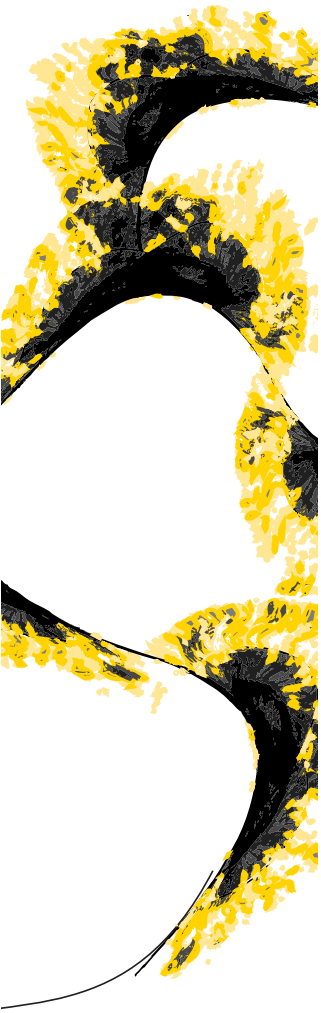

RAM ROBOTICS AND MECHATRONICS

ADAPTIVE CONTROL OF A KINOVA ASSISTIVE ROBOTIC ARM

D. (Damian) Gaethofs

BSC ASSIGNMENT

Committee:
dr. ir. E. Dertien
dr. ir. J. Canyelles Pericàs
dr. F. Nijboer, MSc

february, 2024

006RaM2024
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE. | TECHMED CENTRE UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Summary

The goal of this project was to design and implement a system for Andrei, a patient with severe spasms, to control a robotic arm using an eye tracker. Andrei is not capable of controlling his limbs and is wheelchair bound, therefore his only means of interaction is using his eyes. This system would allow him to be more independent from his caretaker, doing things such as drinking, all on his own. To start off with, research was done into the literature surrounding eye trackers and their application in assistive devices such as robotic arms. From there the requirements for the system were set up.

The robotic arm which was available for this project was the Kinova Jaco, as seen in figure 1a. This is a 6 Degree of Freedom (DOF) robotic arm, meaning it can move in all directions and rotate the hand in three different directions. To control this robotic arm there are several options, in this project it was determined that controlling the robotic arm using robot operating system (ROS) was the best choice. This allows easy integration into existing system surrounding a robotic wheelchair. Then a design proposal was set up for an interface to be used with the system. This interface, alongside the backend of the system, was then realized by creating a ROS package with these aspects. The final interface can be seen in 1b.

The interface was tested in a short experiment where it was used to move the robotic arm and perform some specific commands, such as pick up and a certain object. The completion time was noted for each task and images were taken during the experiment. From these results it became clear that the system is still lacking in usability. The completion time is very high compared to an existing study (305 seconds vs 54 seconds).

In conclusion, the system is a good proof of concept showing that it is very much possible to control a robotic arm using an eye tracker, but there is still a lot to be improved upon.

(a) Kinova Jaco robotic arm

(b) Final interface design

Figure 1: a) The kinova jaco robotic arm, b) the final interface used to control the robotic arm, containing various buttons for controlling the direction the arm moves in. Text on the buttons is in Dutch because the test participant was Dutch.

Samenvatting

Het doel van dit project was om een systeem te ontwerpen en implementeren waarmee Andrei, een patiënt met ernstige spasmes, met zijn ogen een robot arm kan besturen. Andrei kan zijn ledematen niet bewust besturen en zit in een rolstoel, hij kan enkel zijn ogen bewust gebruiken. Dit systeem zou hem in staat stellen om onafhankelijker te zijn van zijn verzorger, zodat hij dingen zoals drinken, helemaal zelfstandig kunnen doen.

Om te beginnen is er onderzoek gedaan naar de literatuur rond eyetrackers en hun toepassing in hulpmiddelen zoals robotarmen. Vervolgens zijn de vereisten voor het systeem opgesteld.

De robotarm die beschikbaar was voor dit project was de Kinova Jaco, zoals te zien is in figuur 1a. Dit is een robotarm met 6 graden van vrijheid (DOF), wat betekent dat hij in alle richtingen kan bewegen en de hand in drie verschillende richtingen kan draaien. Om deze robotarm te besturen zijn er verschillende mogelijkheden, en in dit project werd besloten dat het besturen van de robotarm met behulp van Robot Operating System (ROS) de beste keuze was. Dit maakt integratie in bestaande systemen rondom een robotrolstoel gemakkelijker. Vervolgens is een ontwerpvoorstel opgesteld voor een interface die met het systeem moest worden gebruikt. Deze interface, samen met de achterliggende code van het systeem, werd gerealiseerd door het maken van een ROS-pakket met deze aspecten. De uiteindelijke interface is te zien in figuur 1b.

Deze interface is vervolgens getest waarbij het systeem gebruikt is om de robotarm te verplaatsen en enkele specifieke opdrachten uit te voeren, zoals het oppakken van een bepaald object. De voltooiingstijd is genoteerd voor elke taak en er zijn afbeeldingen gemaakt tijdens het experiment. Uit deze resultaten werd duidelijk dat het systeem nog niet praktisch bruikbaar is. De voltooiingstijd voor een experiment met dit systeem was namelijk aanzienlijk hoger dan dat van een ander onderzoek (305 seconden tegen 54 seconden).

Concluderend is het systeem een goede proof of concept die aantoont dat het heel goed mogelijk is om een robotarm te besturen met behulp van een oogtracker, maar er is nog veel te verbeteren.

Contents

1 Introduction

This project is part of the Ability Tech initiative, a group which was formed to help Andrei, a patient with severe spasms, gain more freedom. Andrei is unable to control his limbs and is only capable of using his eyes to signal his intent. Therefore he has a computer with an eye tracker mounted on his wheelchair. This allows him to not only communicate with people, but also perform other activities, such as creating art, one of his hobbies. However, Andrei is still very dependant on his caretaker, who helps him interact with the rest of the world. One of the projects from Ability Tech is a self-driving wheelchair for Andrei, controlled using an eye tracker. This is a perfect example of granting him more freedom in his life. This project aims to provide another piece of that puzzle, allowing Andrei to control a robotic arm. This would allow him to, for example, independently pick up a cup of water and take a sip, without needing to signal his caretaker.

Thus the objective of this project is designing a system which is capable allowing Andrei to control a robotic arm. More specifically, this project will use a Kinova Gen2 6 degree of freedom (DOF) robotic arm. An image of the robotic arm can be seen in Figure 1.1. To design this system, it is important to understand the specifics about eye tracking systems and the way the robotic arm can be controlled. Therefore, in chapter 2 I will discuss the state of the art research and technology related to eye tracking. Following that, chapter 3 will describe the stakeholders and requirements of the system. Chapter 4 will discuss design specifics of the system, including the subsystems used to control the robotic arm and gain data from the eye tracker. Chapter 5 will then be dedicated to the realization of the system. Chapter 6 discusses the methodology for validating the effectiveness of the system. In chapter 7 I will present the results of the validation and in chapter 8 I will discuss the implications of the results and compare them to existing systems. Lastly chapter 9 will discuss the conclusions and recommendations which can be taken from this project.

Figure 1.1: The Kinova Jaco robotic arm

2 Literature

In this chapter I will discuss the essential literature surrounding eye trackers, covering the basic systems and also the state of the art systems. This will provide a basis upon which requirements can be founded.

2.1 Background literature

In the field of human-machine interaction using eye trackers, the term eye tracking simply refers to following (tracking) the movement of the eye. Following the movement of the eye does not tell us much about what the eye is looking at, which is called the gaze. A more descriptive term would be gaze estimation, which as the name implies, estimates the location of the gaze in 2d or 3d. When this is done in real time, like with most eye trackers, this is called gaze tracking (Cazzato et al., 2020). Commercial eye trackers typically only track the gaze in a 2d plane in space, typically a computer screen. Interacting with the screen is as simple as moving your gaze to a part of the screen and triggering an action.

The triggering of an action using an eye tracker is, however, not always intuitive. Most systems will trigger an action when the user dwells on a button for a certain amount of time, which we call dwell time. When the dwell time is too short however, you can get false activations, which is a problem we call the Midas touch problem (Meena et al., 2017). Simply increasing the dwell time is often not a great solution because it leads to more strain on the users eyes.

A possible solution uses blinking as a trigger for an action. However, people already blink unconsciously, so there would need to be a differentiation between subconscious blinks and blinks as a command. Typically this is done by setting a minimum time which the eyes have to be closed. While this does work, it provides an uncomfortable delay and makes using the system more difficult.

The study done by Sunny et al. (2021) is very similar to this project and has already covered important areas for this project. Like in this project, they use an eye gaze tracker to control a robotic arm and performed tests to validate the system. The most useful part for this project is when they talk about their interface design process. They began with small buttons, but changed to larger buttons, because small buttons are hard to use with eye trackers. They also had a feature which allowed them to run recorded trajectories, useful for repetitive tasks.

2.2 State of the art literature

The previously discussed literature is all relatively well known and a lot of it is already used in commercial products. However commercial products never contain the newest technology. The new frontier into which eye tracker research seems to be going is 3D end point control.

One research group built their own eye tracker using low cost materials and used it to get accurate 3D gaze estimation (Abbott and Faisal, 2012). This means they were able to acquire the 3D coordinates of the point at which a person was looking, which would be incredibly useful in the control of a robotic arm. They followed this up with a paper where they used their system to control a robotic arm by simply looking at the object they wanted see pick up. To do this they used a novel calibration technique, where the robot arm moved along a known path and the user followed the end effector of the arm. This allowed them to determine the relation between the measurement of the eyes and the 3d position the user is looking at, allowing them to get a euclidean error of only 4 cm (Maimon-Mor et al., 2017).

Computer vision has also been used to calculate the 3D coordinates of a person's gaze (Leroux et al., 2015). By combining a camera together with an eye tracker, this group was able to compute the coordinates in 3D with an estimated error of 5.5cm.

Another quite simple, but seemingly novel technique they used was winking one of the eyes as an action trigger (Maimon-Mor et al., 2017). This has a large advantage over blinking as an action trigger, since blinking is something everyone does unconsciously, while winking is a completely conscious action. This means the delay of detecting a conscious blink is not there for winking, allowing much more responsive and intuitive control.

2.3 Interview

A semi structured interview was conducted with a patient and their caretaker, in order to get an idea of the requirements of the system. The patient was in wheelchair with a MyTobii system mounted on it, which allows them do all kinds of things using just their eyes, including communicating. From the interview it became clear that the most important part of the system is allowing independence for the patient. This is seen in daily life all the time, but the example used was being able to drink independently. Controlling the robot joint by joint (joint space) or by direction the end effector (coordinate space) was discussed during the interview and determined that control in coordinate space is a more natural extension. The MyTobii which can be mounted on the wheelchair is not always mounted on the wheelchair, especially in social situation it can block the view of other people. This is also relevant for this project, since the MyTobii could potentially block the view of the robot arm. A suggested solution was showing a camera feed on the MyTobii itself, allowing the patient to always see the robot arm when controlling it. A large problem with eye tracker systems is fatigue, long term usage is straining on the eye so the patient occasionally turned off the eye tracking control to be able to relax their eyes. More details about the interview can be found in Appendix A, but these were the most important points for the next section, in which I will define the requirements of the system.

3 Requirements

In this chapter I will discuss the stakeholders and the requirements of the system. Using the requirements a minimally viable system becomes clear, from which concepts can be generated.

3.1 Stakeholders

The most important stakeholder in this project is the patient who will have to work with the robot arm. For them the system should be easy enough to work with every day and not cause undue fatigue and strain. The caretaker also has a stake in the system, they obviously want it to be safe and reliable so that they do not have to worry about something going wrong.

This project is also part of a larger initiative called Ability Tech. One of the projects of Ability Tech is controlling a wheel chair as a self driving vehicle. To do this there are already some systems in place for controlling a robot, in this case a wheel chair and not a robotic arm. Ideally the system should easily be integratable into the other projects.

3.2 Requirements

Following from the contents of chapter 2, we can determine some requirements of the system.

Requirement 1: *The system must be able to control a Kinova Gen2 robotic arm using an eye tracker in the full 6DOF range*

This is the most important and most basic requirement of the system. It can be fulfilled in various different ways, which will be discussed in the next chapter.

Requirement 2: *The system should induce as little fatigue for the patient as possible*

A big problem with using eye tracking to control a system is fatigue. This is especially true for people whose main means of interacting with the world is through their eyes. Ensuring that the system does not cause too much fatigue is a fundamental requirement if the patient is to use the system for interacting. This requirements came up both during literature research and during the interview from chapter 2, so should be taken as a serious point.

Requirement 3: *The system should show an image of the surrounding area while controlling the arm*

While using the system, the user will be looking at the screen of the computer, which means they cannot clearly see what the robotic arm is doing. This not only makes it hard to direct the robot arm, it also poses possible risks in using the arm. To prevent this from happening, the system should show an image of the surrounding area, including the robotic arm, such that the user can see what their actions are doing.

Requirement 4: *The system should be able to define and reuse set points in space*

Drinking requires the robotic arm to get close to the space of the patient. This is expected to be difficult to control using an eye tracker. The arm could potentially block the view to the screen or other problems could occur. To perform these actions in a more controlled and safe matter, the system should be able to define and reuse set points in space. By doing this, the arm can automatically move to a certain position, such as a drinking position, with minimal interaction by the user. This requirement came up as a result of the interview, where drinking was expressed as one of the desired actions using the arm.

Requirement 5: *Interactable objects on the screen should be large and have clear outlines*

Small objects are hard to interact with using an eye tracker, which would induce a lot of fatigue and provide a bad user experience. Large objects with clear outlines makes it easier to interact with the system and make sure no unwanted actions occur. This requirement is closely related to the fatigue requirement and also came up both in the literature and in the interview.

Requirement 6: *The system should contain an easy to use on and off switch*

In the case of fatigue or something happening around the user, they should be able to quickly turn off the system. This simply means the system will stop sending signals to the robotic arm, allowing the user to move their gaze safely away from the screen. This requirement came forth out of the interview, where the patient displayed them turning of their eye tracker to allow themselves to relax.

3.3 Technical requirements

This section defines the requirements of the previous section into measurable requirements where possible.

Requirement 7: *The system should contain a camera feed of 30 fps and at least 720p (720/1280 pixels)*

30 frames per second and 720p should provide the clear enough image to interact with the surrounding area. A higher frame rate and resolution would be even better.

Requirement 8: *Interactable options should be at least 3x3 cm*

Eye tracker has a particular gaze resolution, having the buttons be at least 3x3 cm ensures that the object is easy to interact with.

4 Design

This chapter contains the essential parts of the design. First I will discuss the hardware used in the project, then I will discuss the different systems for controlling the robotic arm and the ways in which eye tracking can be used to control the arm. After that, I will discuss concepts of how the system front end would look and work. Finally I will present a overview of the internal workings of the system, which leads into the next chapter, where I realize the design.

4.1 Hardware

The robotic arm used in this project is the Kinova Gen2 (Jaco 2). The eye tracker used for creating the system is the TobbiDynavox PCEye Plus, but a final user would likely have a different eye tracker. It is important to note that the eye tracker is only compatible with windows systems and is not compatible with versions past Windows 10. The robotic arm and eye tracker will eventually be used on a wheelchair, which has a so called MyTobii which is a system which runs Windows 10 and has software for helping the patient interact with the world. The MyTobii also has a built-in eye tracker.

As mentioned in the stakeholder analysis, this system would eventually be part of a larger system for controlling the wheelchair. This should be taken into account in order to make the integration of this system into the other systems as easy as possible.

4.2 Robotic arm control

Controlling the robotic arm is normally done using a joystick which connects directly to the robotic arm. The goal of this project is to control it using an eye tracker, not a joystick, so controlling has to be done via a computer. To do this, there are several options:

- Using the API on Windows 10 or Ubuntu 16.04
- Using Robot Operation Software (ROS) kinova package on Ubuntu 20.04 on a laptop
- Using ROS on a Raspberry Pi or another single board computer (SBC)
- Using ROS inside a virtual machine on a Windows 10 system

These options each have their advantages and disadvantages, which I will now discuss. First of all, there is an API made by Kinova for controlling the robot on Windows or Ubuntu. This comes as part of an SDK and is written in C++. Since this option works on Windows, this would allow for a very simple setup on the MyTobii. However, this system would be hard to integrate with already existing projects, since these mostly use ROS, and adding extra sensors is generally harder on a system like this.

Besides the API there is the Robot Operation System (ROS), which a middleware used to control the robotic arm from a computer. Kinova has written a ROS package, which fundamentally builds upon the API, but which makes it possible to control the arm using ROS. ROS is extensively used for robotics and has support for a wide variety of hardware. However the ROS version which runs the Kinova packages has to be installed on Linux and assumes Ubuntu 20.04. This causes a complication in communicating between the eye tracker and ROS, since the eye tracker and MyTobii runs exclusively on Windows. So there would already need to be two separate computer, one which runs the eye tracker software (Windows 1) and one which runs ROS (Linux).

To solve the problem with ROS, it can instead be put on a Raspberry Pi or another single board computer (SBC). This Raspberry Pi then communicates with a Windows system (the MyTobii) from which is gets the signal on what to do. While this would solve the problem, it would require an additional piece of hardware and there needs to be communication between the Pi and the main computer. However, considering the larger scope in which this project resides, this

Table 4.1: Comparison between different possible system designs. Data processing refers to the ease by which data can be used, system integration refers to integrating it into existing solutions, sensor integration relates to adding sensors.

Criteria	Weight	Kinova API on Windows	ROS on a laptop	ROS on a SBC	ROS on a virtual machine in Windows
Simplicity	6	2	1	0	-2
Data processing	4	1	0	0	1
System integration	8	-1	-1	2	0
Sensor integration	6	-2	1	2	0
Future proof	4	-1	1	1	1
Total score	-	-8	8	32	-4
Rank	-	4	2	1	3

problem has likely already been solved. This solution would also provide an easier integration into other systems and make it easier to add extra sensors to the system.

It would also be possible to run a virtual machine on the Windows 10 system, which would run an instance of Linux with ROS installed. This solution would likely provide easier communication than using an external Raspberry Pi. However, running a virtual machine would likely require too much resources for the MyTobii to easily handle. Because it also provides very little integration with other systems, this solution is not a good choice in the long term.

The system types listed above were evaluated based on: simplicity, ease of data processing, system integration, sensor integration and how future proof they are. The evaluation can be found in Table 4.1, from which it becomes clear that running ROS on a single board computer like a raspberry pi is the best choice for this system.

It should be noted that existing or new robotic systems likely use ROS2 while the control of the Kinova Jaco2 uses ROS1, so a bridge would need to be made between the two version.

4.3 Control using an eye tracker

In this section I will discuss and evaluate the ways in which an eye tracker can be used to control a robotic arm.

The following possibilities followed from the literature and the interview:

- Controlling the end effector directly in coordinate space
- Controlling the arm in joint space
- Using 3D gaze tracking for determining the setpoint of the end effector
- Using a camera and computer vision to determine a 3D setpoint

The first way of controlling the robotic arm which comes to mind is simply controlling the end effector in coordinate space. This means you are moving the end effector by having it move in a certain cartesian direction and thus requires inverse kinematics. The inverse kinematics for the Kinova Gen2 has already been worked out and the control systems from the previous section already allow this. A downside of this system is that control is only done in one direction at a time, which is not a very natural way to control something.

An even less intuitive way of controlling the robotic arm would be controlling it joint by joint. This would require no inverse kinematics, but moving an arm joint by joint is a very unintuitive control paradigm. When we move our own arms we do not think about the joint by joint movement, we simply move the hand. Thus, this option would probably require quite a large training time and likely still result in a lesser performance compared to coordinate space control.

Table 4.2: Comparison between different possible eye tracker control designs, complexity is how easy the system is to make, flexibility relates to the amount of control, system integration relates to the implementation into existing eye tracker systems.

Criteria	Weight	Coordinate space	Joint space	3D gaze tracking	Computer vision
Complexity	8	1	2	-1	-2
Ease of use	6	1	-1	2	2
Flexibility	4	2	2	0	0
System integration	4	2	2	-1	-1
Total score	-	30	26	0	-8
Rank	-	1	2	3	4

One state of the art method, previously mentioned in chapter 2, is using 3D gaze tracking to determine the point in 3D space at which a person is looking. This was done using a custom made head mounted eye tracker and a mapping from 2d to 3d using a novel calibration method. The 3D coordinates from the tracking can be used to move the robotic arm to the desired position. The coordinates would be selected using some eye tracked action, like blinking or winking, and the robotic arm would then start moving. This option has obvious advantage in the ease of control, leading to very little fatigue. This option is quite complex however, and requiring low level access to eye tracker data and requiring some complex math. This option is outside the scope of the project but is an interesting option for future research.

Lastly, another state of the art method uses a camera and computer vision alongside the eye tracker to calculate the 3D coordinates of the gaze point. Similar to the previous method, this would intuitive and easy control of the robotic arm by simply looking at an object. At the same time it is also quite a complex solution to implement, requiring extra equipment and a lot extra math. So this solution is also outside the scope of this project but might become a more sensible option in the future or in a larger project.

The different control designs were compared in Table 4.2 . They were evaluated on: complexity, ease of use, flexibility and system integration. Complexity refers to easy the system is to make, flexibility refers to amount of control the user has, system integration has to do with integration the design with current eye tracking systems. From this evaluation it became clear the controlling the arm in coordinate space using an eye tracker is the best solution for this project.

4.4 Interface design

Now that it has been determined how the eye tracker will be used to control the robotic arm, we can design the interface. The main problem which has to be solved is controlling the 3D motion of the robotic arm on a 2D screen. Since we are using coordinate space control, it makes sense to give every direction two buttons, one for positive, one for negative. Then, to make it as least exhausting as possible, the mouse cursor only has to hover over a button to make the robot move in that direction.

To calibrate the robot, it needs to perform a homing action, so there needs to be a home button. An on and off switch, as stated in the requirements in chapter 3, should be included. However, this leaves out an essential part of the robotic arm, the end effector. The end effector is a gripper with 3 fingers, which needs to be opened and closed. The end effector can also be rotated in three different directions, which all also need two buttons.

To satisfy requirement 3, which states that the system should show an image of the surrounding area, there will be a live camera feed behind the buttons. This allows the user to both see what is happening and control the arm at the same time. Lastly, to satisfy requirement 4 from chapter 3, there needs to be a button which leads to submenu for different set points, or there needs to

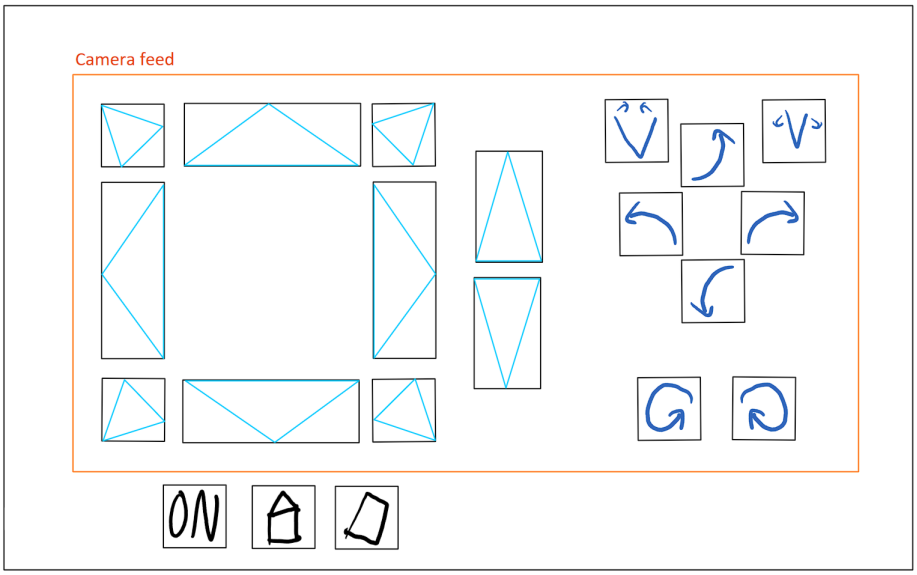

Figure 4.1: A mockup of a potential interface, the left side contains buttons for controlling the end effector in the xyz directions, while the right side has buttons for rotating the end effector and closing the fingers. These buttons are overlaid ontop of a camera feed with the robot arm visible. On and off button, home and set actions will be below the camera feed.

be an area with "shortcut" buttons for performing certain actions and moving to different set points.

A mockup was made of a potential interface design, partially inspired by the work done in Sunny et al. (2021) and is seen in Figure 4.1. The left side contains buttons for controlling the end effector in the xyz directions, while the right side has buttons for rotating the end effector and closing the fingers. These buttons are overlaid ontop of a camera feed with the robot arm visible. On and off button, home and set actions will be below the camera feed.

4.5 System diagram

Figure 4.2 shows the high level organization of the system. It includes possible extensions of the system, like the sensors and emergency stop. The Raspberry Pi is running ROS and receives data from the Windows 10 computer running the interface.

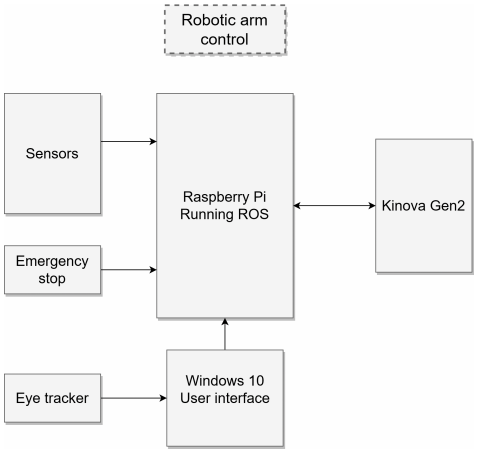

Figure 4.2: A diagram of the full system, the Windows 10 computer has the user interface which is controlled using an eye tracker and sends data to the Raspberry Pi running ROS. The Raspberry Pi can take extra data from sensors, not used in this project and potentially an emergency stop. The Raspberry Pi then sends commands to the Kinova Gen2 robotic arm.

5 Realisation

In this chapter I will discuss the realization of the system. This will cover the final product and any changes that had to be made along the way.

5.1 Working with the Kinova Jaco

To work with the Kinova Jaco, I started with getting it to work using the SDK provided by Kinova. At first this did not work, however after flashing the firmware to the robotic arm it started working.

After that, I had to set up the ROS stack provided by Kinova to work with the Kinova Jaco. This stack is made to run on ROS noetic, on Ubuntu 20.04. After setting everything up, testing began with launching the kinova driver, which sets up all the necessary nodes for controlling the arm. The launch command includes the specific robotic arm used, which in this case is called j2n6s300, which refers to Jaco 2 6 DOF service mode 3 fingers, this name is also used in all the topics, services and actions for controlling the robot. All the topics can be seen at launch, and could now be inspected.

The robotic arm first has to be homed, for which there is a ROS service under $/j2n6s300\_driver/in/home\_arm$. To control the robotic arm there is a topic with takes in a cartesion velocity message, consisting of a twist with linear xyz and angular xyz values. This is published to $/j2n6s300\_driver/in/cartesian\_velocity$, and then the arm starts to move.

The arm can also be controlled in cartesian position, instead of velocity. This is especially useful for moving towards a specific point or moving a specific distance. Kinova provides a demo python file, which takes in the desired cartesian movement as input and then performs that movement. However, you can also use the ROS action that kinova have made for cartesian position control, which is essentially what the demo python file does. Lastly, it is also possible to communicate directly with the Kinova API from within ROS, and there is a function to set the cartesian position. Using the ROS action was the most sensible choice, since this could easily be done within the code of the node. However, full implementation of cartesian control was not achieved in the time of the project.

To close the fingers of the arm, there is a ROS action which can set the state of the fingers. Lastly, there is a service which, when called, essentially stops the robotic arm from moving, until it is started again. This is useful for turning the interface off when not in use, as it blocks *any* command, including homing, fingers and cartesian control.

5.2 Writing a ROS node

Once it was clear what was needed to control the robotic arm, I started on a ROS node which would function as a relay between the interface and the Kinova stack. First of all, I had to make a new ROS package, which I titled eye-control.

This node has several subscribers, but is mainly subscribed to the topic $eye\_control/in$, which takes in a custom message called guiMsg, which contains all information to be send to the Kinova stack. In this case that is mainly the cartesian velocity. A function then converts this message to the message type that kinova expects and publishes it to the corresponding topic.

To control the finger state, I made a simple ROS service server which would send an action to change the fingers to the desired state. The desired state would either be open or closed, no in between, mainly for simplicity.

Figure 5.1: The final interface, buttons contain text describing what they do, written in Dutch, for the target user is Dutch. Behind the buttons is a camera feed, currently off. When the buttons are pressed, the arm move a direction for a certain time. "Aan uit" turns the robotic arm off, meaning none of the buttons will do anything until the arm is turned on.

5.3 Creating the interface

To create the interface, I started with a QT GUI, since this can be fully integrated into a ROS package. However, it soon became clear that this was not going to work, since doing this would require a full install of ROS on the windows computer with the interface. Messing with anything on the windows computer of the user should be avoided, as this is the only device the user can typically use to interact with. Thus the GUI had to be made to work on a webserver, which could be accessed from a web browser.

The backend of the webserver uses python together with flask and flask socketio to host a simple html file with some javascript for the buttons. Flask socketio provides websocket functionality, needed for having the buttons send a signal back to the python server. To communicate from the backend to ROS, I used a python library called roslibpy. This library can communicate with a ROS package called rosbridge, which provides a websocket connection to perform typical ROS operations, such as publishing messages, calling a service etc. To allow the user to connect to the interface, a local network should be set up, to which the hosting computer can broadcast the system. For this project, a hotspot created by a phone was used for simplicity.

The frontend of the interface simply consists of buttons, laid out on the web page in a grid, with text specifying the action of the button. In the background, a camera feed is shown, allowing the user to see what the buttons are doing as they are pressed. The frontend can be seen in Figure 5.1. It is slightly different from the mockup in chapter 4.4, mainly there are no buttons for diagonal movement. The interface right now is already quite busy, adding extra buttons for diagonal movement, while nice, would add too much to the screen, and not allow the eyes to look anywhere safely.

The buttons have to pressed down in order to cause movement, this was chosen over continuous control, where hovering over the button would cause a movement. Only requiring hovering would likely result in a lot of small unintentional movements, as the eyes tend to dart around the screen a lot. How the buttons are pressed depends on the users eye tracker, but in this case simply uses dwell time.

5.4 The code

The code for the realization is not available in this report. It has been uploaded to a github repository as part of the ability tech organization. Thus is can be found under the following name: abilitytech/kinova. It contains documentation on how to get started with the package, including everything needed to run the package.

6 Methodology

In this chapter, I will discuss the methodology for validating the system. I will start by discussing existing methodology from the literature and then propose some improvements. Finally I will discuss the details of the experiment. The next chapter will then discuss the results of the experiment.

6.1 Existing methodology

The system must be validated to 1) show that it works, 2) able to compare it with existing systems. However, there are almost no previous examples of validation done on a system like this. The only notable study was done by Sunny et al. (2021), they had subjects pick up objects from the ground, a table and a shelf. From there they recorded positional, torque and speed data from the robotic arm and kept track of the completion time for each task. They had a sample size of 10 (healthy) participants, who also rated their experience with the system. This study seems like a good starting point for evaluating the system. However some aspects of the tests should be clarified and standardized to make it easy to compare different systems. In the next session I will describe a short proposal for standardizing a few aspects of these tests

6.2 Proposed methodology

The first aspect of the tests which should be clarified is the objects used in the tests. Sunny et al. (2021) did not specify which objects they used and why, while this can have a large impact on the performance of the test, after all a smaller object will be harder to pick up, and thus take more time. The second aspect concerns relative position of the robotic arm to the object. Sunny et al. (2021) made no mention for specific initial position of the robotic arm relative to the object the subjects had to pick up. If the object were to be placed closer to the arm, it would likely take less time and lead to different results, so this too should be standardized.

To solve these two problems, I propose some new standards. First a standard set of objects should be used. I propose the use of a few common objects, first of all a pen of normal size, which, while unlikely to be used by patients, provides an easy example of a small and hard to pick up object. Then a middle sized object like a cup or a bottle of water should be used. Objects larger than this would start to become unwieldy for use with a robotic arm and thus are deemed unimportant.

In terms of distance from the robotic arm, I propose that the object should be placed between 50-60 cm away from the base of the arm in any direction. Then, like in Sunny et al. (2021), it should be tested on the ground, on a table and on an elevated position. The object should be placed in a straight line from the base of the robotic arm, with extra, optional testing for positions deviating from this line. Vertical distance should also be within this 50-60 cm range and should be doable for most robotic arms. These distance are comparable to what a normal person would be able to reach while remaining in the same spot and should provide a reasonable framework for providing tests.

Completion time is the most important data point of these tests and provides easy to understand insight into the efficacy of the system. Recording position, torque and speed of the robotic joints does not provide interesting insights into the working of the system, so should not be needed.

6.3 Experimental setup

Using the previous additions, an experiment was set up. The robotic arm is mounted to a table around 1 meter in front of the subject. First a cup or bottle will be on the table in accordance

with the previous section. The cup will then be placed on the floor next to the table and lastly on an elevated position on the table.

The subject will be asked to grab the cup for all of these situations and the completion time will be measured, stating from the first movement from the home position, to the cup being lifted off the surface it was on.

The same will then be done for a pen, which will provide a harder test. In this case the subject might not be able to lift the object, to give a time limit this should be done within 2 minutes, if two minutes passed the attempt will be deemed a failure. The subject can then do another attempt from start.

After the experiment, the participant will be asked to rate the experience with the system, specifically about fatigue, between 1-10. The participant will also be asked about their experience with the camera feed, did they use it, or did they switch between the screen and looking at the robotic arm. They will be asked about the size of the buttons, were they big enough? Lastly, the participant will be asked about the speed of movement of the robotic arm, was it fast enough or too fast to properly control.

7 Results

This chapter contains the results acquired from the experiment. First there will be some images from the experiment itself. Then there will be a series of images showing the movement of arm in response to the system. The completion time for various tasks will be discussed, lastly the response of the participant to questions after the experiment will be noted. Then in the next chapter, these results will be discussed.

7.1 The experimental setup

Due to time constraints the experiment was not performed by an outside user, but instead performed by myself. Also only part of the experiment was performed, the picking up of a large object at a set distance around the same height of the arm. The experiment was performed in a public space on the University of Twente in DesignLab. The robotic arm was mounted to a table, as shown in Figure 7.1. A spot on the table was marked, this is where the object would be placed. The object used was a large thermo bottle. This was at a distance of 55 cm from the base of the robotic arm. The robotic arm was connected to a laptop running the system. This laptop was connected to a mobile hotspot, to which it was broadcasting the web interface. Another computer then connected to the same mobile hotspot network and entered the IP address with the correct port for connecting to the interface.

7.2 Experimental results

Figure 7.2 shows a sequence of images from one of the tasks, showing how the arm is moving from the home position to the object. The time that it took to get to the object and pick it up was measured to be on average 5 minutes and 5 seconds done by 3 trials. Generally the object was too large to keep good hold of using the arm, often having it slip out or pushing it away.

Figure 7.1: The setup used in the experiment to validate the system. The robotic arm is mounted to the table, a safe distance away from the user. A spot around 55 cm was marked for placing objects.

Table 7.1: Caption

Question	Answer
How fatiguing was it to use the system?	3
Did they use the camera feed?	Yes
Was the size of the buttons good?	Yes
Was the movement of arm fast enough?	No

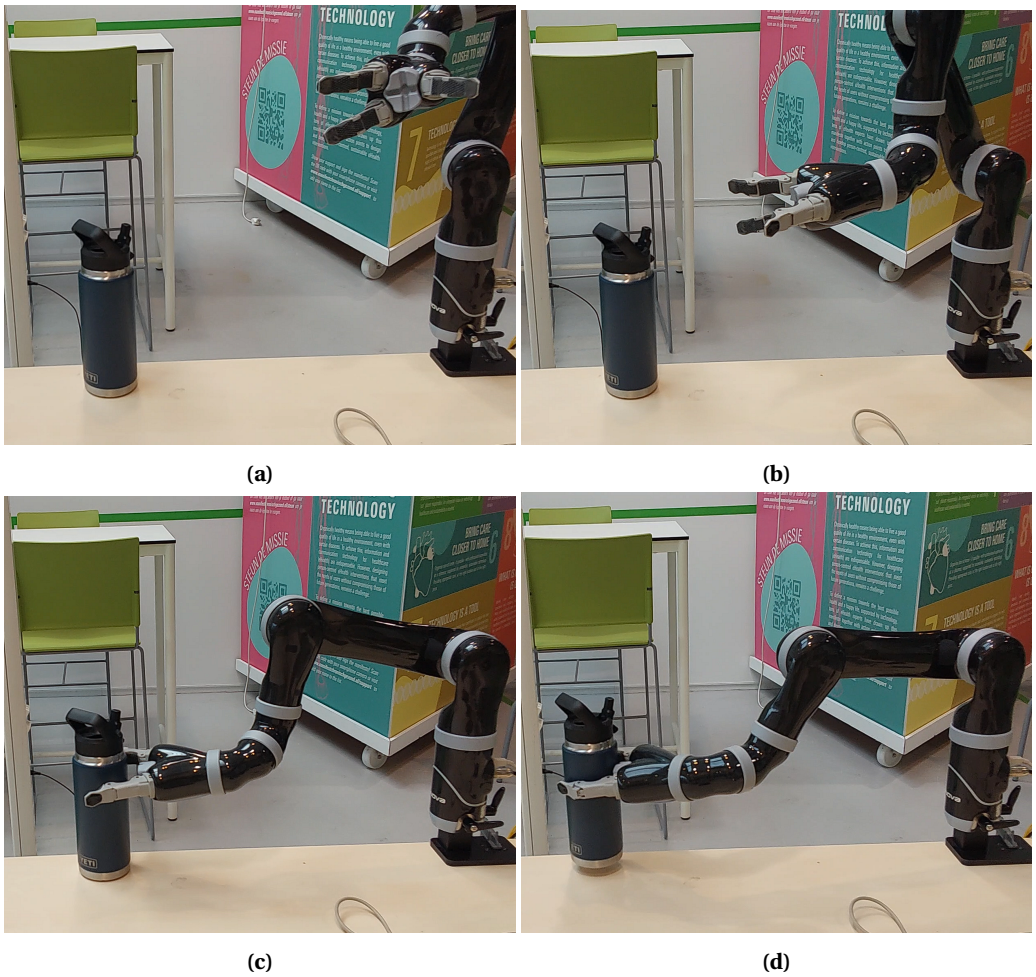

Figure 7.2: Series of images showing the arm being moved by a user using an eye tracker. a) shows the arm in starting position, b) about halfway, c) about to pick up the bottle, d) lifted the bottle

Table 7.1 shows the answers to the questions. First of all, the fatigue, which was rated a 3. While the experiment took quite some time to do, going up to 30+ minutes in total, I felt very little fatigue. This might differ per person, but for myself it was not too bad.

Regarding the camera feed, it certainly has its problems. At times the camera feed would not update, seemingly having too much latency between the two computers, which made it hard to know what was going on. This required me to look away from the screen at times to see how it was going. Another problem with the camera feed was depth perception. It was difficult to see the exact position and rotation of the hand relative to the object that was to be picked up.

While the size of the buttons was fine, it was sometimes hard to get the eye tracker to work at the edges of the screen. Lastly, the movement of the arm was generally not very fast, especially in straight lines, it took too much to move it as much as desired.

8 Discussion

Following from the results, it has become clear that the system works, but still needs refining to be more useful in real situations. Completion time was quite high, in fact being higher than what should have been allowed according to chapter 6, which says 2 minutes should be the maximum. That is definitely a problem which should be solved in a future project. Recommendations for this can be found in the next chapter. This is also strongly related to one of the questions of the experiment, which relates to the movement speed of the arm. It was obvious during the experiment that it should have moved more or moved faster with each button press. This would have also reduced usage time significantly.

Compared to the results from Sunny et al. (2021), the completion time for this system is very high. The median completion time for picking up an object from the table in their experiment was only 54 seconds, significantly lower. However it should be noted that the details of the experiment performed in Sunny et al. (2021) are not very clear, so it is unknown how exactly this compares. Either way, it is clear that the system should be made a lot easier to use.

Lastly, I will check to see whether the system matches the requirements set out in chapter 3. Starting with requirement 1, which says that the system should be able to control a Kinova Gen2 robotic arm using an eye tracker, this has obviously been achieved, as can be seen from the results. Requirement 2 says the system should induce as little fatigue for the patient as possible. From the questions asked after the experiment, it is clear that at least for the duration of experiment, this was not an issue.

Moving on to requirement 3, the system does indeed show an image of the surrounding area, using a camera. However, the camera feed is not ideal, depth perception being difficult also makes it difficult to position the hand correctly infront of an object to pick it up. The fact that the latency for the camera was not great is something which should also be looked into in follow up projects, potentially leaving out the camera feed completely or reducing its use.

Requirement 4 says that the system should be able to define and reuse set points in space. Due to time constraints and the increasing complexity of this requirement, this requirement was not able to be achieved in this project and is left for future projects.

Requirement 5 states that interactable objects on the screen should be large and have clear outlines, the buttons were made to be quite big, and from the question asked to a participant it is clear that this is mostly good enough. Still hitting the edges was an issue, but this is possible solved with a better calibration. More testing would be required to be sure whether this is the case. Requirement 6 says the should contain an easy to use on and off switch. This was achieved by having a simple button on the interface, which was just as easy to use as the rest of the button on the interface.

Lastly the two technical requirements, requirement 7 specifies the quality of the camera feed, while this mainly depends on the camera used, in this case a camera of high enough quality was used to satisfy this requirement, except that the camera often would not update, negating the 30 fps requirement. Requirement 8 specifies the size of the buttons, and by measuring the size of the buttons, this was determined to be achieved.

9 Conclusions and Recommendations

9.1 Conclusions

In conclusion, I proved the viability of a system which allows Andrei to control a robotic arm using just his eyes. There is still a lot to be improved upon, but the proof of concept is there. With the current system, it is quite clear that Andrei would not yet be able to drink independently. So while it might already be used for very simple tasks, there is still a lot to be done. In the next section I describe recommendations for future projects building from this project.

9.2 Recommendations

First of all, the completion time for a task should be lowered significantly. To do this, either a completely different control scheme should be considered, or there should be something like a slider, which sets the amount of time which the robotic arm moves then the button is pressed. This would allow large movements to be done faster, not requiring a lot of clicking on the same button, but rather setting the slider high and then pressing the button.

A completely different control scheme could be something like a 2d plane represented on the screen, where a certain point can be clicked on to move the arm in that direction. Then have a separate slider to control the vertical movement of the arm.

Secondly, the setpoint functionality has not yet been implemented, while it is a crucial part of allowing a user to potentially drink independently. On top of this, the special drink mode available using the Kinova Jaco also has not been implemented yet. There is however no ROS service to enable drink mode, so it needs to be done by directly communicating with the Kinova Jaco API. Communicating with the Jaco API is not too difficult, and the basic setup for doing so is already available in the ROS node for this project, but the functionality still has to be implemented. The (likely) best way to implement this would be to have a single button go through the entire drinking motion, so it turns on drink mode then performs the move to drink and then returns back to its original position. This makes it very easy for the user to take a quick sip.

The system has yet to be put onto a Raspberry Pi or something similar, which was one of the design goals of this system. It works fine on a laptop, but in a potential final product, it would certainly need to be on a SBC. This also means making a local network such that the user can connect to the interface, as this is currently done using a hotspot created by a phone.

Another thing that needs to be considered is placement of the robotic arm on the wheelchair. Where will it be mounted and how will it be mounted? Where will the camera be mounted such that they can clearly see what the robotic arm is doing? These are important questions which were definitely out of scope for this project.

That also brings with it the question of safety around other people. What kind of measures should be taken to ensure that using the robotic arm is safe and what can be done in case of a dangerous situation?

A Use of AI

During the preparation of this work the author(s) used ChatGPT and Github Copilot in order to assist in writing code and translate the summary from English to Dutch. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the work.

B Interview

The interview was conducted in a semi-structured format. It took place in a public space around a large table, on top of which the robotic arm was mounted. The interviewer sat across the table from the caretaker, while the patient sat in wheelchair next to the table, on the side of the robotic arm. The wheelchair of the patient had a MyTobii system mounted on it and they used it to communicate one time. The interviewer had a laptop with him, the interview was not recorded, the answers were written down in a separate document for later processing.

A general discussion was held about the robotic arm and what its capabilities are. Still, a few specific question were asked:

1. Which problems do you face in daily life which could potentially be solved using a robotic arm?
2. Do you have any experience with these types of robotic arms or have seen other people using them?
3. Are there any problems you encounter in the usage of eye trackers?
4. In which way would you like to use the eye tracker to control the robotic arm?

Some examples were given for the first question: independently drinking and being able to participate in social games, like card games, dice games or other social games.

Relating to the second question, they do know someone who has experience controlling a robot, but not using eye trackers. The mentioned patient is still capable of using their hands, so was using a joystick to move the robot, something the patient for this project is not capable of.

The biggest problem they faced with eye trackers was using them outside, where the lighting from the sun would hamper the usage of the eye tracker. While talking about the eye trackers, the patient used the MyTobii system to express their difficulty with using the new system which they had recently adopted, they said that controlling the mouse cursor was more difficult. They also mentioned the difficulty with using an eye tracking system, which can cause fatigue in long term usage. The caretaker explained that in the beginning stages of using the eye tracking system, the patient would have very red eyes after just a few minutes of usage, but this got better over time.

The last problem they had with the MyTobii system relates to social events. The system would block the view of the other people, so they do not mount the system when going out for social events. This is also relevant for the control of the robotic arm, since the arm could be potentially hidden by the MyTobii, or it would simply be difficult to control the arm while focusing the on the MyTobii screen. As a possible solution, the interviewer suggested a camera feed being shown on the screen while the patient controls the robotic arm. This was something they were familiar with from a related project, where they control the wheelchair using eye tracking. That project also contained a camera on the wheelchair.

Lastly, when asked between joint space control versus coordinate space control, they specified a clear preference for coordinate space, which is a more natural way of moving.

During the discussion, they asked to demonstrate the working of the robotic arm. The robotic arm was connected to a laptop which had the Kinova JACO-SDK installed and was capable of moving the arm. Using the keyboard control in the control GUI, the robotic arm picked up a pen, which is quite a small object, and a larger object, a 0.75L thermal bottle. During the demonstration all degrees of freedom were shown and of course the opening and closing of the fingers was shown. Any request for certain movements were performed.

Information about the research was provided, including the goal of the research and interview, the risk and how the data would be processed. An informed consent form was filled. The interview was approved by the ethics committee of the EEMCS faculty.

Bibliography

Abbott, W. W. and A. A. Faisal (2012), Ultra-low-cost 3D gaze estimation: an intuitive high information throughput compliment to direct brain–machine interfaces, **vol. 9**, no.4, p. 046016, ISSN 1741-2552, doi:10.1088/1741-2560/9/4/046016, publisher: IOP Publishing.
https://dx.doi.org/10.1088/1741-2560/9/4/046016

Cazzato, D., M. Leo, C. Distante and H. Voos (2020), When I Look into Your Eyes: A Survey on Computer Vision Contributions for Human Gaze Estimation and Tracking, **vol. 20**, no.13, p. 3739, ISSN 1424-8220, doi:10.3390/s20133739, number: 13 Publisher: Multidisciplinary Digital Publishing Institute.
https://www.mdpi.com/1424-8220/20/13/3739

Leroux, M., M. Raison, T. Adadja and S. Achiche (2015), Combination of eyetracking and computer vision for robotics control, in *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, pp. 1–6, doi:10.1109/TePRA.2015.7219692, iSSN: 2325-0534.
https://ieeexplore.ieee.org/document/7219692

Maimon-Mor, R. O., J. Fernandez-Quesada, G. A. Zito, C. Konnaris, S. Dziemian and A. A. Faisal (2017), Towards free 3D end-point control for robotic-assisted human reaching using binocular eye tracking, in *2017 International Conference on Rehabilitation Robotics (ICORR)*, pp. 1049–1054, doi:10.1109/ICORR.2017.8009388, iSSN: 1945-7901.
https://ieeexplore.ieee.org/document/8009388

Meena, Y. K., H. Cecotti, K. Wong-Lin and G. Prasad (2017), A multimodal interface to resolve the Midas-Touch problem in gaze controlled wheelchair, in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 905–908, doi:10.1109/EMBC.2017.8036971, iSSN: 1558-4615.
https://ieeexplore.ieee.org/document/8036971

Sunny, M. S. H., M. I. I. Zarif, I. Rulik, J. Sanjuan, M. H. Rahman, S. I. Ahamed, I. Wang, K. Schultz and B. Brahmi (2021), Eye-gaze control of a wheelchair mounted 6DOF assistive robot for activities of daily living, **vol. 18**, no.1, p. 173, ISSN 1743-0003, doi:10.1186/s12984-021-00969-2.
https://doi.org/10.1186/s12984-021-00969-2