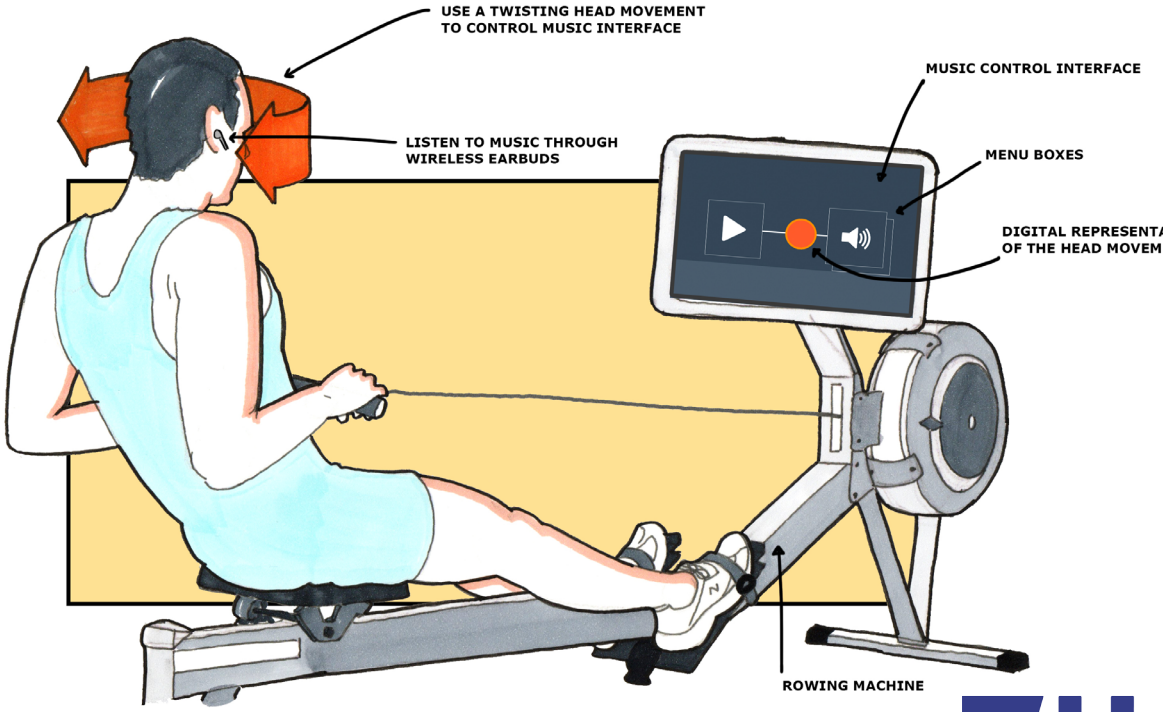
DDM113: Designing with Complex Sensors

ROWbot



Students:

Steffen Hartmeyer
Lara Leijtens
Irina Bianca Şerban
Suzanne Vugs
Ferdi Zwaan

28-06-'18

Table of Contents

The Team

Steffen Hartmeyer *Human-Technology Interaction*
I was responsible for the integration of the application into a coherent whole. Specifically, I developed a modular structure for the application in order to efficiently combine the different interfaces following a model-view-controller pattern. In addition I wrote the code for experiment 2 and 3 and implemented two different filter functions in order to smooth the sensor data. Furthermore, I processed and analyzed the data obtained from experiment 2. My most important learning point was working with the Kinect and dealing with the problems imposed by the sensor.

Lara Leijtens *Industrial Design*
In this project my main focus point was the modelling interaction section. I created the five MPCA models and the model for their relations. I have also written the corresponding part in the report. The models were also explained during the presentation that I created and for most part presented. Apart from this, I designed the first two experiments and was a researcher in all three experiments. I wrote down how the third experiment was set up in the report. Writing the report was a group process, we all helped finalizing it, so I contributed to other sections as well. The main learning point for this course was for me the fact that many design decisions are based on assumptions or guesses, when that is not necessary. Often, the options for these decisions can be tested and analysed through quantitative data collection.

Irina Bianca Şerban *Industrial Design*
My main responsibility was the functionality of the Kinect, making it work with Processing, finding the algorithms for face detection and setting it up for testing and the experiments. I also took part in experiment 2 and 3 as a researcher. Moreover, I helped in the process of developing the concept, shooting the video and writing the final report. My main takeaway from this course was designing and performing experiments which generate quantitative knowledge.

Suzanne Vugs *Industrial Design*
For this project, I was responsible for the speech recognition part. I worked with speech libraries for different programming languages to obtain the optimal result. Then I integrated the code in Processing by writing code that allows them to communicate. Besides that, I was a researcher for experiment 3 for which I also did the analysis. Together with Irina Bianca Şerban, we shoot the material for the concept video. After shooting, I did the editing. Finally, we all worked together to create our concept and write the report. For the report, I was responsible for analyzing the 5 A's, the result section of experiment 3 and the clarification of the speech recognition process. My main learning point are insights in designing (complex) systems in such a way that it makes sense to the user. This is definitely more difficult than I expected.

Ferdi Zwaan *Industrial Design*
For this project, I translated the five A's into a design for the graphical user interface. I was responsible for coding the graphical user interface inside Processing, and continuously tweaking it to specific needs within the process. Furthermore, I helped to conduct experiment 2 and 3, wrote parts of the report, and did the overall design of the report. My main learning point is that no matter how complex a sensor is, by structuring your design process in small steps, you can always make sense of data and use that to improve your design.

Design Brief

The main goal of the elective Designing with Complex Sensors was to learn how to design and test an interactive system that involves complex sensing. At the beginning of the elective, the team was introduced to three aspects that would form the foundation for the upcoming design process; social interaction, modeling interaction, and analyzing sensor.

Social Interaction

In class, the team was introduced to the design framework 'The 5 A's', a framework that helps designers to view intelligent systems as social partners. The 5 A's; Attention, Address, Action, Alignment, and Accident, provide a foundation for the systematic approach to the design of interactive systems. Considering the feedback and feedforward elements of each of these five aspects helps designers to address and tackle challenges that often occur when designing for such systems. [3]

Modeling Interaction

In order to address the 5 A's, and incorporate them into a system, the mirrored perception-cognition-action framework was introduced, that provided the possibility to model intended interactions from either the user– and system perspective. By use of the MPCA template, interaction cycles can be described through the cognitive, physical and digital representations. This helps the designer to see how the user– and system perspective interact with each other in the physical representation of the design.

Analyzing Sensor

Sensors are made to measure a specific variable. When dealing with complex sensors, a range of events can be captured and multiple variables are provided. To implement this in a system, the designer should determine how to embody the sensor and what variables to use to get the best results. Sensor performance can be measured and optimised by collecting data about performance with different conditions. By creating histograms of the responses for all conditions, a mathematical function can approximate the outcomes with a distribution. These distribution can be used to create so-called receiver operator curves, which enable a better sensor performance.Performance: use distributions to create so-called receiver operator curves

Introduction ROWbot

When rowing on a rowing machine, it is not possible for the rower to control music, since his hands are always occupied during the workout. When the rower wants to skip a song, change volume or pause the music for a moment, he will always have to (momentarily) stop his workout to control his music player. For ROWbot, new interaction styles were explored that enable users to interact with a music player system while continuing their workout.

While rowing, the whole body is being used, except for the head which can move freely to a certain extent. Also interaction through speech recognition, a rapidly developing technology, could be seen as an opportunity to interact with the music player. Both interactions - through head-movements and through voice - have benefits and downsides. This project investigated the opportunities and limitations to use both head movements and speech recognition to control a system.

ROWbot utilizes a Kinect (Microsoft Corporation, Redmond, Washington) –a complex motion sensor that enables developers to work with the human body as an input for their systems– to keep track of the user's head-movements and speech. Furthermore, ROWbot uses a screen, attached to the rowing machine, to display a graphical user interface which visualizes the different functions that can be used. The remainder of this report elaborates on the design decisions and the user flow.

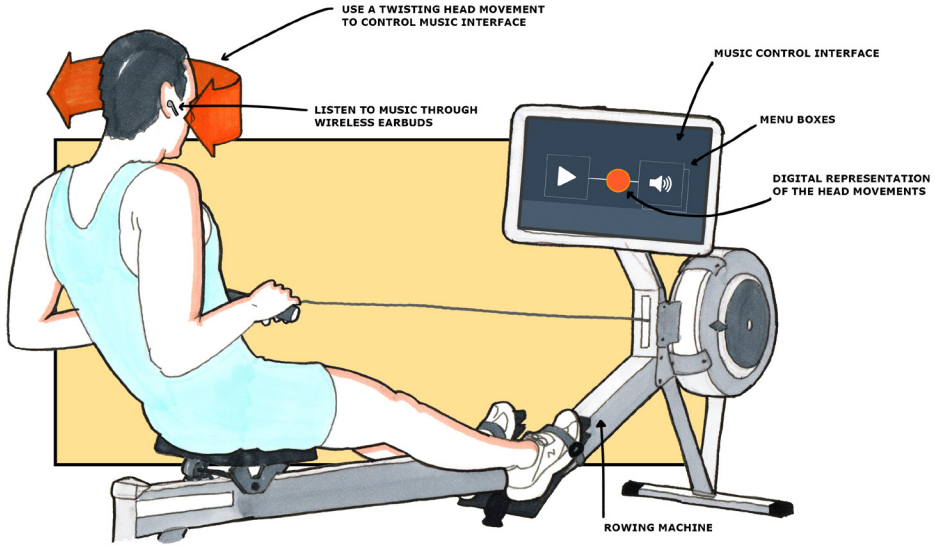


Figure 1: Concept sketch of ROWbot

Social Interaction

Bellotto et al. [3] suggest that designers should take into account five design challenges when designing for human-machine interaction. These five question are based on social studies investigating human-human interactions. This approach tries to create a language between the system and the user that, from the user's perspective, feels intuitive. The next section addresses each design challenge and explains how this relates to the design decisions for ROWbot.

Address

The user addresses the system by voice recognition. In order to communicate this to the user, the GUI displays a mouth with a text cloud stating the words to activate ROWbot (Figure 2). The user cannot have physical interaction with the system's display because their hands are occupied when rowing. Therefore, the options to address the system were limited to speech or body movements. While rowing, the user constantly makes body movements. However, it is not common to constantly talk out loud during a workout. Therefore, the chance that the user would unintentionally activate the system with body movements would be higher than for speech recognition. Thus in order to disambiguate signal-to-noise, the choice was made to use speech recognition to address the system. Besides that, the combination of words, compared to as single word, to activate the system can also reduce unwanted responses. It is more likely that the system incorrectly identifies the speech input when it needs to detect a single word than a combination of words.

The system's display also makes it clear to the user where the intended target system is located: the user knows where to direct his or her focus when they want to communicate with ROWbot.

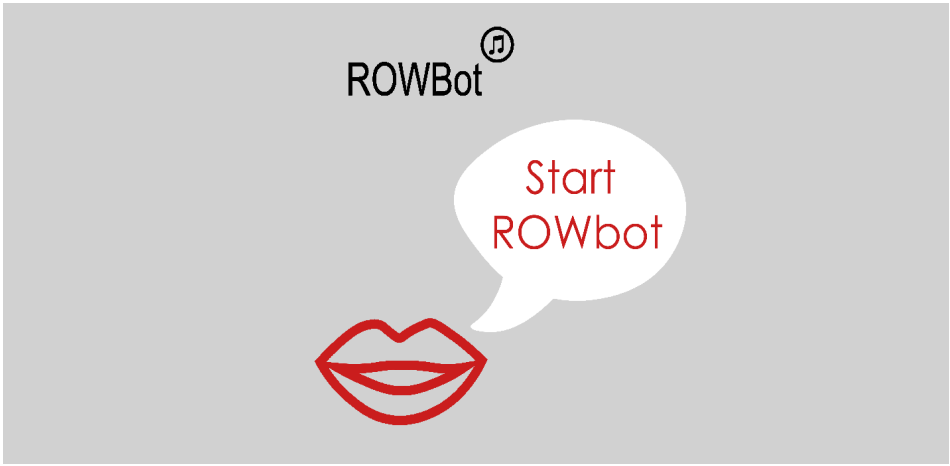


Figure 2: Starting screen of the system. The user needs to say "Start ROWbot" to start controlling the system.

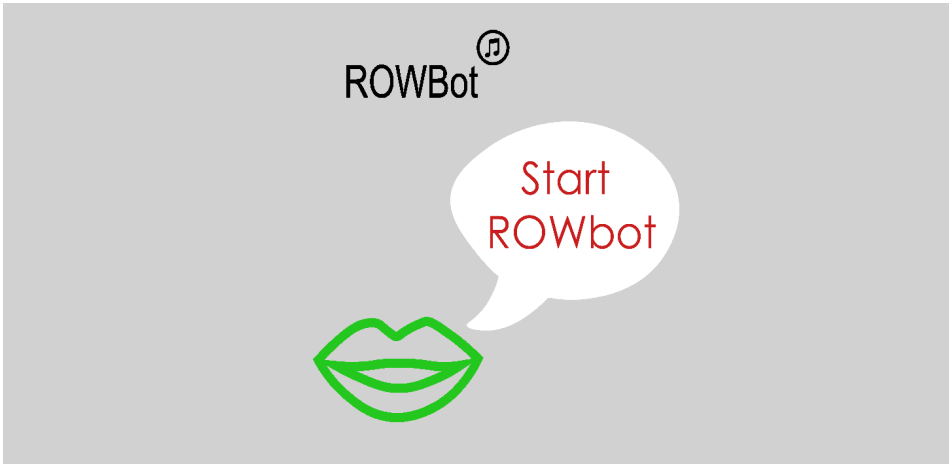


Figure 3: Starting screen of the system. The mouth turns green when the user speaks out loud and the system registers the input.

Attention

This second design challenge relates to the question: how does the user know the system is ready and attending to their actions? The GUI of ROWbot achieves this by providing graphical feedback. When the user speaks out loud, the mouth displayed on the interface turns green (Figure 3). This communicates to the user that the system is paying attention. It needs to be noted that this approach assumes that the user is looking at the display. In order to enable that, the display is integrated with the standard display of the rowing machine (Figure 1). The standard display (already existing display incorporated in the standard row machines) shows the user how their workout is progressing (speed, travelled distance, duration, etc.), which makes it likely that the user pays attention to the display. Besides that, the standard display is located at eye level to draw the user's attention.

Action

The design challenge 'action' targets the issue of how the user controls the system. In the case of ROWbot, this entails the actions to go through the menu and change the volume of the song, skip through the playlist, and/or play/pause the current song.

As explained before, given that the user is rowing, the user can interact with the system by voice recognition or body movements. The first experiment that was conducted for this project, focused on possible body movements that the user could perform to control the system.

Experiment 1

The study was conducted in a rowing center with participants who frequently row. The study consisted of two participants. Each participant was asked to perform certain body movements. These consisted of three types of head movements, squeezing the handle, turning the wrists up and down, moving the knees side to side and moving the shoulders up and down (Figure 4).

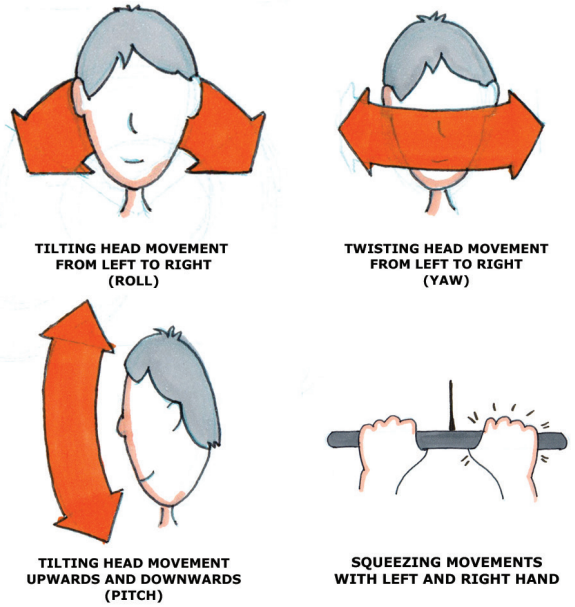


Figure 4: Options to control the system by different movements.

After each session, the researcher interviewed the participant to ask about their experiences. The qualitative data showed that squeezing was not suitable, since you already squeeze the handle firmly while rowing. Also shoulder and knee movements or twisting the wrists were not preferred, because these movements obstructed the rowing flow. The second finding was that participants preferred to twist their head left or right, compared to the other head movement options, because it felt physically more comfortable.

The outcome of the first experiment resulted in the action of moving your head left or right to control the system. In experiment 3, the participants were asked to evaluate their experiences when using only head movements or speech recognition to control the system. The outcomes show that head movements are preferred over speech recognition:

> Participant 1:
> "*Controlling the system with head movement feels more natural than speech. Also because you can perform an action with voice recognition in multiple ways: next, go, up, are all words you can use for the same action. That makes it confusing. Using your head movement as input is less ambiguous.*"
>
> Participant 2:
> "*Using speech to control the system felt more socially uncomfortable. You are already moving on the rowing machine, so adding a small head movement does not make you uncomfortable. If you need to address the system for each action by voice recognition, then it feels like the people take notice.*"

For the testing of experiment 3, the researcher controlled the system and they did not make use of the Kinect sensor due to technical limitations. This meant that the participant might have felt that they were asking the researcher to handle the system. This form of interaction is similar to human-human interaction where you have a conversation with someone and that might have influenced the experience of the participants. However, one of the participants reasoned that if you control the system by voice yourself without the presence of the researcher, then it might becomes strange to 'ask' the system something. Since the system is not human, it feels awkward to talk to a screen. Hence, head movements were again preferred over speech for controlling the system.

Based on the findings of experiment 1 and 3, the design decision was made to control the system with head movements and only use speech recognition for addressing the system.

Alignment

This section addresses the question: how does the user know the system is doing (has done) the right thing? Firstly, graphical feedback communicates to the user what the current state of the system is by displaying that state, and by providing real-time feedback about the user' head movement. Figure X shows an example of a state of the system and it shows the red dot that aligns with the user's head movement. Hence, if the user looks to the left, the red dot mirrors that movement. If a head movement selects a different state then that state will appear on the display. Besides that, when "selecting" a button, the button turns green so that the user knows the button is selected (Figure 5). Secondly, the system provides the user with additional auditory feedback by changing the volume, skipping as song, or play/pause a song based on the user's head movement.

Accidents

An important aspect of the system is how it handles errors. This includes unintended actions and undesirable results from the user's perspective.

The main approach to recover from an error, is that the system goes back to its previous state when it does not detect a head movement to the left or right . For example, if the system determines that the user said "start ROWbot" when that was not the case, then the user will see

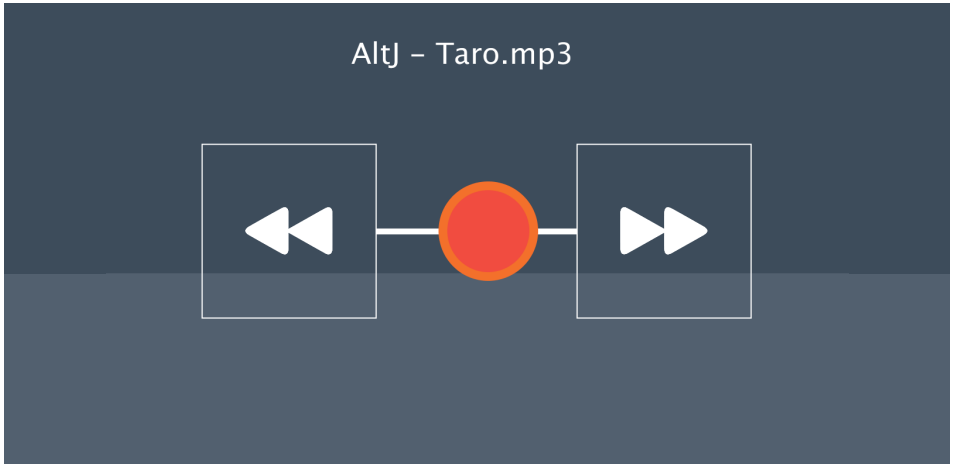


Figure 5: Visualization for skipping a song.

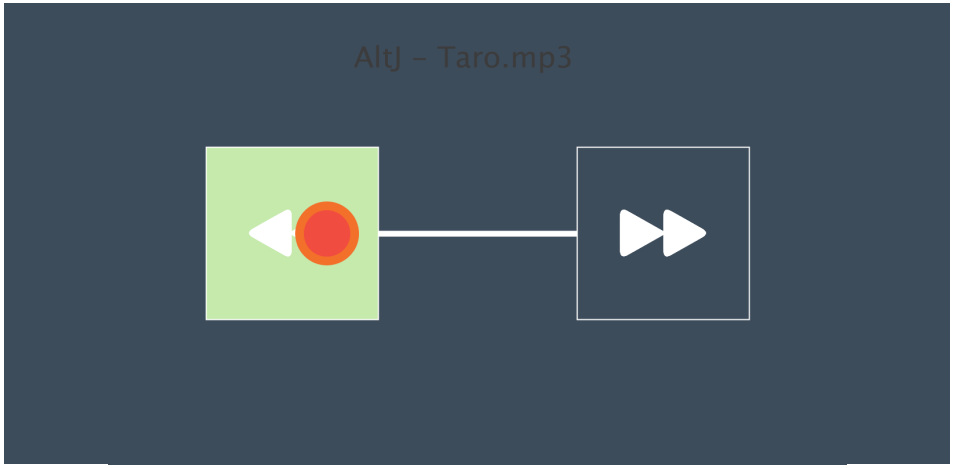


Figure 6: Visualization for skipping a song. The user turned their head to the left, thus the red dot is on the left side of the screen. The green box indicates that the system registered the action to go to the previous song.

the visualisation of the main menu on the display. Because the user did not intend to address the system, he or she continues rowing without performing any action. The system activates a timer that is visualized as a filler (Figure 6). Once that filler fills up the screen, the system goes to its previous state. In this case the system goes back to the state where it waits for speech input. Hence, the user does not have to do anything but to continue their workout while the system recovers from its own error.

This previous example is applicable to every unwanted response of the system when a user performs an action either intended or unintended. By simply keeping their head straight, which is the normal head position when you row, the system goes back to its previous state.

Apart from accidents where the user performs an action, but the system does not recognise the action, there are accidents when the user does not perform an action, but the system thinks that he did. More concretely said: when the system thought the user turned his head to the side, but in reality the user's head was still in the neutral zone. This happened because of technological dysfunctioning of the sensor and related yaw algorithm. A more in depth discussion about this topic is provided in the section 'analysing sensors'.

Modeling Interaction

The mirrored-perception-cognition-action model [4], is a model that describes the process of communication between a system and a user. It gives insights in the effectiveness of implementation of social rules for engagement, from both a user and a system perspective. In the model, a loop describes the several steps of communication from user to system and from system to user. For both user and system, three phases can be distinguished, as can be viewed in figure x. Receiving (perception with senses for users and control with sensors for the system), recognising (cognition through reasoning on the user's side and a model based on calculations on the system's side), and reaction (the user performs an action by using his motoric system and the system uses actuators to change a display), resulting in physical representations.

This cycle is closely related to the 5 A's, explained by Bellotto et al. [3], since the loop can be used to analyse the effectiveness of Address, Attention and Action. Furthermore, when analysing the communication between a system and users with this model, it will become clear whether it is nicely aligned and where accidents may occur.

Using the model (Figure 6), helped to implement the 5 A's in the design concept and to get a proper understanding of the underlying mechanisms. The model was implemented into the design concept, and five loops could be distinguished. They will be discussed in detail in this section. Furthermore, the connection between the five loops is visualised and each loop is discussed in detail for every figure.

The model in Figure x represents the interaction between the user and the system. If the system is not told to do something, it just waits until it receives input from the user. Overall, the system has a passive attitude and the user a proactive attitude.

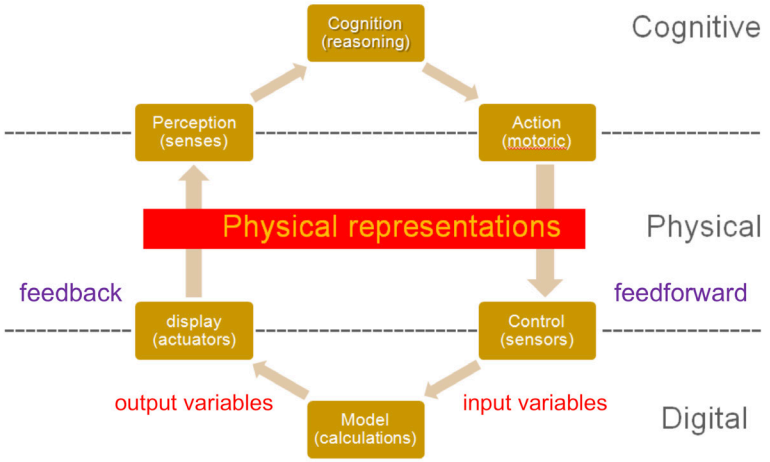


Figure 6: Mirrored-perception-cognition-action model [4]. Image obtained from [2]

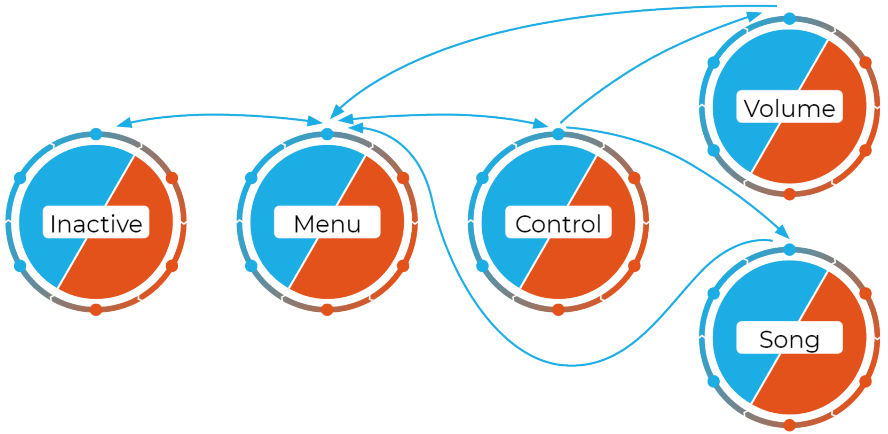


Figure 7: Model of relations between interaction loops.

Each circle of Figure 7 represents a possible action, or a set of closely related possible actions. Furthermore, it also represents a screen in the interface for the user as well as a class that is written in the code. This makes that system very transparent and enables good alignment.

In the model described in Figure 7, it is possible for the user to get into all the loops, but it is impossible to get trapped in a loop. Figure 8 to 13 shows how each interaction loop of the system works.

Optimizing Interaction

There are several points in the system where a decision is made based on keeping the head in the middle (either selecting one item from the "Control" view, or going back to the previous state). Since it is also possible to move the head to the sides and select an option, there should be a time constraint for selecting the middle option. In order to make the design decision about the amount of time for the filler, an experiment was set up to collect data about users head movements.

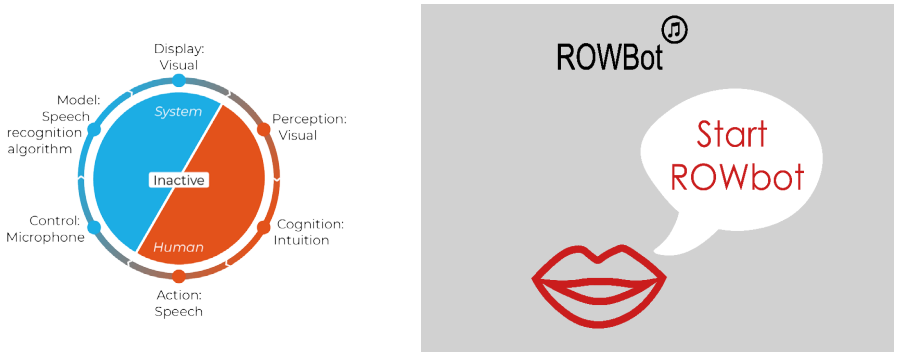


Figure 8: Interaction loop and user interface "Inactive". The system indicates with visual cues that the user should say "start" to activate the system. When the user says something, the system receives the input through the microphone and checks it. If the input is the word "start", the mouth turns green. When the correct word is recognised by the system with a speech recognition algorithm, the next screen appears.

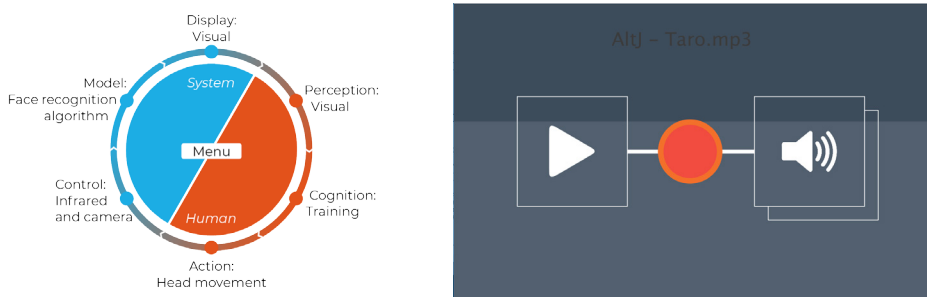


Figure 9: Interaction loop and user interface "Menu". The red dot symbolises the position of the head. The background is filling up with a grey filler, when this is full, the system will go back to the inactive state. By moving the head to the left and right, a decision can be made. The user can decide to play or pause a song, or to perform another action. For now, the only other actions are to change volume or change song, but the system can be extended with other possibilities. For example, setting the BPM of songs or choosing a different playlist. By moving their head to the left, the user can pause or play a song. By moving their head to the right, the user can access the next Interaction loops called "Control". The menu category is added so that it is very easy for users to play or pause a song. The system receives the movements of the user's head with an infrared and a normal camera. Through a face recognition algorithm, the system determines what the position of the head is (see section "Analysing sensor").

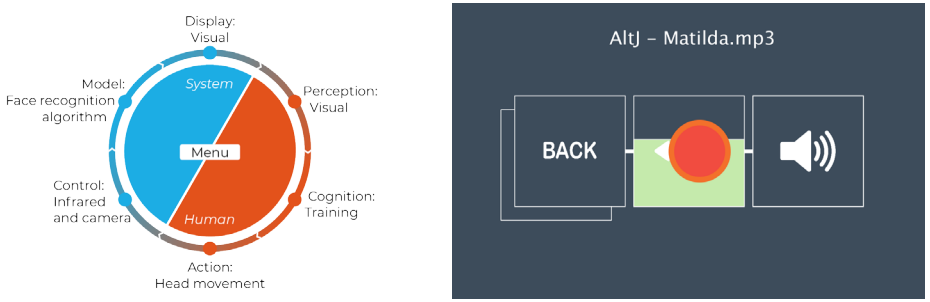


Figure 10: Interaction loop and user interface “Control”. This loop is similar to the menu loop. However, to go back to the “Menu” loop, the head can move to the left and to select an item from the “Control” loop the head should be kept in the middle. Once the green filler is filled, the item is selected. To select a different item, the head can be moved to the right until the prefered item is shown in the middle. (e.g. for selecting volume from Figure x, the user should move his head once to the right; the volume button will move to the middle then; in order to go to the volume view, the user needs to keep his head on the button, in the middle, for as long as the green filler goes up; once the green filler is up, the user will be in the “Volume” view).When more options are added, they can be scrolled through by moving the head multiple times to the right.

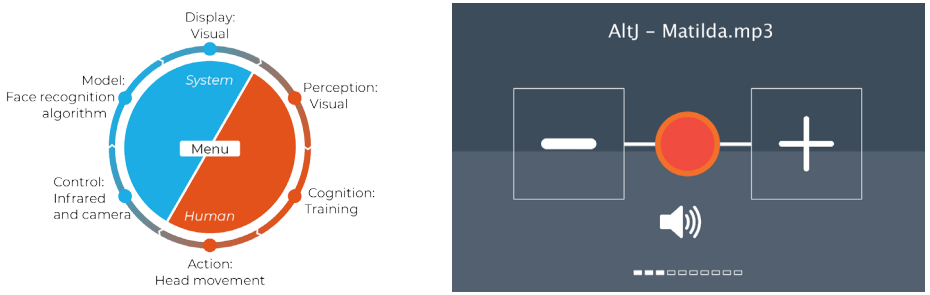


Figure 11: Figure x. Interaction loop and user interface “Volume”. Volume can be turned up or down by turning the head to the right or left, respectively. When the background filler is full, the system will return to interaction loop “Menu”. Apart from visual cues (the buttons + and - turn green when selected and the scale on the bottom fills/ empties by one unit each time the +/- buttons are pressed), the user will receive auditory feedback when the volume is changed.

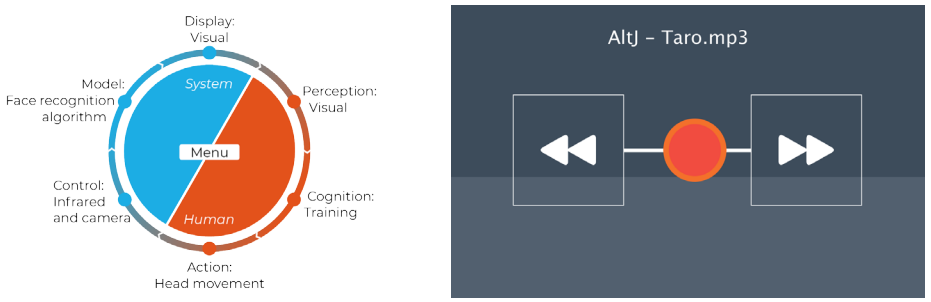


Figure 12: Interaction loop and user interface “Song”. Users can decide to go to the next or previous song by turning the head to the right or left, respectively. When the background filler is full, the system will return to interaction loop “Menu”. Apart from visual cues (the buttons >> or << turn green once touched), the user will receive auditory feedback when the song is changed.

Experiment 3

Setup

Participants who had experience with the system were selected for this experiment, because it would take too much time to let the participants get familiar with the system. Furthermore, by using participants who are familiar with the system, the outcomes will be more realistic.

During the experiment, each participant had to perform several tasks for different filler times. The tasks were: change the volume, change the song, inactivate the system (wait until the system resumes the “Inactive” state). This resulted in four states: Midbox, Volume, Skip and Menu. The tasks were performed in a fixed order, which resulted in the following user flow (based on the states): Start, go to Midbox, go to Volume, go back, go to Midbox, go to Skip, go back, go from Menu to start screen.

By performing these tasks the participants encountered two types of fillers: 1) background fillers that fill the background with a grey filler and once filled, the system goes back to a previous loop, and 2) selection fillers that fill a box with a green filler and once filled, the system goes to a forward loop represented by the box.

There were four filling conditions:

- No timer filling up
- Slow filling speed
- Moderate filling speed
- Fast filling speed

The participants had to perform the three tasks for each of the four conditions. Whenever they encountered a filler, they would say "go" when they wanted to go to the next screen. This could also be when the filler was not filled yet, or after the filler had been filled. The time they wanted the filler to take was logged. By doing so, the prefered time for each filler could be determined, as well as the influence of the filler on the prefered time. In addition, mistakes made - meaning: wanting to select an option to the left or right after the filler was full - were determined.

Results

Figure 13 shows the results for the filler experiment for each participant. The graphs on the left state, for each condition, the prefered time to go to the next state. These graphs indicate that the different conditions influence the preferred time. All participants preferred a shorter time period for condition 4, to go from the state for changing the volume and skipping a song to main, as compared to the other states. This can be explained by the speed of the filler, since this was faster compared

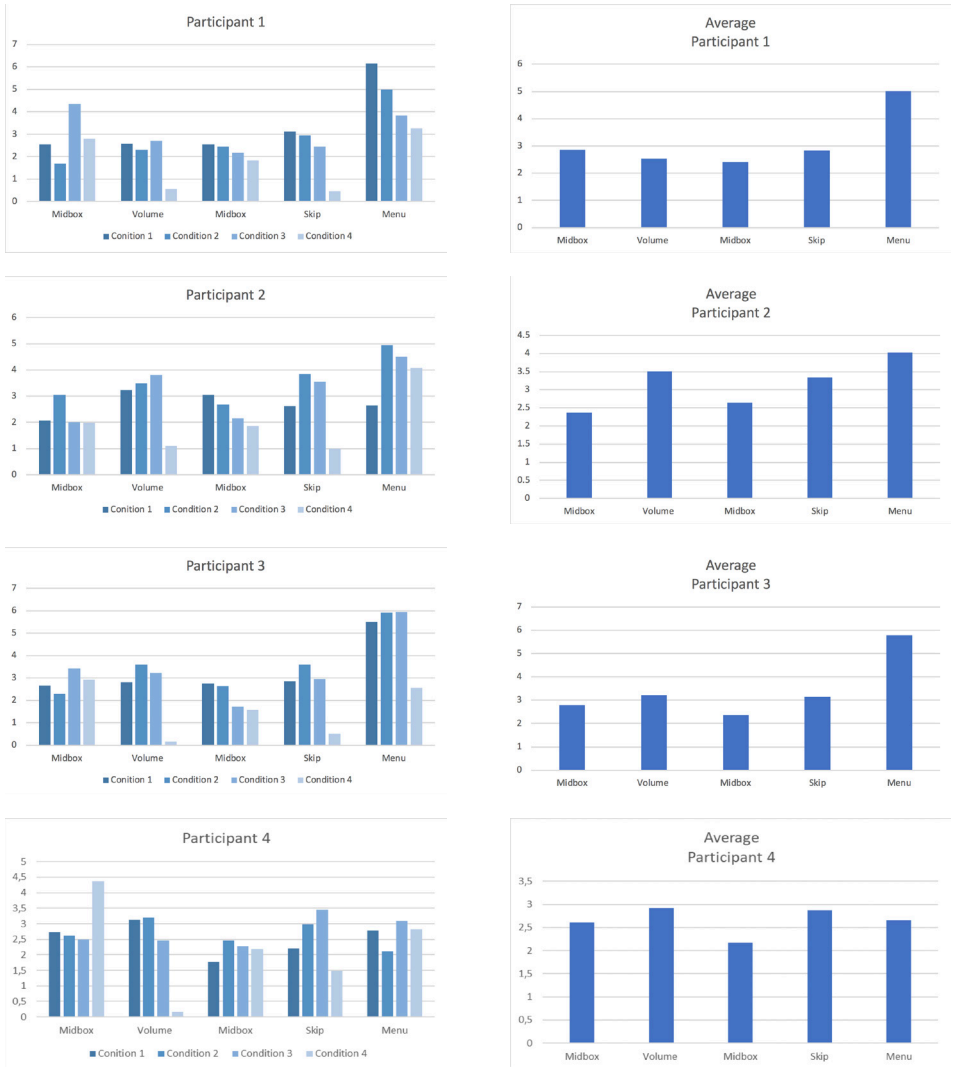


Figure 13: Overview of graphs with the outcomes of experiment 3. The graphs on the left show the preferred times for the fillers for each condition. The graphs on the right is the average of the preferred times of condition 1,2, and 3 for each participant.

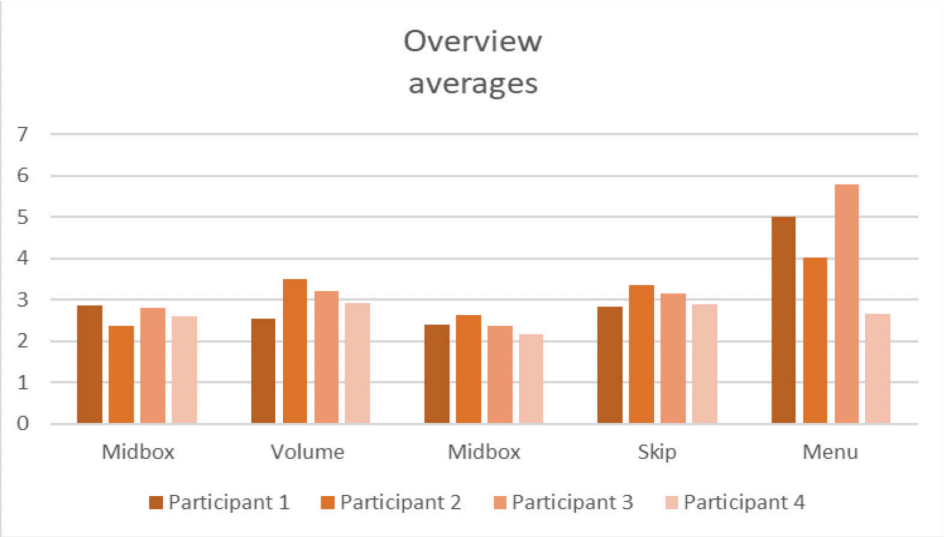


Figure 14: A graph that shows all the averages of condition 1,2, and 3 for all participants.

to the others. Therefore, when determining the a fixed time period for the filler, the designers need to take into account that it influences the user satisfaction with the system and its usability.

Given the small number of participants it cannot be determined if the differences between the preferred times are statistically significant or not. However, conditions 1,2 and 3 do not show extreme differences between each states (Midbox, Volume, Skipping, Menu), compared to condition 4. Therefore, the preferred times of condition 1,2 and 3 were averaged. The results are shown in the right graphs of figure 13. These graphs show for each participant the differences in preferred time depending on the current state. For example, participant 1 and 3 do not show much variation between Midbox, Volume or Skipping, but there is a large difference between Menu compared to the rest. This indicates that these participants prefer a longer time period between the Main Menu view and going back to the view were the systems waits for speech input. This can be because the participants needs more time to determine if he or she wants to stop interacting with the system or not.

Finally, figure 14 shows the comparison between the averages of conditions 1,2 and 3 for each participant. This graph indicates that there are small variation between the states Midbox, Volume, and Skipping for all participants. Therefore, taking the average of them will result in fixed time periods for the fillers of that state. However, the preferred time for Menu differ more between the participants. It would be interesting to explore the possibility to determine this filler based on personal preferences. Nevertheless, given the small number of participants, some of these time periods can be outliers. Further research is needed to see if the filler to go back the the starting screen, can also be averaged or not.

By participating in the Filler experiment, participants had a chance to experience the system as a whole, using a combination of movements and speech. To get qualitative feedback about this experience, a short discussion was held with the four participants at the end of experiment 3. The main outcomes are described in Appendix 1.

Analyzing Sensor

Kinect
Kinect is a motion sensing input device that was produced by Microsoft for Xbox 360 and Xbox One video game consoles and Microsoft Windows PCs. Based around a webcam-style add-on peripheral, it enables users to control and interact with their console/computer without the need for a game controller, through a natural user interface using gestures and spoken commands.

ROWBot uses Kinect for Xbox One, a new version with significantly expanded hardware capabilities. The corresponding Kinect for Windows v2 hardware was released in 2014, along with a supporting SDK. The software behind ROWBot has been programmed in Processing, a simple software sketchbook, which has a library for Kinect V2 called KinectPV2. The library provides built-in methods for face detection and allows for obtaining information from the color camera and the infrared camera of the Kinect, such as the position of the different face features of in a two dimensional space and the general position of the face.

KinectPV2 includes two face detection algorithms: standard face detection and HD face detection. The standard face detection algorithm provides the positions of the eyes, the nose, as well as the edges of the mouth, along with the orientation of the face. In specific, orientations are yaw (rotating head left and right), roll (tilting head left and right) and pitch (tilting head upwards and downwards) of the head expressed as angle in degrees with 0° as the midpoint (keeping head straight, looking straight ahead). The HD face detection algorithm provides a more complex and more accurate description of the face by mapping 1347 vertex points to the entire face of the user, not only the key features. The HD face detection algorithm was provided with the Kinect SDK 2 and was developed to provide a more detailed and accurate representation of a user's face. However, the Processing Kinect API that we used did not provide the face orientation angles (i.e. yaw, roll, and pitch) for HD face tracking. Since we required the yaw value to determine the horizontal head rotation, which was the main interaction method with the system, and due to time constraints and for reasons of simplicity, the standard face tracking algorithm was chosen since it already provided head orientations.

Because the Kinect uses color and infrared cameras, it has its limitations. First, the lighting in the room influences the accuracy with which the sensors detect the face features. Second, for the best detection, the user needs to stay aligned to the middle of the device and look straight into the camera. Third, the Kinect allows to detect multiple faces, which can lead to accidental activation by another user. Since ROWBot is a single user device, the best functionality is obtained by isolating the back of the user. Testing it in a real setting (e.g. in a gym) where people walk behind the user constantly thus posed an additional challenge.

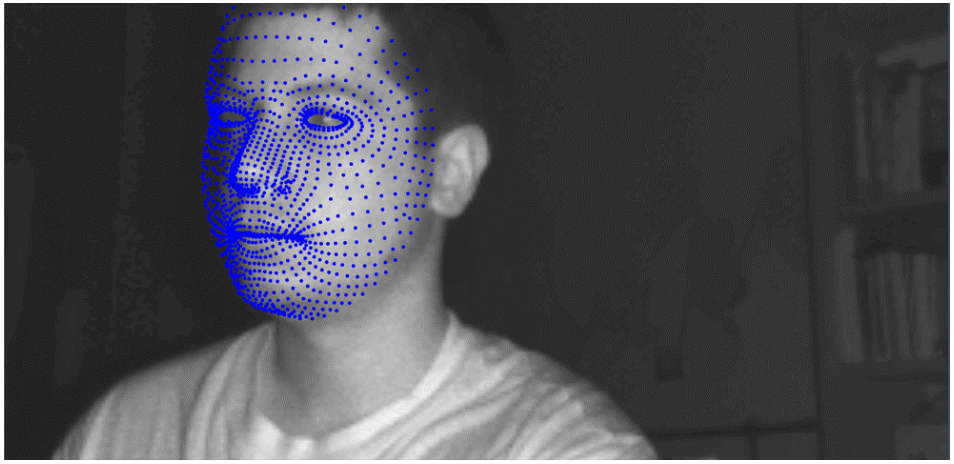


Figure 15: Output of the HD face detection algorithm [5]

Figure 16: Output of the simple face detection algorithm [6]

Measurement and Analysis of the Sensor Data (Experiment 2)

In order to measure and analyze the data provided by the sensor in more detail and to inform about the way the system can recognize when a user selected a function on the interface (e.g. increasing the volume), a short study was performed. Data was collected from seven participants. The study consisted of participants rowing on a rowing machine while their head movements were recorded and tracked by the Kinect sensor. There were three conditions of head movements that were measured. In the first condition participants were instructed to keep their head relatively steady as they would when normally rowing on a rowing machine. In the second condition, participants turned their head either to the left or the right as indicated by the researchers. Importantly, participants did not interact with the system and thus received no feedback about the required magnitude of their head movement. However, they were asked to perform the movements to an extent that was comfortable to them and which they thought would be acceptable to select a function on the interface of the system. In the third condition participants were instructed to turn their head to the left or right and keep it in that positions as long as it felt comfortable to them. Each condition was one minute in length. Throughout the measurement period, the head movements were tracked and the yaw, roll, and pitch values were recorded with a temporal resolution of 50 ms.

The resulting data was analyzed in MATLAB (Version R2017b) and STATA (Version 14.2). Figure # shows yaw as a function of time for condition 1 and 2 for one participant. It can be seen that the measurements obtained from the Kinect are very noisy, with abrupt changes in direction and singular outlying values. Moreover, the neutral position is not always detected as being close to 0°, but shows values up to -20° and 20°. In order to analyze the data further, different smoothing functions were tested. From visual inspection it was determined that two filters provided the best results in terms of filtering a substantial amount of noise while approximating the true signal: A moving median filter and a simplified Kalman filter. Figure 17 also shows the smoothed signal for the median filter (window size = 15) and the Kalman filter. It can be seen that the median filter approximated the underlying signal a bit better than the Kalman filter. Therefore, for further analysis the entire data was smoothed using the median filter with a window size of 15.

Given the noisy signal for the yaw detection, it was subsequently investigated whether pitch and roll could be used in addition to yaw in order to improve the detection accuracy. Figure 17 shows the smoothed yaw, pitch and roll values for condition 1 and 2 for one participant. The figure indicates that despite smoothing the data in the previous

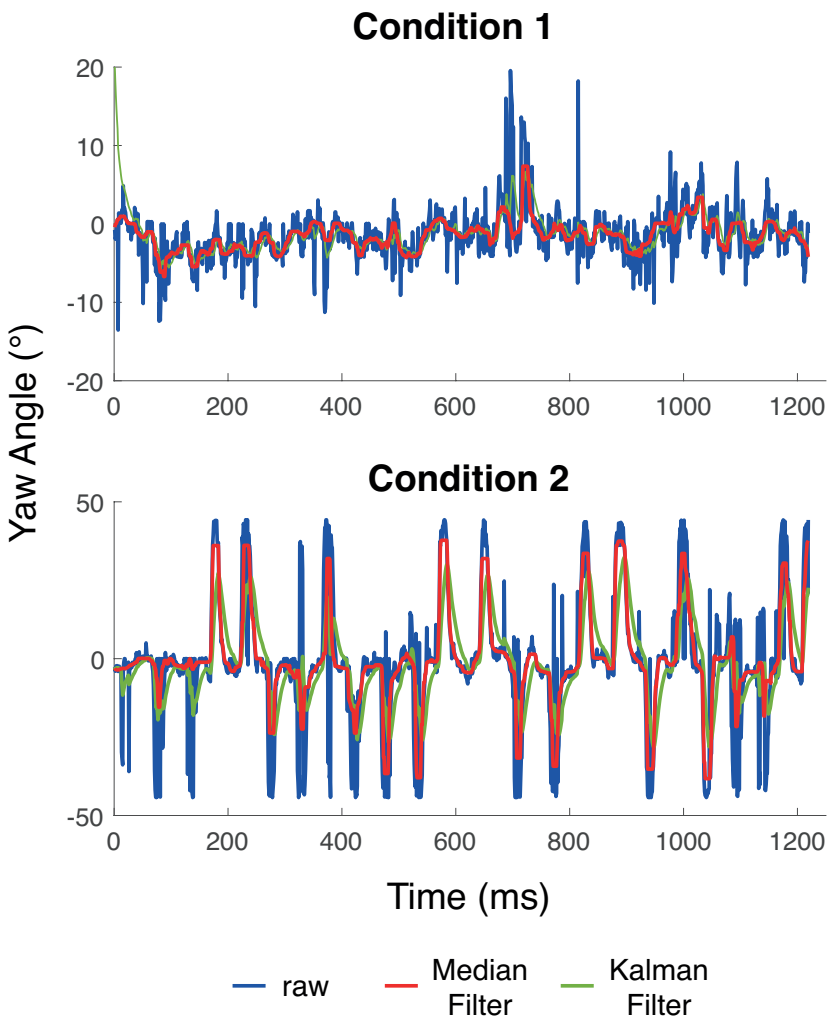


Figure 17: Raw and smoothed yaw values for condition 1 and 2 for one participant.

step, there is still a lot of remaining noise. Note that also pitch and roll are quite noisy. To test whether yaw can be predicted by pitch and roll, a multivariable fractional polynomial regression was performed. The coefficients of the resulting model were all significant, however, only a variance of 11% (R2) could be explained by the model. This indicates that pitch and roll could only marginally be used to improve the accuracy of the detection of the yaw angle.

Initially, it was planned to explore the possibilities of using a machine learning algorithm to process and recognize the head movements (i.e. yaw, pitch and roll together) in order to decide whether a selection occured. However, due to the noise inherent in the signal it was decided to ignore pitch and roll and to use a threshold value for yaw to recognize when a user selects a function on the system interface. In Figure 18 it can be seen that for condition 2 (i.e. where participants were instructed to turn their head), most head rotations resulted in maximum angles of ±40°, whereas in condition 1 (i.e. stable head position) yaw stayed below ±20°. Since the head rotation required to select a function should be as small as possible without compromising the sensitivity of the system and increasing the risk of accidental selections, a threshold value of ±25° was chosen. That is, a user has to turn his head more than 25° to the left or right in order to select a function. As described in Modeling Interaction upon passing the threshold, the function box lights up in green, indicating that a selection occured.

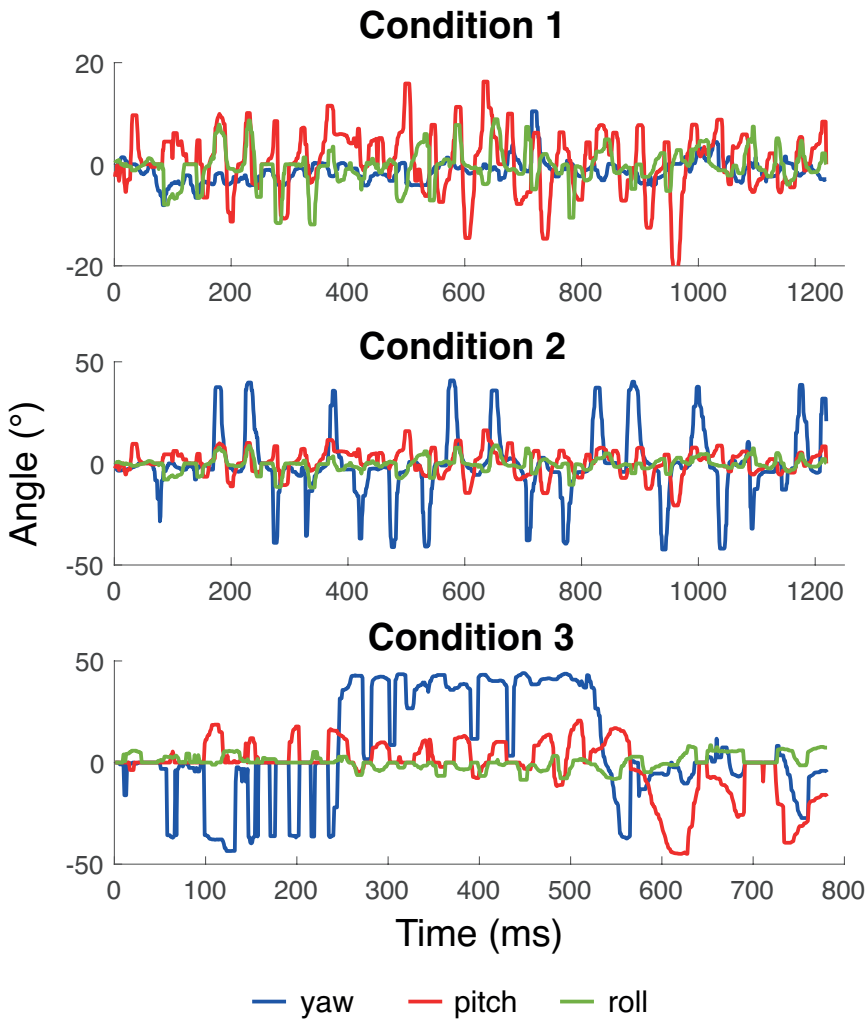


Figure 18: Smoothed yaw, pitch, and roll values for conditions 1, 2, and 3 for one participant.

Speech Recognition

The speech recognition code is written in Python and makes use of the SpeechRecognition package, which uses the Google API to recognise speech. The reason to use Python is because their speech recognition library performed better than the speech recognition library for Processing and Java: it processed speech to text more accurately. The library for Processing was the least accurate, and it was not flexible to use in combination with a GUI. Using Python resulted in a challenge, because the code that controls the Kinect and visualizes the GUI is written in Processing.

A first approach to use the speech recognition library for Python, entailed using Python mode within Processing. Unfortunately, the speech recognition library of Python was not recognized by Processing. The second approach was to use the Java code for speech recognition, since Processing is Java based, but Processing also did not recognize the speech recognition library for Java. The final option was to explore to possibility to call the Python code from the Processing code and see if they could communicate.

A library made it possible to call the Python code within Processing. However, the Python code cannot communicate with Processing while it is still running. Therefore, it cannot give continuous feedback whether or not the user's speech input matches the words to address the system. Only if the speech recognition was successful does the Python code return a statement to Processing. The Python code is called in separate thread. This allows the Processing code to continue, while it keeps checking whether the Python code returned a statement. When this is the case, the system continues and visualizes the main menu.

Discussion

In the previous sections the design concept of the ROWBot was explained in detail and three user studies were described. Specifically, in experiment 1 user preference for different interaction methods was investigated. In experiment 2 the sensor data was measured and analyzed. Finally, in experiment 3 user preferences for different times for switching functions were tested.

The results of experiment 1 show that users preferred horizontal head rotation as the interaction method, as it was most comfortable while rowing. Horizontal head rotations imposed an important design challenge, because by turning his head the user may lose visual contact with the interface. We tried to minimize this problem by selecting a threshold of 25° rotation that is required in order to select a function and which minimizes erroneous selections due to noise in the data. Nevertheless, future studies could further investigate in more detail what magnitude of head rotation would be optimal and preferred by the users, given an improvement of the sensor data.

Similarly, in experiment 3 it was found that users prefer head movements over speech, which is in line with the Principle of Least Effort [1]. This principle is a theory that postulates that humans choose their actions based on the path of least resistance. As explained in earlier in this report, participants find using speech as input more demanding, because they need to think about what to say. For example, going to the next state can be indicated by saying "go", or "next", which makes it easy for the user to say the incorrect word. Especially because the user is also paying attention to his or her workout. Moving your head left or right is less ambiguous and therefore takes less effort.

Moreover, in experiment 3 it was looked at the prefered filler time when the user wanted to go back to the previous view or make a selection from the "Control" view. However, for now, the "Control" view only has two options so the range of choice is limited. For future implementations, if more options would be added, the range of choice would increase and, therefore, the user would need more time to check if they made the right selection when resuming to the neutral position with their head. Therefore, in this case, the selection filler could be slower to allow the user enough time to recover from an unwanted selection and go back to navigating through the options.

The analysis of the sensor data indicates that the standard face detection algorithm provided by the KinectPV2 library for Processing may not be the best suited tracking algorithm for the here presented application. Due to the noise inherent to the data, the interaction with the system is not very smooth and error prone, despite using a smoothing function. Furthermore, as mentioned in the design brief, sensor performance is typically measured and optimised by collecting data about classification performance for different conditions. The outcomes can then be approximated and compared by calculating receiver operator curves. However, given the noise in the sensor data, it was very difficult to distinguish real head movements from random artefacts, which made an analysis of classification performance unfeasible. In order to measure the sensor performance correctly, efforts need to be made to increase the accuracy of the head orientation detection. In the future it could be explored whether better results can be obtained from using HD face detection, for example, by using a recurrent neural network to analyze and classify head movements to determine whether a selection occured. Since HD face detection describes the face with multiple face vertices this would be ideally suited for a machine learning application. Furthermore, additional data such as previous adjustments and user preferences could be used to further increase the accuracy of the system and to prevent errors from occuring.

The system is currently tested as a prototype which uses a laptop's screen to display the interface. In the envisioned system, the screen would be integrated within the rowing machine. Changing the dimensions of the screen has a large effect on the interface's design decisions, including the filler times. Therefore, the outcomes of experiment 3 are now only valid for laptop screens with similar dimensions as the one used in the experiment. Furthermore, currently ROWbot is addressed through voice control. Yet, in further iterations, the possibility to address the system through a specific sequence of head-movements in combination with speech can be explored, to further reduce the possibility of unintentionally addressing the system. Besides that, studies can explore the possibility of addressing and controlling the system entirely by voice recognition or head movements depending on the preference of the user. This will help to make the whole system available for users who prefer to use either interaction through head-movements or interaction through voice-control.

References

[1] Wilson, L. (1949). Human Behavior and the Principle of Least Effort.

[2] Lecture Slides Complex Sensors, by J.B. Martens

[3] Bellotti et al., Making sense of sensing systems: five questions for designers and researchers, 415-422, CHI 2002 (the 5 A's)

[4] Jiachun Du, Thomas van Rooij & J.-B. Martens, Mirrored Perception Cognition Action Model in an Interactive Surgery Assist System, Human Computer Interaction International (HCII) Conference 2017

[5] Vangos Pterneas (2015) How to Use Kinect HD Face, retrieved on 26.06.2018 from https://pterneas.com/2015/06/06/kinect-hd-face/

[6] Works and Portfolio of Thomas Sanchez Lengeling (2015) retrieved on 26.06.2018 from http://codigogenerativo.com/code/kinectpv2-k4w2-processing-library/

Appendices

Appendix 1: Main outcomes of discussion after performing experiment 3

- Middle menu: you need more time to go to a second screen. When you go back from volume/skipping songs, you have already done you're intended action so you want to go back faster.

- When you have a filler that goes too fast, you panic. You depend on the filler, so you want to make you're decision within the given time frame.

- Controlling the system with head movement feels more natural then speech. Also because you can address the system with voice recognition in multiple ways: next, go, up, are al words you can use for the same action. That makes it confusing. Using your head movement as input is less ambiguous.

- So it depends on your action how fast the filler can be. For example, you need more time to process if the action "skipping song" has worked or not. For changing the volume, the time to process if you action was performed correctly by the system, takes less time.

- Using speech to control the system felt more socially uncomfortable. You are already moving on the rowing machine, so adding a small head movement does not make you uncomfortable. If you need to address the system for each action by voice recognition, then it feels like the people take notice.

- Now for the testing, the researcher controlled the system and we did not use the Kinect sensor due to technical issues. This meant that the participant felt that they asked the researcher to handle the system. This form of interaction is similar to social interaction where you have a conversation with someone. However, the participant reasoned that if you control the system yourself by voice, then it becomes strange to 'ask' the system something. Since the system is not human, it feels awkward to talk to a screen.

- People want to take the path of least resistance. Using speech makes you think longer about what you need to say to address the system, while with head movement you only need to look left or right. Therefore, it feels like controlling the system with speech takes more effort. Given the path of least resistance, using head movement is preferred over speech recognition.