KAUNAS UNIVERSITY OF TECHNOLOGY

JULIUS GELŠVARTAS

MULTIFUNCTIONAL USER INTERFACE FOR QUADRIPLEGICS: DEVELOPMENT AND INVESTIGATION

Doctoral dissertation

Technological sciences, informatics engineering (T 007)

2019, Kaunas

This doctoral dissertation was prepared at Kaunas University of Technology, Faculty of Electrical and Electronics Engineering, Department of Automation during the period of 2014–2018. The studies were supported by Research Council of Lithuania.

Scientific Supervisor:

Prof. Dr. Habil. Rimvydas SIMUTIS (Kaunas University of Technology, Technological sciences, Informatics engineering, T 007).

Scientific Advisor:

Dr. Rytis MASKELIŪNAS (Kaunas University of Technology, Technological sciences, Informatics engineering, T 007).

Doctoral dissertation has been published in:
http://ktu.edu

Editor:

Armandas Rumšas (Publishing Office "Technologija")

ISBN 978-609-02-1617-0

The bibliographic information about the publication is available in the National Bibliographic Data Bank (NBDB) of the Martynas Mažvydas National Library of Lithuania.

KAUNO TECHNOLOGIJOS UNIVERSITETAS

JULIUS GELŠVARTAS

DAUGIAFUNKCĖS VARTOTOJO SĄSAJOS PARALYŽIUOTIESIEMS KŪRIMAS IR TYRIMAI

Daktaro disertacija

Technologijos mokslai, Informatikos inžinerija (T 007)

2019, Kaunas

Disertacija rengta 2014-2018 metais Kauno technologijos universiteto Elektros ir elektronikos fakultete Automatikos katedroje. Mokslinius tyrimus rėmė Lietuvos mokslo taryba.

Mokslinis vadovas:

prof. habil. dr. Rimvydas SIMUTIS (Kauno technologijos universitetas, Technologijos mokslai, Informatikos inžinerija, T 007).

Mokslinis konsultantas:

dr. Rytis MASKELIŪNAS (Kauno technologijos universitetas, Technologijos mokslai, Informatikos inžinerija, T 007).

Interneto svetainės, kurioje skelbiama disertacija, adresas:
http://ktu.edu

Redagavo:

Armandas Rumšas (leidykla “Technologija”)

ISBN 978-609-02-1617-0

Leidinio bibliografinė informacija pateikiama Lietuvos nacionalinės Martyno Mažvydo bibliotekos Nacionalinės bibliografijos duomenų banke (NBDB).

Acknowledgement

The author would like to thank his supervisor Rimvydas Simutis. This path would have been much more difficult without Rimvydas' encouragement and support. I would also like to thank my scientific advisor Rytis Maskeliūnas. I am highly grateful to all my family members and friends who have been supporting me along the way. A very special gratitude goes out to my wife Eglė and children Ramunė and Liudvikas for their patience and support. And finally, last but not the least, also to everyone at *Rubedo sistemos* for giving the opportunity to coordinate work with studies.

I dedicate this thesis to my late grandfather Algirdas Karalius. Conversations with him ignited the love for science in me.

"...no-one is actually dead until the ripples they cause in the world die away... The span of someone's life, they say, is only the core of their actual existence."

— Terry Pratchett, Reaper Man

Contents

List of Figures

List of Tables

1. INTRODUCTION

1.1. Problem statement and relevance

Communication is the most important human skill that we learn early in life and use daily to express our thoughts, collaborate and ask for help. Communication problems can lead to social exclusion, stress and anxiety. People suffering from severe motor disabilities often experience speech pathologies and have difficulty communicating. These problems significantly affect the quality of life of patients [Westgren and Levi, 1998] as well as their relatives. This is especially true for people suffering from severe disabilities such as quadriplegia.

Quadriplegia, also known as tetraplegia, is paralysis of all four limbs and torso. Quadriplegia is usually caused by damage to the brain or the spinal cord. Typical causes of this damage (referred to as a *lesion*) are trauma, disease, or congenital disorders. Lesions that cause quadriplegia are situated between C1 (the highest cervical vertebra, located at the base of the skull) and C7. Quadriplegia is classified as either complete or incomplete. Patients with incomplete quadriplegia have a partial motor function, whereas complete form patients possess no sensory or motor function. The outcomes and long term prognosis are therefore highly individual and difficult to predict.

Quadriplegic patients require constant care that is both time consuming and expensive. The majority of quadriplegias are caused by spinal cord injuries (SCI). SCI care costs are very high during the first year due to medical expenses. It is estimated that SCI can cost up to 800,000USD during the first year and up to 200,000USD each consecutive year. Only about 35% of SCI survivors are able to return to work [Lidal et al., 2007]. They also have a shorter life expectancy due to increased risk of respiratory infections and kidney problems. Making patients more independent can therefore improve their quality of life as well as reduce the need for care.

People with disabilities use assistive technology (AT) to perform tasks that they would otherwise not be able to perform. AT consists of two parts, namely, an assistive device and human computer interaction (HCI) interface. Some assistive devices do not have a HCI component and are used to directly interact with the environment. For example, a sip and puff switch can be connected directly to a wheelchair and used for its control. This approach is, however, not versatile. Assistive devices connected to computers can be used to perform many tasks of wide variety.

Each spinal cord lesion is different, therefore, best outcomes can only be achieved by personalizing rehabilitation and AT. Unfortunately, assistive devices in use today are inefficient. Each device has its own user interface (UI) and software, making it very difficult or even impossible to use several devices simultaneously. Moreover, patients have to learn to use a new UI when they change devices. This makes the AT personalization process very difficult. Patients only use one device, whereas changing devices when their abilities change would be more efficient. UI software that integrates multiple devices would make the personalization process easier. Such a system could also improve device efficiency and user experience.

Adaptive AT systems are relevant not only for quadriplegia patients. Such systems could be used by people suffering from other motor disabilities such as Huntington's disease or Cerebral palsy.

1.2. Thesis object

The object of this thesis is consumer grade assistive technology devices and modern HCI methods.

1.3. Aim of the Thesis

The aim of this thesis is to help people suffering from severe motor disabilities to communicate efficiently. The proposed solution concentrates on easy system adaptability for each individual user. Constant system customization ensures efficient Human-Computer interaction (HCI) independently of user's motor disability level.

1.4. Objectives of the thesis

The aim of this thesis shall be achieved by solving the following tasks:

1. To analyze and compare consumer grade AT devices and modern HCI methods;

2. To design multifunctional UI system architecture;

3. To propose real time scene recognition and augmented reality (AR) user interface presentation algorithms;

4. To propose predictive algorithms that improve system interaction time;

5. To develop an adaptive AT user interface application that integrates multiple input devices. The created application also contains an image processing pipeline and AR information presentation component;

6. To conduct experiments and evaluate the developed UI and proposed algorithms.

1.5. Research methodology

The results presented in this thesis have been achieved by using two main methods. First, the participants were asked to use the developed system. The recorded usage experiments were used to evaluate the system efficiency. Secondly, the already existing benchmarks of manually labeled data-sets were used to evaluate the proposed algorithms. Detailed research methodology description can be found in Section 5.1.

1.6. Scientific novelty

The research achieved these novel results:

1. Development of projection mapping based object selection UI. The novel automatic projection mapping is achieved by combining camera-projector system calibration with Color and depth camera (RGB-D) based object detection pipeline. The system automatically detects objects and uses a projector to highlight them in the real world. The user can select the desired object by using this UI.

2. A region-based stereo matching algorithm that can process images in the real time on an embedded computer has been proposed. The novelty of the stereo matching algorithm lies in the use of region based features intended to reduce the stereo matching search space. A stereo camera with an onboard embedded computer is used as an environment sensing device.

3. Camera systems operating in changing lighting conditions require fast and stable camera parameter control algorithms. A cascade camera parameter control algorithm has been proposed. The main algorithm novelty is the use of a cascade controller to control multiple camera parameters simultaneously. Another novelty is using controller parameter gain scheduling to increase the controller's efficiency and stability. The algorithm is used to ensure the reliability of the image processing pipeline.

1.7. Practical significance

There are at least 12,000 cases of SCI in the USA each year. SCI survivors require constant care which is costly and time consuming. This thesis concentrates on developing an adaptable Assistive technology (AT) system. This has several practical benefits. First, the user can start to use the system in the early stages of rehabilitation. This increases the system adoption rate and can help to speed up the rehabilitation process. Second, as the user's abilities change, the system adaptation ensures that the highest possible efficiency is achieved. Both of these factors contribute to costs savings due to the potentially shortening rehabilitation process and the capacity to enable the user to perform tasks independently. System adaptability is also important for degenerative motor disabilities when the system is adapted as the user loses physical abilities.

Multiple device integration has increased the system adaptability. Such a system has some additional practical benefits. Inexpensive assistive device integration would make the system accessible to people from lower income countries. This is especially important as the number of affordable assistive devices increases. Inexpensive devices usually lack specialized software packages whereas the system presented in this thesis aims at addressing this issue.

Finally, this thesis explores how the emerging technologies, such as augmented reality Augmented reality (AR), can be used in AT development. We explored the use of projection mapping to create a natural HCI method for object selection. Current AR technologies still have limitations in various real life scenarios. Computer vision algorithms presented in this thesis address some of these limitations and can be used to create more robust AR applications. Such applications can be used in AT development as well as other domains.

The system described in this thesis has been developed as part of the Agency for Science, Innovation and Technology (MITA) EUREKA project Quadribot. This project also provided recommendations for system personalization and customization. The proposed stereo matching algorithm has been developed in the framework of the Lithuanian Research Council project REP-15114. The projection mapping user interface has been developed as part of European cooperation in science and technology COST Action CA15122, Reducing Old-Age Social Exclusion: Collaborations in Research and Policy (ROSEnet).

1.8. The defended statements

1. Assistive user interfaces that integrate multiple assistive devices are more adaptive. Adaptability ensures that the system is suitable for a larger user group. Assistive devices are used to control the proposed multifunctional user interface.

2. A cascade controller can be used to efficiently control the camera shutter speed and sensor gain parameters simultaneously. Controller stability can be ensured by performing gain scheduling in order to address the non-linearity of the process. This controller is used to improve the multifunctional user interface image processing pipeline reliability.

3. A maximally stable extremal region based stereo matching algorithm can be used to reduce the search space. Reducing the search space can significantly improve the algorithm computation time and reduce memory consumption. These algorithms can be used to perform computations on an embedded Jetson TK1 computer. Such a system is used as a reliable environment sensing device of a multifunctional user interface.

4. A specialized text predictor can help to input words more efficiently than a standard text predictor. Such a predictor is used for efficient text input during topic-specific conversations. Efficient text input is one of the capabilities offered by the proposed system.

1.9. Scientific approval

Results presented in this thesis are original and are outlined across a total of 9 publications. 3 publications have been delivered in “ISI Web of Science” scientific journals with Impact Factor, whereas 1 publications was presented in an “ISI Web of Science” journal. The remaining five publications were printed in conference proceedings.

The research results have been presented in the following international conferences:

1. ‘The 4th International Conference on Information and Communication Technologies for Ageing Well and e-Health’ (ICT4AWE) 2018, Funchal, Portugal;

2. ‘The 20th International Conference on Methods and Models in Automation and Robotics’ (MMAR) 2015, Międzyzdroje, Poland;
3. ‘The 20th International Conference Biomedical Engineering’ 2016, Kaunas, Lithuania;
4. ‘The 21st International Conference Information and Software Technologies’ (ICIST) 2015, Druskininkai, Lithuania;
5. ‘The 9th International Conference Electrical and Control Technologies’ (ETC) 2014, Kaunas, Lithuania;

Academic achievements have been appreciated by two Lithuanian Research Council grants for academic achievements (2016 and 2018) and two Kaunas University of Technology grants for the most active doctoral students. The author participated in VISion Understanding and Machine intelligence (VISUM) summer school at Universidade do Porto, Portugal, 2015. In 2016 and 2017, the author attended the European Robotics Forum in Ljubljana, Slovenia and Edinburgh, United Kingdom and also participated in international conferences ROSCon 2016 Seoul, South Korea and ROSCon 2017 Vancouver, Canada. The results and methods described in this thesis have been presented in the above mentioned events. The author has won the first place in the Hacker Games 2016 Smart Energy contest.

1.10. Thesis structure

The thesis is outlined in 5 chapters. Chapter 2 presents and discusses the state of the art assistive technologies. This chapter also discusses other technologies that can be used to create modern assistive technologies. The detailed architecture of the presented multifunctional user interface is presented in Chapter 3. Chapter 4 describes novel algorithms proposed in this thesis. The experiments performed to evaluate the multifunctional user interface and proposed algorithms are presented and described in detail in Chapter 5. Chapter 6 contains the conclusions.

2. ANALYSIS OF ALREADY EXISTING ASSISTIVE TECHNOLOGY SOLUTIONS

This chapter reviews the state-of-the-art AT related to this thesis. AT is a very broad term covering a range of assistive, adaptive and rehabilitative devices as well as specialized HCI user interfaces. Modern AT are complex systems that integrate simple but reliable assistive devices with advanced UI programs. Patients are able to perform complex tasks by using such systems. We shall mainly focus on five key aspects of AT HCI systems, namely, input modalities, information processing, information presentation, environment sensing and interaction. A more detailed overview of the discussed topics can be found in Fig. 2.1.

The use of AT has a significant impact on the lives of people suffering from quadriplegia. Numerous studies have shown that AT users experience a higher quality of life. This is mainly a result of the increased independence [Manns and Chad, 2001]. AT users participate in more activities and are more independent, whereas non-users spend time in passive recreational activities in residence [Efthimiou et al., 1981].

People with disabilities use a broad range of technologies to perform various activities. Mainstream technologies, such as tablets and smart phones, are very popular due to their availability and low price. Furthermore, such technologies can be used to perform many different activities. Difficulty to customize is the main disadvantage of the mainstream technology. Commercially available AT are specifically designed to be used by individuals with disabilities. Such technologies can be used "off the shelf" and require minimal or no modifications. They are, however, designed for a single activity use-case and suffer from limited functionality. Custom-made AT are created to meet the specific needs of an individual or a small group. Such technologies have a very limited availability and tend to be more expensive. These three types of AT technologies form a continuum that is discussed in more detail in [Cook and Polgar, 2014].

Designing and manufacturing custom hardware devices is a time consuming and expensive process. A preferred way of creating a customized AT is, therefore, to use the already existing devices and create specialized software packages. Intelligent software packages (usually called multifunctional) can be used to perform multiple activities.

2.1. Multifunctional user interface structure

The structure of a multifunctional UI system can be split into several components. All UI systems have at least one input modality. Input modalities are used to express the actions that the user wants to perform. Input modalities are usually hardware devices, but can also be virtual devices. Speech recognition used for HCI is one example of a virtual device. Assistive devices generate action signals that change the state of the UI. A range of different assistive input modalities is discussed in more detail in Section 2.2.

Action signals generated by input modalities are passed to the information processing module of the system. The information processing module is responsible for

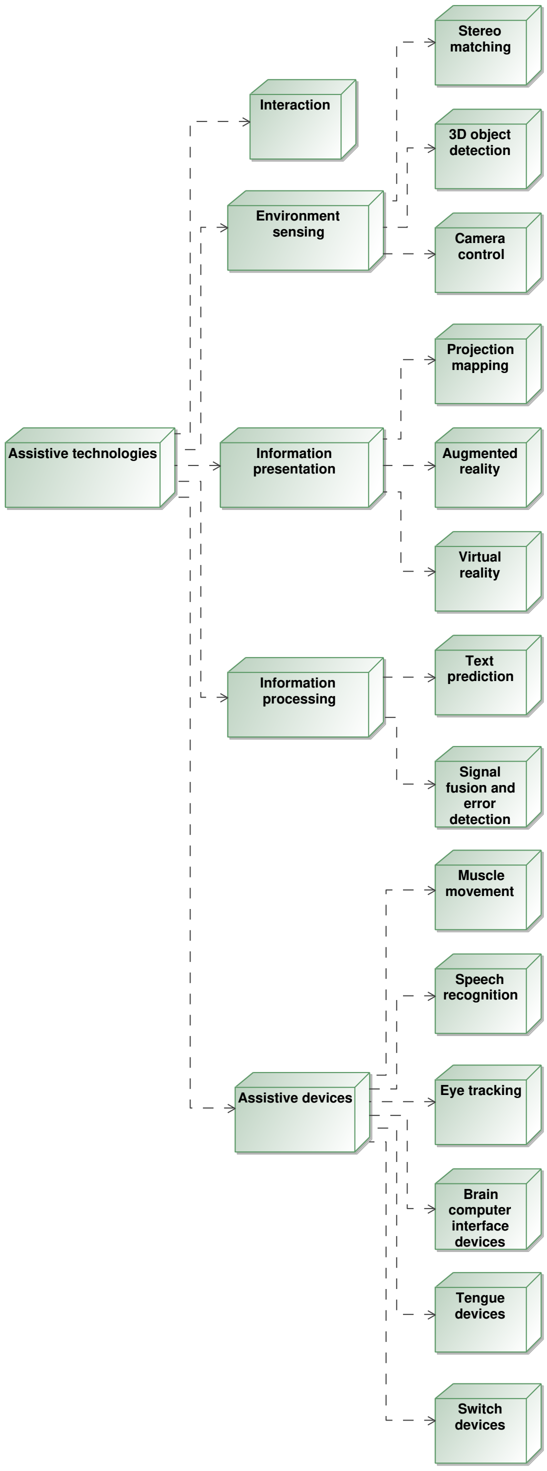


Figure 2.1. Overview of topics covered in this chapter.

mapping input modality signals into changes of the UI internal state. Signals generated by input modalities can also be post-processed in the information processing module. This might be necessary when performing multiple signal fusion or error detection. Relevant information processing algorithms are presented in Section 2.3. In the graphical user interface GUI design stages, this part of the system is often referred to as the business logic. The internal state of this module is usually represented as a state machine.

The internal state of the whole system is presented to the user in the GUI. The design of the GUI determines the system usability and should therefore be very well designed. Several information presentation methods are discussed in Section 2.4. The GUI and internal state separation is a well known Model-View-Controller MVC paradigm [Vlissides et al., 1995]. This paradigm is used in many GUI development frameworks. MVC makes it possible to split the system into several smaller components. The design and development of these components can be done separately and by different people. The View part of the system is often developed in close collaboration with user experience UX experts and designers. The development of the Model part requires more understanding of the internal states and transitions of the system.

Multifunctional user interfaces not only present the state of the system, but can also be used to manipulate the environment. Such systems are connected to active assistive devices also called actuators. These can, for example, be wheelchairs or robotic arms. Some interactive systems are also connected to environment sensing devices. Environment sensing is used to improve the overall system efficiency and increase its safety (see Section 2.5). Assistive interactive systems are discussed in Section 2.6. The information flow between different parts of the multifunctional user interface are illustrated in Fig. 2.2.

The majority of assistive user interfaces are developed for single activity use-cases. The development of a single activity UI is easier than multifunctional UI. A single activity UI usually supports only one assistive device, or may be optimized for controlling a single actuator, such as a wheelchair. Alternatively, an assistive device can be used as a conventional input modality in a regular computer operating system. In this case, no specialized UI is necessary, and the user can access almost all the functionality of the operating system. Unfortunately, using an assistive device without specialized UI is inefficient because operating systems are rarely designed with disabled people in mind.

Multifunctional user interfaces have several advantages over single activity interfaces. First, such interfaces can easily be adapted to the needs of each individual. Second, multifunctional user interfaces have modular architectures that make it possible to add new devices and functionalities. Using multifunctional UI in assistive technologies is not a novel idea. One of the first multifunctional assistive UI architectures was proposed in [Flachberger et al., 1994]. The main purpose of the system was environment control and making telephone calls. Conventional devices, such as a keyboard, a mouse, a joystick and switches have been used for user input. This system was further

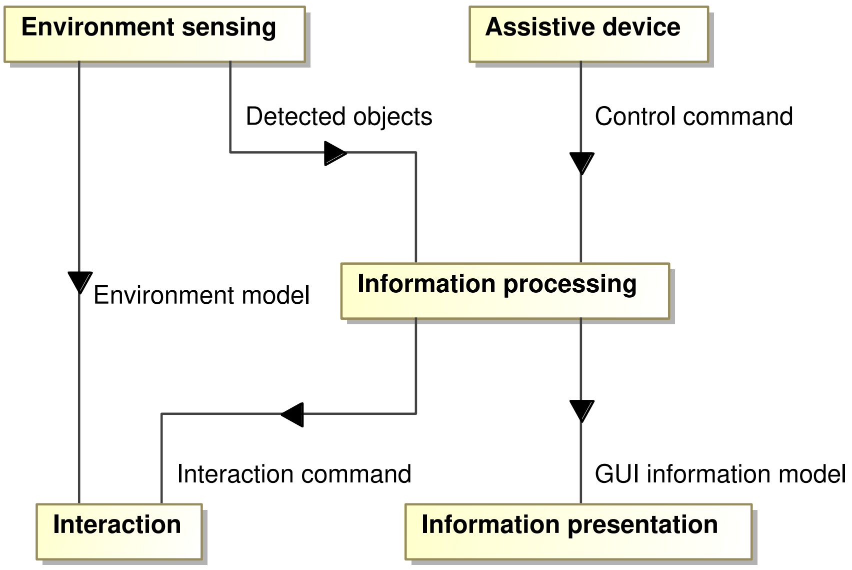


Figure 2.2. Multifunctional user interface information flow.

improved in [Flachberger et al., 1995]. This paper emphasizes the importance of having an easily configurable system that could be changed by the therapist when needed. The development of this system has been halted after initial implementation.

2.2. Assistive devices

Assistive devices are hardware products used by people with disabilities. Users can regain lost abilities or improve performed task efficiency by using these devices. Assistive devices can be split into two categories, namely, input devices and interaction devices. Input devices are mainly used for HCI, while interaction devices are used to change the environment. This section reviews input assistive devices. More detailed information on interaction devices can be found in Section 2.6.

2.2.1. Switch devices

Switch devices are the simplest and cheapest assistive devices and are, therefore, most widely used. A typical switch can generate only one signal when it is pressed. Some switch devices, such as sip and puff devices can have multiple signals, namely, one when sipping air from a tube and one when puffing. Additional signals can also be generated by double clicking or by adding more than one switch. In general, assistive switch devices are very similar to buttons of a conventional computer mouse. Therefore, assistive switch signals are usually mapped to computer mouse click events. The factors that should be considered when choosing a switch as an assistive device are summarized in [Angelo, 2000].

In general, any GUI can be navigated by using at least two input signals. One signal is used for selecting a particular GUI element and at least one more signal is necessary to advance to the next GUI element. The limitation of one signal switch can be overcome by using automatic GUI element advancement. Such systems automatically advance GUI elements at fixed time intervals. Unfortunately, this approach dramatically reduces the system efficiency because the user has to wait a fixed time interval before reaching the desired GUI element.

Despite their limitations, switch devices are being successfully used for assistive and rehabilitation purposes [Rojhani et al., 2017]. The switch device efficiency can also be improved by using intelligent software packages. It was shown in [Venkatagiri, 1993] that lexical predictors can reduce the switch activation effort by as much as 50%. Another possibility is to use additional sensors to make the system aware of its environment so that only a limited number of choices are shown to the user. This approach has successfully been used to create a single switch controlled wheelchair [Ka et al., 2012]. Sip and puff switches have even been used to control robotic arms with many degrees of freedom [Lu and Wen, 2015]. The main limitation of such approaches is that they are very use case specific and cannot be easily reused to perform different tasks.

2.2.2. Tongue devices

Tongue operated assistive devices have been developed in recent years. These devices use an array of teeth-mounted sensors to accurately track a tongue-mounted permanent magnet [Krishnamurthy and Ghovanloo, 2006]. The agility of tongue muscles makes it possible to perform very accurate position control actions. The tongue drive system design was improved in [Huo and Wang, 2008] making it more comfortable. It has been shown that such a sensor could be used to control a smartphone [Kim et al., 2010] as well as a wheelchair [Kim et al., 2012].

Despite the promising results shown by these sensors they are only early prototypes used by researchers. No consumer grade tongue drive devices are available on the market. The expected price range of such devices is also difficult to predict. Moreover, there should be a procedure for disabling the device when the user wants to start speaking.

2.2.3. Brain computer interfaces

The Brain computer interface BCI research field has a large community and is rapidly expanding. BCI devices offer the most natural UX, i.e., being able to perform actions by thinking. Interest in this field has led to the development of numerous BCI devices. These devices are denoted by significantly different designs and performance metrics. BCI devices work by measuring the user's voluntary brain activity (also called brain waves). The electrical activity of the brain is measured by using electrodes. The measured signals are then translated into computer commands or messages.

Many factors have to be considered when designing BCI systems. Most important hardware aspects are: the electrode type and the number of electrodes. Electrodes can be either be invasive (internal) or non-invasive (external). Non-invasive electrode systems are more convenient and are used in consumer grade BCI devices. Non-invasive BCI have several key limitations. First, non-invasive BCI use electroencephalography EEG to record the brain activity. The EEG signal produced by the brain can be influenced by electromyographic EMG activity from the scalp or facial muscles [Wolpaw et al., 2000]. Second, non-invasive BCI cannot effectively use higher-frequency signals because they are dampened by the skull. Invasive BCI, on the other hand, can record brain waves more accurately. Neurosurgery is required to insert invasive electrodes, and such systems are therefore mainly used for research purposes. The electrode location is also important, and the motor cortex is often used for invasive BCI systems.

The number of electrodes used by the BCI system is also an important factor. A higher number of electrodes record more information simultaneously. This information can be used to better decode the control signals generated by the users, but it also means that more information has to be processed by the computer. Non-invasive electrodes are usually distributed evenly on the whole skull, whereas invasive electrodes are mainly focused on a particular brain region. Generally, 8 to 36 electrodes are required to achieve good results [Tam et al., 2011].

While the hardware design of BCI system is important, major advances are made thanks to signal processing breakthroughs. These advances have led to the development of a thought-controlled robotic arm [Hochberg et al., 2012]. In this research, quadriplegic participants were able to directly control the robotic arm by using invasive BCI implanted into the motor cortex. The participants were able to perform three-dimensional reach and grasp movements. The robotic reach and grasp actions were, however, not as fast or accurate as those of an able-bodied person.

This success has led to many other research groups performing similar experiments and improving on the initial results [Schwartz, 2015]. Other groups have been working on solutions that would help to restore motor control as well as sensory feedback of robotic arms [Davis et al., 2016]. More recently, BCI has been used to restore the arm movements of a quadriplegic person [Ajiboye et al., 2017]. Such systems might in future be used to restore the full body movements of a quadriplegic person.

The consumer grade non-invasive BCI devices have been improving at a lower rate. It is difficult and expensive to manufacture BCI devices. These devices, therefore, usually have lower information transfer rates. Consumer-grade BCI systems with fewer than 8 electrodes have been shown to possess limited capabilities [Maskeliunas et al., 2016]. Moreover, non-invasive BCI devices only record lower frequency signals because the skull dampens high frequency signals.

2.2.4. Eye tracking

Eye tracking is also a rapidly developing technology. This technology is used for HCI as well as for studies of the visual system, in psychology, in product design and marketing. There are two main reasons for increasing the use of the eye tracking technology. First, inexpensive consumer-grade eye tracking devices already achieve reasonable performance results [Dalmaijer, 2014]. This makes it possible to easily use the technology not only for research purposes but also as a reliable HCI device for disabled people. Second, eye trackers have a higher information transfer rate and system usability when compared to BCI [Pasqualotto et al., 2015].

The use of eye trackers as an assistive device has led to many advancements over the years [Majaranta and Räihä, 2002]. It has been shown that eye trackers improve the quality of life of severely disabled people [Hwang et al., 2014]. In this paper, eye tracker has been used to control the computer. Eye trackers are also used to control powered wheelchairs [Lin et al., 2006]. The majority of eye tracking devices are positioned on a table in front of the user. Advances in camera technology have made it possible to create head-mounted eye tracking devices [Raudonis et al., 2009]. Head-mounted eye trackers are more convenient because they are not sensitive to head movements.

Despite many advances, eye trackers still have limitations. The Midas touch problem is the major problem associated with using eye trackers for HCI. This problem was discussed as early as 1991 in [Jacob, 1991]. The Midas touch problem arises when the eye gaze is used for HCI. In this situation, the eye is used for both perception and control. This problem has been addressed in different ways. One potential solution is presented in [Istance et al., 2008]. Eye tracking can also fail or work inaccurately when the subject's head is moving. This problem is only present in eye trackers that are not head-mounted. This problem can be solved by performing additional head tracking. In addition, head movements can sometimes be used as an additional control signal generator. A survey of head and eye tracking methods is presented in [Al-Rahayfeh and Faezipour, 2013].

Eye tracking devices can reliably track eye movements. These movements are usually translated into the cursor movement on the screen. Creating a selection command (i.e., equivalent to a mouse click) is more challenging. One approach is to use eye blinks to detect selection events, but this approach may be difficult to implement. Combining eye tracking with blink detection will usually give less accurate results [Pauly and Sankar, 2015]. In addition, involuntary eye blinks would cause problems in this situation. The easiest way to solve this problem is to use an additional assistive device (e.g., switch) to generate the selection commands. Such an approach is sometimes also used to reposition the cursor [Laurutis et al., 2010]. This is especially relevant when the gaze position does not match the cursor position exactly. In this situation, the cursor movement becomes very distracting to the user.

2.2.5. Speech recognition

The speech recognition technology can also be used as an assistive device. A well-designed speech recognition technology can resemble a human conversation and is a natural HCI method. Speech recognition can be classified into two categories, namely, keyword (also called *command*) based, continuous speech recognition. Continuous speech recognition is a more challenging problem. The use of deep neural networks has improved continuous speech recognition in recent years [Palaz et al., 2015]. Despite these advances, continuous speech recognition can only be used for text dictation and is therefore less relevant for assistive UI control. The majority of assistive systems only use command-based speech recognizers. Such a recognizer can be trained to recognize many different commands. An in-depth review of speech recognition technology advancements can be found in [Huang et al., 2014].

One advantage of speech recognition is that it does not require any specialized hardware. Usually, a conventional microphone is used to record speech. The recorded speech can then be processed by using a computer. An increase in mobile computing capabilities has made it possible to use speech recognition in mobile devices. It has been shown that an optimized speech recognizer can produce accurate results on a smart-phone [McGraw et al., 2016]. Using central processing unit CPU resources for speech recognition might not be desirable in some situations. This is especially relevant when other computationally expensive operations are running on the main system computer. Such a situation happens in multifunctional UI described in this thesis.

Speech recognition algorithms can also be implemented on specialized hardware. Field programmable gate array FPGA is often used to optimize the algorithm computation time and power consumption. The FPGA technology can also be used to optimize speech recognition algorithms. One example is a speech recognizer described in [Sledevič and Navakauskas, 2013]. The FPGA implementation significantly reduces the algorithm computation time while at the same time achieving good recognition accuracy.

The speech recognition technology also has several limitations. This technology works unreliably in noisy environments and is only suitable for HCI but not for human-to-human communication assistance. Moreover, speech recognition cannot be used by people who have speech pathologies. Another important factor for speech recognition is the number of recognized commands. Systems that have more commands are easier to use and result in better UX. Increasing the number of recognized commands will, however, reduce the recognition accuracy. One way to improve the recognition accuracy is to use big data sets to train the recognizer. This trained recognizer is later adapted to each individual user by performing additional training [Christensen et al., 2013]. Collecting such data sets and training recognition models is challenging and requires specially trained personnel.

2.2.6. Muscle movement

Electromyography EMG is a technique for recording electrical activity produced by skeletal muscles. EMG is similar to BCI in that it also uses electrodes to record electrical signals. Voluntary muscle movement is intuitive and is not affected by distractions from the environment. Therefore, these signals are easier to detect than brain waves. Moreover, EMG signal decoding does not require high computational power.

Traditionally, EMG has been used in medical research, but technological advance-

Table 2.1. Assistive device comparison.

Assistive device	Advantages	Disadvantages	Main cases of use
Switch	Simple, inexpensive, no signal processing needed	Limited number of signals	GUI navigation, text input
Tongue	Accurate position tracking	Sensors are placed in the mouth	Wheelchair control
BCI	Natural HCI method	Consumer grade devices inaccurate, invasive devices expensive	Robot arm control
Eye tracking	Detects accurate 2D position on the screen	Midas touch problem	UI navigation
Speech	Can recognize many commands	Sensitive to ambient environment noise	Used to execute many commands.
EMG	Can be used as a switch and a position estimate device	Electrodes placed on monitored muscles	Device control and UI navigation

ments have made it possible to create portable EMG devices with built-in processing. EMG devices have similar input capabilities as Switch devices. The user, of course, has to be able to reliably control the muscles that are being monitored.

Assistive EMG-based devices have gained popularity in recent years. Advancements in this area are making it possible to use EMG in many different assistive devices [Gopura et al., 2013]. The used muscles can be chosen in a way so that the HCI becomes reliable and easy to use. For example, eyebrow muscles have been used to control a powered wheelchair [Tsui et al., 2007].

2.2.7. Assistive devices comparison

Each assistive device has its own advantages and disadvantages. There is also significant variability between different devices of the same type. Moreover, each device has a specific use case where it achieves the best results. Table 2.1 summarizes the comparison of assistive devices discussed above.

Sip/Puff is the most commonly used Switch device. There are several reasons for Sip/Puff popularity. First, Sip/Puff devices directly convert user actions (sips and puffs) to computer control signals. This process does not require any sophisticated sensor signal processing algorithms. Second, it is very easy to use. The user can take the device and use it without any former training. The simplicity of the device ensures its reliability and low cost. The main limitation of Sip/Puff is only two control signals that it generates. There is no easy way to overcome this limitation.

Tongue devices are similar to Sip/Puff devices in that they also directly convert user tongue movements into position measurements. This results in accurate measurements that can be used to control a wheelchair or a robotic arm. This device is less convenient than other devices because it is placed in the mount and the user can find it difficult to speak with the device. Moreover, there are not a commercially available device in the market yet.

Brain computer interface (BCI) theoretically is the most convenient device because users can express their intentions by thoughts. In reality, BCI devices require sophisticated information processing algorithms that still are not reliable enough. Good results are also only achieved with very expensive or invasive devices. These limitations still limit the BCI adoption rate.

Eye trackers use image processing to determine the eye gaze coordinates. There are cost effective device available on the market. The main limitation of these devices is that they only generate positional information and there are therefore limitations when using these devices for Graphical user interface (GUI) control.

A speech recognizer does not require any additional devices to operate. This device can be used with only a computer microphone. This method can recognize many different commands, but the amount of recognized commands also affects the recognition quality. Recognition quality can be improved by adapting the recognizer for a particular user, however, this requires significant effort and is not often used. Speech recognition is also sensitive to the ambient environment noise.

Electromyography (EMG) devices are very similar to BCI but they measure muscle activity instead of brain waves. Generally, electric signals generated by muscles are stronger; it is therefore easier to record these signals. Quadriplegic users can only move facial muscles, therefore, EMG electrodes would have to be attached to the users' facial muscles. This makes the device less attractive from the aesthetic point of view.

All the devices expect for Sip/Puff and Tongue devices need additional recorded sensor data processing. The results generated by these devices are therefore less reliable. This is also reflected in the fact that the majority of these devices are still only used for research purposes, and their adoption is therefore limited.

2.3. Information processing

In this section, we shall discuss algorithms that are often used to improve assistive system efficiency. Information processing of the assistive device input signals has been discussed in the previous section. Here, we shall discuss how the commands of several assistive devices can be fused. Signal fusion makes it possible to use several devices simultaneously. In some situations, several devices might be used to generate the same command. In this case, the redundant information is used for error checking. System usability can also be improved by predicting user actions.

2.3.1. Signal fusion and error detection

Signal fusion can be used when there is more than one assistive device attached to the system. A fused signal can then be used as a virtual device that generates commands faster or more reliably. The neural network is a popular approach used for signal fusion. The neural network can take its input from many different modalities and generate a fused signal. Such an approach can extract information from different modalities and produce a better overall result [Han and Wang, 2016]. Obtaining sufficient data for neural network training is usually difficult. Moreover, the exact device combination might not be known in advance, and the use of neural networks becomes impractical.

Probabilistic signal fusion is another method that has been shown to work reliably. In particular, it uses the Bayesian fusion approach to generate a single signal from multiple devices. It was demonstrated in [Leeb et al., 2010] that such a strategy can be used to fuse EMG and BCI signals. Such multiple sensor information fusion can also be used to detect single sensor data errors. In this case, the system would adapt as the quality of the device signal degrades.

2.3.2. Text prediction

Text predictor is a system that predicts the next block of characters (letters, syllables, words, sentences, etc.). When the user sees a suitable prediction, they can complete the remaining block of text. This can greatly enhance the text input rate [Anson et al., 2006]. In recent years, text predictors have been integrated in the majority of text input software. They have been very useful in many areas, including healthcare [Hua et al., 2014]. There are many proprietary and open source text prediction packages available. *Presage* is an example of a popular open source text predictor used in many projects [Vescovi, 2004]. The performance of different text predictors varies and depends on many factors.

All text predictors employ specialized language models. The predictor usefulness depends on the accuracy of the underlying language model. Most basic word predictors use a word frequency model [Venkatagiri, 1993]. Such predictors calculate the most probable word given the characters that the user has already entered. These systems were used because they were easy to implement and train for. The word frequency model assumes that each word is typed independently. This assumption is of course not true. Each word in a sentence is closely related to the words that precede it.

More sophisticated language models incorporate more information to make more accurate predictions. An in-depth review of different prediction techniques and their evaluation can be found in [Garay-Vitoria and Abascal, 2006]. Language models can be classified into three categories, namely, syntactic, semantic, and statistical. Syntactic and semantic models store rules in probability tables or as a grammar. Statistical models use word or character frequencies. These frequencies are often stored in N-grams [Polacek et al., 2017]. The N-gram is a sequence of N words or characters. An N-gram based language model has to be trained. During the training, all non repeating N-grams are extracted, and each N-gram is assigned a frequency. N-gram frequency is the number of times a particular N-gram has appeared in the training dataset. The N-gram model records not only the probabilities of single words but also the relationships between two or more words. Such a model is still relatively easy to implement and train for. Cardinality N of the model determines the maximal number of words that are used to calculate the new prediction. For example, when a model with N=3 cardinality is used, two previously entered words are taken into account when predicting the current word. We should note that a model with N=1 cardinality is identical to the word frequency model.

The N-gram model has several limitations. The predictor completely ignores words that are outside the model cardinality. For example, when the model cardinality is three and the user has already entered four words, the prediction of the new word is calculated only taking two last words into account. Another limitation is that the model cannot differentiate similar phrases when the word order is changed. For example, phrases "a beautiful girl" and "girl is beautiful" are considered different although they are semantically very similar.

More advanced text predictors can be created by using syntactic and semantic language models. Some of these methods use N-gram models but add some additional processing steps [Trost et al., 2005]. Latent semantic analysis methods have been shown to have improved performance over the N-gram model [Wandmacher and Antoine, 2008]. More complex text predictors use classification methods [Van Den Bosch, 2006] or continuous space language models [Mikolov et al., 2013]. The longest common subsequence method [Sandnes, 2015] also produces good results but requires a large training dataset. These methods usually produce better predictions but are more difficult to implement and also require more computational resources. Moreover, constructing such models is difficult and requires manually labeled data, whereas the N-gram model can be trained from regular text samples.

Language models are trained by using training datasets. These datasets are often called *language corpus*. More sophisticated text prediction algorithms are usually trained using a large corpus. This makes it possible to create a model that contains the full representation of a particular language. Such language models are used in general purpose text predictors. A general purpose predictor is expected to return reasonable predictions for any given text. General purpose text predictor might not be suitable for all applications. Obtaining a large corpus for model training is challenging. This is often addressed by using literature books. Such books usually contain large portions of indirect speech. A language model trained on such a corpus will be biased towards indirect speech. People, on the other hand, mainly use direct speech when communicating.

Text predictors can further be improved by enabling them to learn while operating. A learning predictor can adapt to the unique user's writing style. This can be achieved by collecting information about the text that the user has already entered. This information can be incorporated into a language model to create a personalized predictor.

Such a system can learn new words that were not present in the initial language model. Learning makes it possible to improve even an imperfect model by updating it over time. Updating a complex model on-line is difficult or even impossible.

Another text predictor improvement can be made by using an additional semantic language model. Such a model would contain the semantic relationships of the words. Semantic information has been shown to be useful in speech recognition [Grau et al., 2006]. Similar techniques can be used to improve text prediction. Incorporating semantic information into a language model is complicated. One way of incorporating semantic information into a language model is by assigning probabilities $P(w_i|w)$ to all semantic relationships [Cao et al., 2005]. An alternative approach incorporates word meanings into an additional N-gram language model [Verspoor et al., 2008]. Both of these approaches require a semantic database as well as a large training data set. As already mentioned above, obtaining extensive training data set might be difficult.

2.4. Information presentation

Another important UI component is the information presentation method. Conventional UI use displays for information presentation. This method is suitable for simple HCI tasks, such as text input, web browsing, e-mail and Internet calls. These tasks are usually performed individually, and they do not require interactions with the environment. During such a task, users' attention is mainly focused on the computer screen and does not have to switch frequently.

Assistive technologies are used not only for simple HCI tasks. People with severe disabilities rely on assistive technologies for communication and even environment interaction. Such tasks can also be performed by using conventional displays, but the user's attention has to constantly move from the screen to the environment. This method is therefore inefficient.

Recent technological advancements have made it possible to develop alternative information presentation methods, such as augmented and virtual reality. These methods often use novel information presentation devices, such as virtual reality head-set and a head-mounted display HMD. HMD are well suited for augmented reality applications, but they are still expensive. Alternatively, such applications can be developed by using mobile devices or projection mapping (discussed in Section 2.4.3). All of these technologies offer a natural HCI method.

2.4.1. Virtual reality

Virtual reality (VR) is a computer-generated environment that simulates a realistic experience. This technology can be used for assistive purposes. Severely disabled people can perform tasks in VR that they would otherwise not be able to do. For example, a quadriplegic person can experience walking in VR. Moreover, VR can be used to train people with multiple physical and mental disabilities [Cunha and da Silva, 2017]. This study has shown statistically significant cognitive skills improvement. Another recent study has used virtual environments to train quadriplegic patients to use BCI devices more efficiently [Perdikis et al., 2018].

Rehabilitation is another area where VR has been proven to be beneficial. The SCI rehabilitation process is often long and difficult. It was shown in [Hefner et al., 2017] that VR can improve mechanical ventilation weaning rehabilitation process. Post stroke rehabilitation is another area where VR games have successfully been used [Shin et al., 2014].

The use of VR for rehabilitation is becoming more and more popular. VR is, on the other hand, not often used for daily task assistance. VR technology separates the user form the real environment. People suffering from quadriplegia are already separated from their environment by inability to move and communicate. Therefore, VR usage might have adverse effects in such situations.

2.4.2. Augmented reality

Augmented reality (AR) is a technology that superimposes a computer-generated image on a user's view of the real world, thus providing a composite view. Augmented reality applications use either HMD or mobile phones to superimpose additional information onto the real world view. A detailed survey and recent augmented reality advancements can be found in [Billinghurst et al., 2015]. The overall development of augmented reality applications is more difficult than virtual reality because the system has to perform environment tracking and understanding algorithms.

The main advantage of augmented reality is that it provides a very natural HCI method. The user is able to see not only the computer-generated information but also the real world scene simultaneously. The biggest limitation to wider AR adoption is lack of affordable high resolution devices. Despite limitations, augmented reality applications have successfully been used in post-stroke rehabilitation [Hondori et al., 2013]. Moreover, augmented reality is successfully being used in many assistive technology applications [Ong et al., 2011].

2.4.3. Projection mapping

Projection mapping is a form of augmented reality that uses projectors instead of HMD. The main advantage of such an approach is that the projected information is visible for all people observing the scene. Projection mapping is viewpoint sensitive, but it can produce realistic AR experience. Another major projection mapping limitation is that the projected content is difficult to see in very bright environments.

Projection mapping can be either manual or automatic. Manual projection mapping is mostly used in entertainment and art industries where the scene is static. Automatic projection mapping is a more sophisticated method that works in dynamic environments. Automatic projection mapping needs 3D information of the scene obtained with a depth camera. Depth camera information has to be transformed into the projector's optical frame. This transformation is obtained by calibrating the camera-projector system. The method presented in [Kimura et al., 2007] can be used when the camera has already been calibrated. Alternatively, structured light can be used for camera-projector calibration [Moreno and Taubin, 2012]. A more convenient and

Table 2.2. Information presentation method comparison.

Information presentation method	Advantages	Disadvantages	Main use case
Display	Inexpensive, high resolution	GUI separated from the environment	Used in majority of systems due to wide availability
VR	Immersive experience, user experiences fully generated environment	User separated from environment, expensive hardware	Rehabilitation and learning
AR	GUI overlaid on the real environment	Expensive hardware, need 3D environment understanding	Environment interaction
Projection mapping	Experience similar to AR but inexpensive	Not visible in bright environments, viewpoint sensitive	Environment interaction

efficient method is presented in [Yang et al., 2016].

One example of automatic projection mapping is presented in [Benko et al., 2012]. This system contains an algorithm to account for projection deformations caused by occluding objects. The system is mainly used to manipulate virtual objects. More advanced dynamic projection mapping methods have been created in recent years [Sueishi et al., 2015]. Such systems require more expensive hardware setup and are therefore not very practical.

2.4.4. Information presentation method comparison

The use of the particular information presentation method mainly depends on the performed task. The device price is another important factor when choosing an information presentation method. The main advantages and disadvantages of the discussed information presentation methods are summarized in Table 2.2.

2.5. Environment sensing

AR and intelligent AT applications need sensors to be able to understand the environment. 3D sensors are employed to make the sensing useful and complete. An ideal sensor should be very lightweight, consume as little energy as possible, be usable outdoors and have a high frame rate. Currently, there are four types of sensors that are suitable for obstacle avoidance and environment sensing, namely, Light Detection and Ranging (LIDAR), structured-light, Time of flight (TOF), and stereo cameras. 3D sensors that can produce a colored point cloud are often referred to as RGB-D cameras.

Structured light sensors satisfy almost all requirements. This type of sensor projects known patterns onto the environment in order to estimate the scene depth information. This approach, however, has several limitations. First, the projected pattern becomes undetectable in direct sunlight. Second, these sensors have a limited detection range of up to 5 meters. Despite these limitations, structured light sensors have been successfully used for obstacle avoidance in indoor environments [Stowers et al., 2011].

TOF cameras, on the other hand, are less affected by direct sunlight because they use modulated light. Some TOF cameras have a longer range than structured light cameras, but at a cost of increased power consumption. Another limitation of TOF cameras is the low resolution of the currently available sensors. The main advantage of TOF cameras is their very high frame rate (usually more than 100Hz).

3D LIDAR sensors are currently the most widely used type in outdoor environments. These sensors offer a very long range of 100m. or more as well as accurate distance measurements. These sensors, however, are still prohibitively expensive.

All of the discussed sensors are active, i.e., they emit light rays in order to determine the distance to objects. The biggest disadvantage of active sensors is power consumption. Actively emitting light requires a lot of energy. The stereo camera, on the other hand, is a passive distance sensing technology. Using passive environment sensing has several advantages. First, this type of sensor can better adapt to lighting changes in the environment because we can control their exposure time and gain parameters. These parameters can be controlled efficiently by using specialized camera control algorithms (see Section 2.5.1). Second, such a system consumes less energy than active sensing. This thesis investigates the possibility of using a stereo camera as the main environment sensing device for AT.

The accuracy of a stereo camera is highly dependent on the used camera lenses and the sensor resolution. Creating high resolution stereo cameras is challenging due to high computation required to perform stereo matching. Detailed description of stereo matching algorithms is provided in Section 2.5.2.

2.5.1. Camera control

Camera control is a process of setting the camera parameters in order to obtain well-exposed images. Correctly exposed images contain more detailed scene representation. This is important for many computer vision algorithms. For example, it might be easier to detect object boundaries in well-exposed images. Image exposure also plays a significant role in producing accurate stereo matching results.

There are several problems that make the camera control process challenging. First, there are at least two camera parameters that have to be controlled simultaneously, namely, shutter speed and the sensor gain. Other factors affecting image exposure are lens aperture and scene luminance. Some high end computer vision cameras can have an electronically controlled lens aperture. In this case, the aperture control becomes the third parameter that controls the image exposure. Second, the scene luminance may change very quickly. This may, for example, happen when entering or leaving a building, or switching lights on or off. In the majority of situations, the scene luminance cannot be controlled. The main purpose of camera control is therefore to adapt the camera parameters to the changing scene luminance conditions.

The majority of modern digital cameras contain built-in camera control functionality. The quality of these camera control algorithms can vary significantly. Some cameras might not be able to control both shutter speed and sensor gain simultaneously. Moreover, cameras are usually designed for general purpose use cases. Most commonly, cameras try to expose the whole image evenly. This strategy produces reasonable results, but they might not be optimal in all situations.

Environment knowledge can be used to assist the camera control algorithm. For example, some cameras operate in open environments where the top part of the camera image always sees the bright sky. In these situations, it might be useful not to consider the sky when evaluating the image exposure quality. This is often done by providing image masks or regions to the camera control algorithm [Nourani-Vatani and Roberts, 2007]. Such functionality is, however, lacking in the majority of digital cameras.

Another important factor contributing to the camera control algorithm efficiency is the metrics used to evaluate the image exposure quality. A comparison of different metrics used to evaluate the image exposure can be found in [Shirvaikar, 2004]. In general, the image quality metrics are chosen such that they could be computed in real time. Image statistics and histogram based metrics are, therefore, most often used in real time applications and on-board camera control algorithms.

2.5.2. Stereo matching algorithms

Processing a pair of stereo images is a very computationally intensive procedure. Therefore, stereo camera images are usually processed on powerful computers that consume a lot of energy. This approach is not portable and cannot be used in AT systems. In recent years, the latest embedded and mobile processors have become powerful enough to process stereo images. Such a platform is NVIDIA's *Jetson* platform which contains powerful CPU and GPU processors and has low power consumption. This platform can be used to create a portable and low power stereo camera with on-board processing capabilities. A portable stereo camera can be used to create an environment-aware AT.

Stereo matching algorithms can be divided into two categories, namely, local and global. Global algorithms can construct disparity images very accurately [Zhang et al., 2015] but they are computationally intensive and have high memory requirements. Even highly optimized versions of these methods have long computation times and are not suitable for real time applications. Local methods can process images faster and have much higher frame rates. This thesis concentrates on real time applications, and global methods are, therefore, not discussed in detail.

The best way to compare stereo matching algorithms is by using standard datasets and benchmarks. The two most popular stereo matching algorithm evaluation datasets are Middlebury version 3 [Scharstein et al., 2014] and KITTI Vision Benchmark [Menze and Geiger, 2015]. A qualitative comparison of different algorithms is, however, complicated. First, the threshold for bad pixels is different in Middlebury and KITTI datasets. For Middlebury, the dataset pixels whose disparity error is more than 4 pixels are considered bad pixels. Whereas, in the KITTI dataset, bad pixels have a disparity error of more than 3 pixels. Therefore, the percentage of bad pixels should be higher in the KITTI dataset if the algorithm has identical performance. Second, Middlebury and KITTI datasets are very different. The KITTI dataset contains outdoor images, whereas Middlebury contains only indoor images. Moreover, the resolution of images in the datasets is different. Third, a comparison of the algorithm runtime is even more complicated. Both Middlebury and KITTI datasets mainly focus on the accuracy of the reconstructed disparity images. Researchers run algorithms on their own hardware and only submit computed disparity images and computation times. The hardware that was used for computation is different between algorithms, and runtime results are therefore difficult to compare.

Block-matching (BM) is a classical and very simple stereo matching algorithm. Image blocks from two stereo images are compared by using the Sum of Absolute Difference (SAD) error criterion and the full scan-line search for finding the optimal block displacement. Full scan-line search used in BM produces a lot of errors, especially in untextured areas. This is usually addressed by running image pre-filters which remove untextured areas and by adding the matched pixel uniqueness criterion. Even with these measures, the results produced by BM are sparse and have a high bad pixel percentage. BM can be easily parallelized due to its simplicity [Massanes et al., 2010].

Stereo Processing by the Semi-Global Matching and Mutual Information (SGM) method [Hirschmuller, 2008] presents a more sophisticated matching strategy that has become very popular in recent years. The Semi-Global Matching algorithm enforces reconstructed surface smoothness by introducing two matching cost penalties, namely, P1 and P2. P1 is a penalty applied to the matching cost when the disparity value of the current pixel changes by 1 pixel and P2 is applied when the disparity value of the current pixel changes by more than 1 pixel. This technique helps to reconstruct disparity values in the untextured areas of stereo images. Choosing good parameter values can be challenging; it has major impact on the algorithm performance.

Despite its limitations, SGM led to the development of many modern stereo matching algorithms. For example, Semi-Global Block Matching (SGBM) uses Semi-Global Matching from the SGM algorithm, but calculates the matching costs by using the SAD of image patches. At the moment, SGBM is one of the most widely used algorithms in real time applications due to its fast processing times and the quality of the produced images. This is the main reason why SGBM is used for result comparison in this thesis.

Embedded real-time stereo estimation via Semi-Global Matching on the GPU (SGM-GPU) [Hernandez-Juarez et al., 2016] is another method that uses Semi-Global Matching. The matching cost is computed by using Center-Symmetric Census Transform, and, additionally, a Median filter is used to remove outliers. This implementation was optimized for the CUDA platform and achieves very good processing times on NVIDIA Titan X.

In recent years, Convolutional Neural Networks (CNN) have become very popular in solving various computer vision problems, and stereo matching is not an exception. In 2015, Stereo Matching by training a Convolutional Neural Network to Compare Image Patches (CNN-CIP) [Zbontar and LeCun, 2016] was one of the first methods utilizing CNN and achieving high frame rates as well as good accuracy. This algorithm uses CNN for matching cost computation, and matching is performed by using Semi-Global Matching from SGM. These results are achieved by using a very powerful GPU card that has 16 times more GPU cores than the NVIDIA's Jetson TK1 processor. Methods such as [Kowalczuk et al., 2013] and [Kim et al., 2016] also have high stereo matching frame rates on powerful GPU processors. These algorithms would have significantly lower performance on embedded platforms such as NVIDIA's Jetson TK1, or they might not work at all due to high memory consumption.

Efficient Deep Learning for Stereo Matching (EDL-SM) [Luo et al., 2016] is a more advanced method that uses CNN not only for cost computation but also for predicting the disparity image values. The produced output is not competitive compared with other modern methods, and the authors are therefore using additional post-processing stages to improve the quality of the produced disparity image.

A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation (LDTCN-DOSE) [Mayer et al., 2016] is also a CNN-based approach that is used to perform the full disparity estimation task without requiring additional matching algorithms. The main challenge of CNN-based stereo matching algorithms is that they require large datasets for training. This is usually addressed by synthetically generating large datasets. Another disadvantage of most CNN-based methods is that even optimized versions of these algorithms require powerful GPU processors (usually NVIDIA Titan X). Therefore, these methods cannot be used on low power embedded platforms.

Efficient large-scale stereo matching (ELAS) is another algorithm that has high frame rates [Geiger et al., 2010]. This algorithm creates a dense disparity prior image by triangulating robustly matched support points. Having a disparity prior allows to speed-up the computation of the final dense disparity image. Having an algorithm stage for generating a prior makes it sensitive to incorrectly matched support points. The algorithm presented in this thesis was inspired by ELAS. The presented algorithm also generates a disparity prior but does not use support points.

The majority of stereo matching algorithms use fixed size features, such as SAD or center-symmetric census transform [Spangenberg et al., 2013]. Such features are easy to compute, and they produce reasonable results. Fixed size features, however, do not work in large untextured image regions. Using variable size features is not trivial. First, there are not many variable size features available because they require more computations. Second, it is more difficult to match features of different sizes. Maximally stable extremal regions (MSER) [Matas et al., 2004] are one of the most

Table 2.3. Short description of discussed stereo matching algorithms.

Stereo matching algorithm	Algorithm working principle and description
BM	Uses SAD features for cost computation. For each pixel, it finds the best correspondence pixel. Uses per and post filtering to remove noise and untextured areas.
SGM	Uses disparity optimization in 8 directions. Cost is computed by using mutual information metrics for each pixel. Enforces reconstructed surface smoothness.
SGBM	Matching method the same as SGM. Matching cost computed by using SAD features.
SGM-GPU	Parallel implementation of SGM algorithm. Uses center-symmetric census transform for cost computation.
CNN-CIP	Uses CNN for feature matching cost computation. The mathing is performed by using SGM.
EDL-SM	CNN based stereo matching algorithm that performs both cost computation and matching on the network.
LDTCN-DOSE	Another CNN based stereo matching architecture. Achieves good results but requires large amounts of training data and powerful GPU.

popular variable size features. The main advantage of MSERs is that they can be detected very quickly on gray-scale [Nistér and Stewénius, 2008] as well as color images [Forssén, 2007]. MSERs have already been successfully used in a wide-baseline stereo matching setup. Wide-baseline matching is difficult because the epipolar geometry is unknown. A stereo camera can produce rectified images, and in this case matching only has to be performed along the epipolar line.

The majority of stereo matching algorithms concentrate on the accuracy of the reconstructed scene and neglect the algorithm computation time. There are algorithms that are able to achieve accurate results and can operate in real time. Such algorithms perform computations on powerful GPUs. A portable stereo camera suitable for real-time AT applications should use an efficient stereo matching algorithm capable of running on a low power embedded computer. A short summary of the working principles of each algorithm is provided in Table 2.3.

2.5.3. Object detection in 3D images

Object detection is a process of finding the type and 3D position (also called *3D pose*) of an object in 3D point cloud data. The object type is determined by using the object model that is either obtained from object observations or from 3D computer aided design (CAD) models. Object detection is performed by first extracting scene and model features. 3D features can be divided into 2 classes, namely, local and global.

Local features are usually computed on a set of key-points extracted from the original point cloud. Key-points are used to reduce the computation effort required to calculate the features. The most popular method for key-point selection is the point cloud down-sampling by using a voxel grid filter. One example of local 3D features is Fast point feature histograms [Rusu et al., 2009]. Local features are sensitive to the noise and viewpoint, but they can be used to estimate the object pose.

Global features, such as the Viewpoint Feature Histogram [Rusu et al., 2010], are less sensitive to noise, missing information and occlusions. This descriptor is viewpoint-specific and the object model has to be captured from many viewpoints. After object identification, its accurate pose can be computed by using the Iterative closest point algorithm. One variant of this algorithm is presented in [Segal et al., 2009].

Another approach that achieves good results for untextured objects is presented in [Hinterstoisser et al., 2011]. This method also uses object model views generated from different viewpoints. The matching is then performed by running the efficient sliding window approach.

2.6. Interaction

The majority of AT discussed so far are used for communication and HCI. There are also AT that can be used to interact and change the user's environment. Interaction AT can be divided into two categories, namely, passive and active. Passive interaction AT are the devices that can be controlled by using AT. For example, remotely controlled lights and home appliances can be categorized as passive interaction AT. Intelligent home environments can also be used to create passive interaction AT. One example of such a system is described in [Liang et al., 2008].

Active interaction AT are devices that can be used to manipulate and change the environment. Robotics arms and electric wheelchairs are the most commonly used examples of such AT. Active interaction devices are more useful because they give more autonomy to the users. Intuitive device control, however, is still challenging. Connecting an active interaction device with an HCI device does not create a natural user experience. More recent research demonstrates that additional feedback incorporation can greatly improve the control task [Marasco et al., 2018].

2.7. Analysis summary

This chapter reviews the state-of-the-art AT. The majority of this chapter is dedicated to studying HCI methods that are used in AT applications. The reviewed topics have been split into five categories, namely, assistive devices, information presentation, information processing, environment sensing, and interaction. These five topics represent different layers that can be combined together to form multifunctional UI helping disabled people to communicate. These AT layers are visualized in Fig. 2.3.

The HCI are divided into two horizontal layers, namely, Perception and Action. The Action layer transforms the user's intentions into environment or system state changes, while the Perception layer is used to understand the environment and convey the system internal state to the system user. Additionally, five vertical layers are

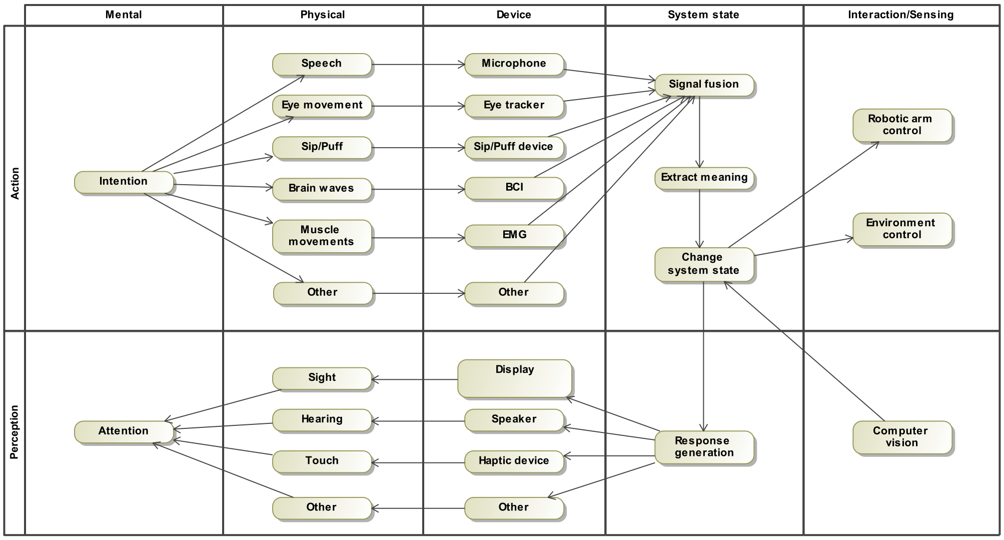


Figure 2.3. Multifunctional AT system layers.

presented, namely, Mental, Physical, Device, System state, and Interaction/Sensing. The Mental layer describes what the user is thinking while interacting with AT. The Physical layer lists the senses and abilities that are used for HCI. User generated actions are processed by using various assistive devices. Another set of devices is used to generate feedback for the user. The focus of this thesis is to use inexpensive devices to create novel HCI methods. One way to achieve this is to use multiple assistive devices simultaneously.

The system state layer represents the AT application. Here, input device signals are handled and processed. This layer also generates GUI that is presented to the user. This layer can also contain prediction algorithms that help the user to use the system more efficiently. One such example is the text prediction algorithm presented in this thesis. AT can be made more useful by adding Interaction and sensing capabilities. Interaction devices empower a quadriplegic user to directly change their environment. Sensors, on the other hand, are used so that the AT system would become aware of the environment. In general, any RGB-D sensor is suitable to capture detailed environment representation. This information can then be used for automatic object recognition. Object recognition adds new capabilities to multifunctional UI and is also an integral part of the proposed projection mapping UI. This thesis will focus on stereo vision based cameras because of their advantages.

3. ARCHITECTURE OF MULTIFUNCTIONAL USER INTERFACE

In this chapter, we will discuss the architecture of multifunctional UI presented in this thesis. The UI was designed taking system reconfigurability and extendability into account. Each of these aspects makes the system applicable for different cases of use. Each aspect is discussed in detail below:

1. **Reconfigurability** makes it possible to change the behavior of the system by modifying the system configuration files. A reconfigurable system can be adapted to the specific needs of each individual user. Such a system can be used by people with different disabilities.

2. **Extensibility** is a system design principle that takes the future system growth into consideration. Extensible systems are expected to be denoted by low extension implementation efforts. In some situations, system extensions can be implemented by using simple scripting languages and additional configuration files.

The following sections will discuss the detailed architecture of system components. Moreover, the intended system use cases are discussed in Section 3.2.

3.1. Multifunctional user interface requirements

The multifunctional User interface (UI) presented in this thesis has been developed mainly according to the requirements gathered for the *Quadribot* project. The following requirements have been used to develop the proposed system:

1. The system should integrate multiple assistive devices.
   (a) The system should integrate EyeTribe eye tracking device.
   (b) The system should integrate Sip/Puff device.
   (c) The system should integrate MindWave BCI device.
   (d) The system should integrate speech recognition technology as an input device.

2. It should be possible to use multiple assistive devices simultaneously.

3. The currently used assistive device should be selected by using system configuration functionality.

4. The system should contain a Yes/No question answering UI.

5. The user should be able to input text by using the system.

6. The text predictor should be used to increase text input efficiency.

7. The system should contain an object selection UI.

8. The system should have functionality to automatically detect known table-top objects by using an RGB-D camera.

9. The system should contain automatic projection mapping user interface for highlighting the detected objects.

10. There should be an item selection UI integrated into the system.

11. It should be possible to generate custom item selection lists.

12. The system should be able to communicate with the outside environment when items in the list are selected.

13. The system should generate log files. These files should contain timestamped system usage information.

14. It should be possible to configure the system and change its functionality without recompiling the application.

15. The system should have well-defined interfaces that can be used to extend system capabilities and integrate new devices.

These requirements have influenced the developed system architecture. As already mentioned, a top priority was given to the system extensibility and reconfigurability. The best practices of system architecture were used to achieve these goals.

3.2. Multifunctional user interface use cases

The proposed system has three user types. Each user type has different system use cases. The use case diagram of the system is provided in Fig. 3.1. User types are discussed in detail below:

1. **Regular** user is a disabled person using the system for assistive purposes. Relevant use cases depend on the disability of the individual user.

2. **Caregiver** is a person who communicates and assists the Regular user. This user is usually an observer of the system. In some situations, the Caregiver might also directly interact with the system.

3. **Administrator** is responsible for setting the system up. System reconfiguration will mainly be performed by the system administrator. The administrator will also have some system extension capabilities.

A detailed description of all system use cases is as follows:

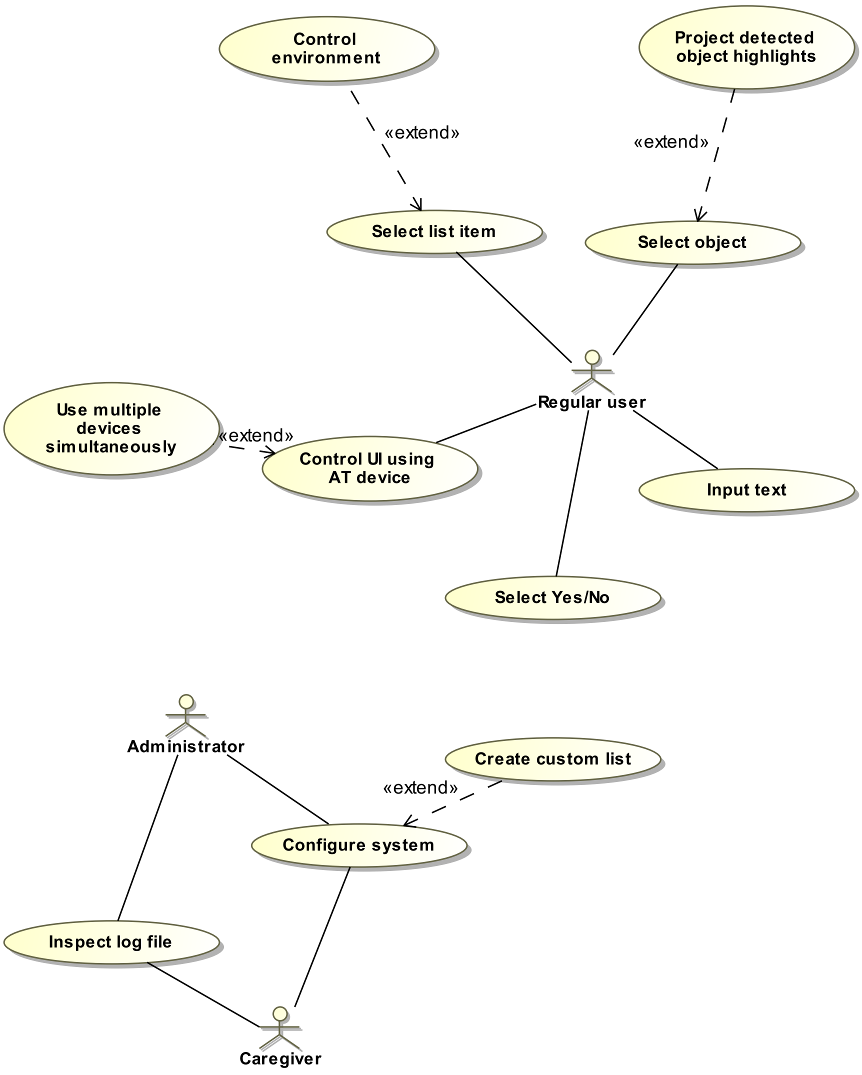


Figure 3.1. The use case diagram of the system.

1. **Control UI using AT device** use case defines that the created multifunctional UI is controlled by using one AT device connected to the system. Each device uses a different user ability, but they all produce identical GUI control signals. Any device should be able to produce at least one control signal.

2. **Use multiple devices simultaneously** is the extension of **Control UI using AT device** use case. Here, more than one device can be used simultaneously. Such an approach helps the user control the UI more efficiently.

3. **Input text** is a use case that is most relevant for people that cannot speak. In this case, the text input is used for communication. This case of use can be extended with additional text to the speech component. This functionality can also be used for document writing.

4. **Select list item** use case makes it possible to select an item from a predefined list. The list can either be text or image-based. Various lists can be created by using the system configuration application. Each list can have a variable amount of items.

   List item selection is a very generic use case and can be used in various situations. For example, there could be a list that helps to identify the user's health condition. In this case, a list could be an image of a human with various body parts indicated as separate list items.

5. **Control environment** use case is an extension of the **Select list item** use case. Here, each list item can be connected to an external actuator that can change the user's environment. For example, a list can be used for wheelchair control where each list item would be a movement direction. Moreover, each list item can be connected to a different external trigger. In this case, each list item could be used to control a different environment device, such as turning lights on and off.

6. **Select Yes or No** is a use case that lets the user answer Yes or No questions. The user can answer several consecutive questions after the answer has been accepted.

7. **Select object** is a more advanced use case. The RGB-D camera is used to automatically detect objects that are placed on a table top in front of the user. In this use case, the detected objects can be displayed in the list similar to the Select list item use case. The user can then select one of the objects.

   There are two ways to use the selection information. First, the selected item information can be indicated to the caregiver. In this case, the caregiver knows that the user wants a particular object. Second, the selection information could be provided to the robotic arm controller. The robotic arm can then manipulate this object. For example, a robotic arm can bring a glass of water close to the user's mouth so that s/he could drink. Using a robotic arm is more complicated, but it gives a greater autonomy to the user.

8. **Project detected object highlights** is an extension of the **Select object** use case. Here, the detected objects are highlighted directly in the scene by using projection mapping. The main advantage of projection mapping UI is the natural HCI. Moreover, the projected highlights are visible to the caregiver. The conventional UI use displays to present information to the user. When using a display, the user has to switch his/her attention from monitor to scene.

9. **Inspect log file** is the use case that is intended for caregivers and system administrators. This use case makes it possible to inspect the actions that were performed while the system was used. Each action is recorded in the log file with the exact time stamp. This information can be subsequently used to analyze the system usage.

10. **Configure system** use case is necessary so that the system could be personalized for each individual user. This use case makes it possible to change various system parameters. The functionality will be implemented as a separate application. This use case will be used by either a caregiver or a system administrator.

11. **Create custom list** is an advanced extension of the system configuration use case. It can be used to create custom lists that are later used in the list selection use case. There is also a possibility to create a hierarchical list where, after selecting an item in the first list, a corresponding second list is displayed. Such lists can be used instead of questionnaires and Yes/No answers by the user.

3.3. System deployment description

The created system contains a number of artifacts implementing system use cases. Some artifacts also communicate with external devices. A detailed system deployment deployment diagram is presented in Fig. 3.2.

The system is deployed on a single computer. The developed applications and libraries are cross platform compatible and can therefore be deployed on either Linux or Windows operating systems. The Quadribot system consists of executable applications and dynamically loaded libraries. Each dynamically loaded library implements a particular system plugin. This plugin-based system architecture helps to satisfy the extensibility and reconfigurability requirements.

The Quadribot system contains the following two executable artifacts:

1. Settings executable that is used to configure all system parameters. These parameters include the main application style and layout settings. Another set of parameters define the plugins that should be loaded into the system. Each plugin also contains its own configuration, for example, the used device port.

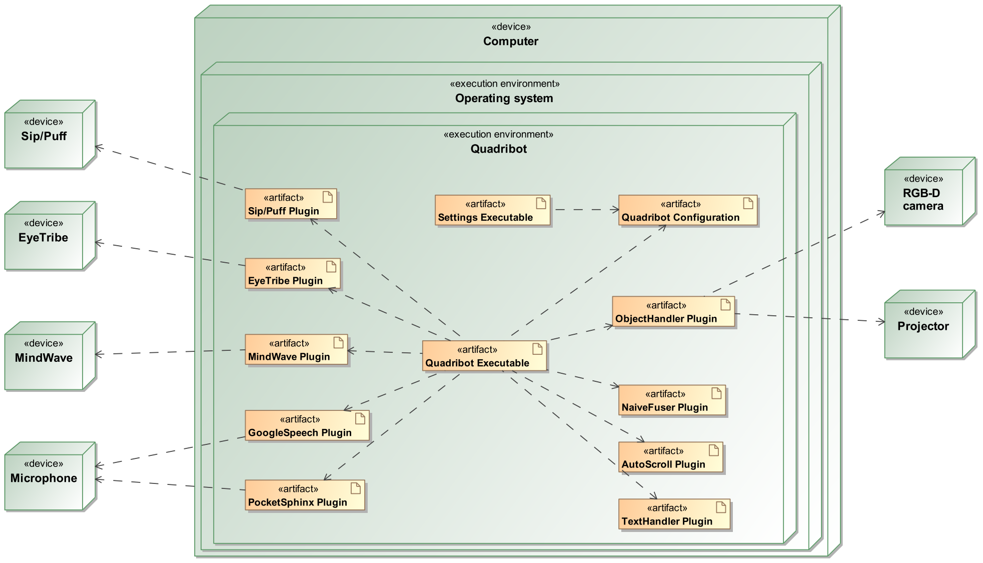


Figure 3.2. The system deployment diagram.

Settings applications store the current system configuration into a configuration file used by the main application.

2. Quadribot executable is the main system application that is started by the regular system user. On startup, this application first reads the system configuration file and loads the configured system plugins. Some system plugins are mandatory, such as the Fuser plugin, others are optional and loaded only if configured. The executable then loads and displays the configured GUI screen to the user.

Plugin architecture used in our system makes it possible to easily extend and change the Quadribot application functionality. Each plugin is deployed as a separate dynamically loaded library. The list of the deployed plugin is as follows:

1. Sip/Puff plugin handles the communication with any Sip/Puff device attached to the computer. The device is expected to support the Human interface device HID protocol.

2. EyeTribe plugin communicates with the EyeTribe device and generated GUI control commands.

3. MindWave plugin implements an interface between the computer and the BCI device from NeuroSky.

4. GoogleSpeech plugin records sounds from the computer microphone and converts them into GUI control commands by using the Google Speech Application programming interface (API) library.

5. PocketSphinx plugin records sounds from the computer microphone and converts them into GUI control commands by using the PocketSphinx library.
6. ObjectHandler plugin implements a projection mapping-based user interface. This artifact communicates with the RGB-D camera and the projector. Projection mapping-based UI is discussed in detail in Section 4.1.
7. NaiveFuser plugin is currently the only Fuser implementation deployed with the system. It implements a simple strategy to propagate GUI control commands from multiple devices running at the same time.
8. AutoScroll plugin implements a virtual device that automatically generates GUI control commands at predefined time intervals.
9. TextHandler plugin implements the functionality of predicting the next word from the text snippet that the user has already entered into the system. This plugin can increase the text input efficiency.

The main executable is able to load one or more device communication plugins. Only one Fuser plugin is loaded at once, but multiple Fuser plugins can be implemented and deployed in a system. This approach makes it possible to easily change the main application behavior. The current Fuser implementation is chosen by using the Settings application.

3.4. Multifunctional user interface architecture

This section describes the software architecture of multifunctional UI. The UI application was implemented by using open source Qt framework. The system has been divided into several components to make it modular. Each component has a well-defined interface that is used for component-to-component communication. Every component can have several different implementations adhering to the same interface. More than one component implementation instance can be started simultaneously. The component instances are loaded dynamically by using the Abstract factory design patter [Wolfgang, 1994]. The list of loaded component instances can be specified by using the system configuration application. The life-cycle of the loaded component instances is handled by a separate component. This architecture ensures the system reconfigurability requirement that has been defined as a system design goal. Figure 3.3. shows the system components, their interfaces and component dependencies.

The presented architecture has been designed by using the Model-View programming paradigm [Vlissides et al., 1995]. This paradigm encourages separation of the GUI components (i.e., View) from the business logic and data handling components (i.e. Model). The Model-View paradigm makes it possible to have several different views for one model. The components developed by using this paradigm have minimal coupling. In addition, the component decoupling is ensured by using Qt Signal-Slot paradigm for object communication. The Qt Signal-Slot paradigm is a safer alternative of the function pointer communication method.

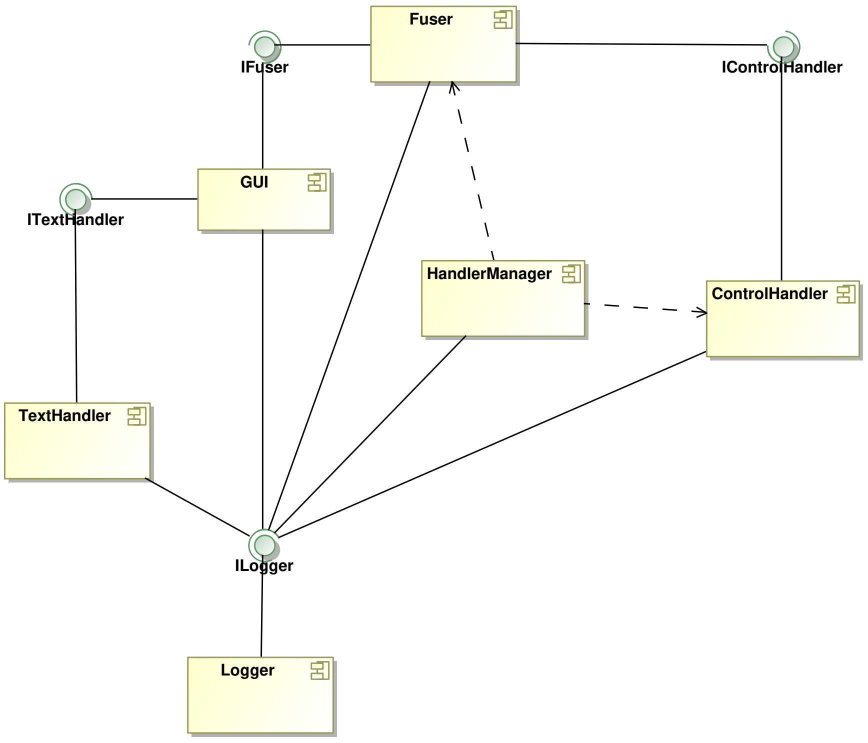


Figure 3.3. The multifunctional user interface system component diagram.

The detailed description of all the system components is provided below:

1. The **ControlHandler** is a component that converts assistive device signals into GUI control commands. This component will have one implementation per input device. Assistive device signal processing algorithms will be implemented in this component. Some assistive devices can directly generate GUI control commands. In such cases, the Control handler is used to forward the generated commands to the GUI.

2. The **Fuser** component can fuse GUI control commands from multiple Control handler instances. The Fuser component can have several implementations for different fusing algorithms. Only one Fuser instance is created during system operation. The desired Fuser instance is set in the application configuration files and loaded during the system start up.

3. The **GUI** component contains the different views that are presented to the user. This component has been implemented by using the Qt Modeling Language QML. QML is a declarative markup language used in the Qt framework for UI development. The system architecture makes it possible to add new QML-based GUI views to the system without the need to recompile the application. New views can be added and used only by changing application configuration files.

4. The **HandlerManager** component is responsible for creating and storing all the Control handler component instances. Moreover, the Fuser component is also created and stored in the HandlerManager. The list of ControlHandler component instances and Fuser component types is loaded from the application configuration files.

   After the creation, each ControlHandler instance is connected to the Fuser component inputs. Fuser outputs are also connected to the current screen from the GUI component.

5. The **TextHandler** component contains text prediction functionality. This component is used to improve the efficiency of the text input use case. This component receives a string representing the text that has already been entered by the user. This string is used to predict the next word that the user wants to enter. The predictor returns a list of words that the user is likely to enter next. This prediction list is displayed to the user. The user can then select one of the predicted words instead of entering all the letters of that word. The prediction list is updated each time when the user enters a new character.

6. The **Logger** component is used to store all the application events and actions in a log file. Each log file entry contains an accurate time stamp. Log file data can be used to examine the system usage patterns as well as unexpected program

crashes. Only a single instance of the logging component exists per application. Each log event also contains an assigned logging level. This component can be configured to only log events of a certain logging level.

All of the system components interact by using well defined interfaces. Different implementations of one component always adhere to the same interface to make the implementations interchangeable. System interfaces are defined below:

1. The **IControlHandler** is an interface implemented by the assistive device handlers. This interface describes the following GUI control command signals:

   (a) **moveForward()** this signal informs GUI that the next list item should be selected. After reaching the end of the list, this command transitions the current item to the first item in the list.

   (b) **moveBackward()** this signal is identical to the **moveForward()** signal but traverses the list in the opposite direction.

   (c) **selectCurrent()** signal is used to select the current item in the list. The selection usually triggers some GUI actions.

   (d) **cursorCoordinates(Point pt)** is a special signal used by eye tracking devices. It contains the two-dimensional screen coordinates. This signal changes the position of the cursor in the GUI. The list item that is located in the *pt* coordinate is automatically made the current item. Navigating the list by using this signal is more efficient because the user does not have to go through the list items one by one.

   We should note that most assistive devices cannot generate all of the control signals. Only the supported signals are generated in this case. This interface is sufficient to support all of the assistive devices currently integrated into the system. In the future, more devices could be integrated into the system. New devices might be able to generate additional GUI control signals. The **IControlHandler** would then have to be extended to support these signals. These signals could, for example, be **moveUp()** or **moveDown()**.

2. The **IFuser** interface is identical to the **IControlHandler** interface. Interfaces are separated to indicate the usage of different deriving objects. Objects deriving from this interface are loaded differently by the **HandlerManager** component. During the creation process, **IFuser** objects are first connected to all the loaded **IControlHandler** instances. The fused signals are then connected to the GUI component.

3. The **ITextHandler** interface is used by text prediction algorithms. This interface has one input slot and one output signal. The **onTextChanged(String text)** slot is invoked by the GUI. The **text** parameter contains the text that has

already been entered by the user. This text changes each time the user enters a new character or when one of the predictions have been selected by the user.

The supplied text is used to generate a list of predictions. The predictions list is supplied to the GUI component by using a **newPredictions(StringList predictions)** signal. The list of predictions is then shown to the user.

4. The **ILogger** interface contains only one method, namely, **loggerCallback(Type type, String message)**. This method is called by all the components that want to log either debugging messages or other information onto the generated log files. The provided type parameter can be used to generate filtered log files.

3.5. Component implementation details

Each system component is implemented by one or more classes. Component classes either strictly adhere to the component interface or are only used internally, withing component boundaries. Components might also have multiple implementations. For example, **ControlHandler** has different implementations for each of the assistive devices. The system class diagram is shown in Fig. 3.4. The implementation details of important classes is provided in the following sections.

3.5.1. Sip/Puff class

This class handles commands generated by using Sip/Puff devices. This class can also handle other devices that support the HID protocol. The supported Sip/Puff device generates left/right mouse button click events. These events are captured by this class and converted into GUI control commands. To make this class more versatile, a simple state machine has been implemented so that this class could handle multiple click actions. For example, a double mouse click can be mapped to a new GUI command. Figure 3.5. shows the diagram of the state machine implemented in the Sip/Puff class.

The state machine contains three states. The SipState and the PuffState states count the number of Sip and Puff commands, respectively. These states are identical and only differ in that they count different commands. There are two criteria for exiting these states. First, if the current state is the SipState, and the Puff command is received and vice versa. Second, the state internal timer elapses. This timer elapses only when no new commands have been received for a configurable period of time. When leaving the SipState or PuffState, an onExit function of the state is called. The current count of the state is used to determine if any GUI commands should be generated. The onExit function also resets the accumulated counts. The NeutralState state is activated when no commands have been received from the Sip/Puff devices for a configurable amount of time.

This state machine makes it possible to create more than two GUI control commands with the Sip/Puff device that only has two commands. For example, the Puff command can be used to select the current GUI elements. Single Sip can be used to

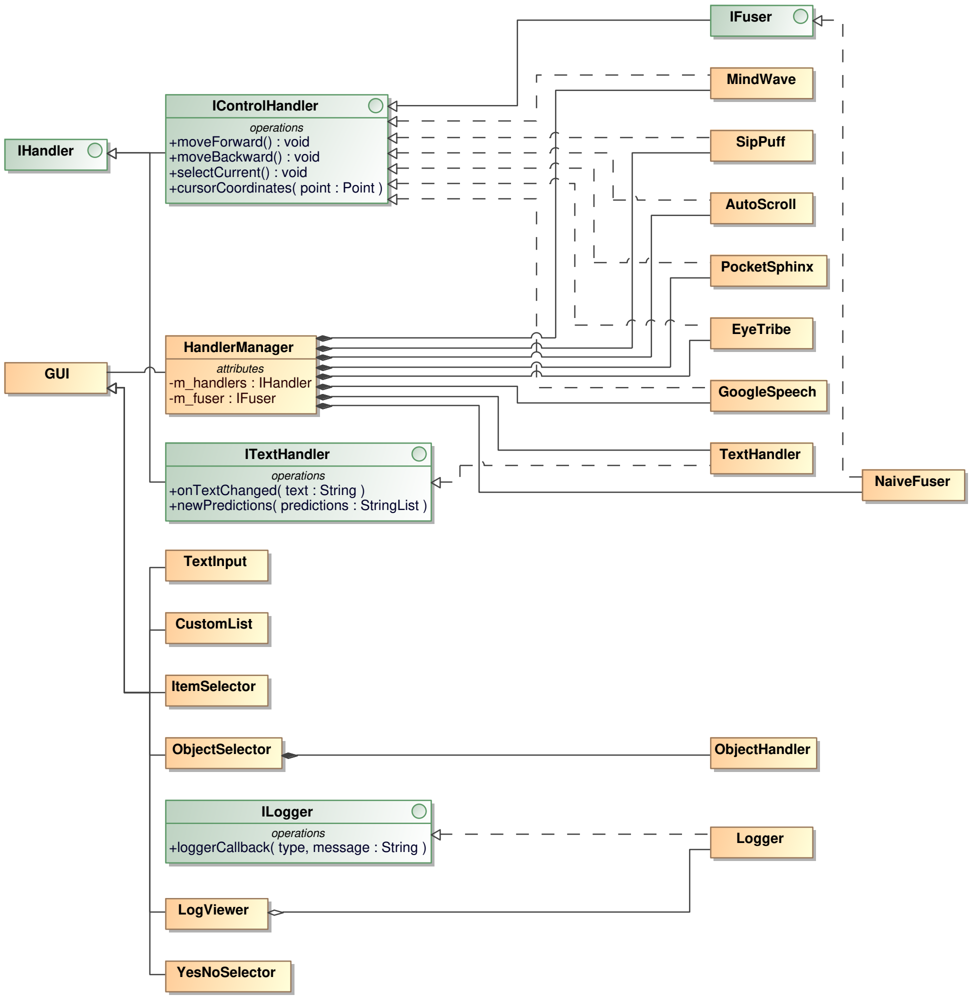


Figure 3.4. The system class diagram.

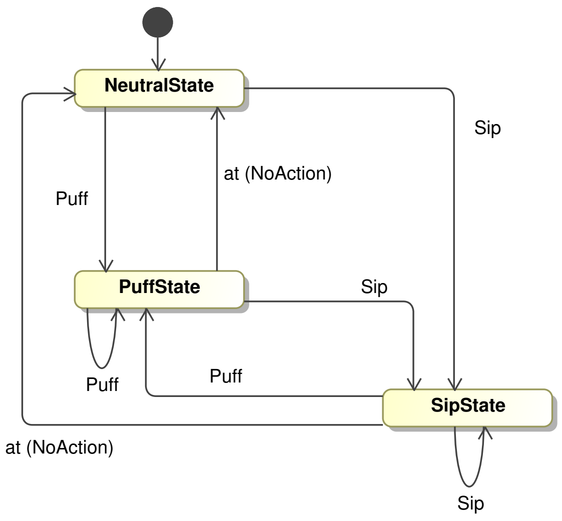


Figure 3.5. The Sip/Puff class state machine diagram.

navigate to next GUI element and double Sip can be used to select the previous GUI element.

3.5.2. MindWave class

This class is used to interface with the MindWave, a brainwave sensing headset from NeuroSky. This class opens a serial port to receive the byte stream from the device. The received byte stream is parsed by using the Parser provided by the device manufacturer. The parsed messages are then handled by the **handleMessage()** method of this class. The device can be used to measure two neurological brain states simultaneously, namely, attention and meditation. Both attention and meditation levels are measured on a scale from 0 to 100. These signals are mapped onto GUI control commands. When either one of the attention or meditation levels exceeds a defined threshold a GUI control command has been generated. The attention level is used to select the current GUI item, and the meditation level is used to move to the next GUI list item. Additional checks are performed to make sure that two or more GUI control commands are not issued simultaneously.

3.5.3. EyeTribe class

The EyeTribe class implements the communication protocol with the EyeTribe eye tracking device. The device exposes its functionality over a proprietary API. The API is accessed by registering a callback function. This function is called each time new gaze data has been calculated by the EyeTribe library. The Gaze data structure contains the eye gaze coordinates in the display coordinate system. The received eye gaze coordinates are translated into **cursorCoordinates(Point pt)** events and are used for the GUI cursor position control.

The empty gaze data structure is returned when no tracking coordinates can be detected. This situation can happen either when the user is not looking into the computer screen or when eyes are not detected in the EyeTribe coordinate system. A short disappearance of the gaze tracking data can be used as an indicator that the user has blinked. The EyeTribe class implements a time based blink detection algorithm. The detected blinks are used as indicators to generate **selectCurrent()** GUI commands.

3.5.4. AutoScroll class

AutoScroll is a specific class that adheres to the **IControlHandler** interface but does not communicate with any assistive or HCI device. The presented UI requires at least two control actions, one for moving to the next element and one for selecting the current element. This action combination can be used to traverse and select items from any list. When the last list item on the list has been reached, the **moveForward()** command would go to the first element on the list i.e., we are using a circular list. As already mentioned, any combination of devices can be used to generate these two mandatory GUI actions. However, there are cases when the user is only able to use a device that generates a single action. In such situations, this action is always used for selection. This class addressed such scenarios by automatically generating list move commands at given time intervals.

The class contains an internal timer. Each time when the timer elapses, a **moveForward()** command is generated. The timer is also reset each time a **selectCurrent()** command is generated by any device. This helps to address the short interval move command after a selection command has been generated. The timer interval of this component can be configured according to the user's needs.

3.5.5. PocketSphinx class

PocketSphinx is an open source speech recognition engine developed by Carnegie Mellon University. It can be used for speech command recognition. This library has been integrated as a virtual assistive device. This class can generate three GUI control commands. Each command is mapped onto one word speech command (also called *keyword*). The keyword to GUI command mapping can be configured according to the user's needs. The speech recognizer library continuously records sounds from the microphone. The recorded sound is divided into utterances by detecting silence-to-speech and speech-to-silence transitions. After a full utterance has been detected, it is processed to generate predictions. When generated predictions match the configured keywords, GUI control commands are generated. The performance of the speech recognizer can be improved by training or adapting a language model. A Speech recognition model training tool has been implemented as part of the proposed assistive GUI.

This implementation of the **IControlHandler** interface is not relevant for communication use cases. Those users who can speak do not require assistive communication UI. Text input might be relevant for those users who can speak, but, in such situations, continuous speech recognition would be more efficient.

3.5.6. GoogleSpeech class

This class implements the functionality that is very similar to the one described in the PocketSphinx class. The main difference of this class is that it uses the Google-Speech API to perform speech recognition. This class also requires a working internet connection and a specific Google Cloud account to operate.

3.5.7. NaiveFuser class

This class implements the **IFuser** interface and is used to propagate GUI control signals from assistive devices to the GUI. NaiveFuser does not perform any additional processing operations on the incoming signals, i.e., each received signal is forwarded to the GUI component. This component is created after **ControlHandler** class instances have been created. After creation, this class is first connected to each **ControlHandler** instance. NaiveFuser is also connected to the current GUI instance. Only one GUI instance is visible to the user at any given time. There is a possibility to change the current GUI instance at the run time. During the GUI instance transition, NaiveFuser is always reconnected to the new GUI instance.

3.5.8. HandlerManager class

HandlerManager instantiates and manages the lifetime of the Fuser and Control-Handler component instances. This class is instantiated during the application start up. During creating, HandlerManager first reads a list of ControlHandler implementations that should be initialized. After all ControlHandlers have been initialized, one instance of the Fuser component is created. The ControlHandler and Fuser types are read from the configuration file. Next Fuser is connected to each ControlHandler instance.

HandlerManager also handles the connection between the Fuser and the current GUI component instance. During the current GUI component change, the Handler-Manager connectGUI() method is called. This method connects the new GUI component with the existing Fuser component.

3.5.9. ObjectHandler class

The ObjectHandler class is connected to the image processing pipeline (described in Section 4.1.2). The image processing pipeline uses the RGB-D camera image to detect table-top objects. The ObjectHandler class receives a list of the currently detected objects. The received objects are converted into a model that is used to visualize the objects in GUI. Two methods are used to visualize the detected objects, namely, displaying objects in a list by using computer display or by projecting the object highlight directly onto the scene by using a projector. The list is updated dynamically each time when a new camera image has been processed.

3.5.10. TextHandler class

This class uses the Presage text prediction library. The text already entered by the user is received from the GUI component. This text is used to generate a list of most likely word predictions. The number of the generated predictions is set in configuration files. The created prediction list is then forwarded back to the GUI component to be displayed to the user. New predictions are generated each time when the user enters a new character or when one of the predicted words has been selected. In the latter case, the selected prediction is used to automatically complete the already entered word. Predictions are also generated when no word letters have been entered. In this case, the prediction list contains the most frequently typed words.

The Presage predictor uses the language model to perform word predictions. The language model is trained before program usage. The language model is also automatically updated when the user uses the program. More information on text prediction is provided in Section 4.4.

3.5.11. Logger class

The Logger class implements program information logging capabilities. Log entries with different types are generated in many system components. Some log entries are used for program debugging while others are used to indicate program state changes. All logging entries are received in the Logger class and saved to the log file with exact timestamps. A generated log file is a convenient data format for analyzing the program usage patterns.

Each log entry also contains a message type. Logging messages received by this component are also forwarded to the GUI and displayed to the user. Only specific type messages are shown to the user. The types of messages that are displayed to the user are configurable.

3.5.12. YesNoSelector class

The YesNoSelector implements a GUI screen used to answer Yes or No questions. The whole screen is divided into two squares, one for the Yes answer, and one for No. The **moveForward()** command moves between these screen options. When the **selectCurrent()** command is called, the current option is displayed on the whole screen and recorded in the log file. The selected option is shown on the screen until any new commands are received. At this point, the screen goes back to the Yes/No selection mode.

This screen is only useful when communicating with humans. Having only two options on this screen improves the answer selection efficiency. This screen therefore does not have any options for the user to transition to other screens. The transition to other screens requires full program restart and can only be performed by the Caregiver. The YesNoSelection screen is most useful when the user has to answer multiple Yes/No questions one after another.

3.5.13. ItemSelector class

The ItemSelector is a generic screen that can be used for item selection and navigation between GUI screens. This screen contains a list of items organized in a grid. Each item in a list has a name, an image and the next screen information assigned to it.

The name and image information of the item is displayed to the user. The next screen information is used to transition to a new screen once that particular item has selected.

The item model is a list of all the currently displayed items. This model is loaded from an Extensible Markup Language (XML) file. The exact ItemSelector model XML file is selected during the screen initialization. This architecture makes it possible to create new lists without modifying the program code and can be performed by the system administrator. This screen is divided into equal size item squares positioned in a grid pattern. It is also possible to create hierarchical lists. For example, the first list could contain types of food such as drinks, snacks, etc. When the user selects drinks, a new list is loaded with types of drinks, such as cola or juice.

3.5.14. ObjectSelector class

The ObjectSelector is a specialized case of ItemSelector screen. This class uses the model provided by the ObjectHandler class instead of loading the model from an XML file. The screen is updated each time the ObjectHandler updates the object list model. This screen makes it possible to select one of the objects. The selected object can be shown to the caregiver. This information can also be sent to external systems, such as the robotic arm controller.

3.5.15. CustomList class

This class is a more generic version of the ItemSelector screen. Items in this list have additional properties, such as the screen position and size. Such items can be positioned anywhere on the screen. In this case, the screen usually also has an image that is rendered below the items as the background. This custom screen can be used to select the image regions. The background image can show a person, and each item can represent a human body part. Such a screen could be used to select an arm and indicate for the caregiver that the user has pain in the arm. This example is illustrated in Fig. 3.6.

3.5.16. TextInput class

The TextInput screen is used for text typing. The screen is divided into two areas. The top area is used to display the already typed text while the bottom area shows the on-screen keyboard. There is an additional one row area at the top of the keyboard for displaying TextHandler predictions. The GUI moving command iterates through both keyboard keys and provided word predictions. When the predicted word has been selected, it is used to automatically complete the current word. The TextInput screen is shown in Fig. 3.7.

The keyboard keys can have various layouts. Currently, the program supports three key layouts, namely, Qwerty, Text on 9 keys (T9) and the Qwerty with Lithuanian letters. Similarly to the ItemSelector, new keyboard layouts can be added with additional configuration files without the need to recompile the program.

The keyboard also has optional keys. Currently, three optional keys are supported. The ‘Clear’ key can be used to remove all the typed text. The ‘Exit’ key can be used

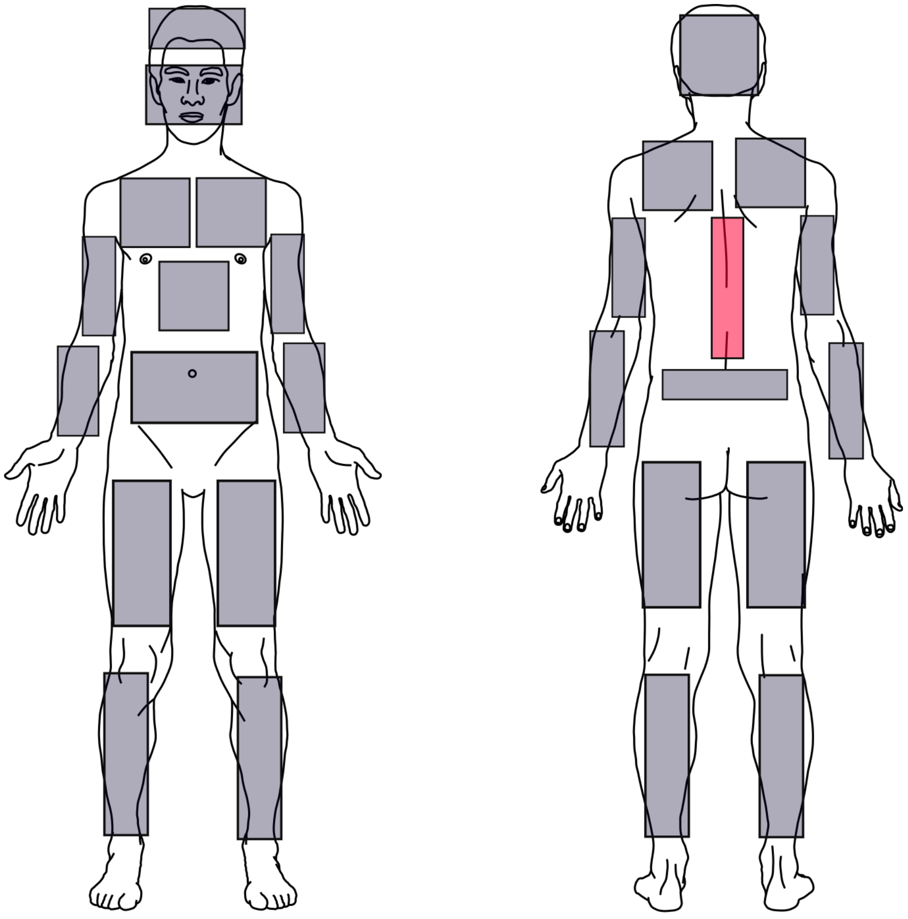
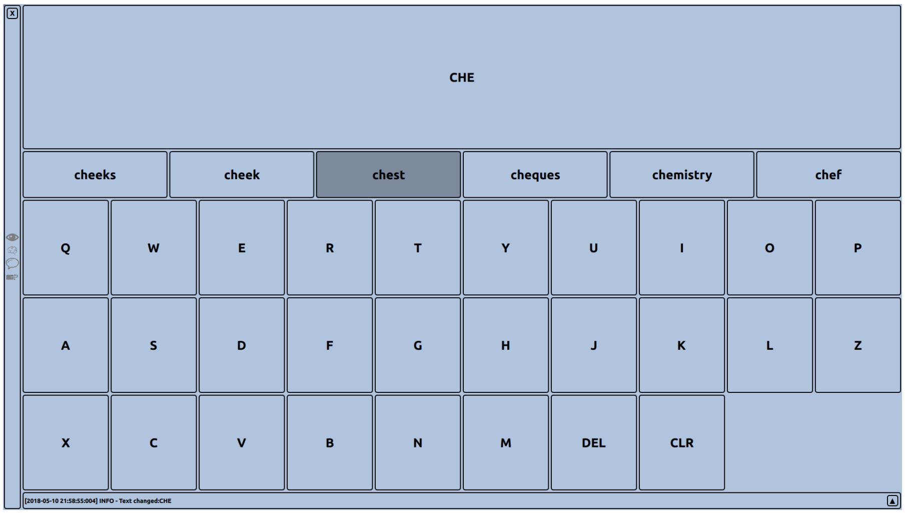


Figure 3.6. An example of CustomList screen with custom item placement. Each gray square represents a different item. The currently selected item is highlighted in red.



```
[2018-05-10 21:58:55:004] INFO - Text changed:CHE
```

Figure 3.7. TextInput screen example. The screen is divided into the entered text area (top), the predictions area (middle) and the keyboard keys (bottom). There are additional status bars at the bottom and left side of the screen.

to navigate from the text input screen to the main ItemSelector menu screen. Delete key makes it possible to erase the last entered text letter. The 'delete' key is usually put at the end of the list. For users that make a lot of incorrect letter choices, there is an additional option making it possible to always reposition the delete key after the last selected letter. With this option, the user has the possibility to delete the letter right after inputting it without needing to go to the end of the key list each time an error has been made.

3.5.17. LogViewer class

This class implements a status bar that can display the latest logging message to the user. The bar can also be extended to display multiple messages if that is necessary. The displayed messages are provided by the Logger class. As already mentioned, only messages that are not filtered by the Logger class will be displayed here. The visibility of this status bar can be changed in the program configuration files.

3.6. Component information flow

The main system components information flow is illustrated in Fig. 3.8. The GUI control commands are generated by assistive devices. These commands are forwarded to the Fuser component. Fused commands are send to the current UI component. Additionally, GUI can receive generated predictions from the text prediction component. The generated UI is presented to the user by using the UI presentation component. This can either be projection mapping interface or conventional computer display.

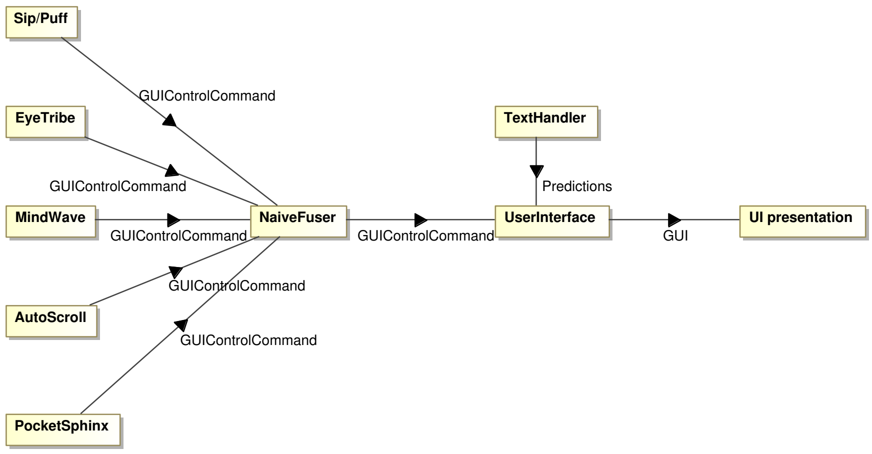


Figure 3.8. Main system components information flow diagram.

3.7. Architecture summary

This chapter presented the detailed architecture of the multifunctional user interface developed for this thesis. The two main principles have been used during the system architecture development, namely, reconfigurability and extensibility. This chapter first details the developed system requirements and use cases, and describes the system deployment. Next, we define the system components and interfaces that are used for information exchange. All the system components have well-defined interfaces. Each system component is dynamically loaded thus ensuring system extensibility. This makes it possible to add a new device and UI screens without the need to change the remaining system code. Lastly, the detailed information of the classes used to implement the components is described, and the system component information flow diagram is presented.

4. PROPOSED ALGORITHMS AND METHODS FOR MULTIFUNCTIONAL USER INTERFACE

This chapter describes the novel algorithms presented in this thesis. The presented algorithms are used to improve the multifunctional user interface for disabled people. The interface is described in more detail in [Gelšvartas et al., 2018]. An overview of the proposed algorithms and their usage in the developed system is illustrated in Fig. 4.1.

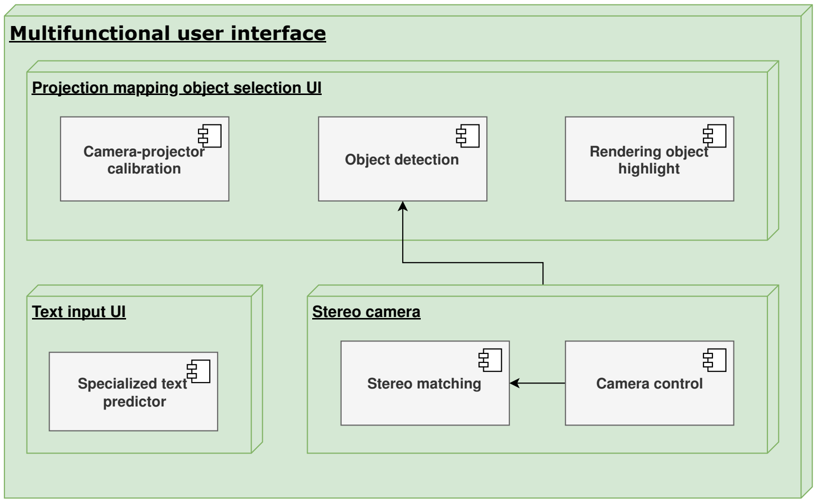

Multifunctional user interface
Projection mapping object selection UI
Camera-projector calibration
Object detection
Rendering object highlight
Text input UI
Specialized text predictor
Stereo camera
Stereo matching
Camera control

Figure 4.1. Proposed algorithm overview and their usage in the developed system.

The developed system contains a projection mapping-based UI used for object selection. This interface offers a natural HCI method because the user sees the detected objects directly in the scene. The projection mapping user interface consists of three parts, namely, camera-projector calibration, object detection and detected object highlight rendering. These algorithms are described in detail in Section 4.1. The main novelty of the proposed projection mapping user interface is that it integrates the object detector and is therefore able to highlight the objects in the scene automatically.

The proposed object detection pipeline can handle data from any RGB-D sensor. Usually, structured light-based sensors are used for capturing RGB-D images. These sensors have limitations, especially in very bright environments. Stereo cameras can also be used to produce RGB-D data, and they are less sensitive to lighting changes. Creating a real-time capable stereo camera is challenging due to the high computational requirements of the stereo matching algorithms. A real-time capable stereo matching algorithm is presented in Section 4.3. The main novelty of this method is its use of

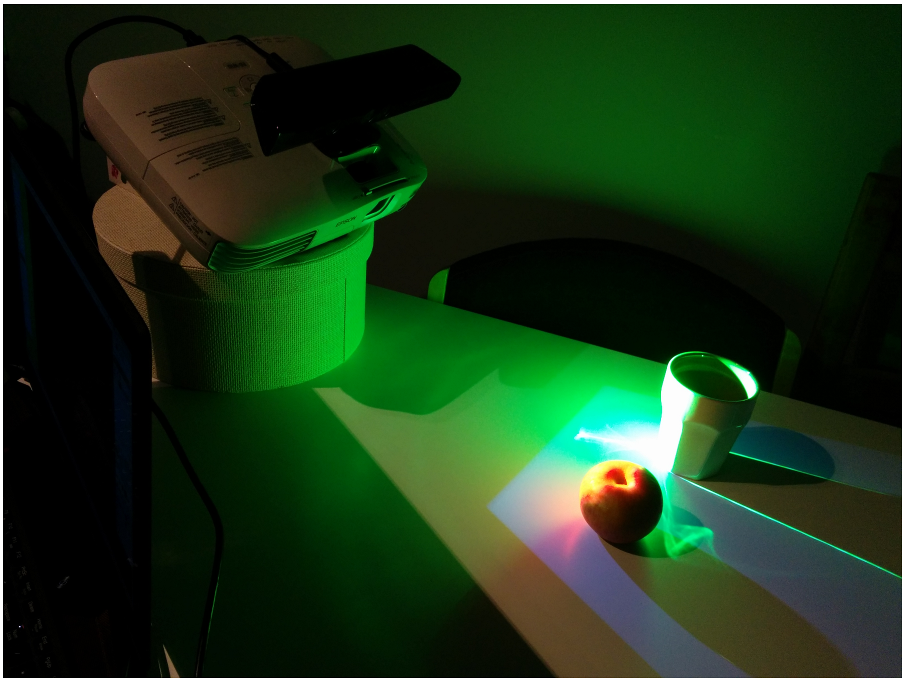


Figure 4.2. Example projection mapping user interface scene setup. The camera-projector system is placed in front of the tabletop scene with objects highlighted in different colors.

region-based features to reduce the stereo matching search space.

The stereo camera robustness is ensured by having a fast camera control algorithm. This algorithms adapts to rapid lighting changes. Detailed camera control algorithm description can be found in Section 4.2. Lastly, we present a specialized text predictor in Section 4.4.

4.1. Projection mapping user interface

In this thesis, we present a novel object selection user interface. This UI works with objects placed on a tabletop in front of the user. The system consists of an RGB-D camera and a conventional projector. The object selection UI highlights the detected objects in the scene by using automatic projection mapping. An example scene highlighted by using automatic projection mapping can be seen in Fig. 4.2.

Projection mapping is a method that is often used in entertainment to display 3D animations on the facades of buildings. This process requires manually calibrating a projector to the specific environment. Automatic object detection in RGB-D images is also a broad field studied by many researchers. The novelty of the projection mapping UI presented in this thesis is to combine the camera-projector calibration procedure presented in [Yang et al., 2016] with the real-time table-top object detection pipeline implemented by using the Point Cloud Library. A combination of these methods makes

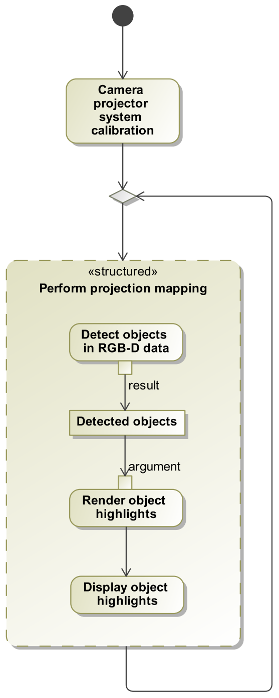

Camera projector system calibration
«structured»
Perform projection mapping
Detect objects in RGB-D data
result
Detected objects
argument
Render object highlights
Display object highlights

Figure 4.3. Activity diagram showing the projection mapping user interface workflow. System calibration is performed only once, while all the other stages are performed continuously during system operation.

it possible to automatically detect 3D objects and highlight them in real time even as their positions are changing.

Automatic projection mapping can be split into three distinct stages. First, the camera-projector system has to be calibrated. In projection mapping, the camera and the projector are often referred to as camera-projector system. In our UI, the RGB-D camera is used instead of a conventional camera. Section 4.1.1. describes the calibration process. Secondly, objects are automatically detected by using an RGB-D camera. The object detection algorithm is described in detail in Section 4.1.2. Objects are detected by using the provided CAD models. These models are also used in the third stage of object highlight rendering and display while using a projector. This process in described in Section 4.1.3. Both second and third steps are performed continuously to create a dynamically changing user interface. The workflow of the projection mapping UI can be seen in Fig. 4.3.

This UI has two main cases of use. First, this UI can be used by a person who is unable to speak but can use an assistive device. The user can select a particular object to inform a caregiver that s/he wants this object. This creates a natural interaction method because both the user and the caregiver see the highlighted object in the scene. Secondly, the selected object could be manipulated by an actuator. This use-case requires an intelligent robotic arm that could manipulate the object when it has been selected. For example, when the user selects a glass of juice, it is grasped by a robotic arm and brought to the user's mouth so that s/he can drink from it.

The main advantage of the proposed object selection UI is natural HCI. The conventional UI use displays to present information to the user. In this case, the user has to switch his/her attention from the monitor to the scene. Another alternative UI presentation method is the use of virtual reality. Severely disabled people have communication barriers, and using a virtual reality headset would further separate the user from the environment. Meanwhile, a projection mapping UI is visible not only for the system user but also for the people nearby.

4.1.1. Camera-projector system calibration

Calibration is a one-time procedure performed during the system setup. This thesis uses the camera-projector calibration procedure described in [Yang et al., 2016]. The calibration process begins by setting up the camera-projector system. The camera and the projector have to be fixed sturdily to each other. Ideally the system should be fixed to a common metal frame or integrated into one case. Camera and projector position or orientation changes with respect to each other; this invalidates calibration, and the process has to be repeated. The camera-projector system position can change after calibration has been finished.

Calibration is performed by placing a small calibration board in front of the camera-projector system. The board has to be stationary for one second before the calibration image has been captured. In each image, the calibration board should be positioned in different locations. The system calibration process is performed once a certain number of images have been captured. Calibration is performed by using a board with a random circular black dot pattern. During calibration, the projector shows a similar white dot pattern that appears on the calibration board. Figure 4.4. shows the calibration procedure. Pattern detection and tracking is performed by using the Local Geometric Consensus algorithm [Yang et al., 2015]. Detected point correspondences are used to estimate the camera-projector system intrinsic and extrinsic parameters.

Both the camera and the projector are modeled by using the pinhole camera model. We should note that while the projector is not a camera, it is still possible to model it by using the same pinhole camera model. In this case, the projector is treated as an 'inverse camera'. The pinhole camera model describes the mathematical relationship between the 3D point coordinates and the point's projection onto an ideal pinhole camera image plane by using perspective transformation (see Eq. 4.1).

$$c = K\,[R|T]\,W \tag{4.1}$$

Here, c is the projected point on the camera image plane, W represents the coordinates of the 3D point in the world coordinate system, and K is the camera matrix. R and T are the rotation matrix and the translation vector that are used to define the

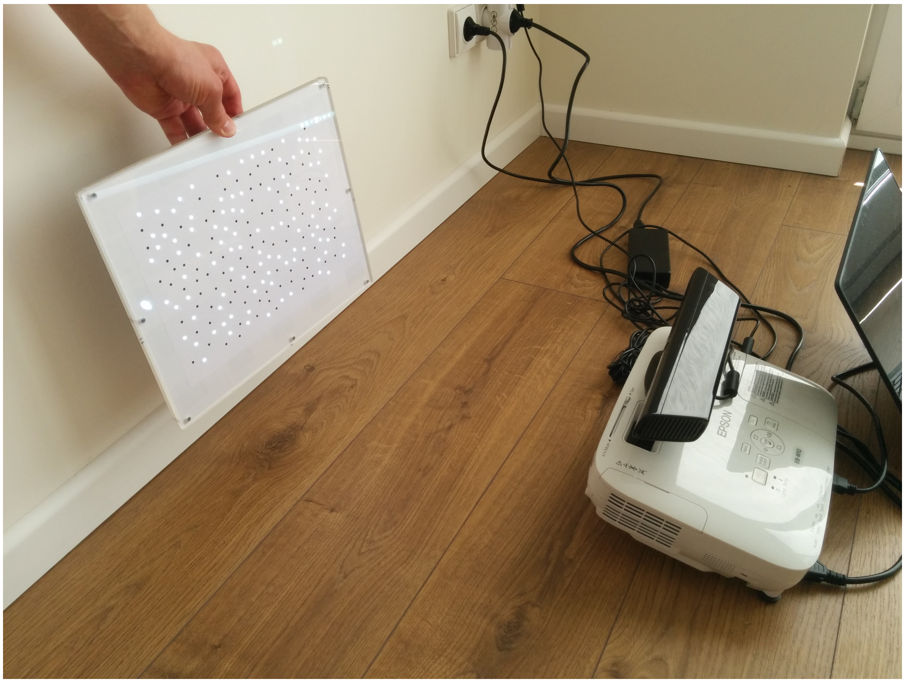


Figure 4.4. Camera-projector system calibration by using a board with a circular black dot pattern. White dots are created by using a projector.

camera motion around a static scene, or vice versa, and the rigid motion of an object in front of a still camera. The above equation can be expanded to

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.2}$$

where f_x and f_y are focal lengths, and (p_x, p_y) is the position of principal point. (X, Y, Z) are the 3D point world coordinates and (u, v) are the projected point coordinates in pixels. Eq. 4.2 is equivalent to

$$\begin{gathered} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \\ x' = x/z \\ y' = y/z \\ u = f_x \times x' + p_x \\ v = f_y \times y' + p_y \end{gathered} \tag{4.3}$$

We should note that Eq. 4.3 can be used only when $z \neq 0$. In-depth description of this camera model can be found in [Hartley and Zisserman, 2003].

This model does not account for geometric distortions caused by the real camera lens or blurring caused by the wide open camera aperture. Moreover, real world cameras only have discrete image coordinates. Therefore, the pinhole camera model is only a first order approximation of the real camera. The simplicity of this camera model makes it very useful in computer vision applications.

Image distortions caused by a camera lens is a major factor contributing to pinhole camera model inaccuracies. These distortions are most noticeable on image edges and are especially significant for the wide field of view (FOV) lenses. Therefore, an additional set of parameters describing the lens distortion are usually estimated during the camera calibration process. The captured camera images can then be undistorted before performing any other computations. Lens distortions are modeled by using distortion coefficient matrix D, namely,

$$D = \begin{pmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{pmatrix} \tag{4.4}$$

where k_1, k_2, k_3 are radial distortion coefficients. p_1 and p_2 are tangential distortion coefficients. Camera matrix K and distortion coefficients D together form the intrinsic camera parameters. A more detailed description of this camera model can be found in [Bradski and Kaehler, 2008]. Distortion coefficients extend Eq. 4.3. to

$$
\begin{aligned}
\begin{bmatrix} x \\ y \\ z \end{bmatrix} &= R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \\
x' &= x/z \\
y' &= y/z \\
x'' &= x'\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2p_1 x'y' + p_2\left(r^2 + 2x'^2\right) \\
y'' &= y'\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_1\left(r^2 + 2y'^2\right) + 2p_2 x'y' \\
r^2 &= x'^2 + y'^2 \\
u &= f_x \times x'' + p_x \\
v &= f_y \times y'' + p_y
\end{aligned}
\tag{4.5}
$$

In the multiple camera setup, R and T from Eq. 4.1. define a transformation from the camera optical origin to the projector optical origin. These parameters are called extrinsic camera-projector system parameters. Both extrinsic and intrinsic camera parameters are estimated during the camera calibration process.

The camera-projector system calibration procedure can be seen in Fig. 4.5. More specifically, the algorithm works as follows:

1. Camera images are continuously recorded to perform circle pattern detection.
2. Detected white and black circle positions are matched with the corresponding circle pattern models. The circle patterns that match the transformed model positions are stored for the camera calibration procedure.
3. Camera calibration is performed by using the [Zhang, 2000] method. A sufficient number of corresponding circles have to be detected before this camera calibration starts. Camera calibration process estimates camera intrinsic parameters i.e. camera matrix K_c and camera distortion coefficients D_c. The estimated D_c are used to create undistorted camera images. This makes projector calibration more accurate.
4. The Board-Camera homography H_{cb} is calculated from the detected circle correspondences. The projected circle locations can then be expressed as $H_{cb}^{-1} y^{(c)}$ where $y^{(c)}$ is the projected circle board viewed from the camera. This creates a new set of correspondences that are used to calibrate the projector, i.e., to obtain projector matrix K_p and projector distortion coefficients D_p.
5. Homography information is also used to estimate the camera-projector system extrinsic parameters, namely, T_{cp} and R_{cp}.

More detailed information about the camera-projector system calibration can be found in [Yang et al., 2016].

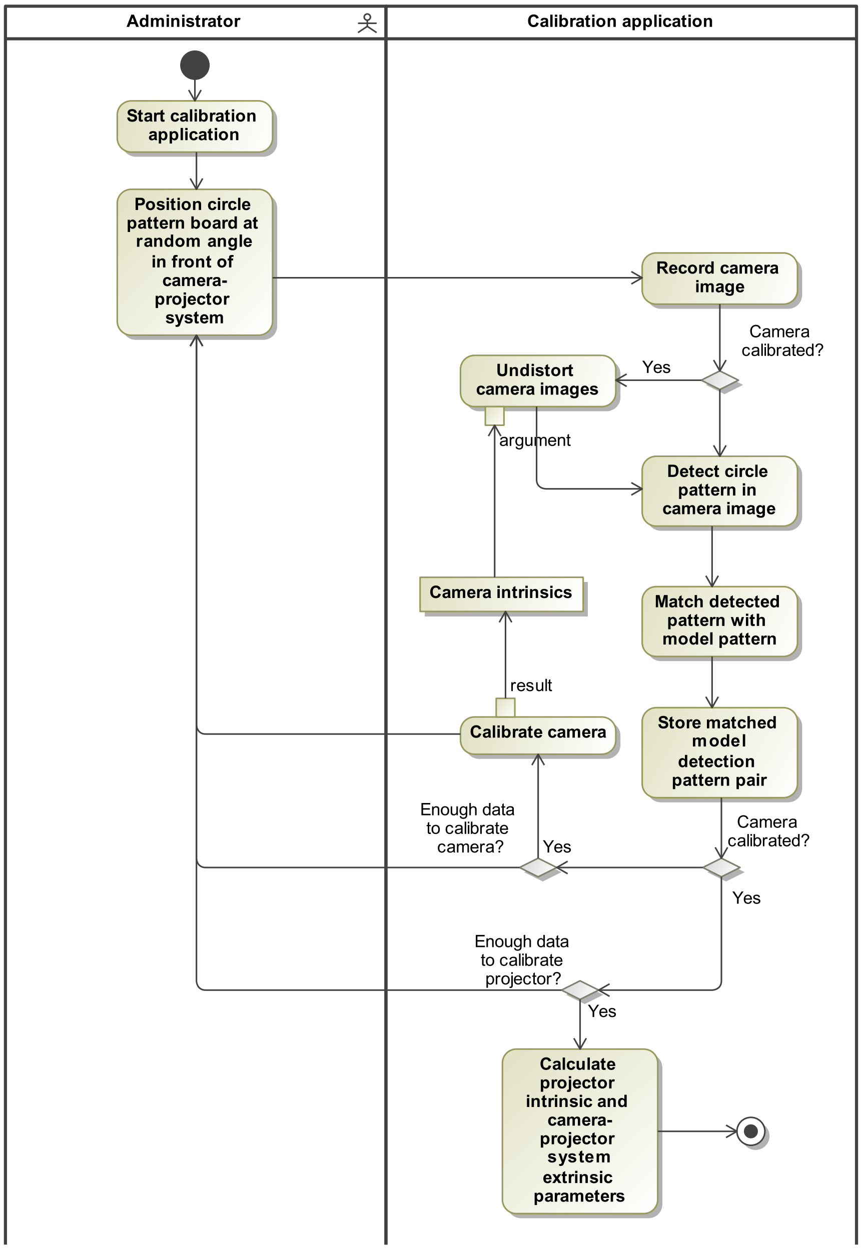

Administrator
Calibration application
Start calibration application
Position circle pattern board at random angle in front of camera-projector system
Record camera image
Camera calibrated?
Yes
Undistort camera images
argument
Detect circle pattern in camera image
Camera intrinsics
Match detected pattern with model pattern
result
Calibrate camera
Store matched model detection pattern pair
Enough data to calibrate camera?
Yes
Camera calibrated?
Yes
Enough data to calibrate projector?
Yes
Calculate projector intrinsic and camera-projector system extrinsic parameters

Figure 4.5. Activity diagram showing the camera-projector system calibration procedure.

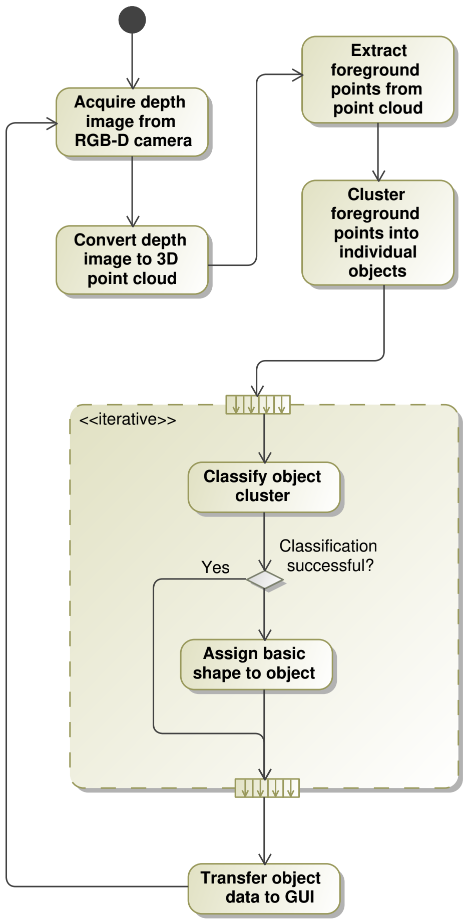

Acquire depth image from RGB-D camera
Convert depth image to 3D point cloud
Extract foreground points from point cloud
Cluster foreground points into individual objects
<<iterative>>
Classify object cluster
Classification successful?
Yes
Assign basic shape to object
Transfer object data to GUI

Figure 4.6. Activity diagram showing the object detection algorithm pipeline.

4.1.2. Object detection pipeline

The object detection pipeline is a sequence of algorithms that are used to detect objects. Each algorithm stage either generates intermittent results, or its output can be used for visualization and algorithm tuning purposes. The object detection algorithm described in this thesis was implemented by combining 3D point cloud processing algorithms implemented in the Point Cloud Library [Rusu and Cousins, 2011]. The object detection algorithm is visualized in Fig. 4.6. Moreover, the pipeline information flow diagram provided in Fig. 4.7. shows the full object detection algorithm pipeline. The object detection pipeline consists of the following stages of the algorithm:

1. The acquisition stage performs sensor data collection and preparation. The output of this stage is a 3D point cloud. Any RGB-D camera can be used for point cloud generation. In this thesis, two types of cameras have been used, namely,

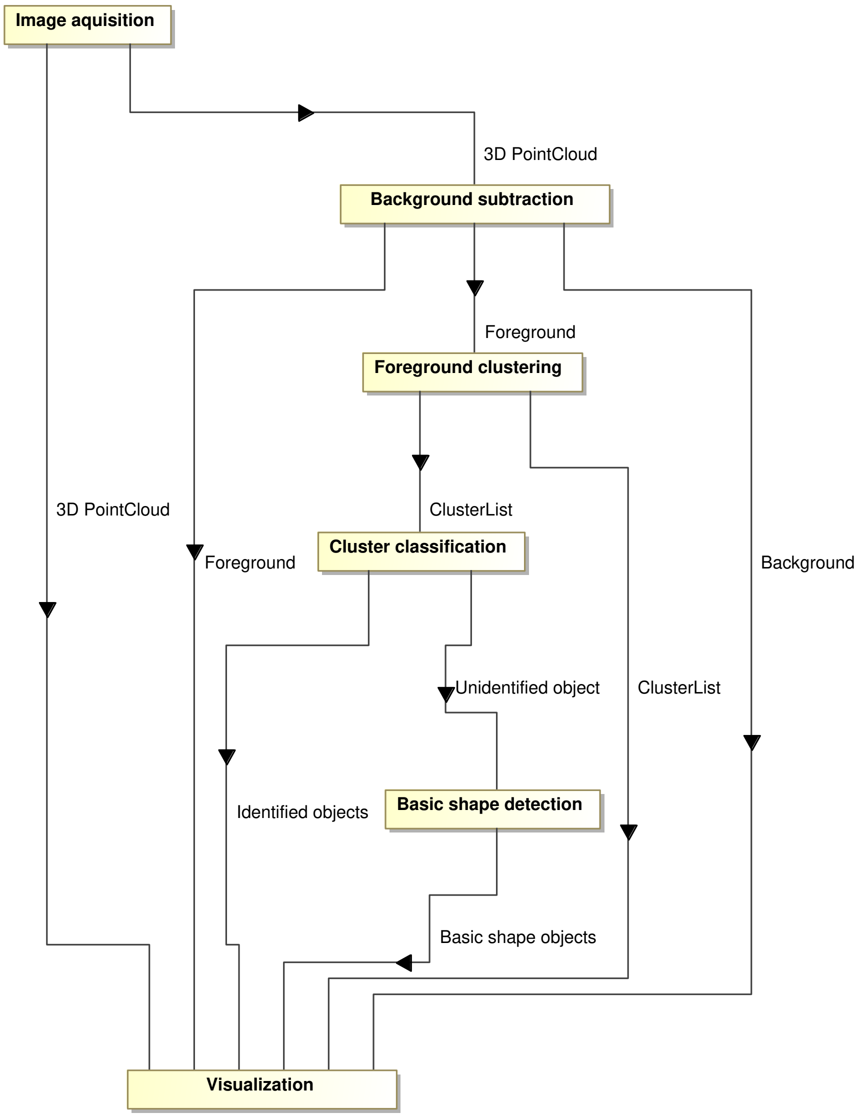


Figure 4.7. Object detection pipeline information flow diagram.

Microsoft Kinect V1 (structured light) and a custom-built stereo camera. The main advantage of structured-light cameras is that they produce depth images internally. This means that their output can be directly used to produce point clouds. The stereo camera output, on the other hand, requires additional processing, namely, image rectification and stereo matching. After these additional stages, stereo camera also produces depth images.

Object detection is performed on a 3D point cloud that is obtained by reprojecting each depth image pixel (u, v, Z) to a 3D point $(X, Y, Z, 1)$. Here, u and v are pixel coordinates along image rows and columns. X and Y are 3D point coordinates along X and Y axes in meters, and Z is the point distance from the camera in meters. This transformation is performed by using intrinsic and extrinsic camera parameters obtained during system calibration.

3D point clouds can be of two types, namely, organized and unorganized. Organized point clouds are generated from depth images. The organized point cloud memory layout resembles a 2D image, i.e., such a point cloud contains width and height. The advantages of an organized point cloud is that it maintains adjacent point relationships. This makes it possible to run some algorithms more efficiently. Unorganized point clouds are stored in a 1D array. Such point clouds are a result of processing algorithms that add or remove some point cloud points.

2. The background subtraction stage divides point cloud points into two point clouds, namely, foreground and background. This process requires some prior environment knowledge. For example, the scene can be set up on a tabletop surface, and the camera position never changes with respect to this surface. In this case, all points belonging to the tabletop surface can be removed from the point cloud. The tabletop surface is considered to be the dominant plane.

   Alternatively, a background model can be created by capturing a background image with no objects. This background model is then used to detect all planar surfaces. A method proposed in [Poppinga et al., 2008] can efficiently estimate planar surfaces in organized point cloud data. During the operation all the scene point cloud points that are close to the detected planes are added to the background point cloud, whereas all the remaining points are considered foreground.

   There are situations when the background model is an unorganized point cloud. This situation happens when the background model is created by fusing several background point clouds. This is useful when the camera is fixed to the end effector of a robotic arm. In this case, planes detected in the fused background model can be used to remove the background from camera observations even when the camera is moving. This algorithm variant requires the current camera pose with respect to the background model. In this case, the background model is an unorganized point cloud, and the previous plane detection algorithm cannot be used. Plane detection on unorganized point clouds is usually performed by using the RANdom SAmple Consensus (RANSAC) algorithm initially proposed in [Fischler and Bolles, 1987].

Further stages of the algorithm are performed only on the foreground point cloud. In practice, only a small portion of points are foreground. This process greatly improves the computation time of the further stages of the algorithm.

3. Foreground clustering. The computation time of the remaining algorithm stages can further be improved by dividing the remaining foreground point cloud into smaller parts. This is achieved by doing Euclidean clustering. A detailed description of this algorithm can be found in [Rusu, 2009]. This stage works well only when certain assumptions can be made about the environment. The algorithm assumes that individual objects will be placed separated from each other, i.e., no two objects will be touching. The algorithm will work with touching objects, but they will be returned as a single cluster. This can affect the computation time of the further stages of the algorithm.

4. Cluster classification. The classification algorithm stage compares each extracted cluster with a list of object models stored in the database. Each cluster is assigned the most probable object type or classified as an unknown object. Unknown objects are objects that are not similar to any of the object models. These objects are assigned a basic shape type in the following stages of algorithms.

   Object classification is performed by first extracting the scene and model features. The Signature of Histograms of Orientations features described in [Tombari et al., 2010] was used in this thesis. A cluster is assigned to a particular object type only when a sufficient number of corresponding features have been detected between the model and the scene. In addition, corresponding features are checked for geometric consistency.

5. Basic shape detection. This stage is a fall-back mechanism used when the classification algorithm fails to assign a cluster to any of the object models. This stage uses RANSAC algorithms to assign the remaining unclassified clusters to one of the two basic object shapes, namely, a cylinder or a sphere. Clusters that are not similar to these basic shapes are assigned a minimal oriented bounding box. These clusters would not have accurate highlights during the projection mapping, but they could still be selected by the user.

4.1.3. Object highlighting by using projection mapping

The detected object models have to be rendered into a 2D image that is shown on the projector. The visualization module of the Point Cloud Library [Rusu and Cousins, 2011] was used to render object highlights. A virtual 3D scene

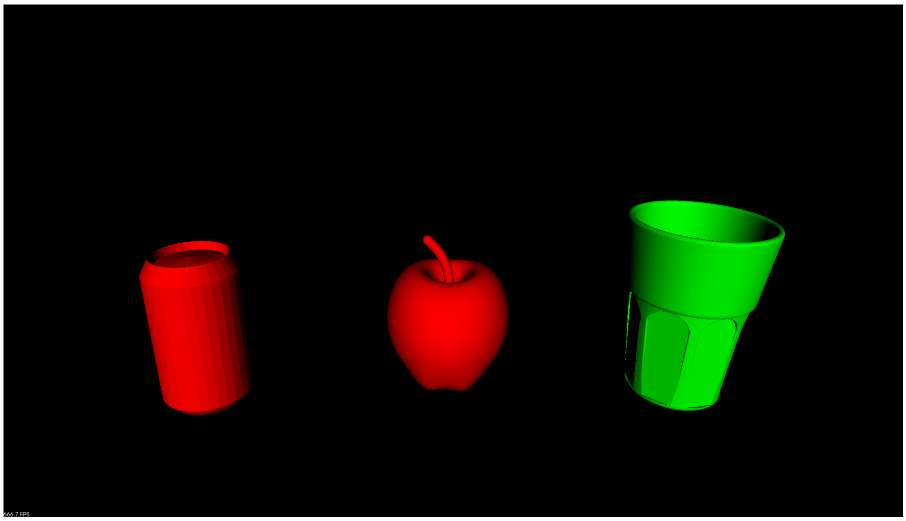


Figure 4.8. Example image of 3D scene with rendered detected object models.

containing a camera is created in a 3D viewer. Camera parameters are chosen such that they match the intrinsic parameters of the projector.

The detected object list is tracked and updated each time a new list of detected objects has been received. Each object is assigned a unique identifier. The index of the current object is also stored so that this object can be highlighted in a different color.

As described in the previous section, objects are detected in point clouds obtained by using an RGB-D camera. The coordinates of these objects are therefore in the camera optical coordinate system. Before rendering, the coordinates of each object are transformed into the projector optical coordinate system. This operation is performed by using the camera-projector system's extrinsic parameters, namely, rotation matrix R_{cp} and translation vector T_{cp}.

Each detected object is inserted into a 3D viewer by using a transformed object's pose coordinates. Inserted object meshes are assigned a uniform color material. When all objects are inserted into the scene, an image generated by a virtual camera is shown on a projector. This image projects object highlights on real scene objects. An example of a rendered 3D scene image can be seen in Fig. 4.8.

4.2. Camera control

This thesis presents a novel camera control algorithm. The algorithm uses a cascade controller to control two camera parameters simultaneously. Moreover, we have performed detailed experiments to study the non-linear nature of the multiple camera parameter control. This non-linearity is addressed by applying controller gain scheduling techniques. The combination of these methods achieves a fast and stable camera parameter controller.

The aim of the algorithm presented in this section is to control camera parameters so that images with good exposure are produced. The image exposure is determined by the amount of light reaching the electronic image sensor. Underexposed images are dark, while overexposed images are very bright. In both cases, some scene information may potentially be lost in either image highlights or shadows. The camera control algorithm should have the following properties to produce well exposed images:

1. Fast. The algorithm should quickly adapt to the changing environment conditions. The lighting condition can sometimes even change instantly, for example, when the lights get turned on or off.
2. Stable. The camera control should be stable, i.e., the produced images should have stable exposure that does not fluctuate between consecutive images. Having images with an oscillating exposure can greatly impact other image processing algorithms.
3. Adaptable. Changing one camera parameter can also affect other camera parameters. The camera control algorithm should dynamically adjust as the camera parameters are changing. Another way that can be used to increase the camera control adaptability is by providing image regions that will be used to measure the current image exposure.

Many internal and external factors affect the exposure of the image produced with a digital camera. An efficient camera control algorithm can be developed by first understanding these factors very well.

4.2.1. Factors affecting image exposure

There are three camera parameters that influence the exposure of the produced images, namely, aperture, shutter speed, and sensor sensitivity. In photography, the relationship between these parameters is often referred to as the exposure triangle. This term signifies that when one parameter is getting changed, another must change to capture the image that has the same exposure. The exposure triangle is illustrated in Fig. 4.9.

There is also a fourth factor influencing the image exposure, namely, scene luminance. Scene luminance can be described as the amount of light that is present in the environment. This factor is not controlled within the camera, unless the camera is equipped with an internal or external flashlight. Scene luminance depends on many things, such as:

1. Amount and intensity of light sources in the environment.
2. Types of surfaces present in the environment.
3. The overall scene geometry.

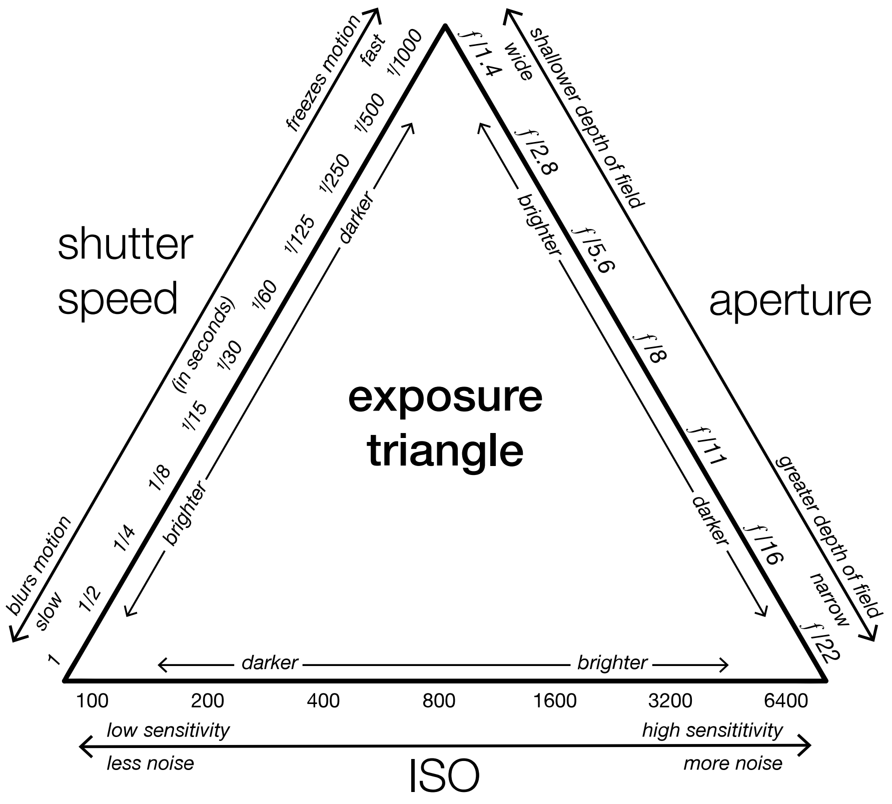


Figure 4.9. Exposure triangle consisting of three camera parameters controlling the produced image exposure. Image created by WClarke [CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0)], from Wikimedia Commons.

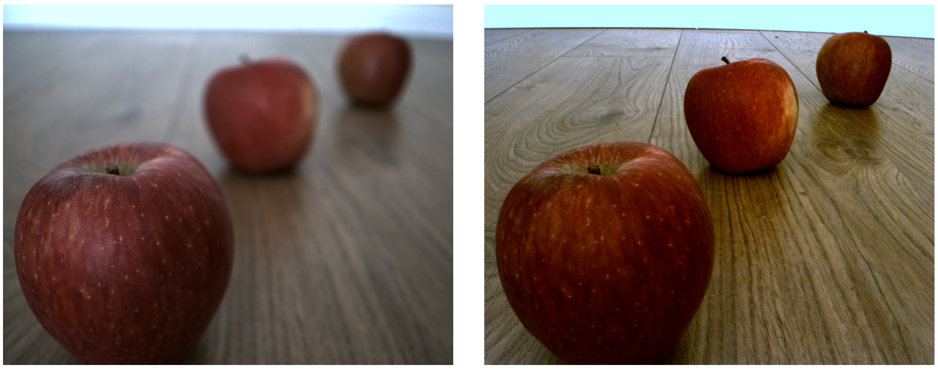

(a) (b)

Figure 4.10. The effects of aperture on the depth of field of the objects. (a) large aperture image with a shallow depth of the field, (b) small aperture image.

Each of the factors described above influence the produced camera image. Each factor can be changed independently in order to study their effects on the image. These effects can be both positive and negative. Choosing good parameter values is always a trade-off that might be use-case specific. Each parameter is discussed in detail below. We should note that in the following paragraphs of this section we shall discuss how a single factor affects the images when all other factors are constant.

As already discussed, the scene luminance is the only factor that is not directly controlled in most situations. In general, it is best to operate the camera in well-lit environments. If possible, the light sources should be distributed evenly throughout the environment. It is best to try to avoid situations where some parts of the image have significantly different lighting. It is difficult to ensure such conditions during very sunny days outside. During such days, there is often significant lighting difference manifested between the sky and the objects that are in the shadows.

The aperture is a variable diameter hole in the camera lens. The aperture can be changed from a large diameter hole to a small diameter hole. More light rays can travel through the camera lens when the aperture is large. A small aperture reduces the amount of light that is passed through the lens. Therefore, a large aperture produces brighter images, while a small aperture produces darker images. The aperture also directly affects the depth of the field. A mall aperture produces a longer depth of the field. This means that objects at a wide range of distances will all be in focus at the same time. Therefore, the aperture is closely related to the focus of the camera lens. This effect is illustrated in Fig. 4.10.

The majority of computer vision cameras used today have lenses with a fixed or mechanically controlled aperture and focus. In this case, the exact aperture and focus values are set during the system setup. The aperture and focus are set depending on the scene lighting and the distance of the observed objects. Some high end cameras have an electronically controlled aperture. The algorithm proposed in this thesis could be

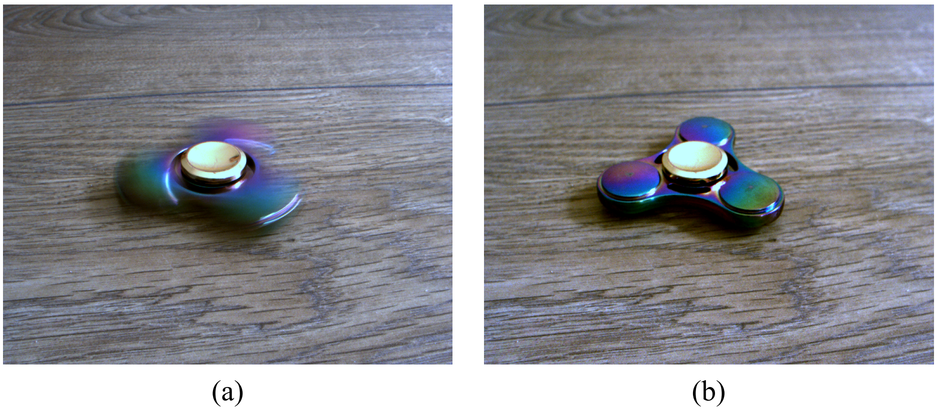


Figure 4.11. The effects of shutter speed on moving objects. (a) long shutter speed image with motion blur, (b) short shutter speed image with no blur.

extended to incorporate the aperture control.

The shutter speed, or the exposure time is the length of time when the camera sensor is exposed to light. A longer shutter speed can be used to create brighter images. The shutter speed also affects how moving objects appear in the produced images. Fast moving objects would be blurred in the images when the shutter speed is long. This is illustrated in Fig. 4.11.

Sensor sensitivity is sometimes also called *gain* or *ISO*. Increasing the signal gain of the digital camera sensor makes it possible to capture brighter images in low light situations. As with other factors, this also has the effect of increasing the noise of the produced images (see Fig. 4.12.).

Each of the discussed factors has advantages and disadvantages. The parameter choice has to be a balance that produces the best quality image.

4.2.2. Control algorithm target

The control algorithm has to measure the current exposure of the image so that parameter values can be adjusted to improve the exposure of future images. This section describes the method used in this thesis. Measuring the image exposure is not a trivial task. Ideally, the exposure measurement should be calculated for each taken image. This implies that measurements should be calculated in real time, and at least as fast as the current camera frame rate. The image histogram is one measurement that can be calculated for each image in real time.

Image histograms is a convenient method to measure the distribution of image pixel intensities. Pixel intensities are directly related to the image exposure. Well-exposed images have a majority of pixels with intensities that are close to the histogram center. Underexposed images have many pixels with low intensity values, while overexposed images have many pixels with high intensity values. We propose to use the image histogram mass center as a way to measure the image exposure. In our case, the histogram

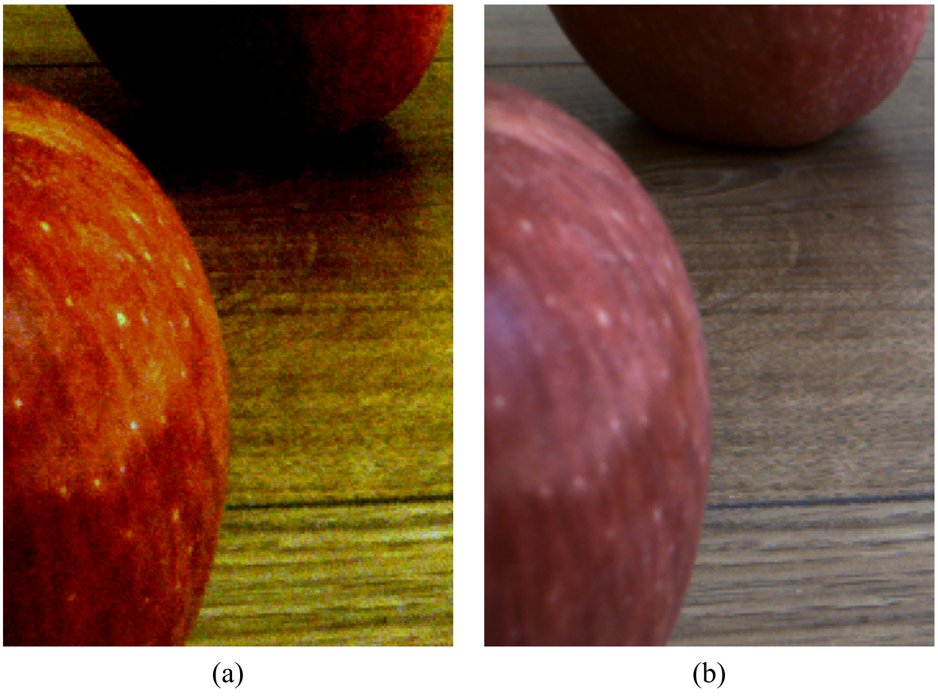


Figure 4.12. Example image captured with high and low camera gain setting. (a) noisy high sensor gain image, (b) low gain image with no noise.

is calculated on 8-bit images and has 256 bins. The histogram center is calculated as follows:

$$H_{ct} = \frac{\sum_{i=0}^{255} i \times H(i)}{\sum_{i=0}^{255} H(i)} \tag{4.6}$$

Here, H_{ct} is the calculated histogram mass center, and function $H(i)$ returns the value of the i-th histogram bin. In this case, the images with H_{ct} value that is close to 128 would be considered as well-exposed.

The proposed approach has one caveat, namely, images with $H_{ct} = 128$ can still be exposed incorrectly. This happens when the camera captures scenes where a half of the image is very bright, and a half is dark. For example, images of the bright sky and the ground in a shade can be exposed incorrectly. This can be partly addressed if such situations are known *a priori*. A histogram-based exposure measurement makes it very easy to incorporate such *a priori* knowledge. The image histogram can be calculated only on some parts of the image. These parts can be specified by providing the image mask. Image mask specification also increases the algorithm adaptability because it can be used to specify interesting ROI to the camera control algorithm. This ROI is usually generated by higher level image processing algorithms.

4.2.3. Algorithm architecture

The camera control algorithm described in this section adjusts the camera parameters so that well-exposed images are created. The image exposure system process and camera parameter relationships are studied in detail in Section 5.4.1. As observed during the experiments, the image exposure system process is mostly linear; therefore, a Proportional Integral Derivative (PID) controller is well-suited to control this process. A conventional PID controller is not able to control multiple output variables simultaneously. This problem has been solved by proposing a novel cascade controller where the output of the first controller is used as an input for the second PID controller.

The controller will adjust two camera parameters, namely, the shutter speed and the sensor gain. Both of these parameters affect the same output process, namely, the image exposure. To create an efficient controller we want to control both parameters simultaneously. To create this controller, we have to select the parameter that is going to be controlled by the inner and outer loops.

The shutter speed is the most important camera parameter. Excessively long shutter speed values can create blurred images. Noisy images created by high sensor gain values usually do not affect image processing algorithms. Moreover, it is possible to reduce the image noise by running image filtering algorithms. The shutter speed is therefore used as a primary control variable controlled by the outer loop. This loop is tuned to be reactive and to quickly adjust to sudden image exposure changes. More formally, the shutter speed control loop error value $e_{ss}(t)$ is calculated as follows:

$$e_{ss}(t) = H_{ct} - H_{target} \tag{4.7}$$

Here, H_{ct} is the calculated histogram mass center of the current camera image, and H_{target} is the desired histogram mass center set point. Typically, it is set to $H_{target} = 128$ for 8-bit images. The shutter speed control function can then be expressed mathematically as:

$$u_{ss}(t) = K_p^{ss} e_{ss}(t) + K_i^{ss} \int_0^t e_{ss}(t')dt' + K_d^{ss} \frac{de_{ss}(t)}{dt} \tag{4.8}$$

Here, $u_{ss}(t)$ is the new shutter speed value that is set on the camera. K_p^{ss}, K_i^{ss} and K_d^{ss} are the proportional, integral and derivative controller coefficients of the shutter speed control loop.

The sensor gain variable is used to slowly adjust to more gradual lighting changes. This loop is tuned to be slower. The current shutter speed value $u_{ss}(t)$ is used as a measured process variable. The shutter speed target can be set as a parameter and should be chosen such that the produced images would have minimal motion blur effects. When the shutter speed controlled by the outer loop exceeds the set threshold, the camera gain will be gradually adjusted so that to reduce the shutter speed. More formally, the gain control loop error value $e_g(t)$ is calculated as follows:

$$e_g(t) = u_{ss}(t) - T_{ss} \tag{4.9}$$

where T_{ss} is the desired shutter speed value. The exact T_{ss} is chosen experimentally such that the noticeable image blur is minimized. This definition leads to the following gain control loop function:

$$u_g(t) = K_p^g e_g(t) + K_i^g \int_0^t e_g(t')dt' + K_d^g \frac{de_g(t)}{dt} \tag{4.10}$$

Here, $u_g(t)$ is the gain value set in the camera. The gain control loop proportional, integral and derivative coefficients are K_p^g, K_i^g and K_d^g, respectively.

As described in Section 5.4.1, the changes in the sensor gain also affect the shutter speed. As the sensor gain is increased, the shutter speed control should become slower. In this thesis, we propose a novel method to increase the efficiency of the cascade camera controller. This is achieved by performing shutter speed control loop gain scheduling. More specifically, the proportional gain component of the shutter speed control loop K_p^{ss} is decreased when the sensor gain increases, i.e.

$$K_p^{ss} = K_{gs} \frac{1}{u_g(t)} \tag{4.11}$$

In this equation, K_{gs} is a constant that can be calculated from the measurements taken in Fig. 5.10. The scheme diagram of the cascade camera controller can be seen in Fig. 4.13.

The proposed algorithm only controls two out of three camera parameters. The aperture of the used camera is manually controlled. The aperture value is set during the camera setup process and does not change while operating. The proposed algorithm

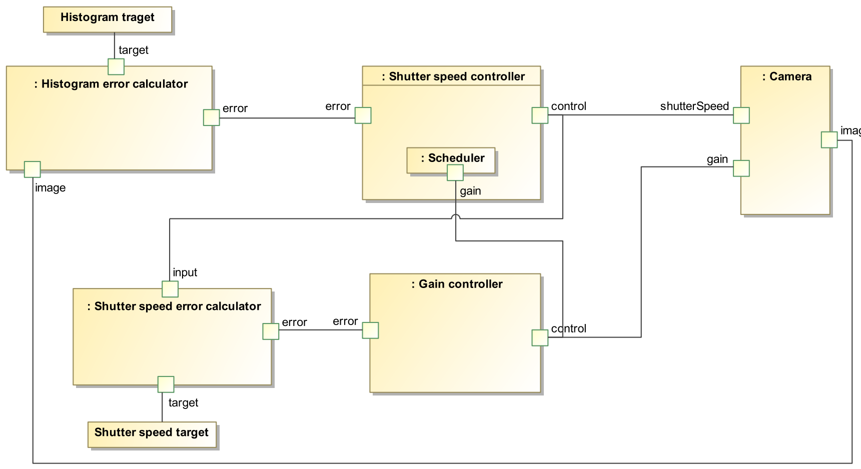


Figure 4.13. The diagram of a cascade controller used to control camera parameters simultaneously.

could be extended to control the camera aperture. This could be done by introducing an additional control loop that measures the current sensor gain and adjusts the camera aperture accordingly.

4.3. Stereo matching by using maximally stable extremal regions

Stereo matching is a process of estimating distances to objects (also called *depth*) in the scene from two or more images produced by cameras that are horizontally shifted. The main advantage of stereo matching is that it is a passive depth estimation method. Stereo matching is a computationally intensive process, but efficient algorithms can perform stereo matching on embedded low power processors. This thesis presents a stereo matching algorithm *Cyclops* that works in real time on the embedded NVIDIA Jetson platform. The algorithms are described in detail in [Gelšvartas et al., 2016a]. The author has also worked on a GPU-based stereo matching algorithm described in [Ivanavičius et al., 2018].

The main novelty of the presented *Cyclops* algorithm is the use of region-based features to reduce the stereo matching search space. This novelty makes the algorithm relevant for the real-time applications on the embedded computers. The algorithm trades the reconstructed disparity map accuracy for the increased computation speed and the reduced memory usage.

Cyclops takes as an input a pair of rectified stereo images and produces a disparity image. The algorithm consists of the following steps:

1. Extract MSERs in each stereo image.

2. Normalize each MSER region so that it could easily be matched.

3. Match MSER regions of the left and the right images.

4. Calculate the disparity image.

Each step is described in more detail in the following sections.

4.3.1. Extracting MSERs

Only several types of variable size features exist. Variable size features are more difficult to compute, but they can better represent the scene object boarders. In this thesis, we have chosen to use MSER for feature extraction. MSER have attracted a lot of research interest because they can be calculated in real time. This property makes MSER suitable for real time applications. Another MSER advantage is that they can be used on both gray-scale and color images.

Each stereo image is processed independently to extract MSERs. MSER extraction algorithm available in the *OpenCV* library [Bradski and Kaehler, 2008] was used. This algorithm can handle both gray-scale and color images. The color image processing algorithm is slower, but MSERs matching is less ambiguous in this case. Gray-scale images can be processed faster, but matching becomes more difficult. The choice of a color or a gray-scale image is a trade-off between the accuracy and speed. Usually, MSERs are detected on regular intensity images (referred to as *MSER+*). Additionally, MSERs can also be detected on the inverted image (MSER-). Performing MSER-detection produces a large number of additional regions for the matching step. Moreover, MSER- detection requires inverted image calculation, which doubles the image processing time.

The main disadvantage of MSERs is that their distribution over the image might be sparse. Additionally, large untextured areas of the image might not be detected as MSER. This is because MSER detection algorithms usually have maximal MSER size restrictions. Even when large MSERs are detected, their matching can be complicated due to occlusions. This can be solved by dividing each image into horizontal sub-images, such that:

$$I_i = I\left([0:w), [i \times h_{sub} : (i+1) \times h_{sub})\right) \tag{4.12}$$

Here, I_i is the i-th sub-image extracted from original image I, w is the original image width, and h_{sub} is the height of each sub-image. The image is divided into horizontal sub-images because the stereo images are rectified such that epipolar lines are aligned with the horizontal axes of the two images. Each sub-image is then processed individually by using the same algorithm steps. Dividing an image has two advantages. First, sub-images split very large MSERs that are much less likely to be matchable due to occlusions. Secondly, since each sub-image is processed independently, the number of matching candidates is reduced, and the matching step is faster. The image dividing procedure could further be improved by creating overlapping sub-images. This would reduce depth discontinuities that appear on image borders. Moreover, this strat-

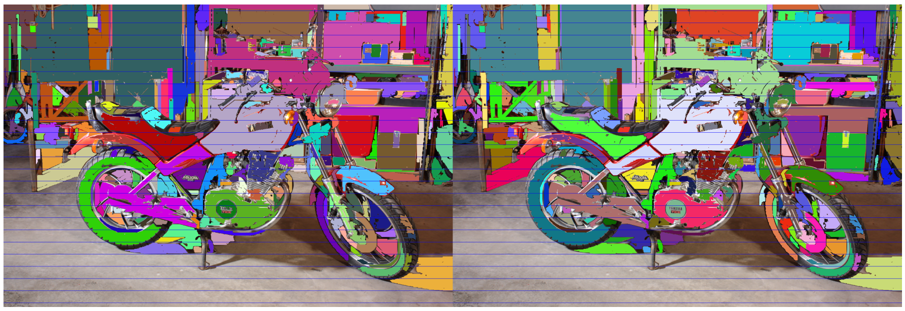


Figure 4.14. Stereo image pair with detected MSER colored randomly. The blue lines show epipolar image lines every 20 pixels.

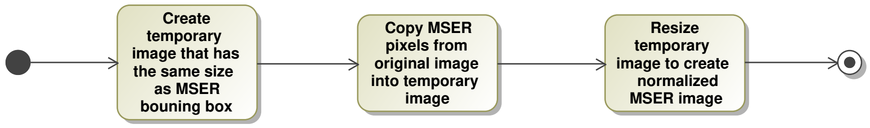

Create temporary image that has the same size as MSER bouning box
Copy MSER pixels from original image into temporary image
Resize temporary image to create normalized MSER image

Figure 4.15. Activity diagram showing the MSER normalization procedure.

egy helps to address problems with small MSERs being split between sub-images. An example image with detected MSERs can be seen in Fig. 4.14.

4.3.2. Normalizing MSERs

The detected MSERs have varying sizes and cannot be directly compared and matched. To overcome this problem, MSERs have to be normalized before matching. In the wide baseline stereo scenario, MSERs are normalized by transforming the region boundary so that the covariance matrix of the region becomes diagonal [Matas et al., 2004]. Wide baseline stereo images are not rectified, and the same region can be arbitrarily rotated in two images. Each normalized region is, therefore, used to extract the rotational invariant (based on complex moments) descriptor. Region descriptors can then be matched. The main disadvantage of this approach is that it uses the region boundary information, but the region internal pixel information is discarded.

Our method uses rectified stereo images. In this case, an MSER region detected in the left stereo image is typically going to have a similar appearance in the right stereo image. Normalization can be performed by scaling each MSER image into a fixed size image. More formally, the normalization input is rectified image I as well as a list of detected MSER in that image. Each MSER is represented by a list of image coordinates that belong to that region $(p_1, p_2, \ldots, p_n)$ and a bounding box of that region b. Each MSER region is normalized independently. The activity diagram explaining the normalization procedure can be seen in Fig. 4.15. The algorithm works as follows:

1. Temporary empty image I_l is created. This image has the same width and height

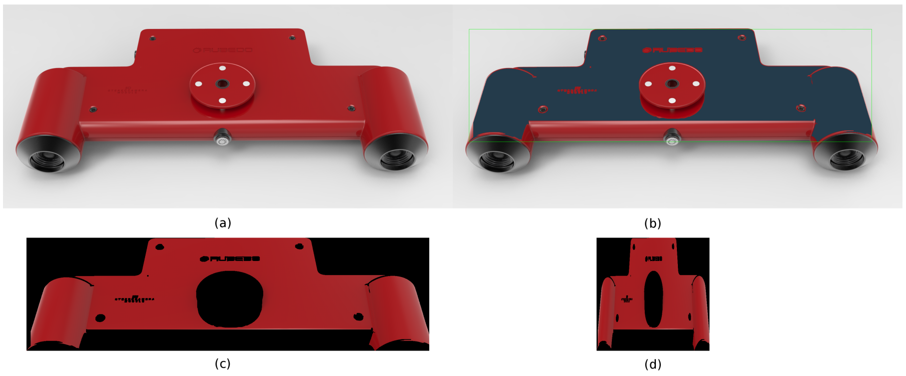


Figure 4.16. MSER region normalization procedure. (a) sample image I, (b) detected sample MSER region (blue) and its bounding box b (green), (c) temporary MSER image I_l, (d) normalized MSER image I_f.

as b.

2. All the MSER points $(p_1, p_2, \ldots, p_n)$ are copied from I to I_l. All points in I_l that do not belong to the MSER region are set to black.

3. Image I_l is resized to a fixed size image I_f. Image I_f is the normalized MSER image that is used in MSER matching.

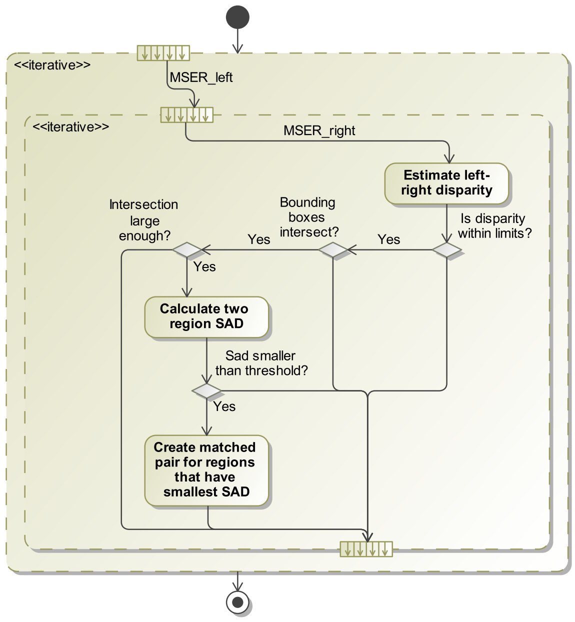
This normalization process is applied to both left and right stereo image MSERs. The normalization procedure is illustrated in Fig. 4.16. The normalized regions can be easily compared and matched.

4.3.3. MSER matching

The matching is performed between two sets of MSERs, namely, $MSER_l = \{mser_{l1}, mser_{l2}, \ldots, mser_{ln}\}$ and $MSER_r = \{mser_{r1}, mser_{r2}, \ldots, mser_{rm}\}$. $MSER_l$ are all regions detected in the left camera image, and $MSER_r$ are regions from the right camera image. Each region $mser_{li}$ is compared to all the regions in $MSER_r$. The majority of regions can be discarded as impossible matches by performing simple checks. We are performing the following checks to discard impossible matches:

1. We estimate the disparity of two regions by calculating the difference of the center points of the MSER bounding boxes. Matching candidates whose estimated disparity is smaller than the minimal disparity or larger than the maximal disparity are discarded. The maximal and minimal disparity values are algorithm parameters.

2. Given two regions $mser_{li}$, $mser_{rj}$ and their bounding boxes b_{li}, b_{rj} we check that their bounding boxes intersect along the O_y axis (O_x axis is along the image



<<iterative>>
MSER_left
<<iterative>>
MSER_right
Estimate left-right disparity
Is disparity within limits?
Yes
Bounding boxes intersect?
Yes
Intersection large enough?
Yes
Calculate two region SAD
Sad smaller than threshold?
Yes
Create matched pair for regions that have smallest SAD

Figure 4.17. Activity diagram illustrating the MSER matching procedure.

columns, and O_y axis is along the image rows). The stereo images are rectified, therefore, the bounding boxes of the matching regions must intersect.

3. The intersection of the bounding boxes b_{li} and b_{rj} along the O_y axis should be higher than a defined threshold.

MSER regions that pass all of the above checks are used for region similarity checks. Region similarity is measured between normalized $mser_{li}$ and $mser_rj$ images by using the sum of absolute differences (SAD) metrics. For each region in $MSER_l$ we select one matching region from $MSER_r$ that has a minimal SAD value.

This approach can introduce false matches when a particular region has no similar regions. To reduce the amount of false matches, the maximal SAD threshold is used. A check is performed to make sure that the matching region SAD does not exceed the maximal SAD threshold. Introducing a fixed maximal SAD threshold does not fully solve the problem. This is because SAD values are dependent on what area of an MSER image is covered with region points. Regions that have square shapes will cover more of the MSER image and will have larger SAD values. Elongated regions, on the other hand, will have more black pixels and smaller SAD values. This is taken into account by scaling the maximal SAD value by the percentage of MSER points in an MSER image. The matching algorithm is shown in Fig. 4.17.

4.3.4. Calculating disparity image

The disparity image is calculated in the left image optical frame. Each pair of matched MSER regions $mser_{li}$ and $mser_{rj}$ is used to estimate the disparity value of the pixels belonging to $mser_{li}$. This paper proposes to use the SGBM algorithm to estimate disparity values inside of the matched regions.

The matched MSER regions are well-localized, and the SGBM algorithm only has to be applied within a very small disparity range. The input for the SGBM algorithm is obtained by extracting image regions of interest (ROI) from the original left and right stereo images. The ROI are defined by the MSER region bounding boxes b_{li} and $b_{r}j$. Disparity values are calculated for all the pixels in the ROI image creating the disparity ROI image. Only disparity values that belong to the $mser_{li}$ are copied from the disparity ROI image into the final disparity image.

4.4. Specialized text predictor

The majority of already existing text predictors are of general purpose. Such predictors are run by using a language corpus consisting of a large number of literature books. The resulting general purpose predictors are versatile and provide good predictions in many situations. These predictors, however, might not be optimal for people with severe disabilities. Moreover, literature book training data tends to contain more indirect speech. Language models trained on such data will therefore be somewhat biased towards indirect speech. People, on the other hand, mainly use direct speech when communicating. Assistive communication technologies therefore should ideally use direct speech language models.

This thesis presents an alternative approach. The language model is trained by using only a limited amount of sentences. Such a model, of course, cannot fully represent the language, but, nevertheless, it has some advantages. The training data for such a model can be prepared for each user individually. A relative or a social worker could type in only the sentences that would be most likely used by the user. The training data could even be generated by using modified sentences from questionnaires used to evaluate the patient’s mental [Saldert et al., 2010] or physical health. Moreover, severe progressive neurological conditions (such as Huntington’s disease) usually cause some degree of mental degeneration [Bates et al., 2014]. In this case, a limited vocabulary predictor can actually help the users express their ideas clearly.

The specialized text predictor described in this thesis uses a standard N-gram-based text predictor. The main contribution of this thesis is to study how a specialized training data set can be used to increase predictor efficiency. This method has the potential to improve user experience and system usage efficiency.

Specialized text predictor presented in this thesis uses a simple N-gram language model. N-gram language models are one of the most popular ways to represent any given language. N-gram is a contiguous sequence of n words extracted from training data. During the training process, the model is created by recording all unique N-grams found in the training data into a database. Each database entry stores the N-gram and its count, i.e., the number of times that this particular N-gram has appeared in the training data set. Such model representation is very convenient for training because N-gram counts can be automatically collected from the training data.

When using the N-gram model an assumption is made that each word depends only on the last $n{-}1$ words. This means that the prediction of word x_i is based on

$$x_i{-}(n{-}1), \ldots, x_i{-}1 \tag{4.13}$$

where n is the cardinality of the model. In practice, a model with cardinality n also contains all models with lesser cardinality, i.e., $n{-}1, \ldots, 1$. In this case, each model returns a defined number of predictions and their probabilities. These probabilities are usually weighted to make sure that predictions from the model with a higher cardinality are preferred. Each prediction returned by the model is a single word.

The proposed text predictor was trained by using a training dataset specifically compiled for conversations about medical health. The training data was in English. Similar training data can be prepared for other languages. It is expected that for simple training sets the prediction quality should be similar for different languages. Medical health assessment questionnaires where used to prepare the training data. These questionnaires contained questions about most common medical conditions and pain descriptions. Each question was manually converted into a statement (i.e., direct speech) because that is what the text predictor is expected to produce. This particular dataset would be suitable for the case of use where a doctor or a relative is trying to evaluate a patient's health condition. Additional datasets could easily be constructed for other use case scenarios. Further algorithm improvements are presented in [Gelšvartas et al., 2016b].

4.5. A summary of proposed algorithms

This chapter presented novel algorithms created specifically for multifunctional user interface. The first three algorithms presented in Sections 4.1, 4.2 and 4.3 are all used in novel Projection mapping-based UI. More specifically, Section 4.1 details the camera-projector calibration process, the image processing pipeline, and the object highlight rendering.

The camera control algorithm is described in detail in Section 4.2. Many cameras have built-in camera control algorithms, but their quality varies significantly. Stable and fast camera control ensures that the results obtained by using image processing are accurate and reliable. The proposed algorithm is therefore important in ensuring the reliability of multifunctional UI.

Stereo matching is another algorithm that is used to improve the Projection mapping UI. Automatic projection mapping used in this thesis requires a RGB-D camera to perform object detection. The proposed stereo matching algorithm can be used to create a stereo RGB-D camera capable of processing images in real time. When combined with the proposed camera parameter controller, such a stereo camera would be able to work outdoors and improve the projection mapping UI reliability.

The final section of this chapter describes the specialized text predictor used in multifunctional user interface. This text predictor can significantly reduce the text input time and improve UX.

5. EXPERIMENTAL SETUP AND RESULTS

This chapter describes the experiments that were performed to evaluate the proposed multifunctional user interface architecture and algorithms. The described experiments measure the performance of individual algorithms as well as the usability of the entire system. The following sections describe the experimental setup and the results of the conducted experiments.

5.1. Experimental methodology

The performed experiments can be classified into three categories, namely, usage efficiency, user experience and algorithm evaluation. Usage efficiency and user experience experiments are performed with people participants. The participant group is described in detail in Section 5.1.1. Algorithm evaluation experiments use devices and data-sets to evaluate the proposed algorithms (i.e, no human participants were involved).

Usage efficiency experiments are performed to evaluate the proposed multifunctional UI. These experiments are described in detail in Section 5.2. The main purpose of these experiments is to measure how fast the users are able to perform various UI operations. The projection mapping UI is evaluated by using a user experience questionnaire. A detailed description of the experiment can be found in Section 5.3.

The remaining sections in this chapter describe the algorithm evaluation experiments. More specifically, the camera control algorithm is evaluated by using the Universal Serial Bus (USB) camera in Section 5.4. All the camera control experimental results have been generated with this camera. The stereo matching algorithm has been evaluated by using an existing data-set with ground truth data. These experiments are described in Section 5.5. Finally, the text predictor has been trained and evaluated by using a specialized handmade data-set. A detailed experiment explanation can be found in Section 5.6.

5.1.1. Participant group

Eleven individuals participated in the system usage experiments. The group consisted of 8 females and 3 males. Participant ages ranged from 16 to 83 years old. The participant age and gender distribution is visualized in Fig. 5.1. The exact age and gender of each participant is also provided in Table 5.1. The participant group was chosen such that it would be as diverse as possible. We can see that the group consists of both male and female participants. The wide age range will ensure that age-related differences can be seen in generated data. The group included a disabled person to demonstrate the developed system's suitability. The study group size is relatively small due to the overall scope of the thesis.

The participant group consisted of 10 healthy individuals and one disabled individual. The disabled participant had deep left hemiparesis caused by neuroinfection polio, chronic cerebral ischemia, and vestibular ataxic syndrome. A high level of spe-

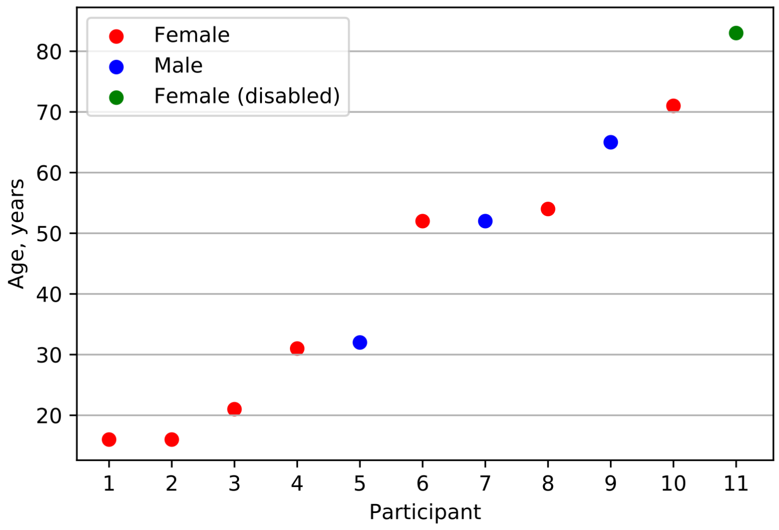


Figure 5.1. Participant age and gender.

Table 5.1. Table containing age and gender of each participant.

Participant	Age (years)	Gender
1	16	Female
2	16	Female
3	21	Female
4	31	Female
5	32	Male
6	52	Female
7	52	Male
8	54	Female
9	65	Male
10	71	Female
11 (disabled)	83	Female

cial needs has been assigned to this participant due to the above mentioned conditions.

The participants performed all the experiments individually. First, the purpose of the created multifunctional UI was presented to the participants. This was followed by the introduction of the actual experiments and used devices. The participants were given the opportunity to play around with the system before performing the actual experiments. The participants were also given some time to relax in between experiments.

5.1.2. Used equipment and software

This section describes the devices that were used during the experiments. Three devices have been used during the experiments, namely, Eye tracker, Sip/Puff and Speech recognition. Detailed information concerning each of the devices is provided below:

1. The Eye tracker from Eye Tribe has been used to perform gaze tracking. Eye tribe is a small form factor (20 x 1.9 x 1,9 cm) device. The device uses an infra-red camera to perform eye detection and tracking. The device has a 20ms response time and 0.5–1 degree accuracy range. The device is operational at the range of 45–75 cm from the camera. The device should be positioned above or below the monitor. In our experiments, we positioned the device below the computer monitor. The device is attached to the computer by using the USB3 port. An image illustrating the device can be seen in Fig. 5.2.

2. The Sip/Puff Breeze device developed by Origin Instruments has been used for Sip/Puff experiments. The device is connected to a computer over the Universal Serial Bus (USB) port and is recognized by the operating system as a Human interface device (HID) device. The device can generate computer mouse left and right button click actions.

3. A conventional laptop computer microphone was used to record speech commands. A standard English language model from the PocketSphinx package was used to recognize the participant's voice commands. The recognizer dictionary has been limited to two words ('Next' and 'OK') to improve the recognition accuracy.

The developed multifunctional user interface has been used during the system evaluation. We used two different interfaces during the evaluation, namely, selection from the list, and text input. The example of the text input screen in provided in Fig. 3.7. The screenshot of the list selection screen can be seen in Fig. 5.4.

5.2. Multifunctional user interface experiments

This section describes the experiments performed to evaluate the performance of the multifunctional user interface architecture. The first experiment tested how efficiently each participant used each device. This was followed with a multiple device experiment. More specifically, we tested the hypothesis that using multiple devices simultaneously can improve the system usage efficiency. A more detailed experiment description can be found in [Gelšvartas et al., 2015].

5.2.1. Multifunctional user interface experimental setup

In all the experiments, the developed UI was presented to the users. The participants were given sufficient time to learn to use the system and ask questions. This time was given to the participants each time they used a new device. This introductory

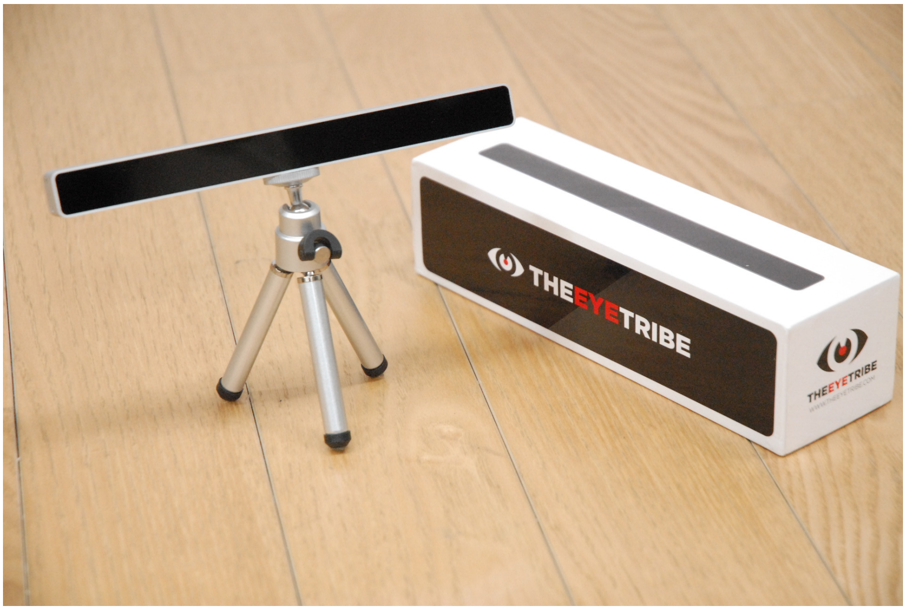


Figure 5.2. The eye tracking device Eye Tribe.

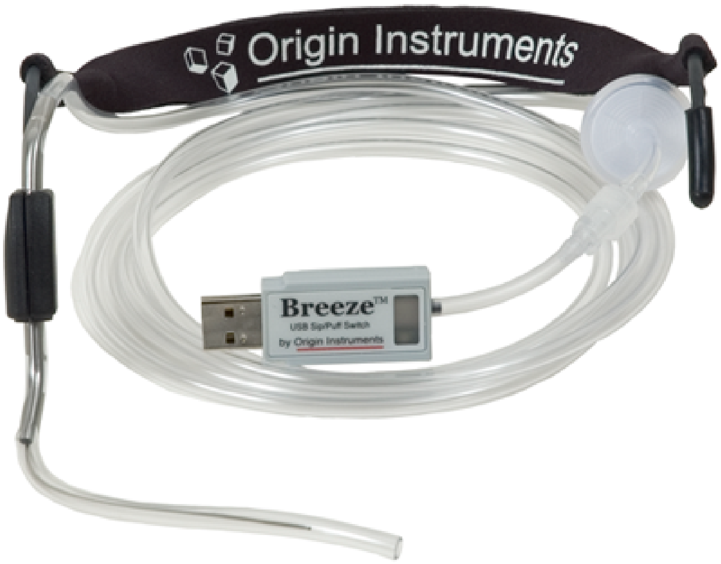
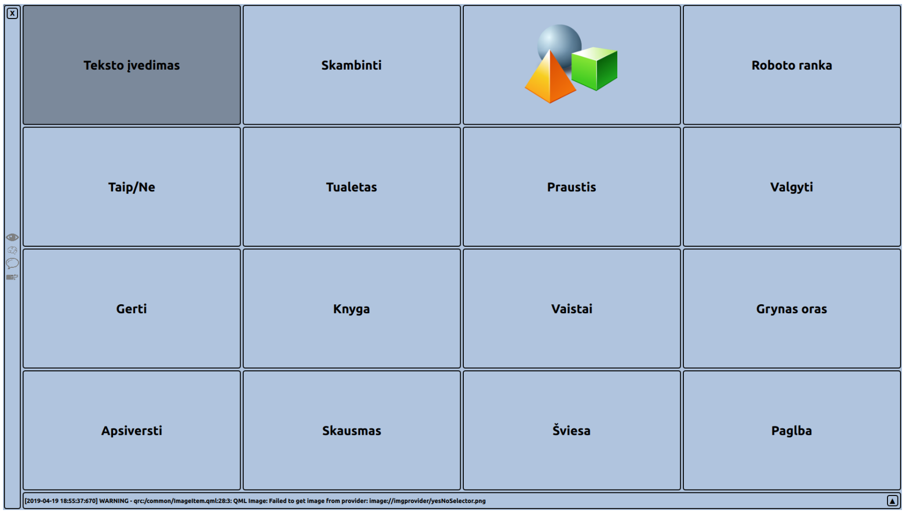


Figure 5.3. The Sip/Puff device from Origin Instruments Sip/Puff Breeze.

period was around 5 minutes per device for all the participants. When asked, all the participants indicated that the time given to learn to use the system was sufficient.



```
[2019-04-19 18:55:37:670] WARNING - qrc:/common/ImageItem.qml:28:3: QML Image: Failed to get image from provider: image://imgprovider/yesNoSelector.png
```

Figure 5.4. A screenshot of the list selection interface. At the start for the program, the first element in the list is selected. During the experiments, the users were asked to select cell number 12.

In the first experiment, the users were presented with a four-times-four grid, where each grid cell represented an action that they could select. The participants were asked to always select grid cell number 12. This means that the participants had to perform 11 actions to select the next grid element, and 1 action to select the desired element (except for the eye tracker where the selection is done by gazing into the desired element). Each participant used three devices, namely, Sip/Puff, SpeechRecognizer and EyeTracker. The experiment with each device was repeated 10 times resulting in 30 data points per participant. All the eleven participants performed this experiment.

The second experiment tested several devices simultaneously. This experiment used a text input task. The participants were asked to input the word ‘chest’. Each participant was asked to input the same phrase by using three device configurations, namely, Sip/Puff, EyeTracker+Spi/Puff, and SpeechRecognizer. When using EyeTracker+Spi/Puff, each device was used to provide one of the necessary input channels. The eye tracker was used to move between keys, and the Sip/Puff to select the desired key. The T9 keyboard layout was used during the experiments. This experiment involved nine participants. Two participants from the study group did not participate in this experiment because the EyeTracker device did not work for them. We present more details in the section below.

The results of the experiment were evaluated on the basis of two criteria, namely, the time it took for the participant to input the desired actions (input time), and the number of errors that were made by the user while inputting the actions. The input time is a convenient way of comparing different input devices. However, measuring only the input time is not sufficient, because we may face situations where two devices indicate the same input time for different reasons. For example, one device can have a big input time because generating each device signal takes a long time, while another device can be much faster at generating signals but often produce erroneous signals. Therefore, we also measure the number of errors made by the user when inputting the test actions. Each action selection experiment is finished once a participant has selected one element. For the text input experiment, the participants are asked to correct any errors that they have made while inputting the text.

5.2.2. User interface experiment results

In the first experiment, each participant used three devices to select the same four-times-four grid cell. Each of the eleven participants repeated every experiment 10 times. This resulted in 330 data points in total. The average per participant/device input times and error rates can be seen in Fig. 5.5. This figure also contains the per-device input time and the error rate averaged over all of the participants. More detailed data of this experiment is provided in Appendix A.

We can see that Speech recognizer had the longest overall input time, but the participants made the fewest errors with this device. Sip/Puff device had a better input rate but with more input errors. Finally, eye tracker had the shortest average input time, but the participants made most errors overall when using this device. We should note that experiments where participants selected an incorrect action are not counted towards the average experiment input time.

When looking at each device in more detail, we can see that all the participants were able to use both Speech recognizer and Sip/Puff device. We observed that Speech recognizer did not work well for one participant. In this case, the participant was saying the correct control commands, but they were not recognized by the system. Similarly, one participant had issues when using Sip/Puff device. The participant tried to issue a Sip command, but the device often issued also a Puff command after some Sips.

We have also observed that the eye tracking device was not working stably for some participants. In these cases, the tracked eye coordinates jumped significantly, resulting either in a long input time or an incorrect action being selected. When using the eye tracker, the participants had to gaze at the desired action for 4 seconds to select it (this 4 second minimal time is indicated in Fig. 5.5e with the red line). There were two participants who could not use the eye tracker device at all. The participants positioned correctly in front of the device, but it failed to detect the participants' eyes completely. For these participants' the error rate was set to 10 (equal to the number of experiments), and the input time was not calculated.

This experiment demonstrates that it is very useful to have a system with multiple input devices. We observed that some users got frustrated when one of the devices was not working for them. This usually led to more errors by that user for the specific device. Such scenarios usually led to the system being rejected by the user after the initial trial. Another observation is that the users preferred more efficient devices even when this led to more errors. This hints that it is worth to implement error recovery scenarios in the UI and recommend the most efficient device for the users. Overall, each participant was able to use at least one of the tested devices efficiently. This clearly demonstrates the advantage of the multiple device assistive UI that can be adapted to each individual user.

The second set of experiments was successfully carried out in three use cases, namely, sip/puff, speech recognizer, and eye tracker + sip/puff. A comparison of the input times of the tested devices is provided in Fig. 5.6.

The SpeechRecognizer has the highest input time because the participant has to pronounce the full word; the processing time of the recorded sound is also high. The input time when using the Sip/Puff switch was significantly lower. The Sip/Puff switch has a minimal processing time; performing sips and puffs is also much faster than pronouncing words. The participants managed to further improve the input time by using the combination of the eye tracker and the Sip/Puff switch. When using the Sip/Puff switch and the SpeechRecognizer, the participants had to move through the whole list of letters if the next letter they had to enter preceded the current letter. The eye tracker gaze coordinates allowed the participants to jump straight to the next letter. This improved the input time in the case of the EyeTribe+Sip/Puff.

The average error rate when performing the text input task is provided in Fig. 5.7. The SpeechRecognizer had the smallest error rate. The few errors that occurred when using this method happened when the recognizer predicted incorrect words. The SpeechRecognizer had to distinguish only between two words; the incorrect recognitions were therefore rare. The Sip/Puff and the EyeTracker+Sip/Puff devices had significantly higher error rates than the Speech recognizer. The Sip/Puff device errors usually occurred when the participants tried to input the words very quickly and accidentally jumped over several keys. The errors of the EyeTracker+Sip/Puff, on the other hand, were usually caused by accidental blinks or a distracted gaze while selecting the keys. However, participants managed to correct these errors much faster than those using the other input devices.

Combining several assistive devices offers even more system adaptation possibilities. This has a benefit of potentially making the system suitable for a larger user group. Finding the best device combination for each user would become complicated. Additional strategies will have to be developed to make this process as efficient as possible.

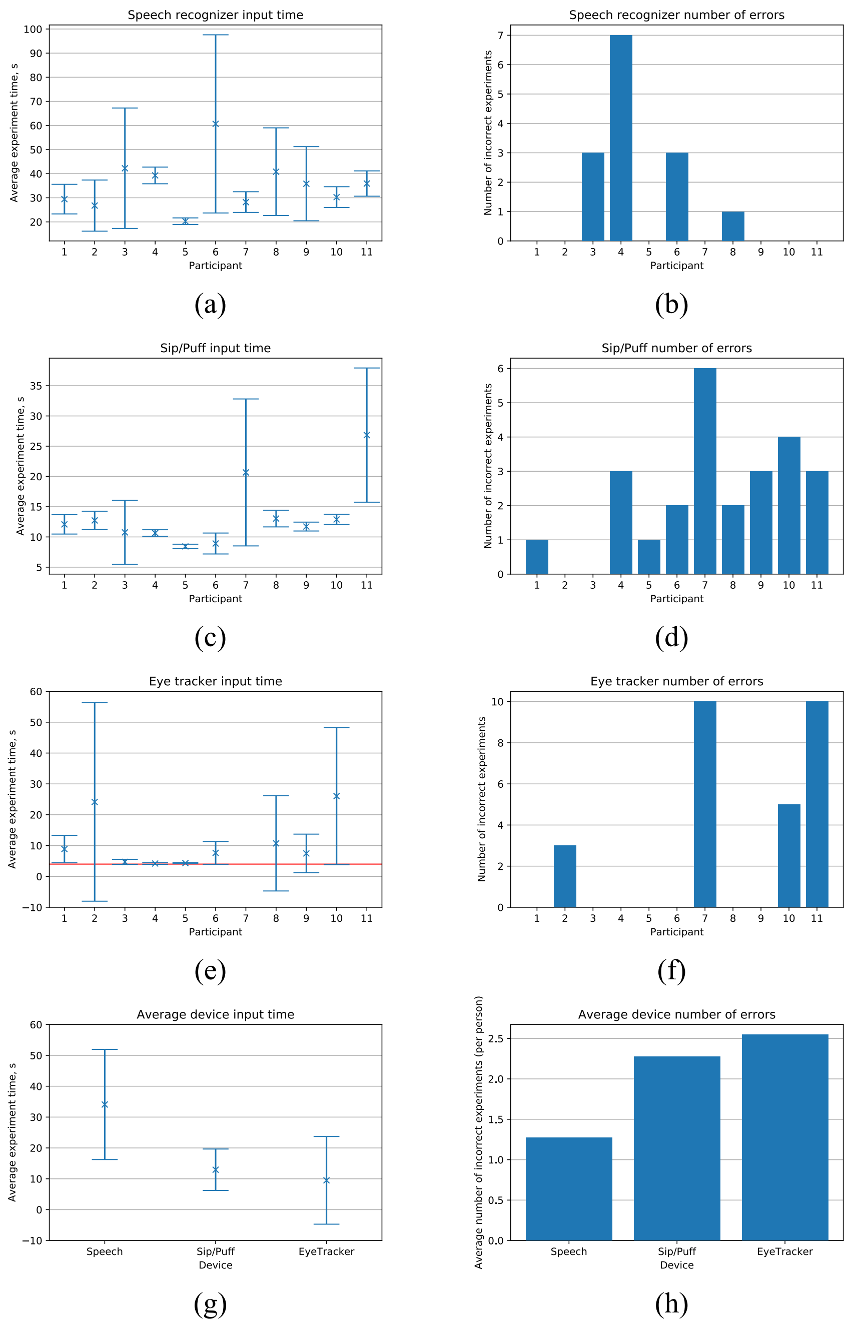


Figure 5.5. The device input time and error rate per participant and averaged for all devices. (a,b) Speech recognizer input time and error rate for each participant; (c,d) Sip/Puff input time and error rate for each participant; (e,f) Eye tracker input rate and error rate for each participant; (g,h) device input time and error rate averaged over all of the participants.

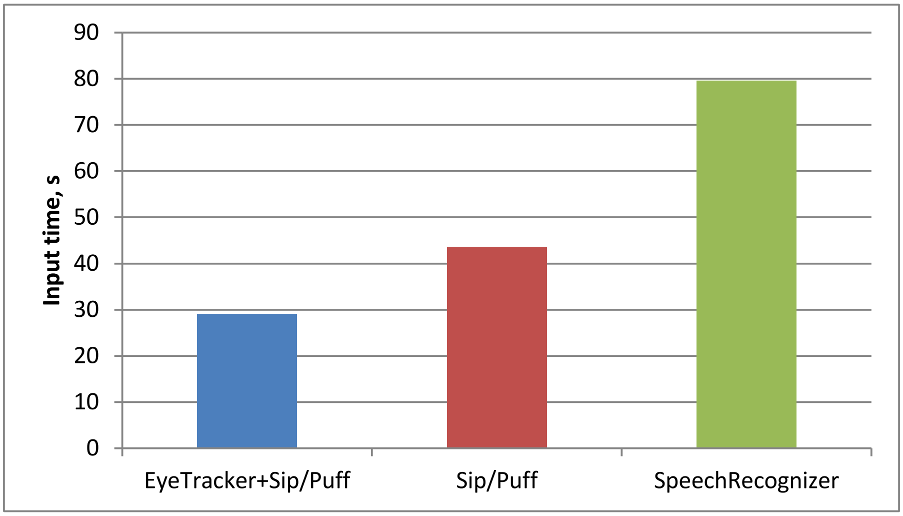


Figure 5.6. Comparison of the input time of the different devices.

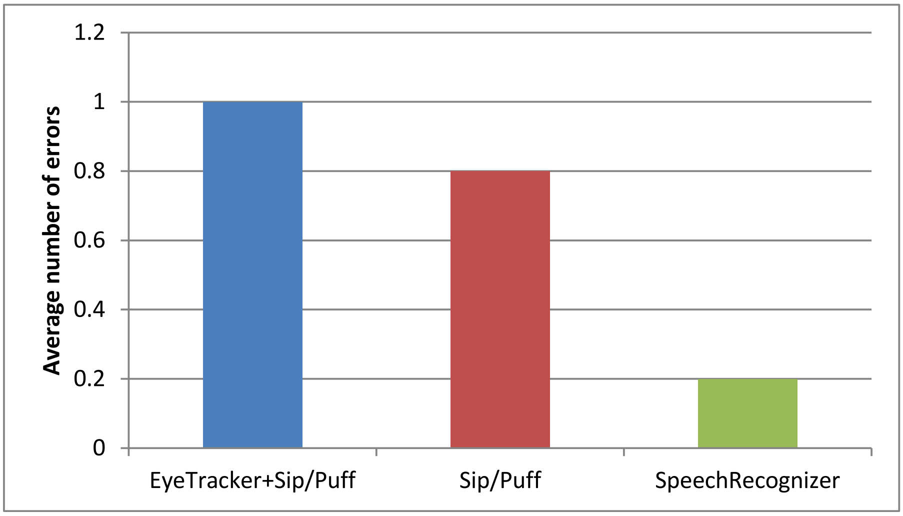


Figure 5.7. The average number of errors performed when using the devices.

5.3. Projection mapping experiments

The usability of the proposed projection mapping user interface has been evaluated in this section. The UI has been used for tabletop object selection.

5.3.1. Projection mapping experimental setup

Object selection UI was evaluated by using a Post-Study System Usability Questionnaire (Post-Study System Usability Questionnaire (PSSUQ)). For this experiment, we used a popular PSSUQ described in [Lewis, 1995]. The system usability study was prepared according to the guidelines presented in [Rubin and Chisnell, 2008]. Projection mapping does not introduce any new functionality to the developed UI but presents information already available in the system in a new way. PSSUQ is therefore the best way to evaluate projection mapping UI.

Eleven participants were asked to use the UI to select one of the three cans placed in front of them on a table top surface. Each individual used the system three times with a sip/puff device for action input. After the experiment, each individual was asked to fill up the system usability questionnaire containing 16 questions. Each question was given a score from 1 to 7, where 1 means the user strongly agrees with the statement, and 7 means strong disagreement. The score scale was chosen according to the questionnaire authors' recommendations. One question was excluded from the questionnaire since it is related to system documentation, and our system is only a prototype.

This thesis evaluated only the usability of the proposed projection mapping UI. The main purpose of this UI is to present the object detection algorithm results in a more natural way. Therefore, we have not presented the object selection time experiments. Moreover, sip/puff device command generation time is independent from the used information presentation method. Measuring this input time would not provide any additional insight into the system usability. We have also confirmed that there was no significant time difference when selecting the item presented with projection mapping versus selecting the same item from a list presented on a computer screen.

The question scores have been divided into three categories, namely, positive, neutral, and negative. The positive scores ranged from 1 to 2, neutral scores from 3 to 5 and negative from 6 to 7. This score scale division was performed to make it easier to interpret and summarize the study results. Ideally, we would like all of the questions to be evaluated positively. Negatively evaluated questions would show the aspects of the system that should be improved first. Enhancements of these system aspects would significantly improve the system usability.

Projection mapping UI implementation used during these experiments did not include the proposed camera control and stereo matching algorithm. These algorithms have been evaluated separately in the subsequent sections. Stereo matching and camera control algorithms will be integrated into the projection mapping UI in the future system implementations.

5.3.2. Projection mapping results

During the experiments each participant was asked to rate all the statements. The results are provided as average score ± standard deviation (STD). The confidence intervals (CI) at 95% confidence level are also provided with the results. The user experience questionnaire results are presented in Table 5.2. Overall, the interface has been assigned positive or close to positive scores by the participants. The majority of the participants learned how to use the system very quickly. As expected, the survey revealed that the interface lacks error handling functionality. This was expected as the interface currently is a prototype, and little attention has been put into error handling scenarios. These limitations will be addressed in further system developments.

Table 5.2. Table containing projection mapping UI user experience questionnaire and results.

Questionnaire statement	Average score ± STD	CI
Overall, I am satisfied with how easy it is to use this system.	1.82 ± 0.39	0.23
It was simple to use this system.	2.36 ± 0.77	0.46
I was able to complete the tasks and scenarios quickly while using this system.	1.55 ± 1.23	0.73
I felt comfortable while using this system.	4.00 ± 2.00	1.18
It was easy to learn to use this system.	2.00 ± 0.43	0.25
I believe I could become productive quickly while using this system.	3.91 ± 0.67	0.39
The system gave error messages that clearly told me how to fix problems.	6.09 ± 0.90	0.53
Whenever I made a mistake while using the system, I could recover easily and quickly.	5.91 ± 0.79	0.47
The information (such as online help, on-screen messages and other documentation) provided with the system was clear.	N/A	N/A
It was easy to find the information I needed.	2.09 ± 0.51	0.30
The information was effective in helping me complete the tasks and scenarios.	1.55 ± 1.72	1.02
The organization of information on the system screens was clear.	1.27 ± 0.62	0.36
The interface of this system was pleasant.	3.27 ± 2.26	1.34
I liked using the interface of this system.	2.73 ± 1.81	1.07
This system has all the functions and capabilities I expected it to have.	2.36 ± 1.30	0.77
Overall, I am satisfied with this system.	1.91 ± 0.51	0.30

The PSSUQ did not reveal any significant gender-related differences. Age-related differences, on the other hand, were more significant. Overall, the younger participants tended to rate the system more positively. Meanwhile, the older participants found the UI less intuitive. Moreover, some older participants indicated that projector-generated UI is sometimes difficult to see. This was one of the main reasons contributing to the lower scores by the older participants.

The results indicated that some participants were not comfortable in using the system. When asked, the users mainly indicated that they were not comfortable in wearing the Sip/Puff assistive device. The experiment had to be performed in a non-brightly lit environment, and this was another reason contributing to user discomfort. Productivity was also indicated as being non-optimal by some users. The main suggestion for improvement from the users was trying to use a different assistive device. Speech recognition was indicated as one of the possible improvements. This would require implementing a speech recognizer that is able to recognize all of the types of objects that can be detected by the object detector. This would also require implementing recognition of the command to distinguish between to objects of the same class (e.g., cup on the left). Lastly, the interface appearance rating has also received some negative feedback. The main suggestion for corrections was trying to play animations to highlight the objects instead of just highlighting the object with a single color.

5.4. Camera control experiments

This section presents experiments performed to study the camera parameter relationships and the proposed cascade camera controller. The camera control algorithm performance has been evaluated by comparing it to the performance of the manufacturer-provided automatic control mode. This section contains the detailed experimental setup and results description. The proposed algorithm is used in the stereo matching pipeline described in this thesis.

5.4.1. Camera parameter relationships

This thesis raises a hypothesis that camera parameters are not independent. This means that the change in one of the camera parameters can affect the behavior of other parameters. Well understood parameter dependence can be used to develop and tune more efficient camera control algorithms. A series of experiments were, therefore, performed in order to measure the camera parameter interdependence.

The experiments were performed with a camera facing a white wall. PointGrey Flea3 (FL3-U3-13E4C-C) with a *Tamron* ($8mm, 1\colon 1.4, \varnothing 25.5$) lens was used during all the experiments. During all the experiments, only two camera parameters were changed at a time. Three experiments have been performed in total, and they measured how the shutter speed is affected by scene luminance, aperture, and sensor gain. During all the experiments, the first one of the three parameters (luminance, gain, aperture) has been fixed at a certain value, and then the shutter speed parameter has been changed in the interval $(0, 0.025)$ s with the step size of $0.001s$. The camera image is captured for each value of the shutter speed, and H_{ct} value is calculated.

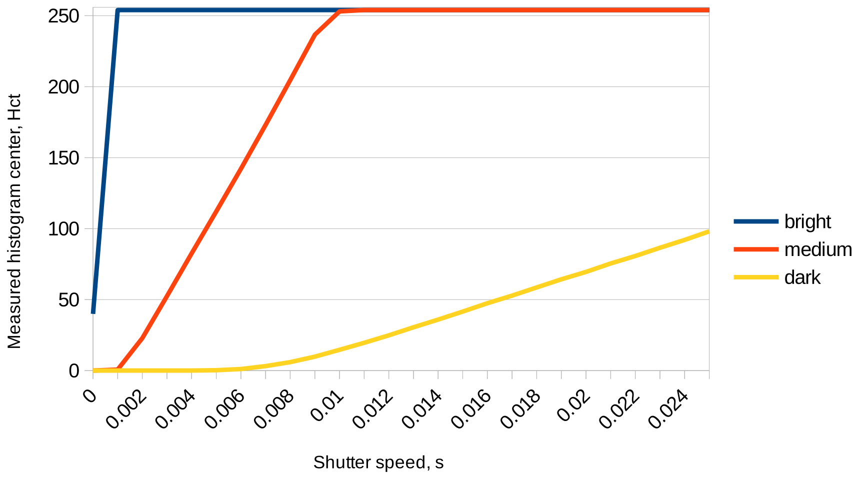


Figure 5.8. Experiments demonstrating scene luminance effects on shutter speed. Three different scene lightness levels were used, namely, bright (blue), medium (red) and dark (yellow).

The first experiment measured the scene luminance and shutter speed dependence. Three different scene luminance scenarios were used, namely, bright, medium, and dark. The aperture was set to $f/4$ and gain to $12dB$. The experiment results are illustrated in Fig. 5.8. We can see that all the shutter speed curves are almost linear. The non-linearities start to become significant at extreme H_{ct} values, where the amount of overexposed and underexposed pixels increases rapidly. We can clearly see that different luminance values do not change the shape of the H_{ct} curve and only change the curve slope.

The second experiment measured the aperture and shutter speed dependence. Three different aperture values were used, namely, $f/2$, $f/4$ and $f/8$. The scene luminance was set to medium brightness, and gain to $0dB$. The experiment results are illustrated in Fig. 5.9. The resulting H_{ct} curve is very similar to that observed when changing the scene luminance. A wider aperture ($f/2$) produced a steeper slope H_{ct} curve.

The third experiment measured the sensor gain and shutter speed dependence. Three different gain values were used, namely, $0dB$, $12dB$ and $24dB$. The scene luminance was set to medium brightness, and aperture to $f/4$. The resulting H_{ct} curves can be seen in Fig. 5.10. Again we observed the same effect that increasing the sensor gain increases H_{ct} curve slope.

We can see that all of the above described experiments produced very similar results. This is not surprising because in practice changing the aperture and sensor gain is equivalent to ‘increasing the amount of light’ available in the scene. Another important observation is that ‘increasing the amount of light’ always makes the H_{ct} curve steeper. This means that the smaller shutter speed value changes have a more significant effect

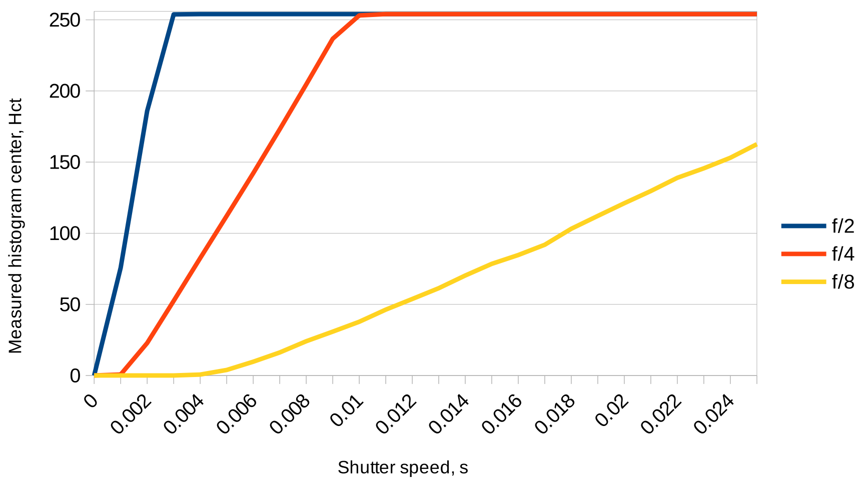


Figure 5.9. Experiments demonstrating aperture effects on the shutter speed. Three different apertures were used, namely, $f/2$ (blue), $f/4$ (red) and $f/8$ (yellow).

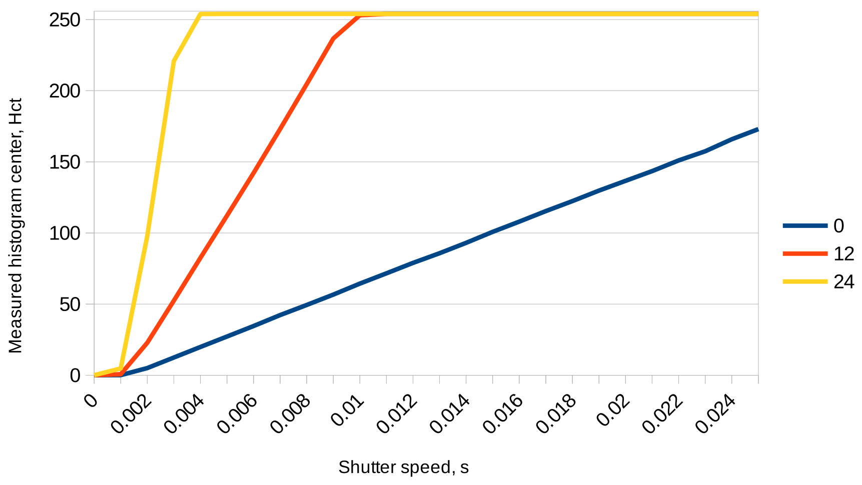


Figure 5.10. Experiments demonstrating aperture effects on the shutter speed. Three different sensor gain values were used, namely, $0dB$ (blue), $12dB$ (red) and $24dB$ (yellow).

on the image exposure when the scene is brighter. In this situation, a controller with a smaller gain should be used to make sure that the controlled image exposure does not oscillate around a set point. On the opposite, when the scene is dark, big shutter speed adjustments will not have a significant effect. Here, a controller with a higher gain value can be used. This observation is very important and is taken into account when designing the camera control algorithm.

5.4.2. Camera control experimental setup

The experiments were performed by using PointGrey Flea3 (FL3-U3-13E4C-C) USB camera with a *Tamron* ($8mm, 1\colon 1.4, \varnothing 25.5$) lens. The lens aperture was set to $f/4$. Two experiments have been performed, one with the proposed camera control algorithm, and another with the automatic shutter speed and gain control of the camera manufacturer. This experiment measured H_{ct} for each captured frame. The camera images were recorded every $0.15s$ (i.e., the camera was running at $\sim 6.66Hz$)

This experiment measures how fast the camera can adapt to the rapid lighting changes. To make the experiments comparable, the following procedure has been used:

1. The camera has been set stationary in front of the outdoor scene with very bright direct sunlight.
2. The camera lens was covered with the lens cap. This ensures that no light is entering the camera sensor.
3. Enough time is given so that both the camera shutter speed and the gain reach the maximal values. At this point, the measured $H_{ct} = 0$ because no light is entering the camera. Moreover, the camera parameters stabilize at maximal values.
4. The lens cap is removed. At this moment, H_{ct} changes from 0 to 255, and the camera control algorithm starts to adapt to the changed lighting conditions.
5. The H_{ct} values are recorded until the camera stabilizes around the target H_{ct}^{target}.

The camera built-in algorithm is closed source, and can only be controlled through the parameters exposed in the manufacturer's camera control console. It is not possible to adjust the H_{ct}^{target} parameter of the PointGrey built-in algorithm. In fact, we cannot guarantee that PointGrey is using exactly this parameter to determine the image exposure. On the other hand, the current H_{ct} value can be measured for both algorithms. To make it easier to compare both algorithm performance results, the following procedure has been used:

1. The PointGrey built-in algorithm has been used, and H_{ct} value recorded.
2. The H_{ct} where the PointGrey algorithm settled after adapting to the lighting conditions was used as a H_{ct}^{target} value for the proposed algorithm.

3. The proposed algorithm experiment was performed with this H_{ct}^{target} value.

Two properties of the algorithms are studied in the course of this experiment, namely, the time it takes for the camera to settle to the target value, and the stability of the algorithm. Previous experiments showed that camera control is more sensitive when more light is entering the sensor. This is the main reason why a very bright scene was used during this experiment.

5.4.3. Camera control results

The described experiment has been performed both using the proposed camera control algorithms as well as the camera algorithm provided by the manufacturer. The results of the camera control algorithm experiments are presented in Fig. 5.11. Both experiments were aligned so that the time of the lens cap removal would match exactly. The lens cap was closed from $0s$ until $0.6s$. At time value $0.75s$, the lens cap is removed, and H_{ct} value jumps to the maximal value of 255. For some time, both algorithms produce the maximal H_{ct} value. During this time, the camera parameters are being adjusted by the camera controller, but they do not affect the measured image exposure. After this time, the image exposure starts to change until it settles around the set point value. The H_{ct}^{target} measured with the manufacturer's software was determined to be 66.7. This H_{ct}^{target} has been set for the proposed algorithm, and the H_{ct} settled at exactly this value.

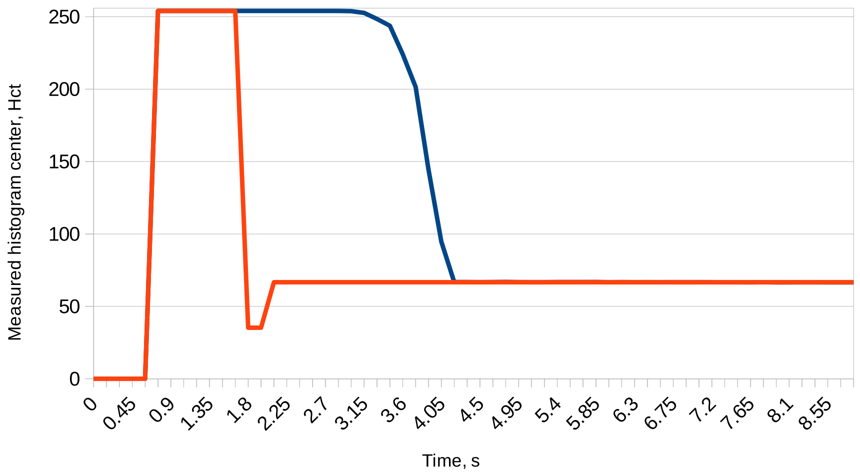


Figure 5.11. The camera parameter control algorithm efficiency experiments. Manufacturer's camera control algorithm (blue), and proposed camera control algorithm (red).

The experiment clearly shows that the presented algorithm converges faster to its set point. More precisely, the proposed algorithm converged in $1.5s$, whereas the manufacturer's algorithm converged in $3.45s$. The proposed algorithm is therefore approximately two times faster than the manufacturer's algorithm. We can see that the proposed algorithm overshoots before settling around the set point. This is the result of non-ideally tuned camera control algorithm gain parameters. The algorithm used during the experiments only used the proportional gain of the PID controller. The algorithm behavior could further be improved by better parameter tuning. Automatic parameter tuning strategies could be used to perform parameter tuning. The manufacturer's algorithm, on the other hand, is closed source, and no parameter adjustments can be performed. Another advantage of having a separate camera control algorithm is that its H_{ct}^{target} can be adjusted depending on the exact case of use.

The proposed algorithms are also relevant when developing a custom stereo vision camera. In this situation, image exposures of both cameras should be identical in order to accurately perform stereo matching. When using an external camera control algorithm, the parameters can be calculated only for one camera and set to both cameras at the same time. The manufacturer's algorithm could set different camera parameters to each camera, which results in differently exposed images.

5.5. Stereo matching experiments

This section describes the proposed stereo matching algorithm evaluation experiments. The output of the stereo matching algorithm can be used for object detection.

5.5.1. Stereo matching experimental setup

All the experiments were performed on a NVIDIA's Jetson TK1 development board. The evaluated algorithms only use CPU computations, and the GPU processor was not used. The performance of the CPU cores was maximized by disabling the CPU power saving functionality. The board was powered by using the default power adapter provided in the development kit. The Linux for Tegra R21.4 operating system provided by NVIDIA was used. During the experiments, only the disparity evaluation algorithm process was running. Each experiment was started via a Secure Shell SSH connection to the board.

Cyclops was compared with two high frame rate algorithms, namely, SGBM and ELAS. These algorithms are at the top of the Middlebury Stereo Evaluation when evaluated according to the algorithm computation time. Both of these algorithms have publicly available implementation source codes. We used the ELAS algorithm implementation provided on the authors' website as well as the SGBM algorithm implementation provided with OpenCV library version 3.1. The algorithms were evaluated by using the tools and images provided by the Middlebury Stereo Dataset version 3. The evaluation was performed only on 15 training images from the Middlebury Dataset. Both ELAS and SGBM algorithms were configured to have the same parameters as those used in the Middlebury Stereo Dataset. The SGBM algorithm was configured to use the 'Full DP' parameter set to false, as this runs the faster version of the SGBM algorithm. 'Full DP' parameter enables the full-scale two-pass dynamic programming algorithm that consumes more memory and is slower.

Cyclops first extracted the MSER regions on sub-images whose width was set to

$h_{sub} = 50$ pixels (px). Each extracted MSER region was normalized to an I_f image whose size was fixed at $50px \times 50px$. The region intersection threshold used during the matching step was set to 0.9. Finally, the maximal SAD threshold was set to 120000.

The algorithms were evaluated by using the following criteria:

1. Algorithm computation time. This was measured without taking into account the time taken to load the input images and save the result disparity image. The computation time was calculated for each image individually. The results section provides the average algorithm computation time per image in seconds.

2. The total memory usage of the process that was executing the stereo matching algorithm. This experiment was performed on the Vintage stereo image pair from the Middlebury Stereo Dataset. The measurements were obtained by using the Massif tool from the Valgrind profiler available in the Ubuntu distribution.

3. The percentage of bad pixels whose disparity error is greater than 2.0. The provided measure is the average percentage of bad pixels per image.

4. The percentage of invalid pixels per image. These are pixels where the stereo matching algorithm was not able to estimate the disparity value. The provided measure is the average percentage of invalid pixels per image.

5. The average error of bad pixels per image. The average disparity value error of pixels that have been classified as bad pixels.

The algorithms were evaluated by using three different resolution image sets, namely, full, half and quarter resolution images. The full resolution images have the size of up to 3000×2000 pixels, and the maximal disparity values of up to 800 pixels. The Middlebury dataset provides the maximal disparity values of each image individually. Half resolution images are two times smaller with up to 1500×1000 pixels and no more than 400 disparity values. The quarter resolution images are again two times smaller than the half resolution images. In the provided results, the used resolution is indicated by adding an underscore and a letter to the algorithm's name. Letter Q indicates quarter resolution images, H stands for half resolution images, and F represents full resolution images. For example, the SGBM algorithm that was run by using the half resolution images would be shown as SGBM_H.

5.5.2. Stereo matching results

The most important criteria used to evaluate the algorithms is the algorithm computation time. *Cyclops* algorithm reduces the search space and is therefore expected to reduce the computation time. The computation time results can be seen in Fig. 5.12.

All the algorithms processed quarter resolution images in very similar computation times. The computation time of the SGBM and ELAS algorithms rapidly increases to

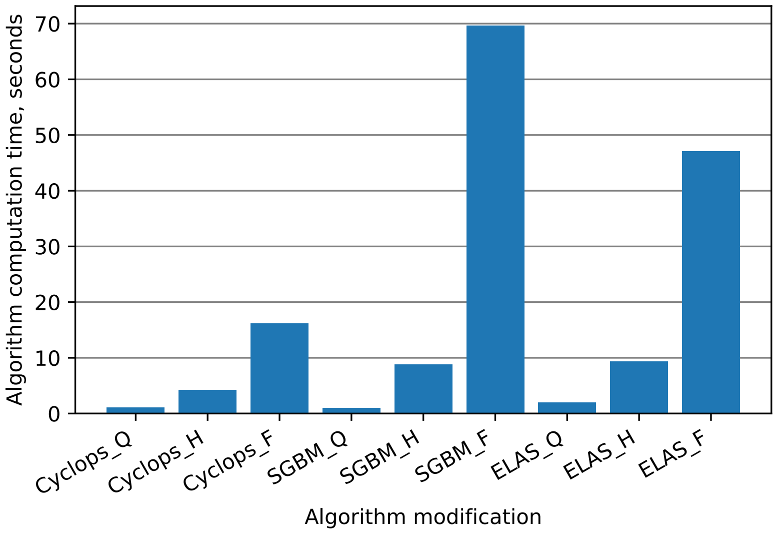


Figure 5.12. The average algorithm computation time per image, seconds.

almost 10 seconds for H images and more than 40 seconds for F images. The computation time of the *Cyclops* does not increase that significantly. The *Cyclops* algorithm is approximately two times faster than SGBM and ELAS for half resolution images. For full resolution images, the speed up is even more significant, approximately three times faster than ELAS and four times faster than the SGBM algorithm.

The percentage of the time spent on algorithm steps is shown in Table 5.3. *Cyclops* spends more than 50% of computation time detecting MSER regions and more than 40% on computing disparity image values. This experiment was performed on the F resolution Vintage stereo image pair. Other image pairs and resolutions showed similar results. MSER regions can be detected in linear time [Nistér and Stewénius, 2008]. This is the main reason why the *Cyclops* computation time increases slower compared to ELAS and SGBM.

Table 5.3. Percentage of time spent on algorithm steps.

Algorithm step	Percentage of total time spent, %
MSER detection	50.18
Normalization	5.18
Matching	0.81
Disparity computation	43.83

The *Cyclops* algorithm not only shows a smaller algorithm computation time but also reduces the overall memory usage of the algorithm. This was tested on Vintage stereo image pair images. The results of the algorithm memory consumption are shown in Fig. 5.13.

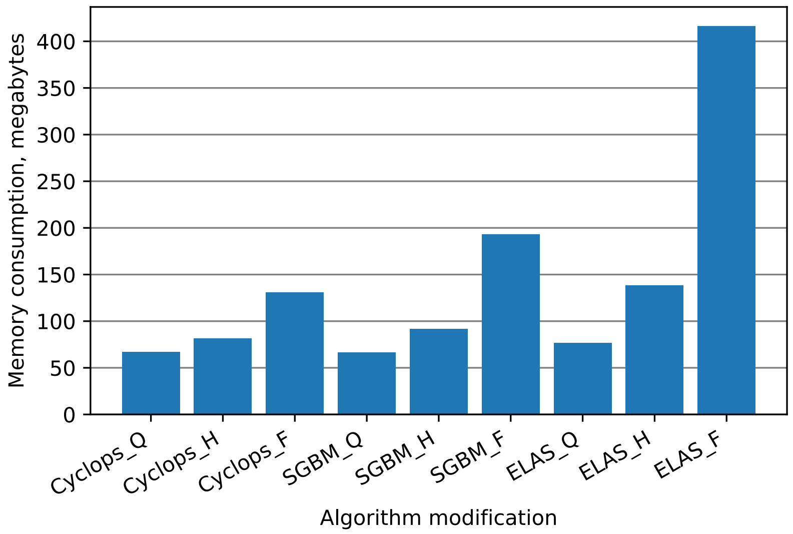


Figure 5.13. Algorithm total memory consumption, megabytes.

Overall, the *Cyclops* algorithm used less memory to process H and F resolution images. Overall, the ELAS algorithm exhibited more than 40% higher memory consumption. All the algorithms used almost the same amount of memory for Q resolution images. On H resolution images, *Cyclops* uses less memory than either SGBM or ELAS. This reduction in memory usage is even more significant on F resolution images, namely, 68.6% less memory than ELAS and 32.3% less memory than SGBM. Detailed results of the computation time and memory consumption are provided in Table 5.4.

Next, the average percentage of bad and invalid pixels in the produced disparity images is evaluated. The results of this experiment are shown in Figure 5.14.

The SGBM algorithm had a similar percentage of bad pixels across all the three resolution images. The percentage of invalid pixels increased approximately by 6% and 11% with the increasing resolution. The main reason is probably the fact that the algorithm parameters were constant across different resolution images. Changing parameters for different resolution images might change the results. The results provided for these algorithms in the Middlebury dataset use constant algorithm parameters.

The ELAS algorithm contained no invalid pixels because the default algorithm parameters for the Middlebury dataset performed disparity image hole filling. There was one exception where a full resolution image contained a single invalid pixel region that was not filled by the algorithm. The cause of this inconsistency was not investigated further. The percentage of bad pixels of the ELAS algorithm was slightly larger (4.7%)

Table 5.4. Algorithm computation time (seconds) and memory consumption (megabytes).

Algorithm name	Time, seconds	Memory consumption, megabytes
*Cyclops*_Q	1.08	67.17
*Cyclops*_H	4.2	81.66
*Cyclops*_F	16.22	130.68
SGBM_Q	1.01	66.36
SGBM_H	8.82	91.73
SGBM_F	69.67	193.07
ELAS_Q	1.99	76.51
ELAS_H	9.37	138.27
ELAS_F	47.04	416.02

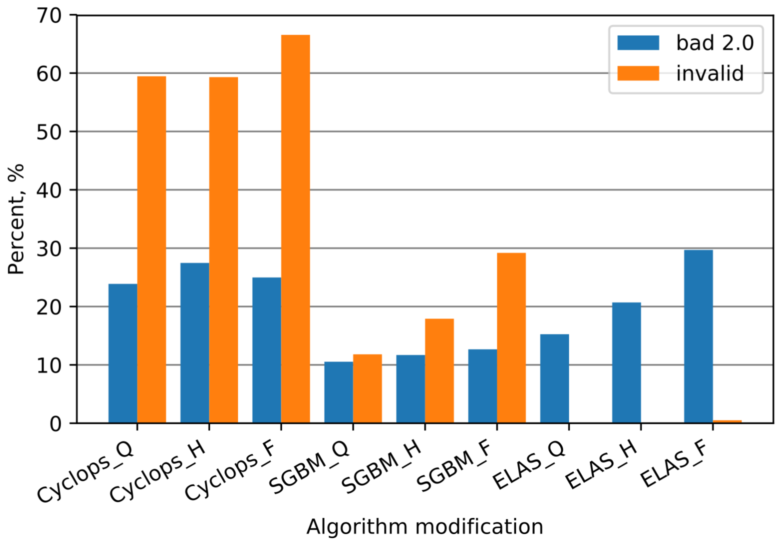


Figure 5.14. Average percentage of bad pixels whose error is larger than 2.0 (blue) and average percentage of invalid pixels (orange).

than SGBM and increased by 5% and 9% for higher resolution images.

The *Cyclops* algorithm has a high percentage of invalid pixels. There are two main reasons for this. First, the detected MSER regions do not cover large untextured regions. This problem was partially addressed by dividing the image into sub-images. Second, no post processing or hole filling algorithms were used to cover the invalid pixel areas. Adding such post-processing algorithms would lower or eliminate the invalid pixels but could also potentially increase the percentage of bad pixels. The percentage of bad pixels of the *Cyclops* algorithm was higher than the SGBM and ELAS algorithm for quarter and half size images. This metric did not change significantly for different resolution images.

Another important metric for evaluating the stereo matching algorithm quality is the average disparity error of bad pixels. The results of this evaluation are provided in Figure 5.15.

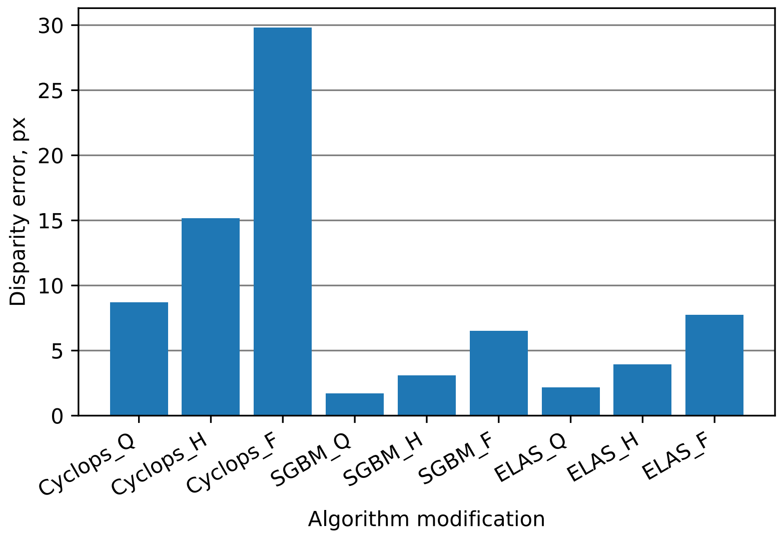


Figure 5.15. The average disparity error of bad pixels.

Both SGBM and ELAS algorithm showed very similar disparity error values. All the algorithms had an increasing disparity error when the image resolution increased. The average disparity error of the proposed algorithm was higher than that of SGBM and ELAS. The main source of error in the *Cyclops* algorithm in comparison with SGBM was that it was used to estimate the final disparity values. The main cause of this is that only the bounding boxes of the matched MSER regions are provided for the SGBM algorithm. The MSER regions usually do not contain enough edges or texture that could improve SGBM matching. This issue can be addressed by providing the SGBM algorithm with regions that are larger than the MSER bounding boxes. Another potential solution is to use a more accurate stereo matching algorithm in the final algorithm stage. The main purpose of the *Cyclops* algorithm is to reduce the search space of the algorithm used in the final disparity evaluation stage. The reduced search space makes it possible to use more advanced algorithms in the final disparity evaluation stage and have high overall algorithm computation frame rates. Detailed results are provided in Table 5.5.

Table 5.5. Percentage of bad pixels whose error is larger than 2.0, percentage of invalid pixels and average disparity error of bad pixels.

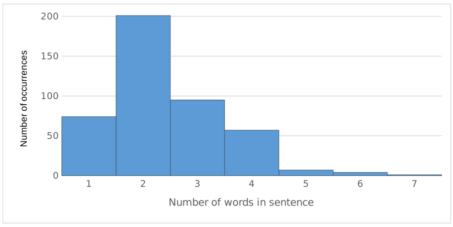
Algorithm name	Bad 2.0, %	Invalid, %	Average disparity error, px
*Cyclops*_Q	23.87	59.45	8.69
*Cyclops*_H	27.46	59.31	15.16
*Cyclops*_F	24.98	66.55	29.81
SGBM_Q	10.54	11.81	1.72
SGBM_H	11.69	17.91	3.09
SGBM_F	12.67	29.19	6.52
ELAS_Q	15.26	0	2.18
ELAS_H	20.70	0	3.94
ELAS_F	29.70	0.52	7.77

5.6. Text predictor experiments

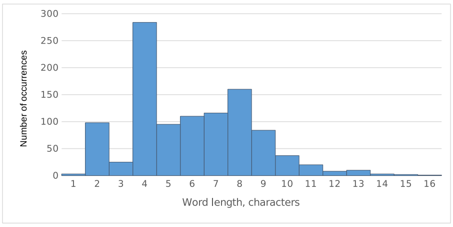
Text predictor experiments investigate whether a specialized text predictor might be more efficient for disabled people. This predictor is compared to the standard general purpose text predictor.

5.6.1. Text predictor experimental setup

The training data consisted of 439 statements (sentences) and 1056 words, of which 420 are unique. The corpora statistics are provided in Fig. 5.16. Longer statements usually describe full symptoms, for example, ‘reduced sensation in hands’. Short statements, on the other hand, mostly describe feelings, for example, ‘weakness’ or ‘intense pain’.



(a)



(b)

Figure 5.16. The statistics of the training data corpora. (a) sentence length histogram and (b) word length histogram.

The model was evaluated by using the same training data set because we only expect the user to communicate about these topics. For each word in the sentence, we measured the percentage of characters (POC) that had to be provided for the text predictor before the predictor returned the correct word in the list of predictions. For each evaluated word, the predictor was given up to $n-1$ words preceding the predicted word. This was done to make sure that the predictor can use the model with cardinality n when calculating new word predictions.

The lowest possible POC for any given word is 0%. We get 0% POC when no letters of a currently predicted word have been entered, and it is already shown in the text predictor predictions list. The highest possible POC, on the other hand, is 100%. In this case, all of the current word letters had to be entered, and it was still not returned by the text predictor. We should note that the lower POC values are better. After calculating the POC of each word in the test data set, we calculated the total percent of characters (TPOC) for the text predictor by using the weighted average formula. Each observation was weighted by the number of characters of that observation.

$$TPOC = \frac{\sum_{i=1}^{m} POC_i \times nch_i}{\sum_{i=1}^{m} nch_i} \tag{5.1}$$

Here, nch_i is the number of characters of the i-th word, and m is the number of words in the data set. We weighted the POC of each word according to the number of characters of that word. This is done to show that the POC of 50% for a 6-character word is better than the POC of 50% for a 2-character word. In the former case, the user could avoid typing 3 characters, whereas, in the latter case, the user only avoids typing 1 character. We also calculated the standard deviation (STD) of the TPOC measurements by using the weighted standard deviation formula.

$$STD = \sqrt{\frac{\sum_{i=1}^{m} nch_i \left(POC_i - TPOC\right)^2}{\frac{(M-1)}{M} \sum_{i=1}^{m} nch_i}} \tag{5.2}$$

Here, M is the number of words used during the testing, namely, 1056. During the experiments, we have also calculated confidence intervals (CI) at 95% confidence level.

5.6.2. Text prediction results

Three different N-gram text predictors were used during the experiments. First, we used the ‘standard’ N-gram language model that is provided with the Presage library [Vescovi, 2004]. This is a general purpose language model that is trained by using literature books. Second, this standard model was ‘adapted’ by using our specialized training data set, i.e., the model was further trained with our data set. The last model was a ‘specialized’ language model that was trained only on the specialized data-set.

Each model contains a data-set of all the unique N-grams that were extracted from the training data-set. Training is performed by finding these N-grams and calculating the number of their occurrences. This data can then be used during the prediction step. We should note that the uni-gram model is only a database of all unique words and their occurrence counts. All other N-gram models generate all possible permutations of n words in each training sentence. For example, if we have a training sentence ‘constant numbness of legs’, the following bi-grams will be generated: ‘constant numbness’,

‘numbness of’ and ‘of legs’. If a generated N-gram already exists in the model, only its count is incremented. The training of ‘adapted’ model is performed by adding new unique N-grams to the model and incrementing the counts for the already existing N-grams.

During the experiments, we calculate the POC for each word in the data-set. When calculating the POC of the first word in each sentence, the letters of that word are provided (one by one) until the predictor returns the word as one of predictions. For example, if the current word is ‘spine’, and we had to provide ‘spi’ before ‘spine’ appears as one of the predictor outputs, then the POC of this word would be 60%. Up to $n - 1$ words (here, n is the model cardinality) are provided as a context for the predictor, but these words do not affect the POC of the current word. POC can also be 0% for some words. There are two situations when this might happen. First, when the first word in a sentence is one of the most frequently found in the training data and is always returned in a list of predictions. Secondly, when, given the context, a word is predicted without providing any letters of that word.

One example sentence from the data-set is ‘sudden weakness in limbs’. This sentence was queried by using the 3-gram model, and the number of the generated predictions was set to 3. For the first word, we need to provide ‘sud’ and ‘sudden’ was among the three predictions resulting in the POC of 50%. For the second word, we only had to provide ‘sudden’, and ‘weakness’ was predicted, i.e., POC equals 0%. ‘in’ also had POC of 0% after providing ‘sudden weakness’. The final word had a POC of 20%, i.e., ‘weakness in l’ was provided for the predictor. The TPOC for this sentence would be 19.05%.

All the three models were evaluated by using the same specialized data-set. Each text predictor was configured to calculate two to six predictions. We should note that increasing the number of predictions will reduce the TPOC of the text predictor. However, it is preferable to have as few predictions as possible because it will make it easier for the user to select the desirable prediction. We believe that providing only one prediction or more than six predictions is impractical. The impact of the model type on the TPOC result is shown in Fig. 5.17.

As we can see, the ‘standard’ language model had very high TPOC levels. Adapting the language model with the specialized training data-set improved the text predictor performance significantly. The difference between the ‘adapted’ model and the ‘specialized’ model was less significant. Nevertheless, the user would have to enter almost two times fewer characters while using the ‘specialized’ model compared to the ‘adapted’ model. We should note that, during these experiments, 3-gram model cardinalities where used because this was the cardinality of the ‘standard’ language model. The results of this experiment are summarized in Table 5.6.

Another set of experiments was performed to study the impact of the model cardinality. We trained five language models that had the cardinalities from 1 to 5. All the models were trained and evaluated by using the specialized data-set. We should note that, for example, a model of cardinality 3 internally also contains models with

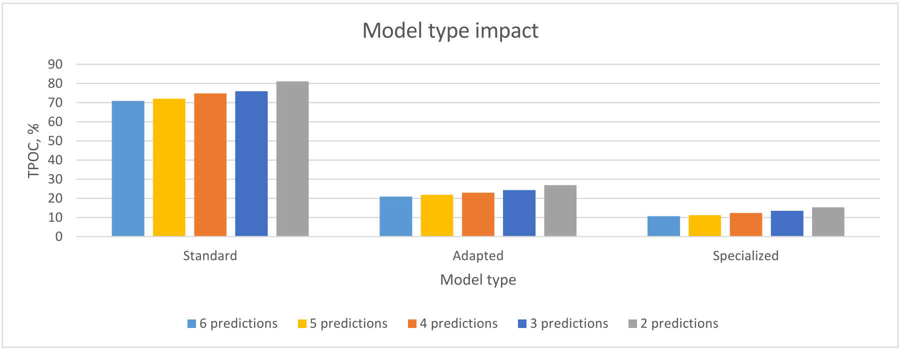


Figure 5.17. The TPOC of standard, adapted and specialized text predictors. Each predictor was configured to return 6 to 2 predictions.

Table 5.6. Model type impact results.

Number of predictions	Measurement	Model type		
		Standard	Adapted	Specialized
2	TPOC, %	81.05	26.9	15.31
	STD, %	25.61	24.93	16.98
	CI, %	1.54	1.5	1.02
3	TPOC, %	75.97	24.33	13.45
	STD, %	27.05	23.41	15.39
	CI, %	1.63	1.41	0.93
4	TPOC, %	74.81	22.93	12.29
	STD, %	28.05	22.72	14.41
	CI, %	1.69	1.37	0.87
5	TPOC, %	72	21.77	11.2
	STD, %	29.66	22.08	13.61
	CI, %	1.79	1.33	0.82
6	TPOC, %	70.89	20.85	10.72
	STD, %	30.31	21.54	13.25
	CI, %	1.83	1.3	0.8

all possible lower cardinalities. The results of these experiments are provided in Fig. 5.18.

As expected, increasing the model cardinality also improves its TPOC. The bi-gram model had a significantly improved TPOC compared to a uni-gram model. The 3-gram model showed substantially lower improvement. Higher cardinality models had almost no noticeable TPOC improvement. Detailed results are provided in Table 5.7.

This experiment clearly shows that expanding the model cardinality beyond 3 is not practical. It is of course worth noting that the majority of phrases used in the

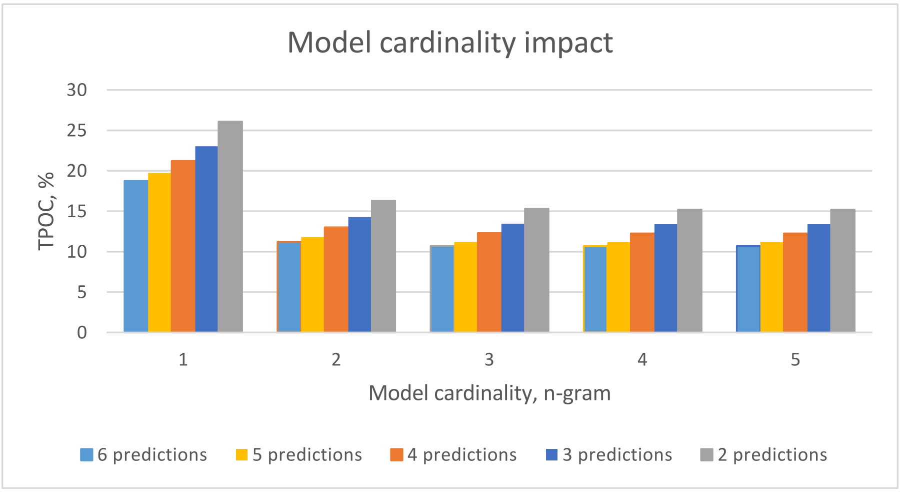


Figure 5.18. The TPOC of five language models with different cardinalities.

Table 5.7. Model cardinality impact results.

Model cardinality	Measurement	Number of predictions				
		2	3	4	5	6
1	TPOC, %	26.09	23.03	21.21	19.76	18.76
	STD, %	16.41	14.18	13.52	13.02	12.59
	CI, %	0.99	0.86	0.82	0.79	0.76
2	TPOC, %	16.31	14.28	13.01	11.8	11.23
	STD, %	16.7	15.23	14.31	13.55	13.22
	CI, %	1	0.92	0.86	0.82	0.8
3	TPOC, %	15.31	13.45	12.29	11.2	10.72
	STD, %	16.98	15.39	14.41	13.61	13.25
	CI, %	1.02	0.93	0.87	0.82	0.8
4	TPOC, %	15.21	13.39	12.24	11.15	10.7
	STD, %	17	15.39	14.41	13.61	13.25
	CI, %	1.03	0.93	0.87	0.82	0.8
4	TPOC, %	15.21	13.39	12.24	11.15	10.7
	STD, %	17	15.39	14.41	13.61	13.25
	CI, %	1.03	0.93	0.87	0.82	0.8

training data set contained 3-word phrases. It is therefore recommended to choose the model cardinality by examining the training data. The sentence length histogram is a convenient method to obtain this data. Using a cardinality higher than 3 is, however, rarely necessary.

The final experiment examined how word lengths influence the performance of the specialized text predictor. A specialized 3-gram text predictor was used during this experiment. The predictor was configured to only return two predictions. The test data-set contained words that had variable lengths from 1 to 15 characters. We should note that during this experiment, we calculated POC instead of TPOC, i.e., results were not weighted by word lengths. The experiment results are shown in Fig. 5.19.

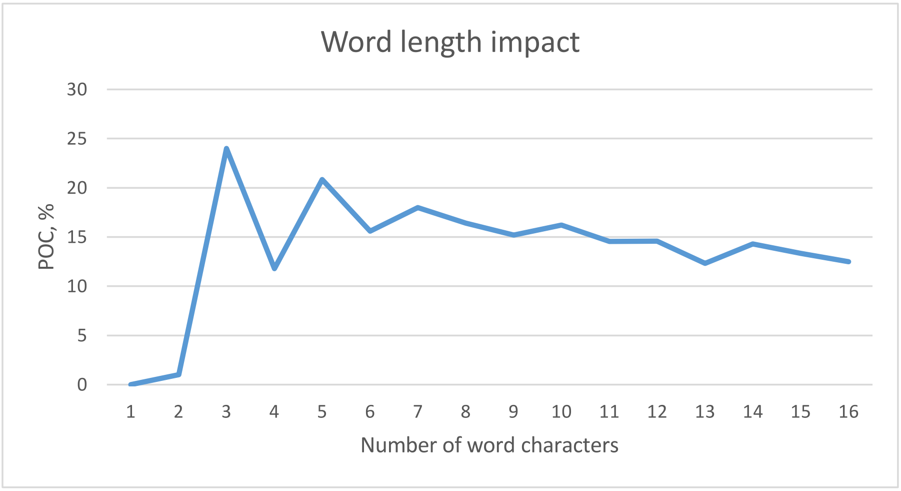


Figure 5.19. Word length impact on text predictor performance.

The experiment has demonstrated that the text predictor has a sufficiently good performance even for words of various length. This shows that the user will be able to type in long phrases efficiently. The POC curve jumps are caused by the varying amounts of different length words present in the training data-set (see Fig. 5.16).

5.7. Experiments summary

This chapter presented the experiments performed in this thesis. The first section describes the experimental methodology and the experiment participant group. The presented experiments can be split into two categories, namely, experiments with participants and experiments used to evaluate the performance of the proposed algorithms.

Participant experiments were used to evaluate the developed multifunctional UI and projection mapping UI. All the remaining experiments were performed to evaluate the proposed algorithms and involved no participants. Camera control experiments used a PointGrey camera to evaluate the algorithm. The Middlebury Stereo Evaluation data-set was used for stereo algorithm evaluation. Finally, a data-set generated from medical questionnaires was used for the text predictor evaluation.

6. CONCLUSIONS

1. This thesis contains a comprehensive analysis of already existing consumer grade AT solutions and HCI methods. The majority of existing solutions are intended for cases of single use. As a result, the user has to learn to use multiple systems to perform different tasks. Moreover, the users would have to change a system as their abilities change as well.

2. The limitations of the presently existing solutions have been addressed by designing a multifunctional UI architecture optimized for system adaptability. The proposed architecture was used to implement a multifunctional user interface that integrates multiple assistive devices. The developed system has been tested by 11 participants. Each participant used the system with a number of assistive devices. The experiments revealed that device suitability varied significantly among the participants. Some participants were not able to use some devices, but every participant managed to use at least one device efficiently. This confirms the advantage of a multiple device adaptive assistive system. Moreover, the participants were able to use the system up to 60% faster with two devices used simultaneously.

3. Automatic projection mapping is a novel, low-cost user interface for object selection. This interface combines an RGB-D camera with a projector to create a real time interactive UX. Main proposed method limitation is projected view brightness in bright environments. These limitations can be addressed with novel HMD devices as they are becoming more widespread. The presented algorithms are suitable for HMD devices. The projection mapping UI has been implemented and integrated into the multifunctional user interface. The experiment participants evaluated the current system by using the PSSUQ questionnaire. The UI received an average score of 2.84 points on the scale of 1 to 7 (where 1 is the highest score).

4. The camera control algorithm adjusts the sensor gain and shutter speed parameters in real time. This algorithm is used to ensure the reliability of the image processing algorithms. This thesis demonstrated that the camera image exposure process is mostly linear when one of the controlled parameters is fixed. A cascaded PID controller is used to control two output signals simultaneously. The non-linearity stemming from the camera parameter interdependence was eliminated by applying Proportional Integral Derivative (PID) gain scheduling techniques. We have demonstrated that the proposed algorithm can adapt to a sudden lighting change in 50% amount of time taken by the manufacturer-provided camera control algorithm. Further algorithm performance improvements could be achieved by using automatic PID tuning techniques.

5. A stereo camera is used for environment sensing and object detection. Such a camera is not sensitive to direct sunlight and can produce high resolution depth images of the environment. This thesis has presented an efficient stereo matching algorithm suitable for embedded computers. The proposed algorithm computes depth images up to two times faster than state-of-the-art efficient stereo matching algorithms. At the same time, this algorithm uses at least 32% less memory compared to the other evaluated algorithms. Future algorithm developments will concentrate on reducing disparity errors produced by the second stage of the algorithm.

6. Text predictors are used to reduce the text input time. This thesis has demonstrated that a conversation-specific text predictor is more efficient than a general purpose text predictor. The proposed specialized predictor has been trained by using a data-set created from medical questionnaire questions. This predictor works up to 60% more efficiently than the standard text predictor. This approach can be used to create several topic-specific predictors and switching them during the system operation.

7. The proposed algorithms have been integrated into a multifunctional UI that enables users to perform novel tasks. In recent years, the amount of internet-connected devices has grown significantly. Such devices can be integrated into the proposed system and enable users to perform more tasks independently. It is recommended to choose the combination of devices for each user individually. This combination should be constantly updated as the users' abilities change.

Bibliography

[Ajiboye et al., 2017] Ajiboye, A. B., Willett, F. R., Young, D. R., Memberg, W. D., Murphy, B. A., Miller, J. P., Walter, B. L., Sweet, J. A., Hoyen, H. A., Keith, M. W., et al. (2017). Restoration of reaching and grasping movements through brain-controlled muscle stimulation in a person with tetraplegia: a proof-of-concept demonstration. *The Lancet*, 389(10081):1821–1830.

[Al-Rahayfeh and Faezipour, 2013] Al-Rahayfeh, A. and Faezipour, M. (2013). Eye tracking and head movement detection: A state-of-art survey. *IEEE journal of translational engineering in health and medicine*, 1:2100212–2100212.

[Angelo, 2000] Angelo, J. (2000). Factors affecting the use of a single switch with assistive technology devices. *Journal of rehabilitation research and development*, 37(5):591.

[Anson et al., 2006] Anson, D., Moist, P., Przywara, M., Wells, H., Saylor, H., and Maxime, H. (2006). The effects of word completion and word prediction on typing rates using on-screen keyboards. *Assistive technology*, 18(2):146–154.

[Bates et al., 2014] Bates, G., Tabrizi, S., and Jones, L. (2014). *Huntington's disease*. Number 64. Oxford University Press (UK).

[Benko et al., 2012] Benko, H., Jota, R., and Wilson, A. (2012). Miragetable: freehand interaction on a projected augmented reality tabletop. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 199–208. ACM.

[Billinghurst et al., 2015] Billinghurst, M., Clark, A., Lee, G., et al. (2015). A survey of augmented reality. *Foundations and Trends® in Human–Computer Interaction*, 8(2-3):73–272.

[Bradski and Kaehler, 2008] Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.".

[Cao et al., 2005] Cao, G., Nie, J.-Y., and Bai, J. (2005). Integrating word relationships into language models. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 298–305. ACM.

[Christensen et al., 2013] Christensen, H., Casanueva, I., Cunningham, S., Green, P., and Hain, T. (2013). homeservice: Voice-enabled assistive technology in the home using cloud-based automatic speech recognition. In *4th Workshop on Speech and Language Processing for Assistive Technologies*, pages 29–34.

[Cook and Polgar, 2014] Cook, A. M. and Polgar, J. M. (2014). *Assistive Technologies-E-Book: Principles and Practice*. Elsevier Health Sciences.

[Cunha and da Silva, 2017] Cunha, R. and da Silva, R. L. d. S. (2017). Virtual reality as an assistive technology to support the cognitive development of people with intellectual and multiple disabilities. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 28, page 987.

[Dalmaijer, 2014] Dalmaijer, E. (2014). Is the low-cost eyetribe eye tracker any good for research? Technical report, PeerJ PrePrints.

[Davis et al., 2016] Davis, T., Wark, H., Hutchinson, D., Warren, D., O'Neill, K., Scheinblum, T., Clark, G., Normann, R., and Greger, B. (2016). Restoring motor control and sensory feedback in people with upper extremity amputations using arrays of 96 microelectrodes implanted in the median and ulnar nerves. *Journal of neural engineering*, 13(3):036001.

[Efthimiou et al., 1981] Efthimiou, J., Gordon, W., Sell, G., and Stratford, C. (1981). Electronic assistive devices: their impact on the quality of life of high level quadriplegic persons. *Archives of physical medicine and rehabilitation*, 62(3):131–134.

[Fischler and Bolles, 1987] Fischler, M. A. and Bolles, R. C. (1987). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision*, pages 726–740. Elsevier.

[Flachberger et al., 1994] Flachberger, C., Panek, P., and Zagler, W. L. (1994). Autonomy—a flexible and easy-to-use assistive system to support the independence of handicapped and elderly persons. In *International Conference on Computers for Handicapped Persons*, pages 65–75. Springer.

[Flachberger et al., 1995] Flachberger, C., Panek, P., and Zagler, W. L. (1995). Compose autonomy!–an adaptable user interface for assistive technology. In *Proceedings of the 2nd TIDE Congress (The European Context for Assistive Technology), IOS Press, Paris*, pages 413–416.

[Forssén, 2007] Forssén, P.-E. (2007). Maximally stable colour regions for recognition and matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

[Garay-Vitoria and Abascal, 2006] Garay-Vitoria, N. and Abascal, J. (2006). Text prediction systems: a survey. *Universal Access in the Information Society*, 4(3):188–203.

[Geiger et al., 2010] Geiger, A., Roser, M., and Urtasun, R. (2010). Efficient large-scale stereo matching. In *Asian conference on computer vision*, pages 25–38. Springer.

[Gelšvartas et al., 2016a] Gelšvartas, J., Simonavičius, H., Lauraitis, A., Maskeliūnas, R., Cimmperman, P., and Serafinavičius, P. (2016a). Realtime stereo matching using maximally stable extremal regions. *Transylvanian Review*, 24(1):15–27.

[Gelšvartas et al., 2015] Gelšvartas, J., Simutis, R., and Maskeliūnas, R. (2015). Multifunctional user interface implementation details and evaluation. In *Methods and Models in Automation and Robotics (MMAR), 2015 20th International Conference on*, pages 501–504. IEEE.

[Gelšvartas et al., 2016b] Gelšvartas, J., Simutis, R., and Maskeliūnas, R. (2016b). User adaptive text predictor for mentally disabled huntington's patients. *Computational intelligence and neuroscience*, 2016:2.

[Gelšvartas et al., 2018] Gelšvartas, J., Simutis, R., and Maskeliūnas, R. (2018). Projection mapping user interface for disabled people. *Journal of healthcare engineering*, 2018.

[Gopura et al., 2013] Gopura, R., Bandara, D., Gunasekara, J., and Jayawardane, T. (2013). Recent trends in emg-based control methods for assistive robots. In *Electrodiagnosis in new frontiers of clinical research*. InTech.

[Grau et al., 2006] Grau, S., Segarra, E., Sanchis, E., Garcia, F., and Hurtado, L. F. (2006). Incorporating semantic knowledge to the language model in a speech understanding system. *IV Jornadas en Tecnologia del Habla, Zaragoza, Spain*, pages 145–148.

[Han and Wang, 2016] Han, Z. and Wang, J. (2016). Feature fusion algorithm for multimodal emotion recognition from speech and facial expression signal. In *MATEC Web of Conferences*, volume 61, page 03012. EDP Sciences.

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

[Hefner et al., 2017] Hefner, J., O'Connor, K., and Binder, D. (2017). Virtual reality in mechanical ventilation weaning after spinal cord injury. *Iproceedings*, 3(1):e2.

[Hernandez-Juarez et al., 2016] Hernandez-Juarez, D., Chacón, A., Espinosa, A., Vázquez, D., Moure, J. C., and López, A. M. (2016). Embedded real-time stereo estimation via semi-global matching on the gpu. *Procedia Computer Science*, 80:143–153.

[Hinterstoisser et al., 2011] Hinterstoisser, S., Holzer, S., Cagniart, C., Ilic, S., Konolige, K., Navab, N., and Lepetit, V. (2011). Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 858–865. IEEE.

[Hirschmuller, 2008] Hirschmuller, H. (2008). Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341.

[Hochberg et al., 2012] Hochberg, L. R., Bacher, D., Jarosiewicz, B., Masse, N. Y., Simeral, J. D., Vogel, J., Haddadin, S., Liu, J., Cash, S. S., van der Smagt, P., et al. (2012). Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–375.

[Hondori et al., 2013] Hondori, H. M., Khademi, M., Dodakian, L., Cramer, S. C., and Lopes, C. V. (2013). A spatial augmented reality rehab system for post-stroke hand rehabilitation. In *MMVR*, pages 279–285.

[Hua et al., 2014] Hua, L., Wang, S., and Gong, Y. (2014). Text prediction on structured data entry in healthcare. *Applied clinical informatics*, 5(01):249–263.

[Huang et al., 2014] Huang, X., Baker, J., and Reddy, R. (2014). A historical perspective of speech recognition. *Communications of the ACM*, 57(1):94–103.

[Huo and Wang, 2008] Huo, X. and Wang, J. (2008). Introduction and preliminary evaluation of the tongue drive system: wireless tongue-operated assistive technology for people with little or no upper-limb function. *Journal of rehabilitation research and development*, 45(6):921.

[Hwang et al., 2014] Hwang, C.-S., Weng, H.-H., Wang, L.-F., Tsai, C.-H., and Chang, H.-T. (2014). An eye-tracking assistive device improves the quality of life for als patients and reduces the caregivers' burden. *Journal of motor behavior*, 46(4):233–238.

[Istance et al., 2008] Istance, H., Bates, R., Hyrskykari, A., and Vickers, S. (2008). Snap clutch, a moded approach to solving the midas touch problem. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 221–228. ACM.

[Ivanavičius et al., 2018] Ivanavičius, A., Simonavičius, H., Gelšvartas, J., Lauraitis, A., Maskeliūnas, R., Cimmperman, P., and Serafinavičius, P. (2018). Real-time cuda-based stereo matching using cyclops2 algorithm. *EURASIP Journal on Image and Video Processing*, 2018(1):12.

[Jacob, 1991] Jacob, R. J. (1991). The use of eye movements in human-computer interaction techniques: what you look at is what you get. *ACM Transactions on Information Systems (TOIS)*, 9(2):152–169.

[Ka et al., 2012] Ka, H. W., Simpson, R., and Chung, Y. (2012). Intelligent single switch wheelchair navigation. *Disability and Rehabilitation: Assistive Technology*, 7(6):501–506.

[Kim et al., 2010] Kim, J., Huo, X., and Ghovanloo, M. (2010). Wireless control of smartphones with tongue motion using tongue drive assistive technology. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 5250–5253. IEEE.

[Kim et al., 2012] Kim, J., Huo, X., Minocha, J., Holbrook, J., Laumann, A., and Ghovanloo, M. (2012). Evaluation of a smartphone platform as a wireless interface between tongue drive system and electric-powered wheelchairs. *IEEE transactions on biomedical engineering*, 59(6):1787–1796.

[Kim et al., 2016] Kim, S., Kang, S.-J., and Kim, Y. H. (2016). Real-time stereo matching using extended binary weighted aggregation. *Digital Signal Processing*, 53:51–61.

[Kimura et al., 2007] Kimura, M., Mochimaru, M., and Kanade, T. (2007). Projector calibration using arbitrary planes and calibrated camera. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–2. IEEE.

[Kowalczuk et al., 2013] Kowalczuk, J., Psota, E. T., and Perez, L. C. (2013). Real-time stereo matching on cuda using an iterative refinement method for adaptive support-weight correspondences. *IEEE transactions on circuits and systems for video technology*, 23(1):94–104.

[Krishnamurthy and Ghovanloo, 2006] Krishnamurthy, G. and Ghovanloo, M. (2006). Tongue drive: A tongue operated magnetic sensor based wireless assistive technology for people with severe disabilities. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4–pp. IEEE.

[Laurutis et al., 2010] Laurutis, V., Niauronis, S., and Zemblys, R. (2010). Alternative computer cursor shifts for large amplitude eyesight jumps. *Elektronika Ir Elektrotechnika*, 105(9):61–64.

[Leeb et al., 2010] Leeb, R., Sagha, H., Chavarriaga, R., et al. (2010). Multimodal fusion of muscle and brain signals for a hybrid-bci. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 4343–4346. IEEE.

[Lewis, 1995] Lewis, J. R. (1995). Ibm computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7(1):57–78.

[Liang et al., 2008] Liang, C.-H., Hung, W.-S., Hsieh, M.-C., Wu, C.-M., and Luo, C.-H. (2008). A multi-agent based architecture for an assistive user interface of intelligent home environment control. In *Intelligent Systems Design and Applications, 2008. ISDA'08. Eighth International Conference on*, volume 1, pages 335–338. IEEE.

[Lidal et al., 2007] Lidal, I. B., Huynh, T. K., and Biering-Sørensen, F. (2007). Return to work following spinal cord injury: a review. *Disability and rehabilitation*, 29(17):1341–1375.

[Lin et al., 2006] Lin, C.-S., Ho, C.-W., Chen, W.-C., Chiu, C.-C., and Yeh, M.-S. (2006). Powered wheelchair controlled by eye-tracking system. *Optica Applicata*, 36:2–3.

[Lu and Wen, 2015] Lu, L. and Wen, J. T. (2015). Human-robot cooperative control for mobility impaired individuals. In *American Control Conference (ACC), 2015*, pages 447–452. IEEE.

[Luo et al., 2016] Luo, W., Schwing, A. G., and Urtasun, R. (2016). Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703.

[Majaranta and Räihä, 2002] Majaranta, P. and Räihä, K.-J. (2002). Twenty years of eye typing: systems and design issues. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 15–22. ACM.

[Manns and Chad, 2001] Manns, P. J. and Chad, K. E. (2001). Components of quality of life for persons with a quadriplegic and paraplegic spinal cord injury. *Qualitative Health Research*, 11(6):795–811.

[Marasco et al., 2018] Marasco, P. D., Hebert, J. S., Sensinger, J. W., Shell, C. E., Schofield, J. S., Thumser, Z. C., Nataraj, R., Beckler, D. T., Dawson, M. R., Blustein, D. H., et al. (2018). Illusory movement perception improves motor control for prosthetic hands. *Science translational medicine*, 10(432):eaao6990.

[Maskeliunas et al., 2016] Maskeliunas, R., Damasevicius, R., Martisius, I., and Vasiljevas, M. (2016). Consumer-grade eeg devices: are they usable for control tasks? *PeerJ*, 4:e1746.

[Massanes et al., 2010] Massanes, F., Cadennes, M., and Brankov, J. G. (2010). Cuda implementation of a block-matching algorithm for multiple gpu cards. *Journal of Electronic Imaging*, pages 1–17.

[Matas et al., 2004] Matas, J., Chum, O., Urban, M., and Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767.

[Mayer et al., 2016] Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., and Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048.

[McGraw et al., 2016] McGraw, I., Prabhavalkar, R., Alvarez, R., Arenas, M. G., Rao, K., Rybach, D., Alsharif, O., Sak, H., Gruenstein, A., Beaufays, F., et al. (2016). Personalized speech recognition on mobile devices. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 5955–5959. IEEE.

[Menze and Geiger, 2015] Menze, M. and Geiger, A. (2015). Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070.

[Mikolov et al., 2013] Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

[Moreno and Taubin, 2012] Moreno, D. and Taubin, G. (2012). Simple, accurate, and robust projector-camera calibration. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pages 464–471. IEEE.

[Nistér and Stewénius, 2008] Nistér, D. and Stewénius, H. (2008). Linear time maximally stable extremal regions. In *European Conference on Computer Vision*, pages 183–196. Springer.

[Nourani-Vatani and Roberts, 2007] Nourani-Vatani, N. and Roberts, J. M. (2007). Automatic camera exposure control. In *Proceedings of the Australasian Conference on Robotics and Automation 2007*, pages 1–6. Australian Robotics & Automation Association ARAA.

[Ong et al., 2011] Ong, S.-K., Shen, Y., Zhang, J., and Nee, A. Y. (2011). Augmented reality in assistive technology and rehabilitation engineering. In *Handbook of augmented reality*, pages 603–630. Springer.

[Palaz et al., 2015] Palaz, D., Doss, M. M., and Collobert, R. (2015). Convolutional neural networks-based continuous speech recognition using raw speech signal. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4295–4299. IEEE.

[Pasqualotto et al., 2015] Pasqualotto, E., Matuz, T., Federici, S., Ruf, C. A., Bartl, M., Olivetti Belardinelli, M., Birbaumer, N., and Halder, S. (2015). Usability and workload of access technology for people with severe motor impairment: a comparison of brain-computer interfacing and eye tracking. *Neurorehabilitation and neural repair*, 29(10):950–957.

[Pauly and Sankar, 2015] Pauly, L. and Sankar, D. (2015). A novel method for eye tracking and blink detection in video frames. In *Computer Graphics, Vision and*

Information Security (CGVIS), 2015 IEEE International Conference on, pages 252–257. IEEE.

[Perdikis et al., 2018] Perdikis, S., Tonin, L., Saeedi, S., Schneider, C., and Millán, J. d. R. (2018). The cybathlon bci race: Successful longitudinal mutual learning with two tetraplegic users. *PLoS biology*, 16(5):e2003787.

[Polacek et al., 2017] Polacek, O., Sporka, A. J., and Slavik, P. (2017). Text input for motor-impaired people. *Universal Access in the Information Society*, 16(1):51–72.

[Poppinga et al., 2008] Poppinga, J., Vaskevicius, N., Birk, A., and Pathak, K. (2008). Fast plane detection and polygonalization in noisy 3d range images. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3378–3383. IEEE.

[Raudonis et al., 2009] Raudonis, V., Simutis, R., and Narvydas, G. (2009). Discrete eye tracking for medical applications. In *Applied Sciences in Biomedical and Communication Technologies, 2009. ISABEL 2009. 2nd International Symposium on*, pages 1–6. IEEE.

[Rojhani et al., 2017] Rojhani, S., Stiens, S. A., and Recio, A. C. (2017). Independent sailing with high tetraplegia using sip and puff controls: integration into a community sailing center. *The journal of spinal cord medicine*, 40(4):471–480.

[Rubin and Chisnell, 2008] Rubin, J. and Chisnell, D. (2008). *Handbook of usability testing: howto plan, design, and conduct effective tests*. John Wiley & Sons.

[Rusu, 2009] Rusu, R. B. (2009). *Semantic 3D Object Maps for Everyday Manipulation in Human Living environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany.

[Rusu et al., 2009] Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3212–3217. IEEE.

[Rusu et al., 2010] Rusu, R. B., Bradski, G., Thibaux, R., and Hsu, J. (2010). Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2155–2162. IEEE.

[Rusu and Cousins, 2011] Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4, Shanghai, China.

[Saldert et al., 2010] Saldert, C., Eriksson, E., Petersson, K., and Hartelius, L. (2010). Interaction in conversation in huntington´s disease: An activity-based analysis and the conversation partner's view of change. *Journal of Interactional Research in Communication Disorders*, 1(2):169–197.

[Sandnes, 2015] Sandnes, F. E. (2015). Reflective text entry: a simple low effort predictive input method based on flexible abbreviations. *Procedia Computer Science*, 67:105–112.

[Scharstein et al., 2014] Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nešić, N., Wang, X., and Westling, P. (2014). High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition*, pages 31–42. Springer.

[Schwartz, 2015] Schwartz, A. (2015). Recent advances in brain-controlled prosthetics for paralysis: Friday keynote. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pages 313–313. ACM.

[Segal et al., 2009] Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-icp. In *Robotics: science and systems*, volume 2, page 435.

[Shin et al., 2014] Shin, J.-H., Ryu, H., and Jang, S. H. (2014). A task-specific interactive game-based virtual reality rehabilitation system for patients with stroke: a usability test and two clinical experiments. *Journal of neuroengineering and rehabilitation*, 11(1):32.

[Shirvaikar, 2004] Shirvaikar, M. V. (2004). An optimal measure for camera focus and exposure. In *System Theory, 2004. Proceedings of the Thirty-Sixth Southeastern Symposium on*, pages 472–475. IEEE.

[Sledevič and Navakauskas, 2013] Sledevič, T. and Navakauskas, D. (2013). Fpga based fast lithuanian isolated word recognition system. In *EUROCON, 2013 IEEE*, pages 1630–1636. IEEE.

[Spangenberg et al., 2013] Spangenberg, R., Langner, T., and Rojas, R. (2013). Weighted semi-global matching and center-symmetric census transform for robust driver assistance. In *International Conference on Computer Analysis of Images and Patterns*, pages 34–41. Springer.

[Stowers et al., 2011] Stowers, J., Hayes, M., and Bainbridge-Smith, A. (2011). Altitude control of a quadrotor helicopter using depth map from microsoft kinect sensor. In *Mechatronics (ICM), 2011 IEEE International Conference on*, pages 358–362. IEEE.

[Sueishi et al., 2015] Sueishi, T., Oku, H., and Ishikawa, M. (2015). Robust high-speed tracking against illumination changes for dynamic projection mapping. In *Virtual Reality (VR), 2015 IEEE*, pages 97–104. IEEE.

[Tam et al., 2011] Tam, W.-K., Tong, K.-y., Meng, F., and Gao, S. (2011). A minimal set of electrodes for motor imagery bci to control an assistive device in chronic stroke subjects: a multi-session study. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 19(6):617–627.

[Tombari et al., 2010] Tombari, F., Salti, S., and Di Stefano, L. (2010). Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer.

[Trost et al., 2005] Trost, H., Matiasek, J., and Baroni, M. (2005). The language component of the fasty text prediction system. *Applied Artificial Intelligence*, 19(8):743–781.

[Tsui et al., 2007] Tsui, C. S. L., Jia, P., Gan, J. Q., Hu, H., and Yuan, K. (2007). Emg-based hands-free wheelchair control with eog attention shift detection. In *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, pages 1266–1271. IEEE.

[Van Den Bosch, 2006] Van Den Bosch, A. (2006). Scalable classification-based word prediction and confusible correction. *Traitement Automatique des Langues*, 46(2):39–63.

[Venkatagiri, 1993] Venkatagiri, H. (1993). Efficiency of lexical prediction as a communication acceleration technique. *Augmentative and Alternative Communication*, 9(3):161–167.

[Verspoor et al., 2008] Verspoor, K. et al. (2008). A semantics-enhanced language model for unsupervised word sense disambiguation. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 287–298. Springer.

[Vescovi, 2004] Vescovi, M. (2004). Soothsayer: un sistema multi-sorgente per la predizione del testo. *Master's thesis. Politecnico di Milano, Dipartimento di elettronica e informazione*, pages 1–137.

[Vlissides et al., 1995] Vlissides, J., Helm, R., Johnson, R., and Gamma, E. (1995). Design patterns: Elements of reusable object-oriented software. *Reading: Addison-Wesley*, 49(120):11.

[Wandmacher and Antoine, 2008] Wandmacher, T. and Antoine, J.-Y. (2008). Methods to integrate a language model with semantic information for a word prediction component. *arXiv preprint arXiv:0801.4716*, pages 1–10.

[Westgren and Levi, 1998] Westgren, N. and Levi, R. (1998). Quality of life and traumatic spinal cord injury. *Archives of physical medicine and rehabilitation*, 79(11):1433–1439.

[Wolfgang, 1994] Wolfgang, P. (1994). Design patterns for object-oriented software development. *Reading Mass*, page 15.

[Wolpaw et al., 2000] Wolpaw, J. R., Birbaumer, N., Heetderks, W. J., McFarland, D. J., Peckham, P. H., Schalk, G., Donchin, E., Quatrano, L. A., Robinson, C. J., Vaughan, T. M., et al. (2000). Brain-computer interface technology: a review of the first international meeting. *IEEE transactions on rehabilitation engineering*, 8(2):164–173.

[Yang et al., 2015] Yang, L., Normand, J.-M., and Moreau, G. (2015). Local geometric consensus: a general purpose point pattern-based tracking algorithm. *IEEE transactions on visualization and computer graphics*, 21(11):1299–1308.

[Yang et al., 2016] Yang, L., Normand, J.-M., and Moreau, G. (2016). Practical and precise projector-camera calibration. In *Mixed and Augmented Reality (ISMAR), 2016 IEEE International Symposium on*, pages 63–70. IEEE.

[Zbontar and LeCun, 2016] Zbontar, J. and LeCun, Y. (2016). Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2.

[Zhang et al., 2015] Zhang, C., Li, Z., Cheng, Y., Cai, R., Chao, H., and Rui, Y. (2015). Meshstereo: A global stereo model with mesh alignment regularization for view interpolation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2057–2065.

[Zhang, 2000] Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334.

THESIS-RELATED PUBLICATIONS OF THE AUTHOR

ISI Web of Science journals with impact factor

1. Ivanavičius, A., Simonavičius, H., Gelšvartas, J., Lauraitis, A., Maskeliūnas, R., Cimmperman, P., & Serafinavičius, P. (2018). Real-time CUDA-based stereo matching using Cyclops2 algorithm. EURASIP Journal on Image and Video Processing, 2018(1), 12.

2. Gelšvartas, J., Simutis, R., & Maskeliūnas, R. (2018). Projection Mapping User Interface for Disabled People. Journal of healthcare engineering, 2018.

3. Gelšvartas, J., Simutis, R., & Maskeliūnas, R. (2016). User adaptive text predictor for mentally disabled Huntington's patients. Computational intelligence and neuroscience, 2016, 2.

Other scientific journals

1. Gelšvartas, J., Simonavičius, H., Lauraitis, A., Maskeliūnas, R., Cimmperman, P., & Serafinavičius, P. (2016). Realtime Stereo Matching Using Maximally Stable Extremal Regions. Transylvanian Review, (1).

Conference proceedings

1. Gelšvartas, J., Simutis, R., & Maskeliūnas, R. (2018). Augmented reality object selection user interface for people with severe disabilities. 4th International Conference on Information and Communication Technologies for Ageing Well and e-Health (pp. 156-160).

2. Gelšvartas, J., Simutis, R., & Maskeliūnas, R. (2015, August). Multifunctional user interface implementation details and evaluation. In Methods and Models in Automation and Robotics (MMAR), 2015 20th International Conference on (pp. 501-504). IEEE.

3. Gelšvartas, J., Simutis, R., & Maskeliūnas, R. (2015, October). Text Predictor for Lithuanian Language. In International Conference on Information and Software Technologies (pp. 460-468). Springer, Cham.

4. Gelšvartas, J., Simutis, R., & Maskeliūnas, R. (2014). Multifunctional user interface to control robotic arm for paralyzed people. Electrical and Control Technologies, 22.

5. Gelšvartas, J., Lauraitis, A., Simutis, R., & Maskeliūnas, R. Review of assistive technologies for disabled people. Biomedical engineering 2016, 20(1).

A. APPENDIX

In this section, we present the detailed results of the multifunctional user interface experiments presented in Section 5.2. The mean, Standard deviation (STD) and the confidence intervals (CI) at 95% confidence level of device input times averaged over all of the experiments are provided in Table A.1. More detailed information for each participant is provided in Table A.2.

Table A.1. Device input time average over all of the experiments.

Experiment	Measurement	Device		
		Speech recognition	Sip/Puff	Eye tracker
Average over all participants	Mean time, s	34.09	12.94	9.50
	STD, s	17.85	6.71	14.21
	CI, s	3.34	1.25	2.66

Table A.2. Per-user statistics of the input time for each device.

Participant	Measurement	Device		
		Speech recognition	Sip/Puff	Eye tracker
1	Mean time, s	29.44	12.08	8.88
	STD, s	6.12	1.60	4.44
	CI, s	3.80	0.99	2.75
2	Mean time, s	26.78	12.73	24.14
	STD, s	10.60	1.52	32.16
	CI, s	6.57	0.94	19.94
3	Mean time, s	42.25	10.76	4.72
	STD, s	24.99	5.28	0.81
	CI, s	15.49	3.27	0.50
4	Mean time, s	39.28	10.64	4.20
	STD, s	3.47	0.54	0.27
	CI, s	2.15	0.34	0.17
5	Mean time, s	20.28	8.43	4.31
	STD, s	1.39	0.37	0.17
	CI, s	0.86	0.23	0.10
6	Mean time, s	60.66	8.91	7.63
	STD, s	36.95	1.73	3.69
	CI, s	22.90	1.07	2.29
7	Mean time, s	28.20	20.66	N/A
	STD, s	4.30	12.14	N/A
	CI, s	2.67	7.53	N/A
8	Mean time, s	40.82	13.04	10.73
	STD, s	18.20	1.38	15.44
	CI, s	11.28	0.85	9.57
9	Mean time, s	35.84	11.72	7.47
	STD, s	15.38	0.73	6.24
	CI, s	9.53	0.45	3.87
10	Mean time,	30.26	12.89	26.04
	STD, s	4.33	0.84	22.20
	CI, s	2.68	0.52	13.76
11	Mean time, s	35.91	26.84	N/A
	STD, s	5.25	11.09	N/A
	CI, s	3.26	6.87	N/A

SL344. 2019-05-31; 15,5 leidyb. apsk. l. Tiražas 14 egz.
Išleido Kauno technologijos universitetas, K. Donelaičio g. 73, 44249 Kaunas
Spausdino leidyklos „Technologija“ spaustuvė, Studentų g. 54, 51424 Kaunas