to access their patients or relatives?**

From the literature study, it was found out that all elder care-centric robotic services should follow a user-centric design approach involving the caregivers. To evaluate how remote caregivers perceive, accept and intend to use a care telepresence robotic application, a user evaluation model subjecting caregivers was constructed basing on some well-adopted technology acceptance models for welfare technologies.

A telepresence robotic application intended for elderly care named as *HWU Telecare* was proposed in Chapter 4. The application was developed using the RetraDev framework, addressing **RQ-2** in the process as well. The conducted HRI evaluation study on this proposed application focused on evaluating the quality of the user interface and their experience while using the different telepresence components. Although the evaluation study was conducted with a limited number of participants, the participants provided valuable feedback for improving the proposed application.

RQ-4: **Besides teleoperation and teleconferencing, what additional telehealth features needs to be added in a telepresence robotic application to aid remote caregiving tasks?**

The answer to this question was presented in chapter 2 (section 2.4.4, 2.4.1), chapter 4 and chapter 5. To aid remote caregivers to perform their care-centric jobs using telepresence robotics, researchers and robotic developers are integrating several telehealth tools to their user interfaces, such as visualizing physiological data of the elderlies, vital sign monitoring, mental health monitoring and reporting, reminder setting and writing e-prescriptions through their user interfaces. In the proposed *HWU Telecare* application, two experimental telehealth features on the user interface were proposed as Wizard-of-Oz (WoZ) styled components to evaluate the attitude and acceptance of remote caregivers towards these proposed telehealth features on their user interfaces. The participating caregivers provided positive feedback on the inclusion of these telehealth features, and expressed their intention to use these features.

6.1.2 Addressing the extended research questions

The extended research questions were formed once the scope of the primary research questions expanded during the course of this project. The ERQs have been addressed in this study as follows:

ERQ-1: **Does the usage of a social robot for telepresence bring additional benefits for elderly care applications?**

From the literature study in chapter 2, it was found that using social robots for telepresence can bring additional social benefits while engaging with the elderlies. To verify this proposition, in chapter 5, the conducted user evaluation study contained three specific constructs in its evaluation model addressing the social perspective of caregivers using Pepper and the proposed telecare application. Since the pilot study was conducted with a limited number of participants and only one participant completed the evaluation questionnaire, the result to the social perspective evaluation through the questionnaire was inconclusive. However, the one participant who submitted a response perceived the social benefits of using Pepper as a social care robot (response-mean=+2.25). In addition to that, one participant provided an open-ended feedback on their perceived ability to move Pepper's head around and close-in with a robotic teleconference user for enhanced social engagement using Pepper.

ERQ-2: **What are the autonomous features of a humanoid telepresence robot which can contribute to elder care?**

Some useful autonomous features for telepresence robots were studied and discussed in chapter 2. Following up on that, several key autonomous behaviors useful for care telepresence robots were identified, such as autonomous navigation, autonomous docking, conversation following and face tracking during teleconferencing. In this study, the face tracking feature was added to the Pepper robot, so that it could track its local user's face during a teleconference call. The face tracker was designed using the built-in *ALFaceTrackerAPI* of Pepper to track people's face during conversation. This feature was included to the user evaluation study for the participants for evaluation. However, the participants did not notice or understood the face tracking feature properly during their evaluation, probably due to some technical issues that caused the face tracker not to operate. In addition to the face tracker, attempts were made to incorporate autonomous navigation alongside manual teleoperation to *HWU Telecare* using Pepper, however due to conducting the project remotely and not having sufficient access to the robot this feature could not be incorporated.

ERQ-3: **What are the possible constraints robotic engineers have to face when they work with a remotely located telepresence robot?**

The possible constraints robotic engineers face when they work with a remotely located social telepresence robot were addressed in chapter 1 and 2. Some of the constraints identified are: (i) Not having physical access to the telepresence robots (ii) Complex and expensive cloud infrastructure configuration for connecting telepresence robots over the cloud, (iii) Connectivity bottlenecks, and (iv) Requiring DevOps knowledge to develop and deploy cloud-based telepresence robotic applications. These issues were addressed and resolved by **RQ-2**.

6.2 Limitations

The impact of COVID-19 pandemic limited the scope of contribution to this project because of not having accessibility to the lab facility and the robot. As a result, the scope of the project was altered from an initial broad context, so that the project could be completed within the deadline. In addition to that, there are some limitations to the proposed robotic framework, proposed care telepresence robotic application and the HRI user evaluation study conducted in the project. They are as follows:

6.2.1 No autonomous navigation

One of the key initial scopes of this study was to explore and implement a cloud-based semi-autonomous navigation functionality to the experimented robot Pepper. The idea was to explore suitable SLAM approaches for mapping and localizing Pepper in an unknown environment using ROS, then build up an autonomous navigational component for the proposed telepresence robotic application. The experimentation of SLAM using the physical Pepper robot over the cloud, however, was not successful. There were several reasons for not being able to implement a ROS-based SLAM algorithm for localization and mapping remotely using the Pepper available at the lab, such as:

- There were limited available durations and schedules of access to the lab's Pepper while working remotely. Accesses to the lab's Pepper were received by the kind assistances of the members of the RALT lab. Lab members upon arriving to the lab would switch on the Pepper for the researcher, and they would also spare time to troubleshoot any issues on behalf of the researcher since they had the access to the Pepper. However, the lab members themselves had limited schedules and durations each day available for their own lab experiments, which further limited the access to Pepper to conduct extensive experiments.
- The lab's Pepper did not support connecting to rosbridge using roslibpy due to its outdated security certificates.
- It was not possible to cross-compile ROS packages for Pepper remotely and then deploy to the robot.

6.2.2 Bastion Host security risk

While Bastion Host was an integral part of the RetraDev framework to power it into a fully-functional cloud-communication framework, Bastion Host itself has some limitations. If not configured properly, Bastion Host can bring potential security risk for the local PCs as well as the cloud server. The security of the Bastion Host was hardened following standard security measures for Bastion Hosts. However, security strength evaluation of the Bastion Host was not conducted during the course of the study.

6.2.3 Issues with the WebRTC connection for the teleconferencing microservice

There were several issues that were observed and addressed in the teleconference call microservice. In a few occasions it was found that the WebRTC based teleconference call sessions were poor, both video and audio. On top of that, there were lags that were observed between the video and audio during some teleconference sessions. This impacted the quality of experience of the participating users.

6.2.4 ROS compatibility with RetraDev

The compatibility of ROS with the RetraDev framework needs to be further explored. In this study, only one potential use case for the RSR component of RetraDev was explored. However, further compatibility tests and experiments are necessary to incorporate ROS further inside the RSR architecture of RetraDev.

6.2.5 Lags in teleoperation

There were several cases when lags and latencies were reported in the transmission of teleoperation command to Pepper by the users. In a couple of cases, although there were no lags from Pepper's camera stream, some significant delays observed in the transmission of teleoperation commands to the robots (>200ms). This impacted the conducted experiments and also the quality of experience of the user evaluation participants.

6.2.6 Issues with the implemented face tracker

The implemented face tracking feature for Pepper and HWU Telecare did not perform satisfactorily during the user evaluation experiment. Further investigation is necessary to identify the cause and solution to this issue.

6.2.7 Population validity for the HRI study

The population size for the pilot user study was not sufficient enough to draw any conclusive interpretations. Finding participants remotely and scheduling within the limited time was difficult, as the target potential participants were professionals. A large scale study could provide adequate findings for validating and evaluating the proposed telepresence robotic telecare solution, and could also provide further insights on how to plan the future progress of its development

6.3 Future Works

This project was conducted as a pilot study for a larger scale HRI research on care telepresence robots. The future work of this project would involve exploring the potential expansion of the proposed RetraDev framework on a larger scale, and perform high-level technical evaluation of it. In this study, the technical evaluation of the RetraDev framework was limited to benchmarking. In the future, additional tests will be conducted, such as unit test, integration test and functionality test.

Additionally, the ROS compatibility with the proposed framework will be explored, so that the integration between them can be strengthen. The proposed HWU Telecare application would be expanded in the future, to integrate robotic autonomous features to it. The work for this is already in progress. Dashboard v2.0, which has been mentioned in chapter 4 along with Dashboard v1.0, incorporates a point-and-click map based navigation using an RSR component in its UI, along with a robot management feature and implementation of various telehealth features among others. The implementation of these telehealth features need to be done using physical sensors and readers. Therefore once the robot is accessible for physical laboratory works, these sensors would be integrated with Pepper, removing the current WoZ-styled telehealth components on the UI and replacing them with actual integrated telehealth sensors.

Improvements will be made to the teleoperation microservice, by identifying the issues with the current setup and solving them. Issues related to latency and lags, video streaming quality needs to be addressed and will be identified. Additionally, the quality of the teleconferencing microservice will be improved. Currently, there are several reducers added to the teleconference module to optimize the video and audio quality for faster processing on both the remote user and local user's side. However, these reducers might be contributing to the connectivity issues of the microservice. These will be resolved for the future. A new session management system for the teleconference module will be added to keep track of the teleconference sessions and manage accessibility of the potential users.

The user evaluation study conducted for HWU Telecare was limited. Therefore, this study involving the caregivers will be continue beyond this project to conduct on a larger scale with the same hypothesis proposed in this study. A larger group of participants and their inclusion can provide significant insights to the application's development, and allow this research work on elder care telepresence robotics to progress further.

Appendix A

RetraDev LocalCloud installation

(The source code of this section can be found at **https://github.com/tunchunairarko /retradev-localcloud**)

This appendix lists the steps needed to setup the RetraDev LocalCloud in a local PC for development. The process of converting the LocalCloud into a cloud framework using RetraDev Bastion Host is discussed in Appendix B. The steps mentioned here should work both in Windows 10 and Ubuntu Linux.

A.1 Pre-requisites

The following software components are required before installing the LocalCloud:

- Nodejs version 14.x or greater
- npm version 6.x or greater
- Python 3.7 or greater
- pip version 18.0 or higher
- MongoDB Community Server

A.1.1 Installing the dependencies

For windows

1. Download the latest Windows build of nodejs from **here** and install. It should install both nodejs and npm in the process.

2. Download the latest version of Python 3 from **here** and install. It should install both python3 and pip in the process. Make sure that the python path is added to the system's $PATH environment variable.
3. Download the MongoDB community server's latest version from **here** and install.
4. To add a GUI for the MongoDB installation for operating with the database visually, install MongoDB Compass from **here**. This is an optional requirement.

For Ubuntu

1. The nodejs and npm installation instructions can be found from the official Github repository (**here**).
2. The detailed instruction to install Python 3 and pip can be found (**here**).
3. The detailed instruction to install mongodb can be found from the official website (**here**).
4. To add a GUI for the MongoDB installation for operating with the database visually, install an Ubuntu Debian MongoDB Compass package from **here**. This is an optional requirement.

A.2 Installation

The installation process for the LocalCloud and its microservices is handled by pre-written automated scripts. Follow these steps to install the LocalCloud in your pc:

1. Open up a bash terminal (Ubuntu)/Powershell (Windows) in your desired folder for installation.
2. Clone the Github repository of the RetraDev LocalCloud:

```
 git clone https://github.com/tunchunairarko/retradev-localcloud.git

```

3. (For windows), run the following automated script to install all the Node modules, python libraries and Python flask server:

```
 localcloud_install.bat

```

4. (For linux), run the following automated script to install all the Node modules, python libraries and Python flask server:

```
    localcloud_install.sh

```

5. Install these additional npm module globally

```
    npm i nodemon pm2 -g

```

For Ubuntu, additional permission may be required to execute the script. Make sure the directory has write and execution permission for the current user using the *chmod* command. Details regarding this step can be found **here**.

A.3 Running the LocalCloud service

To run the LocalCloud service, two sets of environment variable files need to be configured, one for the NodeJs server and another for the Flask server. Templates for the environment files are provided in the Github repository as (.env.template) and (/teleoperation_mcs/.env.template). Copy the contents of these two files and paste them in two separate .env files at: */.env* and */teleoperation_mcs/.env*.

The NodeJs .env file should look something like this:

```
MONGODB_CONNECTION_STRING="mongodb://localhost:27017/{SET_DB_NAME_HERE}"
JWT_SECRET="XYZ" (PLACE YOUR JWT_SECRET HERE. You can generate a
    JWT_SECRET from https://passwordsgenerator.net/ and validate from here
     https://jwt.io/)
PYTHON_PATH=..."python.exe" (UPDATE YOUR PYTHON EXECUTABLE PATH HERE)
PORT=XXXX (Choose your desired port for running the LocalCloud central
    server. Default is 9000)
FLASK_API="http://localhost:YYYY" (Local URL for the teleoperation server
    )
```

The Flask /teleoperation_mcs/.env file should look something like this:

```
RETRADEV_SERVER_PORT="XXXX"
PORT="YYYY" (PORT FOR RUNNING THE TELEOPERATION MICROSERVICE. Default is
    12432)
HOST=127.0.0.1
```

Once both the .env files are configured, we are ready for starting and using RetraDev LocalCloud.

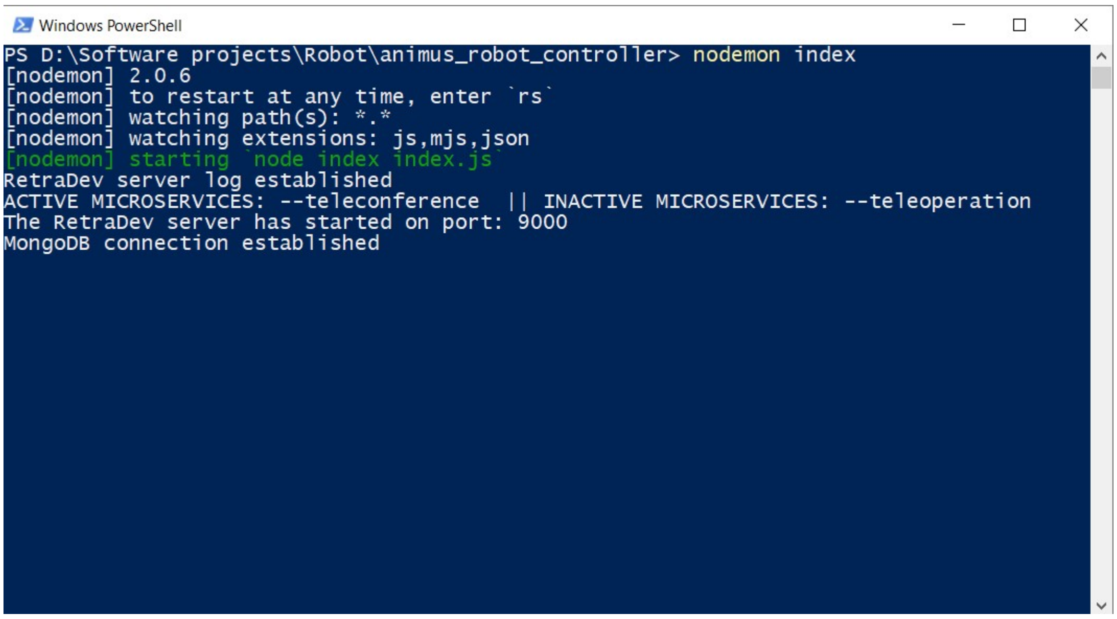
To start the RetraDev server using pm2 in the background, run:

```
pm2 start index.js --name="RetraDev server"
```

To start the RetraDev server using nodemon, run:

```
nodemon index.js
```

When running using Nodemon, the console should look like this (Figure A.1):



```
PS D:\Software projects\Robot\animus_robot_controller> nodemon index
[nodemon] 2.0.6
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
RetraDev server log established
ACTIVE MICROSERVICES: --teleconference  || INACTIVE MICROSERVICES: --teleoperation
The RetraDev server has started on port: 9000
MongoDB connection established
```

Figure A.1: RetraDev server started

To start the teleoperation server in Ubuntu, run:

```
cd teleoperation_mcs; python3.x teleop_server.py
```

To start the teleoperation server in Windows, run:

```
cd teleoperation_mcs; py teleop_server.py
```

To start the teleoperation server in using pm2, run:

```
cd teleoperation_mcs; pm2 start teleop_server.py --name="
    RetraDev_teleop_service" --interpreter=python3.x
```

Appendix B

Using the RetraDev Bastion Host

(The source code of this section can be found at **https://github.com/tunchunairarko /retradev-bastion-server-configuration**)

This appendix lists the steps needed to setup a RetraDev Bastion Host for configuring the full RetraDev framework.

B.1 Pre-requisites

To follow or go through with the RetraDev Bastion Host configuration and usage, the following things are required:

- **A FQDN (Fully Qualified Domain Name):** It can be any domain name from any vendors, as long as the DNS records are accessible.
- **Linux virtual private server with Ubuntu installed:** This can be acquired from any vendors. Some examples of Linux cloud VPS are: AWS EC2, Linode VMs, Digitalocean Droplets etc. The minimum required configuration are as follows: - RAM: 1GB - CPU: 1 core - Network In: 40GB - Network Out: 1000GB - Bandwidth: 1TB. The IPv4 address or the reverse DNS record is required for configuring the domains.
- **Firewall daemon:** Access to the server's firewall is required to open the required tunneling ports. The port 80 and 443 needs to be open to expose LocalCloud application and services to remote users and robot clients.

Creating DNS records for the RetraDev components Let's assume our domain name is mydomain.com Let's assume that the reverse DNS of your server is something like this:

rvdns.members.linode.com

We will need to create 4 subdomains from your FQDN corresponding to the various components of the RetraDev framework. To create the subdomains:

- Go to your domain management portal
- Add CNAME records as the following:
  1. Host: localcloudserver, Points to/target: rvdns.members.linode.com, TTL: 5 minutes
  2. Host: teleoperation, Points to/target: rvdns.members.linode.com, TTL: 5 minutes
  3. Host: teleconference, Points to/target: rvdns.members.linode.com, TTL: 5 minutes
  4. Host: relayserver, Points to/target: rvdns.members.linode.com, TTL: 5 minutes

Then we'll have the following domain names for each of our services:

- localcloudserverer.mydomain.com
- teleoperation.mydomain.com
- teleconference.mydomain.com
- relayserver.mydomain.com

We're going to use these 4 names across the rest of the tutorial

Once these resources are secured, we are ready to go with the next steps. Assuming that the cloud server can be accessible via SSH, the following Linux packages/software components need to be installed:

- NodeJs v14.x or higher
- npm v6.x or greater
- Nginx

- LibSSL
- Certbot
- ssh-keygen
- pm2

Instructions to set the dependencies:

The nodejs and npm installation instructions can be found from the official Github repository **here**.

To install the rest of the components, use the following code:

```
sudo apt-get update && sudo apt-get upgrade -y
sudo apt-get install ssh-keygen
sudo apt-get install nginx -y
sudo systemctl status nginx
sudo systemctl start nginx
sudo systemctl enable nginx

#if you skipped installing build-essential and libssl-dev during nodejs
    installation, install it now
sudo apt-get install build-essential libssl-dev

npm i pm2 -g
sudo snap install core; sudo snap refresh core
sudo snap install --classic certbot
sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

Once we are done with these installation, we are ready to start with the RetraDev Bastion Host configuration.

B.2 Installing and configuring the RetraDev Bastion Host

Clone the github repo containing the necessary configuration files for the Bastion Host:

```
cd /var/www
git clone https://github.com/tunchunairarko/retradev-bastion-server-
    configuration

cd retradev-bastion-server-configuration
```

Once we're inside the retradev-bastion-server-configuration folder, our first task will be to set the 4 Nginx configuration files. The Nginx configuration files would look like this:

```
server {
    listen 80;
    server_name test.mysite.com www.test.mysite.com;
    location / {
        proxy_pass http://127.0.0.1:9000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_redirect off;
     }
}
```

By default, the following ports have been set for the RetraDev Bastion host:

- LocalCloud server: 9000
- Teleoperation microservice: 12432
- Teleconference microservice: 11300
- Relay server: 9543

You can change these ports as you like from the nginx files. Now enter each of the 4 configuration files, and change each of the domain names like this:

- For the file with port 9000, *test.mysite.com* to *localcloudserver.mydomain.com*
- For the file with port 12432, *test.mysite.com* to *teleoperation.mydomain.com*
- For the file with port 11300, *test.mysite.com* to *teleconference.mydomain.com*
- For the file with port 9543, *test.mysite.com* to *relayserver.mydomain.com*

Once we have updated these 4 files, our next step would be to install these to the nginx configuration of the server and configure SSL/TLS for each of these subdomains. The cloned git folder already contains an automated script to take of the installation. Just run:

```
sudo bash installnginxfiles.sh
```

Then the necessary subdomains for RetraDev configuration is ready.

Installing the relay server

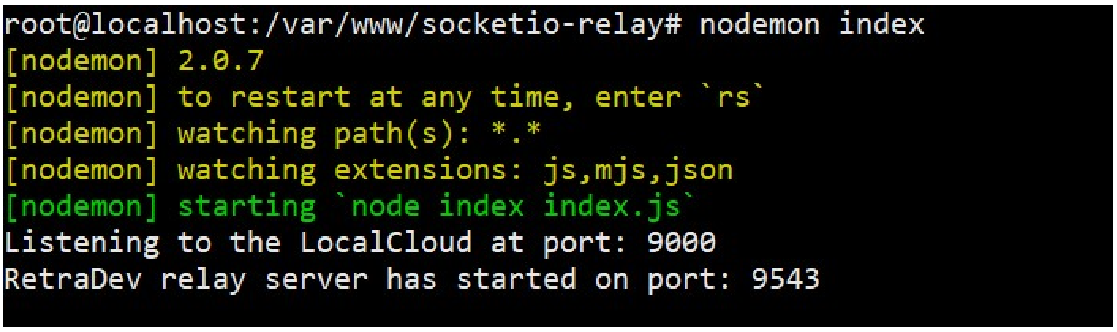
Assuming we are inside the clone git folder, run the following:

```
cd relayserver
npm i
npm i nodemon -g
```

Relay server will be installed. To test whether the relay server is working or not, type:

```
nodemon index
```

The console should look like this:



```
root@localhost:/var/www/socketio-relay# nodemon index
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index index.js`
Listening to the LocalCloud at port: 9000
RetraDev relay server has started on port: 9543
```

FIGURE B.1: Relay server started with Nodemon

If no error occured, we're ready to make the relay server go live. Close the nodemon using Ctrl+C, and type:

```
pm2 start index.js --name="RetraDev_Relay"
```

The relay server will be live. To check the status of the relay server, type:

```
pm2 status
```

B.3 Process of starting up Bastion Tunnels with the LocalCloud

Minimize the SSH terminal. Go to the installation directory of LocalCloud: /path/to/retradev-localcloud/servicerunners. You will find 2 automated scripts for starting up the Bastion tunnels for the RetraDev central server and the teleoperation microservice. We'll start with the LocalCloud server at first.

(For Ubuntu) Open the localcloudrunner.sh script, it will look like this:

```
#!/bin/bash
ssh -i "YOUR_KEY_PAIR_NAME.pem" -N -T -R 9000:localhost:9000 user@rvdns.
    members.linode.com
```

(For Windows) Open the localcloudrunner.sh script, it will look like this:

```
@ECHO OFF
ssh -i "YOUR_KEY_PAIR_NAME.pem" -N -T -R 9000:localhost:9000 user@rvdns.
    members.linode.com
pause
```

Change "YOUR_KEY_PAIR_NAME.pem" to your SSH key_pair file name (DO NOT USE PASSWORD-BASED AUTHENTICATION FOR TUNNELING, IT IS NOT SECURED). Then, change your server username and server reverser DNS/IP address. Once you are done, just execute the script in bash terminal, and your locally installed RetraDev LocalCloud server will go live. Follow the same steps to make the teleoperation microservice live using the *teleoperationrunner.sh* or *teleoperationrunner.bat* script.

Running the teleconference server The teleconference microservice does not run locally. For that, we need to install and run it from our cloud server. We need to follow the following steps:

- SSH to the Bastion Host
- Navigate to:

  ```
      cd /var/www

  ```

- Download the files for the teleconference microservice from **here**
- Install the teleconference server:

  ```
      cd teleconference
      npm i
      npm run build

  ```

- Test the installation by running:

  ```
      npm start

  ```

  Go to your web browser, and type teleconference.mydomain.com. The teleconference service should show up.

- Once we're ready deploy, enter:

  ```
      pm2 start index.js --name="RetraDev_Teleconference"

  ```

The teleconference server is already pre-programmed to communicate with the LocalCloud, so no additional configuration is required to establish its communication with the LocalCloud.

Appendix C

Installing and Serving HWU Telecare

(The source code of this section can be found at **https://github.com/tunchunairarko/hwu_telecare**)

This appendix shows how we can use the RetraDev framework to serve a telepresence robotic application proposed in this study called ”HWU Telecare”. The installation process will involve how to serve it and connect it with the LocalCloud microservices:

C.1 Download and building the application

- Navigate to your LocalCloud root directory: /path/to/LocalCloud
- Clone the github repo

```
git clone https://github.com/tunchunairarko/hwu_telecare

```

- Type the following commands to build the React project

```
cd hwu_telecare
npm i
npm run build

```

- Once they are completed, we need to update the LocalCloud server script. Go back to the LocalCloud root directory and open index.js
- Find the following line:

```
    const root = require('path').join(__dirname, 'client', 'build')
```

Change 'client' to 'hwu_telecare'

- Restart your LocalCloud server and Bastion Tunnel. The **HWU Telecare** application will go live and should be accessible from anywhere around the world.

Configuring the teleconference microservice for HWU Telecare

- Go to the following file: **/src/components/layout/Dashboard/dashboard.js**
- Find the following variable and update it using the teleconference microservice URL:

```
    const teleConferenceEmbedSource = "{PUT THE URL OF
   TELECONFERENCE WIDGET HERE}"
```

Appendix D

Questionnnaire

D.1 Perceived Enjoyment

1. I enjoyed interacting using the telecare robot

☐	☐	☐	☐	☐
Strongly Agree	Partially Agree	Neither	Partially Disagree	Strongly Disagree

2. I think that interacting with me via the telecare robot was enjoyable to the other person

☐	☐	☐	☐	☐
Strongly Agree	Partially Agree	Neither	Partially Disagree	Strongly Disagree

3. I would find HWU Telecare a useful and enjoyable tool to help me carrying out my care duties

☐	☐	☐	☐	☐	☐	☐
Extremely Likely	Quite Likely	Slight Likely	Neutral	Slightly Unlikely	Quite Unlikely	Extremely Unlikely

D.2 Perceived Sociability

1. It was possible to have a pleasant conversation and interaction

☐	☐	☐	☐	☐
Strongly Agree	Partially Agree	Neither	Partially Disagree	Strongly Disagree

2. It was easy for me to talk and move using the robot while having a conversation

☐	☐	☐	☐	☐
Strongly Agree	Partially Agree	Neither	Partially Disagree	Strongly Disagree

3. When interacting and chatting using the telepresence robot I felt like I was physically present in the room

☐	☐	☐	☐	☐
Strongly Agree	Partially Agree	Neither	Partially Disagree	Strongly Disagree

D.3 Social Influence

1. I think my organisation and colleagues or other carers will have a good impression of me if I use the telecare robot system

☐	☐	☐	☐	☐	☐	☐
Extremely Likely	Quite Likely	Slight Likely	Neutral	Slightly Unlikely	Quite Unlikely	Extremely Unlikely

2. I think it would give a good impression to our residents/patients if I use the telecare robot

☐	☐	☐	☐	☐
Strongly Agree	Partially Agree	Neither	Partially Disagree	Strongly Disagree

D.4 NASA-TLX

1. Were the tasks too demanding (mentally, physically, or both)?

☐	☐	☐	☐	☐	☐	☐
Very low	Quite Low	Slightly low	Neutral	Slightly high	High	Very high

2. How hurried or rushe was the pace of the tasks of stage 2 (registration,familiarizing with the interface, navigation testing, finding objects)?

☐	☐	☐	☐	☐	☐	☐
Very low	Quite Low	Slightly low	Neutral	Slightly high	High	Very high

3. How hurried or rushed was the pace of the task of stage 3 (connection testing, teleconference call, following the local user when he moved, taking pulse reading and notes)?

☐	☐	☐	☐	☐	☐	☐
Very low	Quite Low	Slightly low	Neutral	Slightly high	High	Very high

4. How would you rate your success you have been to complete the two stages?

☐	☐	☐	☐	☐	☐	☐
Very low	Quite Low	Slightly low	Neutral	Slightly high	High	Very high

5. How hard did you have to work to accomplish your level of performance?

☐	☐	☐	☐	☐	☐	☐
Very low	Quite Low	Slightly low	Neutral	Slightly high	High	Very high

6. I felt discouraged, insecured, irritated, stressed and/or annoyed during completing my tasks

☐	☐	☐	☐	☐	☐	☐
Very low	Quite Low	Slightly low	Neutral	Slightly high	High	Very high

D.5 Perceived Barrier

1. I am concerned about my privacy invasion while using this application

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

2. I am worried about the confidentiality, data storage mechanism and security of this system

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

3. The design of the system is too complex for care professionals

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

4. It will take a lot of time to completely learn this system

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

5. I am concerned about the quality of the remote connection to the robot

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

6. Do you have any additional concerns regarding our system?

D.6 Perceived Ease of Use

1. Learning to operate the system on a regular basis would be easy for me

☐	☐	☐	☐	☐	☐	☐
Extremely Likely	Quite Likely	Slight Likely	Neutral	Slightly Unlikely	Quite Unlikely	Extremely Unlikely

2. It would be easy for me to become skillful at using the system

☐	☐	☐	☐	☐	☐	☐
Extremely Likely	Quite Likely	Slight Likely	Neutral	Slightly Unlikely	Quite Unlikely	Extremely Unlikely

3. My interaction with HWU Telecare would be clear and understandable

☐	☐	☐	☐	☐	☐	☐
Extremely Likely	Quite Likely	Slight Likely	Neutral	Slightly Unlikely	Quite Unlikely	Extremely Unlikely

4. I would find HWU Telecare flexible to interact with

☐	☐	☐	☐	☐	☐	☐
Extremely Likely	Quite Likely	Slight Likely	Neutral	Slightly Unlikely	Quite Unlikely	Extremely Unlikely

5. I would find the HWU Telecare and the telecare robot system easy to use

☐	☐	☐	☐	☐	☐	☐
Extremely Likely	Quite Likely	Slight Likely	Neutral	Slightly Unlikely	Quite Unlikely	Extremely Unlikely

D.7 Perceived Usefulness

1. Using HWU Telecare would improve the quality of care

☐	☐	☐	☐	☐	☐	☐
Extremely Likely	Quite Likely	Slight Likely	Neutral	Slightly Unlikely	Quite Unlikely	Extremely Unlikely

2. Using HWU Telecare would increase caregiver's productivity

☐	☐	☐	☐	☐	☐	☐
Extremely Likely	Quite Likely	Slight Likely	Neutral	Slightly Unlikely	Quite Unlikely	Extremely Unlikely

3. Using HWU Telecare would enhance caregiver's effectiveness

☐	☐	☐	☐	☐	☐	☐
Extremely Likely	Quite Likely	Slight Likely	Neutral	Slightly Unlikely	Quite Unlikely	Extremely Unlikely

4. Using HWU Telecare would make the life of caregivers easier

☐	☐	☐	☐	☐	☐	☐
Extremely Likely	Quite Likely	Slight Likely	Neutral	Slightly Unlikely	Quite Unlikely	Extremely Unlikely

5. What aspect of the platform did you find the most useful?

D.8 Care Robotic Telepresence Questionnaire

1. Were you able to find the three objects (coffee machine, TV and the couch) in the apartment?
   - ☐ Coffee Maker
   - ☐ Couch
   - ☐ TV
2. Were you able to mute/unmute and hide/unhide your camera during the call?
   - ○ Yes
   - ○ No
3. In the case of call/connect drop, were you able to reconnect with the robot using the refresh button?
   - ○ Yes
   - ○ No
   - ○ No connection drop occurred

1. It was easy to create account during registration

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

2. I was able to follow the on-screen instructions clearly to complete my tasks

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

3. I was able to connect to and disconnect from the robot without any issue

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

4. I was able to navigate the robot using the onscreen controls

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

5. I was able to take a pulse measurement during the call easily using the provided button

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

6. I was able to take notes and set guidelines to the patient during the call

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

7. The video quality was clear enough

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

8. It was easy to use the different tools on the dashboard

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

9. There was too much latency during the telepresence connection

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

10. Some buttons appeared too big on the screen

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

11. There were audio noises and distortions during the call

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

12. I was able to hear things clearly

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

13. There was too much latency while I was driving the robot

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

14. It was difficult to drive the robot to turn and move left or right

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

15. It was difficult to control the head of the robot

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

1. What was the most interesting feature of our robot telecare system in your opinion?

2. Did you face any major issues during you session?

D.9 PS-SUQ

D.9.1 Information Quality (INFQ)

1. The application gave error messages that clearly told me how to fix problems

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

2. Whenever I made a mistake using HWU Telecare, I could recover easily and quickly

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

3. The information (such as introduction, onscreen messages, and participant information) provided with this study was clear

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

4. It was easy to find the information I needed

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

5. The information was effective in helping me complete the tasks

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

6. The organization of information on the application was clear

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

D.9.2 Expectations (E)

1. It was simple to use HWU Telecare

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

2. I was able to complete the tasks quickly and effectively

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

D.9.3 Intention To Use (ITU)

1. I felt comfortable while using this application

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

2. I believe I could become productive quickly if I use HWU Telecare

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

D.9.4 Satisfaction (S)

1. I liked the user interface, and it was pleasant to use

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

2. Overall, I am satisfied with this system

☐	☐	☐	☐	☐	☐	☐
Strongly Agree	Agree	Partially Agree	Neutral	Partially Disagree	Disagree	Strongly Disagree

D.9.5 Open-ended questions

1. Do you want to share any additional feedbacks regarding our platform so that we can improve further?

2. How would you use our robot telecare system in your daily life / job situations? Can you provide specific examples?

References

[1] N. G. Hockstein, C. Gourin, R. Faust, and D. J. Terris, "A history of robots: from science fiction to surgical robotics," *Journal of robotic surgery*, vol. 1, no. 2, pp. 113–118, 2007.

[2] U. DESA, "World Population Ageing 2020 Highlights | Population Division," Feb 2021, [Online; accessed 12. Feb. 2021]. [Online]. Available: https://www.un.org/development/desa/pd/news/world-population-ageing-2020-highlights

[3] N. Davis, "Falling total fertility rate should be welcomed, population expert says," *The Guardian*, vol. 26, 2018.

[4] S. Koceski and N. Koceska, "Evaluation of an assistive telepresence robot for elderly healthcare," *Journal of medical systems*, vol. 40, no. 5, p. 121, 2016.

[5] C. A. Holveck and J. Y. Wick, "Addressing the shortage of geriatric specialists," *The Consultant Pharmacist®*, vol. 33, no. 3, pp. 130–138, 2018.

[6] J. Abdi, A. Al-Hindawi, T. Ng, and M. P. Vizcaychipi, "Scoping review on the use of socially assistive robot technology in elderly care," *BMJ open*, vol. 8, no. 2, p. e018815, 2018.

[7] M. Kyrarini, F. Lygerakis, A. Rajavenkatanarayanan, C. Sevastopoulos, H. R. Nambiappan, K. K. Chaitanya, A. R. Babu, J. Mathew, and F. Makedon, "A survey of robots in healthcare," *Technologies*, vol. 9, no. 1, p. 8, 2021.

[8] T. B. Tuli, T. O. Terefe, and M. M. U. Rashid, "Telepresence mobile robots design and control for social interaction," *International Journal of Social Robotics*, pp. 1–10, 2020.

[9] A. Cesta, G. Cortellessa, A. Orlandini, and L. Tiberio, "Long-term evaluation of a telepresence robot for the elderly: methodology and ecological case study," *International Journal of Social Robotics*, vol. 8, no. 3, pp. 421–441, 2016.

[10] A. Kiselev, A. Kristoffersson, F. Melendez, C. Galindo, A. Loutfi, J. Gonzalez-Jimenez, and S. Coradeschi, "Evaluation of using semi-autonomy features in mobile robotic telepresence systems," in *2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. IEEE, 2015, pp. 147–152.

[11] S. Khan and C. Germak, "Reframing hri design opportunities for social robots: Lessons learnt from a service robotics case study approach using ux for hri," *Future internet*, vol. 10, no. 10, p. 101, 2018.

[12] K. Jovanovic, A. Schwier, E. Matheson, M. Xiloyannis, E. Rozeboom, N. Hochhausen, B. Vermeulen, B. Graf, P. Wolf, Z. Nawrat *et al.*, "Digital innovation hubs in health-care robotics fighting covid-19: Novel support for patients and health-care workers across europe," *IEEE Robotics & Automation Magazine*,

vol. 28, no. 1, pp. 40–47, 2021.

[13] A. M. Panchea, D. Létourneau, S. Brière, M. Hamel, M.-A. Maheux, C. Godin, M. Tousignant, M. Labbé, F. Ferland, F. Grondin *et al.*, "Opentera: A microservice architecture solution for rapid prototyping of robotic solutions to covid-19 challenges in care facilities," *arXiv preprint arXiv:2103.06171*, 2021.

[14] F. Melendez-Fernandez, C. Galindo, and J. Gonzalez-Jimenez, "A web-based solution for robotic telepresence," *International Journal of Advanced Robotic Systems*, vol. 14, no. 6, p. 1729881417743738, 2017.

[15] M. J. Johnson, M. A. Johnson, J. S. Sefcik, P. Z. Cacchione, C. Mucchiani, T. Lau, and M. Yim, "Task and design requirements for an affordable mobile service robot for elder care in an all-inclusive care for elders assisted-living setting," *International Journal of Social Robotics*, vol. 12, no. 5, pp. 989–1008, 2020.

[16] N. Charness and T. S. Jastrzembski, *Gerontechnology*, 2009, p. 1–29.

[17] A. Kristoffersson, S. Coradeschi, and A. Loutfi, "A review of mobile robotic telepresence," *Advances in Human-Computer Interaction*, vol. 2013, 2013.

[18] Z. Li, *Development and Implementation of Behaviours for a Socially Assistive Robot for the Elderly.* University of Toronto (Canada), 2015.

[19] S. Laniel, D. Létourneau, F. Grondin, M. Labbé, F. Ferland, and F. Michaud, "Toward enhancing the autonomy of a telepresence mobile robot for remote home care assistance," *Paladyn, Journal of Behavioral Robotics*, vol. 12, no. 1, pp. 214–237, 2021.

[20] A. Ghiță, A. F. Gavril, M. Nan, B. Hoteit, I. A. Awada, A. Sorici, I. G. Mocanu, and A. M. Florea, "The amiro social robotics framework: Deployment and evaluation on the pepper robot," *Sensors*, vol. 20, no. 24, p. 7271, 2020.

[21] D. Moher, L. Shamseer, M. Clarke, D. Ghersi, A. Liberati, M. Petticrew, P. Shekelle, and L. A. Stewart, "Preferred reporting items for systematic review and meta-analysis protocols (prisma-p) 2015 statement," *Systematic Reviews*, vol. 4, no. 1, p. 1, 2015. [Online]. Available: https://dx.doi.org/10.1186/2046-4053-4-1

[22] A.-W. Harzing, *The Publish or Perish tutorial: 80 easy tips to get the best out of the Publish or Perish software.* Tarma Software Research, 2016.

[23] M. K. Simon, L. W. Dietz, T. Diez, and O. Kopp, "Analyzing the importance of jabref features from the user perspective." in *ZEUS*, 2019, pp. 47–54.

[24] J. Rathbone, M. Carter, T. Hoffmann, and P. Glasziou, "Better duplicate detection for systematic reviewers: evaluation of systematic review assistant-deduplication module," *Systematic reviews*, vol. 4, no. 1, pp. 1–6, 2015.

[25] H. B. J. G. J. L. Fozard, Jan Rietsema, "Gerontechnology: Creating enabling environments for the challenges and opportunities of aging," *Educational Gerontology*, vol. 26, no. 4, pp. 331–344, 2000.

[26] J. A. Graafmans, V. Taipale, and N. Charness, *Gerontechnology: a sustainable investment in the future.* IOS press, 1998, vol. 48.

[27] "ABOUT US - AAL Programme," Oct 2020, [Online; accessed 17. Apr. 2021]. [Online]. Available: http://www.aal-europe.eu/about

[28] P. Rashidi and A. Mihailidis, "A survey on ambient-assisted living tools for older adults," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 3, pp. 579–590, 2013.

[29] S. Blackman, C. Matlo, C. Bobrovitskiy, A. Waldoch, M. L. Fang, P. Jackson, A. Mihailidis, L. Nygård, A. Astell, and A. Sixsmith, "Ambient assisted living technologies for aging well: a scoping review," *Journal of Intelligent Systems*, vol. 25, no. 1, pp. 55–69, 2016.

[30] D. Monekosso, F. Florez-Revuelta, and P. Remagnino, "Ambient assisted living [guest editors' introduction]," *IEEE Intelligent Systems*, vol. 30, no. 4, pp. 2–6, 2015.

[31] D. Singh, J. Kropf, S. Hanke, and A. Holzinger, "Ambient assisted living technologies from the perspectives of older people and professionals," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction.* Springer, 2017, pp. 255–266.

[32] A. Ylisaukko-oja, E. Vildjiounaite, and J. Mantyjarvi, "Five-point acceleration sensing wireless body area network - design and practical experiences," in *Eighth International Symposium on Wearable Computers*, vol. 1, 2004, pp. 184–185.

[33] K. Chaudhary and D. Sharma, "Body area networks: A survey," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, pp. 3319–3323.

[34] L. Parra, S. Sendra, J. M. Jiménez, and J. Lloret, "Multimedia sensors embedded in smartphones for ambient assisted living and e-health," *Multimedia Tools and Applications*, vol. 75, no. 21, pp. 13 271–13 297, 2016.

[35] I. Bisio, F. Lavagetto, M. Marchese, and A. Sciarrone, "Smartphone-centric ambient assisted living platform for patients suffering from co-morbidities monitoring," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 34–41, 2015.

[36] B. Graf, C. Parlitz, and M. Hägele, "Robotic home assistant care-o-bot® 3 product vision and innovation platform," in *Human-Computer Interaction. Novel Interaction Methods and Techniques*, J. A. Jacko, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 312–320.

[37] C.-A. Smarr, C. B. Fausset, and W. A. Rogers, "Understanding the potential for robot assistance for older adults in the home environment," Georgia Institute of Technology, Tech. Rep., 2011.

[38] J. van Heek, M. Ziefle, and S. Himmel, "Caregivers' perspectives on ambient assisted living technologies in professional care contexts." in *ICT4AWE*, 2018, pp.

37–48.

[39] E.-M. Schomakers and M. Ziefle, “Privacy perceptions in ambient assisted living.” in *ICT4AWE*, 2019, pp. 205–212.

[40] C. Jaschinski and S. B. Allouch, “Listening to the ones who care: exploring the perceptions of informal caregivers towards ambient assisted living applications,” *Journal of ambient intelligence and humanized computing*, vol. 10, no. 2, pp. 761–778, 2019.

[41] A. Costa, P. Novais, and R. Simoes, “A caregiver support platform within the scope of an ambient assisted living ecosystem,” *Sensors*, vol. 14, no. 3, pp. 5654–5676, 2014.

[42] R.-M. Johansson-Pajala, K. Thommes, J. A. Hoppe, O. Tuisku, L. Hennala, S. Pekkarinen, H. Melkas, and C. Gustafsson, “Care Robot Orientation: What, Who and How? Potential Users’ Perceptions,” *Int. J. Social Rob.*, vol. 12, no. 5, pp. 1103–1117, Nov 2020.

[43] C. Harrefors, S. Sävenstedt, and K. Axelsson, “Elderly people’s perceptions of how they want to be cared for: an interview study with healthy elderly couples in northern sweden,” *Scandinavian journal of caring sciences*, vol. 23, no. 2, pp. 353–360, 2009.

[44] B. Hofmann, “Ethical challenges with welfare technology: a review of the literature,” *Science and engineering ethics*, vol. 19, no. 2, pp. 389–406, 2013.

[45] M. Goeldner, C. Herstatt, and F. Tietze, “The emergence of care robotics—a patent and publication analysis,” *Technological Forecasting and Social Change*, vol. 92, pp. 115–131, 2015.

[46] R. Bemelmans, G. J. Gelderblom, P. Jonker, and L. De Witte, “Socially assistive robots in elderly care: A systematic review into effects and effectiveness,” *Journal of the American Medical Directors Association*, vol. 13, no. 2, pp. 114–120, 2012.

[47] T. Jacobs and G. S. Virk, “Iso 13482-the new safety standard for personal care robots,” in *ISR/Robotik 2014; 41st International Symposium on Robotics*. VDE, 2014, pp. 1–6.

[48] P. Fiorini, K. Ali, and H. Seraji, “Health care robotics: A progress report,” in *Proceedings of International Conference on Robotics and Automation*, vol. 2. IEEE, 1997, pp. 1271–1276.

[49] P. Khosravi and A. H. Ghapanchi, “Investigating the effectiveness of technologies applied to assist seniors: A systematic literature review,” *International journal of medical informatics*, vol. 85, no. 1, pp. 17–26, 2016.

[50] T. Vandemeulebroucke, B. Dierckx De Casterlé, and C. Gastmans, “The use of care robots in aged care: A systematic review of argument-based ethics literature,” *Archives of Gerontology and Geriatrics*, vol. 74, p. 15–25, 2018.

[51] K. Fischer, J. Seibt, R. Rodogno, M. K. Rasmussen, A. Weiss, L. Bodenhagen, W. K. Juel, and N. Krüger, "Integrative social robotics hands-on," *Interaction Studies*, vol. 21, no. 1, pp. 145–185, 2020.

[52] García-Soler, D. Facal, U. Díaz-Orueta, L. Pigini, L. Blasi, and R. Qiu, "Inclusion of service robots in the daily lives of frail older users: A step-by-step definition procedure on users' requirements," *Archives of Gerontology and Geriatrics*, vol. 74, p. 191–196, 2018. [Online]. Available: https://dx.doi.org/10.1016/j.archger.2017.10.024

[53] Z. Z. Bien and D. Stefanov, *Advances in rehabilitation robotics: Human-friendly technologies on movement assistance and restoration for people with disabilities.* Springer Science & Business, 2004, vol. 306.

[54] K. Caine, S. Šabanovic, and M. Carter, "The effect of monitoring by cameras and robots on the privacy enhancing behaviors of older adults," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, 2012, pp. 343–350.

[55] P. Lin, K. Abney, and G. A. Bekey, *Robot ethics: the ethical and social implications of robotics.* Intelligent Robotics and Autonomous Agents series, 2012.

[56] J. Parviainen and J. Pirhonen, "Vulnerable bodies in human–robot interactions: embodiment as ethical issue in robot care for the elderly," 2017.

[57] A. Sharkey and N. Sharkey, "Granny and the robots: ethical issues in robot care for the elderly," *Ethics and information technology*, vol. 14, no. 1, pp. 27–40, 2012.

[58] J. Hudson, M. Orviska, and J. Hunady, "People's attitudes to robots in caring for the elderly," *International Journal of Social Robotics*, vol. 9, no. 2, pp. 199–210, 2017.

[59] E. R. Nilsen, J. Dugstad, H. Eide, M. K. Gullslett, and T. Eide, "Exploring resistance to implementation of welfare technology in municipal healthcare services–a longitudinal case study," *BMC health services research*, vol. 16, no. 1, pp. 1–14, 2016.

[60] T. B. Sheridan, "Teleoperation, telerobotics and telepresence: A progress report," *Control Engineering Practice*, vol. 3, no. 2, pp. 205–214, 1995.

[61] M. Minsky, "Telepresence," 1980.

[62] M. R. Sarder, F. Ahmed, and B. A. Shakhar, "Design and implementation of a lightweight telepresence robot for medical assistance," in *2017 International Conference on Electrical, Computer and Communication Engineering (ECCE).* IEEE, 2017, pp. 779–783.

[63] A. Chibani, Y. Amirat, S. Mohammed, E. Matson, N. Hagita, and M. Barreto, "Ubiquitous robotics: Recent challenges and future trends," *Robotics and Autonomous Systems*, vol. 61, no. 11, pp. 1162–1172, 2013.

[64] T. Tsai, Y.-L. Hsu, A.-I. Ma, T. King, and C. Wu, "Developing a telepresence robot for interpersonal communication with the elderly in a home environment." *Telemedicine journal and e-health : the official journal of the American Telemedicine Association*, vol. 13 4, pp. 407–24, 2007.

[65] I. Rae, L. Takayama, and B. Mutlu, "The influence of height in robot-mediated communication," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2013, pp. 1–8.

[66] M. Niemelä, L. Van Aerschot, A. Tammela, I. Aaltonen, and H. Lammi, "Towards ethical guidelines of using telepresence robots in residential care," *International Journal of Social Robotics*, pp. 1–9, 2019.

[67] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Transactions on automation science and engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[68] J. Wan, S. Tang, H. Yan, D. Li, S. Wang, and A. V. Vasilakos, "Cloud robotics: Current status and open issues," *IEEE Access*, vol. 4, pp. 2797–2807, 2016.

[69] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: architecture, challenges and applications," *IEEE network*, vol. 26, no. 3, pp. 21–28, 2012.

[70] M. Waibel, M. Beetz, J. Civera, R. d'Andrea, J. Elfring, D. Galvez-Lopez, K. Häussermann, R. Janssen, J. Montiel, A. Perzylo *et al.*, "Roboearth," *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[71] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel, "Rapyuta: A cloud robotics platform," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 481–493, 2014.

[72] G. Toffetti and T. M. Bohnert, "Cloud robotics with ros," in *Robot operating system (ROS)*. Springer, 2020, pp. 119–146.

[73] W. Chen, Y. Yaguchi, K. Naruse, Y. Watanobe, K. Nakamura, and J. Ogawa, "A study of robotic cooperation in cloud robotics: Architecture and challenges," *IEEE Access*, vol. 6, pp. 36 662–36 682, 2018.

[74] "Robotics Roadmap for US Robotics: From Internet to Robotics, 2020 Edition CCC Blog," Apr 2021, [Online; accessed 24. Apr. 2021]. [Online]. Available: https://cccblog.org/2020/09/09/robotics-roadmap-for-us-robotics-from-internet-to-robotics-2020-edition

[75] J. Mišeikis, P. Caroni, P. Duchamp, A. Gasser, R. Marko, N. Mišeikienė, F. Zwilling, C. de Castelbajac, L. Eicher, M. Früh, and H. Früh, "Lio-a personal robot assistant for human-robot interaction and care applications," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5339–5346, 2020.

[76] "Home | Lhf Connect," Apr 2021, [Online; accessed 25. Apr. 2021]. [Online]. Available: https://en.lhfconnect.net

[77] M. R. Fossati, M. G. Catalano, M. Carbone, G. Lentini, D. Caporale, G. Grioli, M. Poggiani, M. Maimeri, M. Barbarossa, C. Petrocelli *et al.*, "Lhf connect: a diy telepresence robot against covid-19," *Strategic Design Research Journal*, vol. 13, no. 3, pp. 418–431, 2020.

[78] "GoBe Robots Launches New Telepresence Robot to Reduce CO2 Emissions by Thousands of Tons," Jul 2020, [Online; accessed 25. Apr. 2021]. [Online]. Available: https://www.businesswire.com/news/home/20200701005119/en/GoBe-Robots-Launches-New-Telepresence-Robot-to-Reduce-CO2-Emissions-by-Thousands-of-T

[79] G. Cortellessa, F. Fracasso, A. Sorrentino, A. Orlandini, G. Bernardi, L. Coraci, R. De Benedictis, and A. Cesta, "Robin, a telepresence robot to support older users monitoring and social inclusion: Development and evaluation," *Telemedicine and e-Health*, vol. 24, no. 2, p. 145–154, Feb 2018.

[80] B. Jones, Y. Zhang, P. N. Wong, and S. Rintel, "Vroom: Virtual robot overlay for online meetings," in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–10.

[81] K. Abe, M. Shiomi, Y. Pei, T. Zhang, N. Ikeda, and T. Nagai, "ChiCaRo: telepresence robot for interacting with babies and toddlers," *Adv. Rob.*, vol. 32, no. 4, pp. 176–190, Feb 2018.

[82] J. Ruiz-del Solar, M. Salazar, V. Vargas-Araya, U. Campodonico, N. Marticorena, G. Pais, R. Salas, P. Alfessi, V. C. Rojas, and J. Urrutia, "Mental and emotional health care for covid-19 patients: Employing pudu, a telepresence robot," *IEEE Robotics & Automation Magazine*, vol. 28, no. 1, pp. 82–89, 2021.

[83] M. Lombard and K. Xu, "Social Responses to Media Technologies in the 21st Century: The Media are Social Actors Paradigm," *STARS*, vol. 2, no. 1, p. 2, 2021.

[84] A. Lotfi, C. Langensiepen, and S. W. Yahaya, "Socially assistive robotics: Robot exercise trainer for older adults," *Technologies*, vol. 6, no. 1, p. 32, 2018.

[85] G. Zhang, J. P. Hansen, and K. Minakata, "Hand-and gaze-control of telepresence robots," in *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, 2019, pp. 1–8.

[86] "Pepper Robot Features - Why Pepper?" Feb 2017, [Online; accessed 29. Apr. 2021]. [Online]. Available: https://www.gwsrobotics.com/why-pepper-robot

[87] N. T. Fitter, Y. Joung, M. Demeter, Z. Hu, and M. J. Matarić, "Design and evaluation of expressive turn-taking hardware for a telepresence robot," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2019, pp. 1–8.

[88] "Telepresence Robots | Remote Presence Robots , Virtual Presence Robots, Telebots," Apr 2021, [Online; accessed 26. Apr. 2021]. [Online]. Available: https://telepresencerobots.com

[89] "Padbot Telepresence Robot | Best Virtual Remote Presence | Mobile Robots," Jul 2020, [Online; accessed 26. Apr. 2021]. [Online]. Available: https://www.padbot.com/padbotp2

[90] J. González-Jiménez, C. Galindo, and J. Ruiz-Sarmiento, "Technical improvements of the giraff telepresence robot based on users' evaluation," in *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, 2012, pp. 827–832.

[91] V. Ahumada-Newhart and J. S. Olson, "Going to school on a robot: Robot and user interface design features that matter," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 26, no. 4, pp. 1–28, 2019.

[92] T. Tanioka, "Nursing and rehabilitative care of the elderly using humanoid robots," *The Journal of Medical Investigation*, vol. 66, no. 1.2, pp. 19–23, 2019.

[93] M. Miyagawa, Y. Kai, Y. Yasuhara, H. Ito, F. Betriana, T. Tanioka, R. Locsin *et al.*, "Consideration of safety management when using pepper, a humanoid robot for care of older adults," *Intelligent Control and Automation*, vol. 11, no. 01, p. 15, 2019.

[94] A. Gardecki and M. Podpora, "Experience from the operation of the pepper humanoid robots," in *2017 Progress in Applied Electrical Engineering (PAEE)*, 2017, pp. 1–6.

[95] C. &. F. Mark W. Westlake, "Ohmni Robot," *Gearbrain*, Nov 2020. [Online]. Available: https://www.gearbrain.com/ohmni-labs-robot-home-review-2582464625.html

[96] J. R. Wilson, "Designing a socially assistive robot to preserve the dignity of older adults," Ph.D. dissertation, Tufts University, 2017.

[97] T. L. Mitzner, T. L. Chen, C. C. Kemp, and W. A. Rogers, "Identifying the potential for robotics to assist older adults in different living environments," *International journal of social robotics*, vol. 6, no. 2, pp. 213–227, 2014.

[98] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, "Rosbridge: Ros for non-ros users," in *Robotics Research.* Springer, 2017, pp. 493–504.

[99] G. Wilson, C. Pereyda, N. Raghunath, G. de la Cruz, S. Goel, S. Nesaei, B. Minor, M. Schmitter-Edgecombe, M. E. Taylor, and D. J. Cook, "Robot-enabled support of daily activities in smart home environments," *Cognitive Systems Research*, vol. 54, pp. 258–272, 2019.

[100] N. Pavón-Pulido, J. A. López-Riquelme, and J. J. Feliú-Batlle, "Iot architecture for smart control of an exoskeleton robot in rehabilitation by using a natural user interface based on gestures," *Journal of Medical Systems*, vol. 44, no. 9, pp. 1–10, 2020.

[101] N. Chivarov, D. Chikurtev, S. Chivarov, M. Pleva, S. Ondas, J. Juhar, and K. Yovchev, "Case study on human-robot interaction of the remote-controlled

service robot for elderly and disabled care." *Comput. Informatics*, vol. 38, no. 5, pp. 1210–1236, 2019.

[102] P. Hintjens, *ZeroMQ: messaging for many applications.* " O'Reilly Media, Inc.", 2013.

[103] "23/ZMTP," Jul 2021, [Online; accessed 20. Jul. 2021]. [Online]. Available: https://rfc.zeromq.org/spec/23

[104] E. Coronado and G. Venture, "Towards iot-aided human–robot interaction using nep and ros: A platform-independent, accessible and distributed approach," *Sensors*, vol. 20, no. 5, p. 1500, 2020.

[105] L. Danter, S. Planthaber, A. Dettmann, W. Brinkmann, and F. Kirchner, "Lightweight and framework-independent communication li-brary to support cross-platform robotic applications and high-latency connections," in *Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2020.

[106] R. Rai, *Socket. IO Real-time Web Application Development.* Packt Publishing Ltd, 2013.

[107] socketio, "socket.io," Jul 2021, [Online; accessed 12. Jul. 2021]. [Online]. Available: https://github.com/socketio/socket.io

[108] R. Suenaga and K. Morioka, "Rowma: A reconfigurable robot network construction system," in *2021 IEEE/SICE International Symposium on System Integration (SII).* IEEE, 2021, pp. 537–542.

[109] Y. Jin, Y. Deng, J. Gong, X. Wan, G. Gao, and Q. Wang, "Oyaya: A desktop robot enabling multimodal interaction with emotions," in *26th International Conference on Intelligent User Interfaces*, 2021, pp. 55–57.

[110] "Cyberselves Animus," Jul 2021, [Online; accessed 13. Jul. 2021]. [Online]. Available: https://www.cyberselves.com/animus

[111] W. Huang, Y. Qi, W. Zhang, L. Pang, and P. Fan, "Remote data transmission of intelligent cloud robot based on google protobuf," in *Journal of Physics: Conference Series*, vol. 1721, no. 1. IOP Publishing, 2021, p. 012034.

[112] S. E. Reppou, E. G. Tsardoulias, A. M. Kintsakis, A. L. Symeonidis, P. A. Mitkas, F. E. Psomopoulos, G. T. Karagiannis, C. Zielinski, V. Prunet, J.-P. Merlet, M. Iturburu, and A. Gkiokas, "Rapp: A robotic-oriented ecosystem for delivering smart user empowering applications for older people," *International Journal of Social Robotics*, vol. 8, no. 4, pp. 539–552, 08 2016. [Online]. Available: https://www.proquest.com/scholarly-journals/rapp-robotic-oriented-ecosystem-delivering-smart/docview/1799002811/se-2?accountid=16064

[113] "RAPP-Empowering robotics for social inclusion," Jan 2017, [Online; accessed 13. Jul. 2021]. [Online]. Available: https://rapp-project.github.io

[114] R. Toris, J. Kammerl, D. V. Lu, J. Lee, O. C. Jenkins, S. Osentoski, M. Wills, and S. Chernova, "Robot web tools: Efficient messaging for cloud robotics," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS).* IEEE, 2015, pp. 4530–4537.

[115] E. S. Short, D. Short, Y. Fu, and M. Matarić, "Sprite: Stewart platform robot for interactive tabletop engagement," *arXiv preprint arXiv:2011.05786*, 2020.

[116] M. J. Sobrepera, V. G. Lee, and M. J. Johnson, "The design of lil'flo, a socially assistive robot for upper extremity motor assessment and rehabilitation in the community via telepresence," *Journal of Rehabilitation and Assistive Technologies Engineering*, vol. 8, p. 20556683211001805, 2021.

[117] P. M. Grice and C. C. Kemp, "Assistive mobile manipulation: Designing for operators with motor impairments," in *RSS 2016 Workshop on Socially and Physically Assistive Robotics for Humanity*, 2016.

[118] V. Dawarka and G. Bekaroo, "Cloud robotics platforms: review and comparative analysis," in *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC).* IEEE, 2018, pp. 1–6.

[119] Y. Liu and Y. Xu, "Summary of cloud robot research," in *2019 25th International Conference on Automation and Computing (ICAC).* IEEE, 2019, pp. 1–5.

[120] "The Construct: A Platform to Learn ROS-based Advanced Robotics Online," Jun 2021, [Online; accessed 13. Jul. 2021]. [Online]. Available: https://www.theconstructsim.com

[121] F. Bazzano, F. Lamberti, A. Sanna, G. Paravati, and M. Gaspardone, "Comparing usability of user interfaces for robotic telepresence," in *International Conference on Human Computer Interaction Theory and Applications*, vol. 3. SCITEPRESS, 2017, pp. 46–54.

[122] N. D. X. Hai, L. H. T. Nam, and N. T. Thinh, "Remote healthcare for the elderly, patients by tele-presence robot," in *2019 International Conference on System Science and Engineering (ICSSE).* IEEE, 2019, pp. 506–510.

[123] M. Desai, K. M. Tsui, H. A. Yanco, and C. Uhlik, "Essential features of telepresence robots," in *2011 IEEE Conference on Technologies for Practical Robot Applications.* IEEE, 2011, pp. 15–20.

[124] P. Björnfot, "Evaluating input devices for robotic telepresence," in *European Conference on Cognitive Ergonomics 2021*, 2021, pp. 1–8.

[125] V. C. Pham, Y. Lim, H.-D. Bui, Y. Tan, N. Y. Chong, and A. Sgorbissa, "An experimental study on culturally competent robot for smart home environment," in *International Conference on Advanced Information Networking and Applications.* Springer, Cham, 2020, pp. 369–380.

[126] "What is Bastion Host Server - Learning Journal," Jul 2021, [Online; accessed 13. Jul. 2021]. [Online]. Available: https://www.learningjournal.guru/article/

public-cloud-infrastructure/what-is-bastion-host-server

[127] G. Ercolano, P. D. Lambiase, E. Leone, L. Raggioli, D. Trepiccione, and S. Rossil, "Socially assistive robot's behaviors using microservices," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2019, pp. 1–6.

[128] A. Kapitonov, S. Lonshakov, V. Bulatov, B. K. Montazam, and J. White, "Robot-as-a-service: From cloud to peering technologies," *Frontiers in Robotics and AI*, vol. 8, p. 95, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2021.560829

[129] "How an SSH tunnel can bypass firewalls, add encryption to application protocols, and help access services remotely." Jul 2021, [Online; accessed 14. Jul. 2021]. [Online]. Available: https://www.ssh.com/academy/ssh/tunneling

[130] "Top 4 BEST Ngrok Alternatives In 2021: Review And Comparison," Jul 2021, [Online; accessed 20. Jul. 2021]. [Online]. Available: https://www.softwaretestinghelp.com/ngrok-alternatives

[131] inconshreveable, "ngrok - secure introspectable tunnels to localhost," Jul 2021, [Online; accessed 14. Jul. 2021]. [Online]. Available: https://ngrok.com/product

[132] E. T.-H. Chu, K.-H. Lin, S.-Y. Chen, J. Hsu, and H.-M. Wu, "Sbot: A social media based object tracking system," in *2020 Indo–Taiwan 2nd International Conference on Computing, Analytics and Networks (Indo-Taiwan ICAN)*. IEEE, 2020, pp. 132–137.

[133] C. Deuerlein, M. Langer, J. Seßner, P. Heß, and J. Franke, "Human-robot-interaction using cloud-based speech recognition systems," *Procedia CIRP*, vol. 97, pp. 130–135, 2021.

[134] "Using ngrok with ROS," Apr 2021, [Online; accessed 14. Jul. 2021]. [Online]. Available: https://ibrahimessam.com/blog/2020-12-25-ngrok-ros-bridge

[135] R. Shtylman, "Localtunnel ˜ Expose yourself to the world," Feb 2018, [Online; accessed 14. Jul. 2021]. [Online]. Available: http://localtunnel.github.io/www

[136] Archiveddocs, "Designing Your Cloud Infrastructure," Jul 2021, [Online; accessed 20. Jul. 2021]. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/hh831630(v=ws.11)

[137] osrf, "rvizweb," Jul 2021, [Online; accessed 18. Jul. 2021]. [Online]. Available: https://github.com/osrf/rvizweb

[138] F. Lier and S. Wachsmuth, "Towards an open simulation environment for the pepper robot," in *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, 2018, pp. 175–176.

[139] M. Labbé and F. Michaud, "Memory management for real-time appearance-based loop closure detection," in *2011 IEEE/RSJ international conference on intelligent*

robots and systems. IEEE, 2011, pp. 1271–1276.

[140] "RTAB-Map," Jun 2021, [Online; accessed 20. Jul. 2021]. [Online]. Available: http://introlab.github.io/rtabmap

[141] manoelpla, "pepper_sim_ws," Jul 2021, [Online; accessed 18. Jul. 2021]. [Online]. Available: https://github.com/manoelpla/pepper_sim_ws

[142] chaolmu, "gazebo_models_worlds_collection," Jul 2021, [Online; accessed 18. Jul. 2021]. [Online]. Available: https://github.com/chaolmu/gazebo_models_worlds_collection

[143] A. Dworak, P. Charrue, F. Ehm, W. Sliwinski, M. Sobczak, A. Dworak, P. Charrue, F. Ehm, W. Sliwinski, and M. Sobczak, "Middleware trends and market leaders 2011," in *13th International Conference on Accelerator and Large Experimental Physics Control Systems*, 2011, pp. 1334–1337.

[144] ITU-T, "Itu-t g.114 (05/2003), series g: Transmission systems and media, digital systems and networks, international telephone connections and circuits – general recommendations on the transmission quality for an entire international telephone connection," 2019. [Online]. Available: http://handle.itu.int/11.1002/1000/6254

[145] T. Arbuthnot, "What are Thresholds for Good and Poor Network Packet Loss, Jitter..." *Tom Talks*, Jul 2019. [Online]. Available: https://tomtalks.blog/2018/05/what-are-thresholds-for-good-and-poor-network-packet-loss-jitter-and-round-trip-time-for-unifie

[146] T. Ohata, K. Ishibashi, and G. Sun, "Non-contact blood pressure measurement scheme using doppler radar," in *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC).* IEEE, 2019, pp. 778–781.

[147] M. Garbey, N. Sun, A. Merla, and I. Pavlidis, "Contact-free measurement of cardiac pulse based on the analysis of thermal imagery," *IEEE transactions on Biomedical Engineering*, vol. 54, no. 8, pp. 1418–1426, 2007.

[148] D. K. Ng, C.-H. Chan, R. S. Lee, and L. C. Leung, "Non-contact infrared thermometry temperature measurement for screening fever in children," *Annals of tropical paediatrics*, vol. 25, no. 4, pp. 267–275, 2005.

[149] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology.* Elsevier, 1988, vol. 52, pp. 139–183.

[150] E. Ammenwerth, "Technology acceptance models in health informatics: Tam and utaut," *Stud Health Technol Inform*, vol. 263, pp. 64–71, 2019.

[151] A. Chang, "Utaut and utaut 2: A review and agenda for future research," *The Winners*, vol. 13, no. 2, pp. 10–114, 2012.

[152] J. Han and D. Conti, "The use of utaut and post acceptance models to investigate the attitude towards a telepresence robot in an educational setting," *Robotics*, vol. 9, no. 2, p. 34, 2020.

[153] J. R. Lewis, "Psychometric evaluation of the pssuq using data from five years of usability studies," *International Journal of Human-Computer Interaction*, vol. 14, no. 3-4, pp. 463–488, 2002.

[154] A. Baharum, S. M. Amirul, N. M. M. Yusop, S. Halamy, N. F. Fabeil, and R. Z. Ramli, "Development of questionnaire to measure user acceptance towards user interface design," in *International Visual Informatics Conference.* Springer, 2017, pp. 531–543.

[155] A. Reis, R. Xavier, C. Macedo, T. Costa, V. Rodrigues, and J. Barroso, "A roadmap to evaluate the usage of telepresence robots in elderly care centers," in *2018 2nd International Conference on Technology and Innovation in Sports, Health and Wellbeing (TISHW).* IEEE, 2018, pp. 1–6.

[156] A. Cao, K. K. Chintamani, A. K. Pandya, and R. D. Ellis, "Nasa tlx: Software for assessing subjective mental workload," *Behavior research methods*, vol. 41, no. 1, pp. 113–117, 2009.