Proceedings
of the
ISY Workshop

Issue #1, 07-24-2013

Faculty of Technology
Summer term 2013, Bielefeld University

Editors: Thomas Hermann, Florian Lier & Eckard Riedenklau

Index

INTELLIGENT SYSTEMS PROJECT: SMARTPHONE LOCATION CLASSIFICATION BASED ON WIFI FINGERPRINTS

S. Jebbara, F. Tristram, C. Poggemeier

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: J. Wienke, J. Moringen, S. Wrede

ABSTRACT

This paper describes an indoor localization system based on WiFi signal strength fingerprinting. Mobile devices are used to measure the strength of available WiFi signals which are then collected externally for classification - we chose Android smartphones, since they are widely available and easy to program. The system reaches high degrees of classification accuracy with relatively small sets of training data, and is easy to use.

1. INTRODUCTION

A smart home needs to know the position of it's users. A lot of conceivable features of an intelligent room or a smart home depend on it. Activating the lights around a person at night, turning off the stove when she leaves the room or activating the coffee machine when she sits down at the kitchen table in the morning are all dependent on said person's position.

Yet the exact location in XYZ-coordinates is irrelevant for majority of applications. If there are only two different light sources in a room you do not need to know an agent's exact location to switch on the right one, you only need to know which of the two is closer. For a lot of practical uses it is only important to know whether the agent is in a given room at the moment or not. In addition to that, [1] shows that the nature of WiFi signals being distorted by walls, antenna orientation, rain or just people walking by makes acquiring exact room coordinates rather difficult. [2] describes a coordinate based localization system, but it requires the user to create a signal strength map for the entire desired area.

This is why we opted for a classification based approach for localization. Another reason was of course that it should work with the one feature nearly all smartphones have installed: A WiFi-antenna. The idea is that the combined individual signal strength from multiple access points throughout a building should be unique for a limited number of important positions in said building. By recording a sample set of WiFi signal strength scans at a couple of points of interest it should be possible to use classification methods to later determine which of these sets a given scan would most likely belong to and therefore which point of interest (POI) the agent is closest to. [3] explores a similar approach, but arranges the classified points closely together and in a straight line.

The remaining paper is organized as follows. In section 2 we will give an overview of the features required for this project. Section 3 gives a description of the framework which implements these features. Section 4 shows the experiments we conducted to test our implementation and the results will be evaluated in section 5. We conclude this paper with a discussion of our findings in section 6.

2. REQUIREMENTS

A classifier-based localization service consists of various different software components. Required are:

i. A framework which handles the network-based data transfer between the different components. While it would be in theory possible to do the location classification on a phone, it seems advantageous to instead have a server do the calculation. Apart from saving battery charge and being faster, it is usually not as interesting for the device to know it's own location as it is for an external observer. With a server-based architecture you can set up a single server for every room which will work with every device instead of having to have a representation for every possible room ahead of time build into the device's app. For a real world application it would also be important for the system to be energy efficient and to handle the simultaneous input from an arbitrary number of devices.

ii. An application that runs on the device to be localized. It has to acquire the measured WiFi signal strength from the device and send it to the classification application.

iii. An application that works as an interface for the classifier. It should receive the WiFi data and provide the

location information to other clients.

iv. A software module which implements actual classification functions. Optimally the actual classification algorithm is easily exchangeable. The spacial resolution can be relatively low, as for many applications one class per room is sufficient. It is more important to get a high classification accuracy.

Points i. to iii. are covered by our SSOSDAD-Framework, which will be presented in detail in the next chapter.

For i. we chose to use the Robotic System Bus (RSB) framework [4]. It enables the instantiation of logical buses with network-wide visibility, which basically allows us to broadcast the data from one device to a number of different processes.

For ii. we wrote an Android-App in Java, since Android offers an easy to use API to access a device's hardware and compatible devices are widely spread. It is included in our SSOSDAD-Framework in section 3.

For iii. we created a filter for our SSOSDAD-Framework that is linked between a device and a client, taking the WiFi data from the former and providing the location to the latter. Section 4 provides a more in-depth look on this.

Point iv. is covered by a Support Vector Machine from the Weka-library. More on this in section 4.

3. SSOSDAD-FRAMEWORK

The *Server-based Streaming Of Sensory Data from Android Devices* (SSOSDAD) Framework consists of three components: An app which runs on an Android device and provides data from said device on demand, Client-Software which requests various data from mobile devices, and a server which manages the communication between all the other system components and keeps track of all participants. It also offers the ability to create a filter, which alters the data send from a device in a given way to prevent multiple clients having to do the same computation.

The relation between server and devices/clients is (1, N), meaning there can be arbitrary number of devices and clients registered at a server, but they can only register with a single server at a time. There is a predefined bus in the network for registration while the server is running which a device/client writes to in order to register. The server then saves the available clients/devices for as long as they send an alive-signal in steady intervals.

A client can then request a list of registered devices from the server, which also includes the available data streams from each device (mostly hardware sensors like a gyroscope, but also information like incoming calls). The client then makes a requests for a specific data stream from a specific device with the server. The server forwards the streaming request to said device, and upon confirmation provides the client with the bus address for the stream. The device continues writing into that bus until a stop command comes from the server. Multiple clients can read from one bus, so we have a (N, N) relationship between the clients and the devices.

Filters are (usually) small programs that are linked between a smartphone and one of it's output buses. It takes the data sent from the phone, modifies it (for example by smoothing) and writes it to the output bus. To the client it seems like the now refined data is coming directly from the phone.

In our specific case, the devices perform a WiFi scan and forwards the data to a location filter. The filter wraps the actual localization and provides it device's location to the clients.

The architecture of this framework allows for easy use by third party developers, since they only need to write their own clients without having to deal with how the data is actually transferred. Running only a single server per network reduces registration and communication protocol overhead. And since the server has only an administrative role it needs relatively low amounts of processing power.

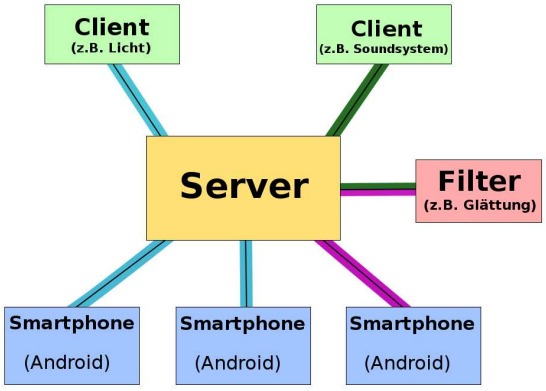


Figure 1: Illustration of SSOSDAD

4. SOFTWARE COMPONENTS

The heart of our software consist of a three parts: a smartphone app for Android devices, a location service, which handles incoming data and the localization component, which uses machine learning techniques to estimate the users position. We implemented these parts with JAVA 1.7 and Android 4.1.2. See fig. 2 for a schematic overview of the system.

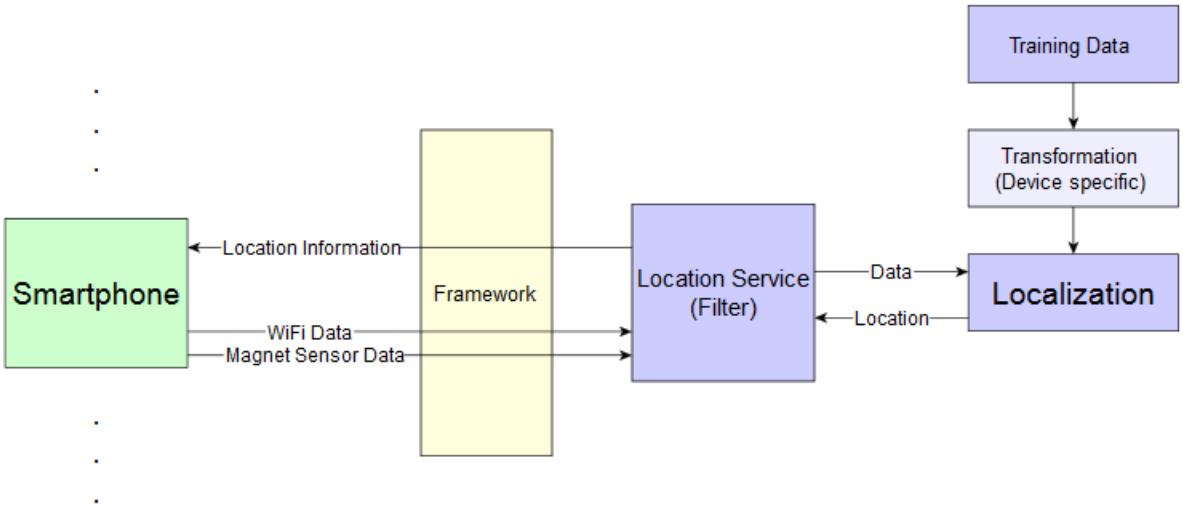


Figure 2: System Components

For our goal, to enrich the framework by positional information, we built a new smartphone app. This app was created on the basis of our former framework smartphone app. We added a few features to record WiFi training data more conveniently. In general, the smartphone app registers with the server of our framework and publishes a list of provided sensors. In our scenario, we are interested in the*received signal strengths* (RSS) of nearby access points. The location service (which previously connected to the server as a filter) is notified of the new device and requests the streaming of the WiFi signals. Since the location service is registered as a filter, it creates a pseudo sensor for each processed smartphone. Precisely, that means, that the list of provided sensors is extended by the new location data, which is subsequently available for other clients.

As soon as the smartphone begins streaming of the WiFi data, the location service receives this data and passes it to the localization component. This part implements the classification[1]. Now the localization component computes the input vector for the classification of each new scan. In principal, this vector contains the RSS of each known access point. Since the signals of all access point are not available at each location and for each individual measurement (due to reflection and absorption), some values are missing. Those errors in measurement were fixed by assigning the lowest possible signal strength to the missing values: -100 dbm. With the computed vector, the SVM[2] can estimate a location based on the training data. To deal with hardware variance in the RSS, which arises when using different smartphones, optionally, a transformation can be applied to the input vector beforehand, so that it better matches the training data. The articles [6] and [7] describe a linear transformation between the measured WiFi data of different devices. This topic is briefly discussed in section 6.2 later.

After the localization component estimates a location, the result is passed back to the framework, where it serves as the earlier described pseudo sensor for the specific smartphone. The values of the pseudo sensor can be accessed by other clients exactly like a those of real sensor, which are implemented on the device itself. As future work, the filter can send the information of the location to a client on the smartphone where it is visualized properly. This could be used as an indoor navigation system.

5. EXPERIMENTAL SETUP

For our experiments we chose two different locations: a medium-sized computer science laboratory and one floor segment of the technical faculty wing. For the laboratory we defined three points of interest (kitchen, couch and desk), for the floor we defined 14 POIs in different rooms respectively (5).

We recorded the data with our Android-app while holding the smartphone in one hand. The reason for this is that the users would usually navigate while looking at their phone. We also made sure to hold the phone in a natural fashion. Keeping the orientation of the phone constant

[1]We used the library WEKA[5] to have a pool of implemented machine learning algorithms, but the SVM showed the best results for our scenario. Thus, we will focus on this technique.

[2]Here we use the SMO implementation of the WEKA library. The SVM uses a 2-degree polynomial kernel and a complexity parameter $C = 1$. The multi-class problem is solved by the SMO using pairwise classification.

would reduce the noise of the data, but would also not reflect the way an end-user would use his device.

During both experiments we placed one additional router in the respective location which we needed to connect the phones to the server. The signal strength of this router was measured in the experiments as well as the signal strength of all other routers in the vicinity. We did not know the location of any of these routers. In the lab we received signals from 18 different access points, on the faculty floor we received more than 100. This is likely due to the fact that the routers here broadcast multiple WiFis simultaneously, which are all counted as single access points.

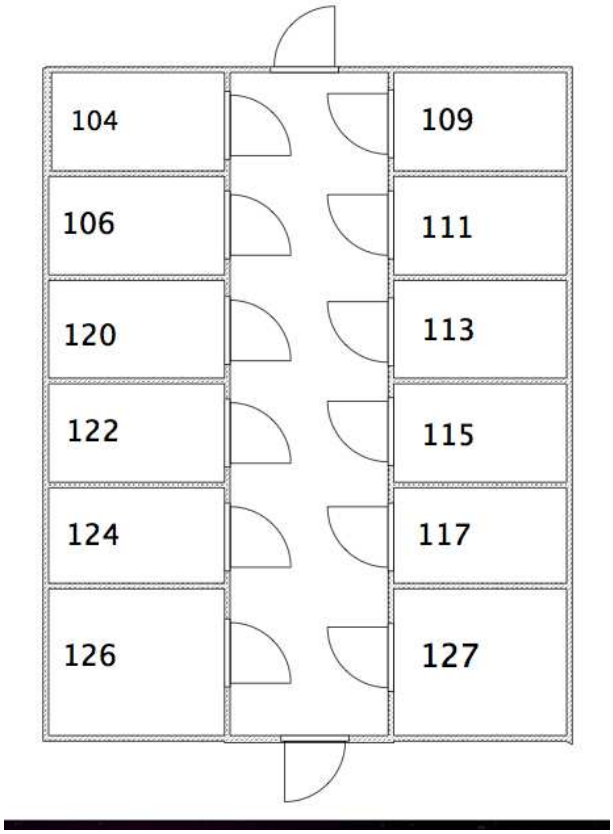


Figure 3: Floor N5

The lab experiment was conducted with three different smartphones, two of which were of the same type (1x Galaxy Nexus and 2x Galaxy SII), while the measurements on the faculty floor were only performed by a Galaxy Nexus type phone.

6. EVALUATION

6.1. Results

Before we present the results of our classification, we will discuss the recorded data. Fig. 4 shows the received signal strengths at each location for each access point. The data was recorded at three different positions in an office room (see fig. 5) of approximately $7m \times 12m$ and contains the RSS of 18 access points. Generally, the measured signal values show a relatively low variance. As we described in section 4, some of the received data is corrupted, due to missing signal values of access points. Since we correct these errors by assigning -100 dbm to the missing signal, the representations of the RSS for the access points 5 and 14 to 18 appear in some cases distorted.

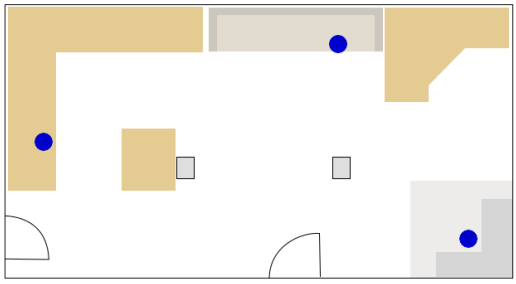


Figure 5: Room drawing of the office room. The blue dots mark the locations where we recorded the data. From left to rigth: Desk, Couch and Kitchen.

Based on this WiFi data, we trained a classifier as described in section 5. Since the number of training samples per location is not obvious to choose, we conducted a series of experiments. In each experiment we trained our classifier with a different number of randomly selected training samples, starting with 1 training sample per location and ending with 20 samples. The results of each single experiment were averaged over 40 iterations. Fig. 6 shows the results, displaying the number of samples on the x axis and the achieved F-measure for the classification on the y axis. It is interesting to see, that with just one training samlpe per location we can achieve a classification accuracy of 71%. Further, we can dramatically increase the accuracy to 90% if we use 5 samples for each location and even to 96%, if we raise the number of samples to 20 points per position.

We confirmed these high accuracies by conducting another experiment. The test environment was the office floor mentioned in section 5. Fig. 3 shows the corresponding floor plan. As described, we recorded data for 11 locations, which were distributed inside the office rooms, as well as in the corridor. Again, we achieved promising results, with 92% correctly classified samples.

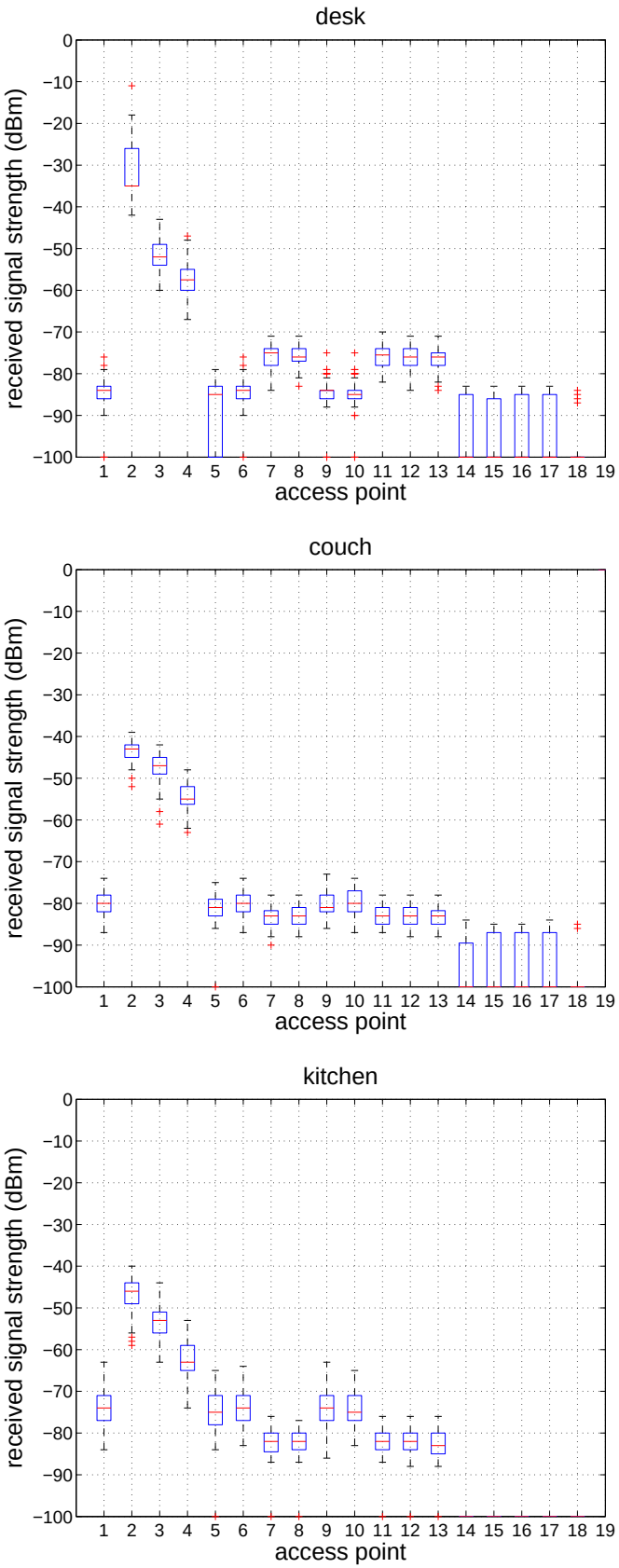


Figure 4: Received signal strength per location and access point.

Training \ Testing	Galaxy Nexus	Galaxy SII (A)	Galaxy SII (B)
Galaxy Nexus	96.48%	77.81%	78.43%
Galaxy SII (A)	62.69%	91.97%	90.85%
Galaxy SII (B)	66.14%	88.42%	90.24%

Figure 7: Variance in classification accuracy due to hardware variance in the training and test phase.

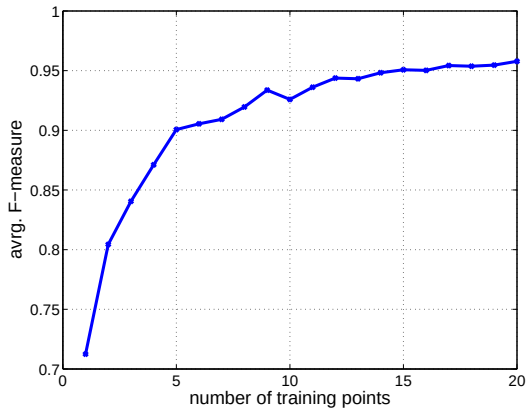


Figure 6: Accuracy in relation to the used number of training samples per location

In the above described experiments, we used the same smartphone (Galaxy Nexus) for the recording of the training data as well as for the test data. Since we intend our software to be independent of the smartphone model, we investigated the performance of our software with other Android devices in additional experiments. We were especially interested in the performance for smartphones, for which we do not have any training data. Table 7 shows the accuracies for different devices as training and test devices.

The results show, that the variance between similar smartphone models is, as expected, low. However, we detected an unpleasant variance in the cases, where the training and test devices are different models. In these cases, the classification is more reliable, if the training data was recorded with the Galaxy Nexus. This may be for the reason, that the data recorded with the Galaxy Nexus is less noisy than that of both of the other smartphones.

6.2. Discussion

The previous section shows, that a localization system based on the received WiFi signals, yields utilizable results. The design of the framework as a *Point-of-Interest* localization, leads to even more robust location estimations compared to computing the position of a device in some Euclidean coordinate system [8]. The high accuracy of our localization service could be even more enhanced, if we smooth each location estimation by performing a fusion of the last k classification results, as done in [3].

Since the location service is implemented as a filter component, according to our SSOSDAD framework (see section 3), it is (in theory) capable of processing the data of an arbitrary number of smartphones. As our primary use case for the software is home automation and enrichment of an intelligent room, the number of simultaneously processed smartphones is in general rather low, say 5. For these low numbers of participants, the systems reaction time is still a fraction of a second.

To use this software in a real world scenario, the problem of hardware variance (as depicted in table 7) has to be solved. [6] and [7] propose a linear transformation, which can align the RSS of an ”unknown” device to those of the training device.

$$rss_{training} = c_1 \cdot rss_{unknown} + c_2 \tag{1}$$

An unsupervised learning algorithm could determine the slope c_1 and the intercept c_2 of the linear shift on runtime, aligning the new device to the trained model ([6],[7],[9] and [10]).

7. CONCLUSION

Our main goal, estimating what point of interest a device is closest to at a given time, has been achieved. As it had been shown in our experiments, we were able to correctly classify the phones position with an accuracy of more than 90%, needing only 5 points of data for training. If the points of interest would be higher in number and/or closer together, results might diminish, but as it was stated before we only aimed to have relatively few points of interest at different locations at the site of deployment. For what we set out to do we developed a modular simple-to-use framework which could be extended for custom purposes.

8. REFERENCES

[1] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye, "Push the limit of wifi based localization for smartphones," in *Proceedings of the 18th annual international conference on Mobile computing and networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 305–316. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348581

[2] P. Claro and N. B. Carvalho, "Local positioning system based on artificial neural networks," in *Proceedings of the 17th international conference on Artificial neural networks*, ser. ICANN'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 699–708. [Online]. Available: http://dl.acm.org/citation.cfm?id=1778066.1778148

[3] P. Menendez, C. Campomanes, K. Trawinski, and J. Alonso, "Topology-based indoor localization by means of wifi fingerprinting with a computational intelligent classifier," in *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, 2011, pp. 1020–1025.

[4] J. Wienke and S. Wrede, "A middleware for collaborative research in experimental robotics," in *System Integration (SII), 2011 IEEE/SICE International Symposium on*, 2011, pp. 1183–1190.

[5] G. Holmes, A. Donkin, and I. Witten, "Weka: A machine learning workbench," in *Proc Second Australia and New Zealand Conference on Intelligent Information Systems, Brisbane*, 1994.

[6] A. W. Tsui, Y.-H. Chuang, and H.-H. Chu, "Unsupervised learning for solving rss hardware variance problem in wifi localization," *Mob. Netw. Appl.*, vol. 14, no. 5, pp. 677–691, Oct. 2009. [Online]. Available: http://dx.doi.org/10.1007/s11036-008-0139-0

[7] C. Figuera, J. L. Rojo-Álvarez, I. Mora-Jiménez, A. Guerrero-Curieses, M. Wilby, and J. Ramos-López, "Time-space sampling and mobile device calibration for wifi indoor location systems," *IEEE Transactions on Mobile Computing*, vol. 10, no. 7, pp. 913–926, July 2011. [Online]. Available: http://dx.doi.org/10.1109/TMC.2011.84

[8] Y. Kim, H. Shin, Y. Chon, and H. Cha, "Smartphone-based wi-fi tracking system exploiting the rss peak to overcome the rss variance problem," *Pervasive Mob. Comput.*, vol. 9, no. 3, pp. 406–420, June 2013. [Online]. Available: http://dx.doi.org/10.1016/j.pmcj.2012.12.003

[9] M. Kjaergaard and C. Munk, "Hyperbolic location fingerprinting: A calibration-free solution for handling differences in signal strength (concise contribution)," in *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, 2008, pp. 110–116.

[10] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Practical robust localization over large-scale 802.11 wireless networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking*, ser. MobiCom '04. New York, NY, USA: ACM, 2004, pp. 70–84. [Online]. Available: http://doi.acm.org/10.1145/1023720.1023728

INTELLIGENT SYSTEMS PROJECT: AUDEYE

V. Losing, L. Rottkamp, M. Zeunert

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: Dr. T. Pfeiffer, Dr. K. Essig

ABSTRACT

This project aims to assist a chess player in a real-world chess game against a real opponent. We constructed a system in which this player wears eye-tracking glasses (ETG) and an ear-piece. The ETG record both his eye movements and the chess scene itself. This input is then processed, including image recognition to obtain the state of the game. The player communicates with the system by executing eye gestures. In this way he can control the system and request hints for good moves which are calculated by a chess engine. He may also choose to be informed about the usefulness of executing a move with the chess piece he looks at. These values are made audible by the means of sonification. Additionally, a player-vs-computer mode was implemented, the game's protocol is saved and several statistics are collected.

1. INTRODUCTION

In chess, inexperienced players are usually feeling unconfident when facing more experienced players. This is clearly rooted in the fact that these more experienced players can *see* more than a beginner, even if both have full knowledge of the rules, which we will assume here. The beginner can *see* correct moves, but only the experienced can *see* the value of those moves. Thus the beginner will feel quite blind—and most people don't enjoy playing games with unlike preconditions.

Having a supporter would certainly ease the situation by increasing confidence and therefore prevent beginners from early quitting. This supporter may also help in the aftermath of the game by providing a protocol of moves and other observations. Creating such a supporter with the help of electronic devices was our aim during the project described in this paper, which is organized as follows:

We start by illustrating the utilization of our system in section 2, followed by brief introductions into the system design in section 3 and the hardware setup in section 4. We then describe the software components created to reach our goal in section 5 and proceed by presenting the results of a short evaluation of our gesture recognition component in section 6. In section 7 we discuss our project and related work before finally giving a conclusion in section 8.

2. INTERACTION DESIGN

Our electronic supporter must be able to follow the game, receive commands from the player and then communicate his advice. Despite being in a public environment, this conversation is kept private: The player commands with nothing but his eye movements, while the supporter's response is made audible through an ear-piece—hence the project's name Aud(ible)Eye(d). Therefore, the player wears eye-tracking glasses (ETG), which record both his eyes and the image of the scene, in particular the chessboard and its pieces. This data is then processed by our system and the response is delivered in natural language or as non-speech audio ("sonification").

Initially, the supporter follows the game passively by just stating recognized moves, but is already watching out for certain eye gestures. Four different gestures are introduced as shown in Figure 1. The u-gesture enables *Sonification Mode* if disabled and disables it if enabled. Likewise, the x-gesture toggles the *Solo Mode* setting. The two remaining gestures trigger actions: The n-gesture can be used to ask the system for the next best move and the α-gesture to undo the last recognized move—a feature which comes in handy in case a move shall be taken back in a training game.[1]

To indicate the beginning of a gesture, the player fixates

[1] This feature has another application which we will discuss in section 7.

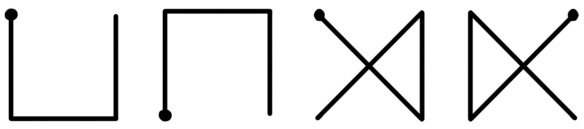


Figure 1: The four gestures from left to right: u, n, x, α. The beginning of each gesture is indicated by filled circles.

a point of his choice for an unnaturally long time.[2] The system will then inform him to begin the gesture and react if one of the four gestures was classified with sufficient certainty.

3. SYSTEM DESIGN

Physical constraints of the project were well-defined before the project was started: The ETG and the notebook (both described in the next section) were provided by our supervisors. The software for connecting the ETG is only available for Microsoft Windows and C++, so at least the parts of our system making use of the ETG had to be written in C++ on Microsoft Windows, too.

One of the main themes of the project was to make use of the Robotics Service Bus (RSB)[3] for communication between programs. RSB is a message-oriented bus developed at the Cor-Lab[4] at Bielefeld University. While RSB is capable of linking components running on different operating systems (and written in different programming languages), we decided to stick with a single computer to avoid potential network problems and additional work in development and deployment. Consequently, we were able to manage our whole code-base in one Microsoft Visual Studio workspace, which also accelerated the development process.

Apart from hardware constraints, other physical constraints are those of the environment in which the user operates the system. Our setting involves playing chess in a natural way, so the given situation is characterized by the user of the ETG sitting in a chair, with the chessboard placed on a table in front of him as depicted in Figure 2. When the image recognition proved to be rather difficult, the conventional chess pieces were replaced by flat tokens to avoid overlapping pieces and allow a more robust detection of the chessboard itself.

4. HARDWARE DEVICES

The whole system is running on a single notebook, which is connected to the ETG via USB. The ETG are a product of SensoMotoric Instruments (SMI)[5] and are designed to be operated in a Microsoft Windows environment. Gaze data is provided with a temporal resolution of 30Hz while a front-facing camera delivers a video stream with a resolution of 1280x960 pixel at 24Hz [1].

The ETG are operated via drivers by SMI, which can can be controlled via the C++ API also provided by SMI via

Figure 2: Two of the authors playing a game of chess with the aid of the electronic supporter.

the iViewNG SDK. To setup the ETG, the iViewETG software is used to connect to the device and then to calibrate the glasses. Calibration is a crucial point because otherwise the delivered gaze data would be inaccurate. To calibrate, the user is advised to look at certain points (as marks on a blackboard) while the operator selects the corresponding points in the video image. To obtain a good calibration, a the three-point calibration tool is used, the three points being placed in a way that resembles the letter L. These points lie on a plane perpendicular to the user's viewing direction.

5. SOFTWARE COMPONENTS

Numerous programs are running simultaneously, each contributing one or more important functionalities. As mentioned before, communication is handled via RSB: In this way the data from the ETG is processed step-by-step until finally a sound feedback is generated. An overview of the components is shown in Table 1.

Despite the fact that our system is decentralized in the way that every component may communicate directly with other components, a controller is helpful to coordinate them. For example, when a component is started, it should firstly obtain the global configuration of the system, for example whether *Solo Mode* is enabled or disabled. In a modular environment, it should not be the responsibility of input devices as *GestureDetector* to check if new components were started and then resend previously recognized gestures: It should only publish detected gestures and nothing else. Following this logic, it is also not responsible for interpreting gestures.

Therefore the component **ActivityController** was introduced to act as a central contact point. It maintains the current configuration and publishes configuration changes and actions via RSB. It also listens for configuration requests:

[2] Two seconds proved to be a good compromise between a low number of false positives and operating comfort

[3] Homepage of RSB: https://code.cor-lab.org/projects/rsb

[4] Research Institute for Cognition and Robotics, http://www.cor-lab.de

[5] SensoMotoric Instruments, http://www.eyetracking-glasses.com

Name	Description	RSB Inputs	RSB Outputs
ActivityController	Maintains configuration, interprets gestures and triggers actions	ConfigurationRequest, Gesture	Configuration, Action
RSBEyeTracking	Connects to ETG and delivers eye-tracking data and scene images		ImageFixation, GazePoint
ChessBoardFinder	Image recognition: Extracts occupied fields and fixated field from scene image	ImageFixation	OccupiedFields, FixatedField
GestureDetector	Recognizes eye gestures from gaze points	GazePoint	Gesture
ChessExtractor	Extracts game situation from occupied fields	OccupiedFields	Chess
ChessStatistics	Records and visualizes player's eye movements	Chess, FixatedField	
Sonificator	Generates sound from move values	FieldRating	
Speaker	Generates natural speech output	SentenceToSpeak	

Table 1: Components of the system. *Inputs* are RSB channels the component listens to; *Outputs* are RSB channels published to. Configuration and Action inputs are not shown here, neither are ConfigurationRequest and SentenceToSpeak outputs.

When a component starts, it firstly sends a configuration request which is then answered by the *ActivityController* with the current configuration. Secondly, the *ActivityController* listens for recognized gestures and then is the one who interprets and publishes the user's wishes by changing and publishing the current *Configuration* or publishing an *Action*.

5.1. Basic Components

The component **RSBEyeTracking** establishes an interface to the ETG and publishes both video data and fixation data. The Artificial Intelligence Group[6] recently started to develop an interface which will be able to abstract from different ETG. This interface makes use of the also new iView API and we successfully transitioned to this new system during the course of our project. In the future, the group will finish their own RSB component, which could then replace *RSBEyeTracking*.

Human eye movement consists of fixations, periods of almost no eye movement, which are separated by fast movements called saccades [2]. In our application, we are only interested in fixations, particularly fixations long enough to indicate that the user is actively looking at this point. To detect fixations, we implemented a simple velocity-based algorithm which basically monitors the velocity of the user's gaze path and triggers a fixation when the velocity rises after a period of little movement as indicated in [2].

When we detect a fixation, an *ImageFixation* is published to RSB. It consists of the current scene image combined with the fixation's coordinates and duration. To recognize eye gestures, we do not need the scene image but a higher publishing frequency in order to recognize fast gestures. Therefore *GazePoint*s are also published at a fixed and higher rate (30Hz), containing only the coordinates of the user's gaze points.

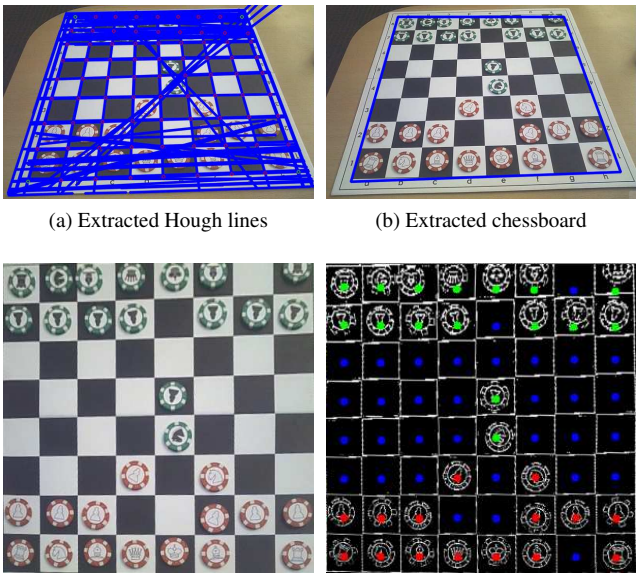
*ImageFixation*s are processed by the component **ChessBoardFinder**, which has several functionalities, mostly analyzing the obtained scene image as shown in Figure 3: Firstly, it finds the chessboard and rectifies the image so that the chessboard has a quadratic appearance, as if the photo was taken from a bird's perspective. Then, occupied fields are extracted and published to RSB. Finally, the user's fixation is projected onto the chessboard and the currently fixated field is published. As the camera is always moving and the chessboard may not always be visible in the scene image—for example when the player talks to his opponent—the chessboard detection can not be reduced to a tracking-problem.

At the beginning of our project, the recognition of individual chess pieces proved difficult, so we decided to split the task of recognizing the current state of the game: *ChessBoardFinder* just tells us *which* fields are occupied. In a second step, changes of this result are analyzed by the component **ChessExtractor** which then calculates the move that must have happened. Obviously, this strategy only works when the system is able to follow the game from the very beginning.

Technically, *ChessBoardFinder* makes heavy use of the open source computer vision library OpenCV[7]. Although OpenCV provides a build-in function to detect chessboards for calibration purposes, this method proved to be unreliable when chess figures populate the field and also to be not fast enough for real-time applications. In our approach for chessboard detection, we use Hough transforms to detect lines similar to Escalera and Amingol in [3]. These lines are then filtered by various heuristics using background knowledge as the typical structure of a chessboard. In short, the board will be represented by a perspective-distorted grid of 9x9 lines which can then be rectified to ease further processing: The individual fields will be scanned for *edges*, curves

[6] Artificial Intelligence Group, Bielefeld University, https://techfak.uni-bielefeld.de/ags/wbski/

[7] Homepage of OpenCV: https://http://opencv.org



(a) Extracted Hough lines (b) Extracted chessboard

(c) Rectified chessboard (d) Occupied fields (Red/green dots)

Figure 3: Stages of extracting occupied fields.

in the image at which the brightness changes sharply as depicted in Figure 3d. As these edges are introduced by the pieces themselves, we are reliably able to discriminate between empty and occupied fields. Finally the color of each piece is determined and the occupied fields are sent out via RSB.

5.2. Interaction Components

The component **GestureDetector** detects gestures in the user's eye movements. For classifying gestures, we employ the *$1 Unistroke Recognizer* by Wobbrock, Wilson and Li [4]. Bayor Wetzel[8] translated the reference implementation to C++ code, which we use in our system. The gesture recognizer matches the recorded gesture with stored templates, making custom gestures possible. We chose this recognizer because it is well-documented, established, fast and accurate. Additionally it proved to be easy to include due to its small footprint and minimal dependencies. It was primarily designed to match mouse gestures or gestures painted on a touchscreen displays, both having explicit gesture beginnings. As there are no intrinsic beginnings in our application scenario, we let the user signify the start of an eye gesture by a long fixation.

The component **ChessExtractor** maintains the state of the game and publishes the data type *Chess* describing the state of the game. As described above, *ChessBoardFinder* publishes a list of occupied fields when a move occurs. This list is then analyzed by *ChessExtractor*, which knows the actual chess rules. When a valid move is recognized, it is communicated to the user via a *SentenceToSpeak*.

Additionally, this component is connected to the chess engine "Brutus"[9] by Stephan Vermeire to provide hints and move verification. Brutus is provided as open source C++ code and as a consequence we were able to modify it to fit our needs, for example accessing the value of certain moves. When a user requests a hint, the chess engine is given a few moments to calculate the next best move which will then be communicated. Also, if *Sonification Mode* is enabled, the value of each piece is calculated and cached to allow a timely sonification.

5.3. Output Components

If *Sonification Mode* is active, the component **Sonificator** is used to convert numerical values into sounds: When the user looks at a chess piece, *ChessExtractor* calculates the value of the best move possible with this piece. This value is then obtained and made audible for the user. The higher the value, the higher the corresponding sound, so he can easily find out which pieces would be good candidates for moving. To generate the sounds, portions of the open source Synthesis ToolKit in C++ (STK)[10] are included in this component. We use a Xylophone-like sound for our sonification to minimize the generated annoyance.

After the game has ended, the user may find himself wanting to analyze the game. Therefore the component **ChessStatistics** records statistics about how often each field was fixated. Additionally, fixations of individual chess pieces are counted.

The component **Speaker** receives arbitrary English sentences and uses the Microsoft Voice text-to-speech API (included in Microsoft Windows) to speak them out.

6. EVALUATION

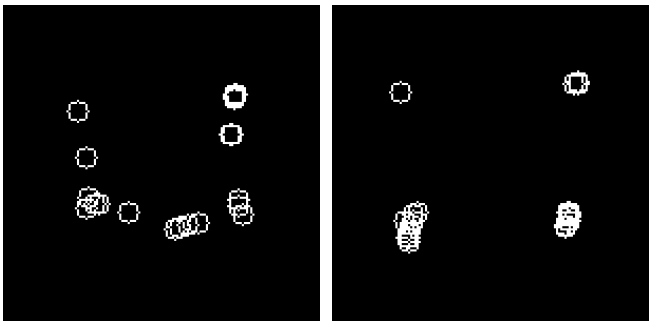
For evaluating the project, we focus on our gesture recognition component. Firstly the other components proved to be working as expected, on the other hand the gesture recognition failed to classify gestures correctly more often than not. As the only method of communicating with the system, this component acts as a central point in user interaction and therefore requires special attention. In our short experiments, we therefore asked five persons to execute 120 gestures each. We only evaluated the u- and x-gesture to allow more tries of the same gestures. As these are just mirrored versions of the n- and α-gesture respectively, we believe this will not significantly affect the obtained insights.

In **experiment I**, we placed the subject in front of a computer screen showing a single dot. After setting up the

[8] Bayor Wetzel: http://www-users.cs.umn.edu/~wetzel/

[9] Brutus: http://home.xmsnet.nl/vermeire/brutus.html

[10] Synthesis ToolKit in C++: https://ccrma.stanford.edu/software/stk/



(a) Continuous (b) Jerky

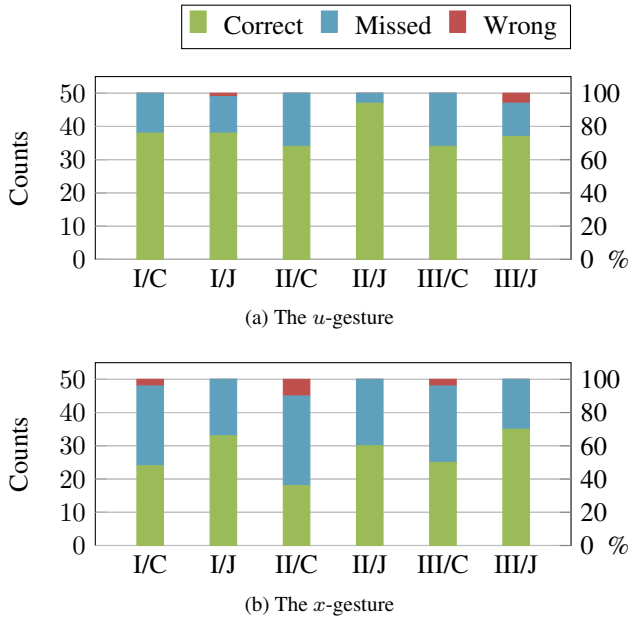
Figure 4: Typical gaze points while executing the u-gesture.

ETG and relevant parts of our system, we asked the subject to constantly fixate the dot. Every time the gesture recognition indicated that a gesture beginning was found, the dot started to move: Firstly (I) it continuously followed the path of the u-gesture as shown in Figure 1 while maintaining constant speed. After a gesture was finished, we noted the outcome of the gesture recognizer: Either a **correct** classification, **missed** classification (no gesture recognized) or **wrong** classification. This process was repeated ten times. We then repeated this procedure showing the x-gesture (I/C). Then we again presented the u-gesture, but now not by continuously moving the dot but letting it "jump" from corner to corner (I/J), leading to gaze point patterns as shown in Figure 4b. This was also carried out using the x-gesture (I/J), therefore obtaining a total of 40 gesture outcomes in this experiment.

Experiment II essentially consisted of the same task, but this time without moving the dot. Instead we asked the subject to execute both gestures at his own speed and fashion, albeit sticking to the pattern described before and therefore again obtaining 40 classification results (II/C and II/J).

In **experiment III**, we asked the subject to play a game of chess with a member of our team, thereby executing gestures from time to time. To correlate the outcomes, we also asked to announced the gesture targeted (again u or x) and whether it was executed in a continuous or jerky fashion. Again we noted 40 classification results (III/C and III/J).

The results of the experiments are shown in Figure 5. Some observations can be made: Firstly, the x-gesture generally performed poorer than the u-gesture, indicating that it may be harder to recognize by our system. On the other hand, the x-gesture is more complex and therefore its execution itself may likely be more difficult and error-prone. Secondly, the continuous way of executing the x-gesture seems to be somewhat harder than the jerky version as it is showing less correct classification results. Here, the subject's eyes have to move both horizontally and vertically simultaneously, which may be especially difficult for humans



(a) The u-gesture

(b) The x-gesture

Figure 5: Results of experiments I, II and III, each with continuous and jerky gaze paths.

when following a path at constant speed. When looking at the u-gesture, the difference between continuous and jerky execution seems to have only little influence. An anomaly is II/J in Figure 5a which shows a surprisingly high number of correct classifications. We believe this is mostly due to the fact that the subjects repeated the exact same gesture rapidly without thinking too much about it. In contrast, the presumably more difficult x-gesture seems to require more practice to be reproduced in a likewise sovereign way.

In every experiment, the number of false positives was relatively low, which is crucial for an interface through which commands are given. However this was payed for by the relatively high rates of missed classifications, as the corresponding gestures were often correct classified but not exceeding the certainty threshold we determined in previous experiments. As a fourth result, one can see that classification performance has not significantly decreased in experiment III. This is especially remarkable because of the fact that the subject's focus was repeatedly distracted in this scenario mimicking the real-world application.

During and after the experiments, we asked the subjects about their impressions: Most subjects agreed that the continuous way of executing gestures was more demanding. This is most likely due to the fact that a high number of fixations needs to be consciously enforced in this case. In contrast, looking at only three or four imagined points is both less effort and probably also easier to remember.

7. DISCUSSION

In the following, we will discuss some topics touched while working at this project. In general, our system is able to support a chess player as planned and most of our initial goals have been met. We have a few more features in mind, which could not be implemented due to time restrictions: An automatic color calibration would improve robustness in varying scenes. Also, an automatic ETG calibration would remove the need for a second person to assist with calibrating the ETG before the game starts. Despite this task being simple—it may be performed by the player's opponent even if he has no further experience with eyetracking software—asking the opponent for such assistance may not be exactly what one would call a convincing introduction of a new artificial assistant.

7.1. Eye gestures

During the first half of the project, we used so-called visual buttons to control our system. These physical tokens were located near the chessboard and triggered actions when fixated by the user. This proved to be quite effective, but the tokens had to be carried around and placed again when the user moved to a different scene. In a futuristic setting, our system should be able to be used without any scene preparations: The player would just be wearing glasses and an ear-piece and start playing.[11] Eye gestures were found to be an alternative, having the additional advantage of allowing an—in principle—arbitrary number of different gestures.

When considering to use a mouse gesture recognizer for recognizing gestures, we were skeptical—the gestures may be the same, but the way of "drawing" them is certainly different. However, Drewes and Schmidt successfully applied a mouse gesture algorithm in eye gesture recognition in [5]. On sighting possible gesture recognition libraries, we also found that the *$1 Unistroke Recognizer* internally resamples the gesture path, so mouse and eye gestures are not processed much differently.

In general, our gesture recognition component proved to be quite usable. During our evaluation, gestures were often not classified ("missed") due to the certainty threshold being set towards the safe side. However, even having to repeat every second gesture is worth the observed low false positive rate, as a false positive will cause an unwanted command, while a false negative needs no further actions from the user but repeating the gesture more clearly. It needs to be mentioned that the commands associated with the more complicated x- and α-gesture—activating solo mode and undoing a move—are seldom used in a normal game of chess. In contrast, the more reliable u- and n-gesture are typically applied more often.

[11] Surely the notebook we need now would then be included in the ETG

Controlling a program only with eye gestures may turn out to be quite exhausting if such commands are often used as we experienced ourselves while developing our recognizer. Luckily, when our program is used in a real-world application, such commands are typically rare enough to not being uncomfortable. Even in our evaluation, no subject complained about having to execute 120 gestures in less than an hour.

To increase recognition performance, further training examples could be obtained during user-specific training. Additionally, certainty thresholds could also be tuned individually: Following a targeted trade-off between false negatives and false positives, the thresholds could be adjusted independently. Yet it is not clear if these new thresholds would perform well when playing a game of chess.

7.2. Chessboard detection

The problem of following a game of chess is not new in the field of image recognition, despite often at laboratory conditions not given in our setting. For example, Piskorec et al.[6] relied on a fixed camera and chessboard, while additionally placing the camera directly over the chessboard. Another system by Matuszek et al.[7] used a 3D camera also sensing the height of chess pieces. Apart from that, they followed a strategy similar to ours: Observing the game from the beginning and updating the state of the game by watching the changes of occupied fields. While their camera was not filming the scene from directly above, it should be noted that it was mounted on a robotic arm which assumed the same position after each move, therefore providing the same viewing angle and distance. We in contrast have to detect the chessboard in various distances and angles. Matuszek et al. used traditional chess pieces and detected occupied fields with the means of machine learning, an approach leading to good results.

Our system is not able to extract field information when parts of the chessboard are not visible in the scene image. This shortcoming requires the user to take care of capturing the whole chessboard at least after every move, which could be inconvenient and distracting.

In section 2 we mentioned another use of the *undo* action. Indeed, in the beginning of our project, the image recognition would occasionally fail and deliver wrong piece positions. In some cases these led to the erroneous registration of moves which never occurred. The system was then stuck in this error state, as moves once done can't be reverted. The *undo* action could then be used to go back and finish the game with the help of the supporter. However, after some tweaks of our system—as more robust piece detection and requiring multiple consistent processing results before a move is accepted—the need for this function vanished.

7.3. Performance aspects

Concerns regarding the feasibility of running the whole system on a single notebook—which also hosts the ETG itself—turned out to be no issue. It should however not be concealed that it would probably not perform well on a low-end machine. The most demanding components are the ETG and its connector, *ChessBoardFinder*, *GestureDetector* and *ChessExtractor* when calculating piece values. As expected, components as *Sonificator* and *Speaker* only cause very little CPU load.

7.4. Sonification

The sonification used provides helpful, but limited feedback. It is possible to extend our Sonificator with further "instruments" which then could output additional dimensions. For example, one could calculate both a defensive score and an offensive score, indicating whether a figure is in danger of being captured or may itself be a good choice for attacking. These values could provide additional insight in the mechanics of chess and give the player a better feeling for certain situations. However, Flowers warned in [8] that sonification of multiple variables needs careful adjustment in order to provide good insight in the data to be displayed.

The biggest issue with the current sonification is imprecise calibration of the ETG: When looking at the chessboard from a natural sitting position, the distance between two fields is often very small. As a result, even a slightly off calibration will cause wrong fields to be visually selected and then made audible, causing unexpected and irritating sensations. Here, the proposed automatic calibration could be accompanied by a component to detect imprecise calibration—maybe by recognizing phases in which the player is wanting to get sonification output but is often looking at empty fields. Then a new calibration could be initiated to increase the fidelity of the ETG data.

8. CONCLUSION

During this project, we successfully created an artificial supporter for chess players, consisting of eye-tracking glasses and a notebook. Our system is able to follow a real-world chess game, enabling the player to interact with the system by requesting hints or the evaluation of selected pieces. This interaction is done by recognizing eye gestures which showed good performance in our evaluation. Additionally, a game protocol and usage statistics are printed at the end of each game, which may then be used to further analyze the game. The image recognition of common chess pieces proved to be rather difficult, so we decided to use colored flat tokens instead, which do not overlap and therefore can be separated more robustly. Transitioning to real chess pieces would be the next step to a universal chess supporter.

9. ACKNOWLEDGEMENT

We would like to thank our supervisors for their encouraging feedback, trusting relationship and also providing the necessary hardware. Additionally, Kai Harmening (and to a lesser extend Patrick Renner) helped us by modifying our ETG component to fit in the eye-tracking library currently being developed by the Artificial Intelligence Group at Bielefeld University.

10. REFERENCES

[1] SMI Product Sheet: Eye Tracking Glasses, 2012, http://www.eyetracking-glasses.com/fileadmin/user_upload/documents/smi_etg_flyer.pdf.

[2] D.D. Salvucci and J.H. Goldberg, "Identifying fixations and saccades in eye-tracking protocols," in *Proceedings of the 2000 symposium on Eye tracking research & applications*, ETRA, 2000, pp. 71-78.

[3] A. De la Escalera, J.M. Armingol, "Automatic Chessboard Detection for Intrinsic and Extrinsic Camera Parameter Calibration," in *Sensors 10, no. 3*, 2010, pp.2027-2044.

[4] J.O. Wobbrock, A.D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes," in *Proceedings of the 20th annual ACM symposium on User interface software and technology*, ACM, 2007, pp. 159-168.

[5] H. Drewes, A. Schmidt, "Interacting with the computer using gaze gestures," in *Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction - Volume Part II*, IFIP, 2007, pp.475-488.

[6] M. Piskorec, N. Antulov-Fantulin, J. Curic, O. Dragoljevic, V. Ivanac, L. Karlovic, "Computer vision system for the chess game reconstruction," in *MIPRO, 2011 Proceedings of the 34th International Convention*, MIPRO, 2011, pp.870-876.

[7] C. Matuszek, B. Mayton, R. Aimi, M.P. Deisenroth, B. Liefeng, R. Chu, M. Kung, L. LeGrand, J.R. Smith, F. Fox, "Gambit: An autonomous chess-playing robotic system," in *2011 IEEE International Conference on Robotics and Automation*, ICRA, 2011, pp.4291-4297.

[8] J.H. Flowers, "Thirteen years of reflection on auditory graphing: Promises, pitfalls, and potential new directions," in *Proceedings of the 11th International Conference on Auditory Display*, ICAD, 2005, pp.406-409.

INTELLIGENT SYSTEMS PROJECT: BIOSIGNAL FEEDBACK FOR MULTIPLE-CHOICE LEARNING

P. Blöbaum, F. Grimm, D. Wigand

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: A. Finke, N. Hachmeister, H. Riechmann

ABSTRACT

Current low-cost, consumer-available measurement technologies enable intelligent room setups that monitor several biological properties. We introduce a system that uses a galvanic skin-response (GSR) sensor to infer a person's stress level and electroencephalography (EEG) measurements to classify error-potentials when giving incorrect answers in a multiple-choice quiz. These assessments are applied to a multiple-choice learning scenario for language training. Though some technical obstacles exist, results indicate that such a system can improve the learning experience or results of participants.

1. INTRODUCTION

The goal behind every intelligent room design is to aid or support the inhabitants in some way that would not be possible without some specific technology. Most regular living environments have a desk that provides a working space to study or perform other tasks that are primarily made up of mental work.

During the project, conducted for the "Intelligent Room" seminar at Bielefeld University, we focused on supporting participants in a study setting. Normally learning tasks can only be monitored by observing a-posteriori achievements, like test results. When studying vocabulary for example, a pen-and-paper based system will only provide limited feedback to the participant and the overall result can only be evaluated in a written examination. Digital alternatives offer quiz-style computer programs that provide direct feedback if a question was answered incorrectly and can keep track of progress and common mistakes.

Our project focuses on additionally monitoring different unconscious modalities during such activities to provide even more feedback and leverage the capabilities of an intelligent room. By adding different sensors to the participant's computer workstation, we aim to infer information on current stress levels and participant's certainty when answering quiz questions.

Two kinds of sensors are used in our system. First, a galvanic skin-response (GSR) sensor is attached to the participant's hand. This sensor allows to measure changes in conductivity of the participant's skin which can, over time, indicate a changed stress level (raised tension or relaxation). Second, a consumer-grade electroencephalography (EEG) headset is mounted on the participant's head. While such a device does not offer measurements as precise as those of medical EEG setups, it takes less time to equip and provides at least some mobility to the participant. This makes it feasible to apply even in a private learning setting. Our system uses the EEG sensors to measure error-related signals in the participant's neural activity. These signals indicate whether the participant was, consciously or unconsciously, aware of an erroneous answer (see section 2 for details).

For every quiz session, GSR and EEG data is recorded in a baseline phase. A combination of both measurements is then used in a learning phase to update probabilities for questions that have been asked in the past. The main goal of these measurements is to repeat questions that have been answered incorrectly even more frequently if the participant was not aware of an error. Questions that have been answered correctly will receive lower probabilities, while erroneous answers are scored higher depending on the participant's "mental workload", a combination of stress and error-potentials. Details of calculating these scores and how they are converted to question probabilities are available in the software section 5 of this paper.

2. RELATED WORK

For a proper understanding of the research context, related work corresponding to the current state-of-the-art will be presented in this section.

Different studies deal with either the use of error-related negativity (ERN) or the evaluation of mental workload to improve human-computer interaction (HCI) or to maintain mental health respectively. Study [1] uses the evaluation of a drivers' mental workload, using the *"Driving Activity Load Index"* (DALI), to improve the usability of mo-

bile phones and navigation systems whole lowering the driver's mental workload. Medical applications use continuous monitoring of mental workload – as it is described in [2] – to prevent mental disorders and maintain mental health. Study [2] evaluates the classification of discriminating different classes of workload in the daily life solely by using data from lab experiments.

Regarding Galvanic Skin Response (GSR), there are various studies dealing with different methods of detecting the degree of stress by using skin conductance [3]. In study [4] the focus has been set on the diagnosis of sudomotor dysfunction by detecting particular sweat levels. According to [4], this could improve the diagnosis of diabetes. Furthermore, the stress module described in study [5] is based on GSR. With that it is possible to distinguish between different degrees of effort or stress respectively. It could be used to detect situations with a high level of stress, which increase the risk of possible cardiac problems.

As mentioned before, there are various studies that deal with error-related negativity (ERN) itself [6, 7] and the degree to which it provides information about how people are biased to learn [8]. The ladder examines the relation between ERN and the bias to learn either more from mistakes or correct choices. The studies [9] and [10] consider the ERN from a more practical point-of-view. The usage of ERN in relation to HCI has priority in this study. In [9], off-the-shelf headsets like the Emotiv EPOC ™ are evaluated in HCI scenarios. Those scenarios are generally made up of multiple-choice reaction time tasks, in which the participant has to press the announced button under time pressure. The intended purpose is to equip HCI-modules with the ability to improve the interaction, so that - for instance - misunderstandings can be handled in a more automatic and efficient way.

3. SYSTEM DESIGN

A **quiz** program (see fig. 1) was developed that displays questions, four potential answers and a timer that is counting down for a configurable amount of time per question (5 seconds during the experiments). The user interacts with the quiz through the numerical keys one through four to allow four possible answers for each question. Pushing these keys requires only minimal motor movement in the fingers, which is important for clean EEG measurements. Especially the brain signals in the regions of the cortex that are interesting for error-related potentials are easily disturbed by muscular activity, leading to artifacts in the EEG signal[11, 12].

Initial plans for the question sets used Japanese characters and their English meaning as the question set for this program. The question set was quickly discarded as the number of wrong answers was high and participants have shown very little learning effect during early tests.

Figure 1: Screenshot of the quiz program

Bilingual English-Spanish wordlists, available for different knowledge levels through Cambridge University Press[13], have then been selected. The source data was transformed into pairs of a question, the correct answer and three random incorrect answers. These questions have been exported as XML files, which can be read by the quiz program. Making the question set selectable on a per-participant basis is necessary because the required EEG-based classifications require a high amount of correct answers to produce good classification results[14]. The question set for each participant had to be selected considering his/her prior knowledge to achieve more than 90% correct answers.

The timer that is visible in the quiz application is solely used for reporting and to give a visual incentive to the participant to provide an answer within a reasonable time limit. Exceeding the time limit does not have any negative impact, the whole system can be used in a self-paced way simply by not answering a question. Self-paced operation allows for breaks if the participant cannot concentrate on the task or sensors need to be adjusted during an experiment.

The buttons in the quiz application can take four different states:

- **Neutral** Initial state for each of the four available answers.
- **Locked** Whenever a participants enters an answer by pressing one of the keys "1" to "4", the button turns orange. Further input is disabled until the answer is revealed.
- **Correct** After a fixed interval of two seconds, the correct answer is revealed. This turns the background of the button to a bright green.
- **Wrong** If the locked answer was incorrect when the correct answer is revealed, it's background color is turned to red. This allows direct visual feedback for the participant.

Whenever an event occurs in one of the software components, network communication is used to exchange relevant information. For the quiz program, timestamps and results are emitted and received by a central controller that is responsible for data aggregation and calculates updated question scores.

During the first semester of the project, we developed a system to monitor levels of relaxation using a consumer-grade **EEG** headset[1]. An integral part of the second semester project was to modify this system in order to use it to measure and classify error-related potentials.

Error-related potentials have been reported to occur locked[14] to the time of a response or the correct answer to a previous question is revealed to the participant. The system stores timestamps for both events in-memory. As the revelation timestamp has not shown to be very effective in this setup, offsets from the participant's response are currently used when segmenting the EEG data for classification. As the offset after which the potential occurs depends highly on the participant and a few environmental factors, it could not be hardcoded into the system[14]. Data gathered in a baseline phase of 75 questions is used to perform an automatic cross-validation for the classification pipeline. During each cross-validation the classifier is trained with data of 200 samples at a different offset following the timestamp of each response, ranging from 50 to 450ms. The best-performing offset is then used to train the classifier, now using all available baseline data. Afterwards, the component is restarted in classification mode, which outputs classifications on incoming data from the EEG. Currently, Fisher's linear discriminant analysis is used for classification of unfiltered data on all connected EEG electrodes. Though the initial processing pipeline contained steps for performing a fast Fourier transform (FFT) and deleting unrelated channels, those steps did not seem to improve the overall classification performance. The pipeline was implemented using the *Brain-Computer Interface Framework UBiCI* [2] that was developed by the neuroinformatics group at Bielefeld University. More details are available in the software section 5 of this paper.

Data gathered through the **GSR** device is analyzed in a simpler way, where no long baseline phase is required. A Python component, *GSREval*, is used to gather and aggregate data from the device and provide access to data at a given point in time. This component, which was implemented as part of the project, is used to compare the GSR values at different time intervals preceding an answer to classify changes in the participant's stress level.

For regular *UBiCI*-based projects, the stimuli generation (quiz) would take place inside of a *UBiCI* processing pipeline itself. We decided on a more modular approach as we initially planned to use multiple instances of the framework to capture data for more than one modality. The final software architecture could be easily extended by other sensors or modalities, independent of the *UBiCI* framework. As the quiz application in this architecture is not tied to any specific framework as well, it could also be used for quick ad-hoc sessions without even using the external EEG and GSR data sources.

The process- and phase-controller components we introduced to achieve this are used to orchestrate the different classification pipelines and the quiz program itself. The controllers act as a router for all component events and question scores. After completing baseline and learning phases, a simple reporting phase is executed. The reporting script exports some information on the questions that were answered incorrectly as a takeaway for the participant (see figure 4). Details on all phases can be found in section 6.

4. HARDWARE

This sections provides details on the hardware that was used to capture EEG and GSR data.

4.1. EEG headset

The system uses an Emotiv EPOC™ EEG headset[15]. This consumer EEG device is mobile, in that it is battery-powered and connected wirelessly to a USB dongle. It features 14 measurement electrodes at international 10-20 system locations $AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8$ and $AF4$. Two additional electrodes act as online references for the measurement electrodes. Data on head movement is recorded by a two-dimensional gyroscope that is incorporated into the headset as well[15].

Equipping the headset requires preparation by applying $NaCl$ solution to each of the 16 wet electrodes. APIs are available to access raw data coming from the device. These APIs have been interfaced with the *UBiCI* framework, for details see section 5.

The participant is instructed to remain relatively still and keep body movements to a minimum in order to limit the occurrence of EEG motor artifacts[11] during an experiment.

4.2. GSR sensor

When planning the project a commercial GSR sensor[3] by *g.tec* was available that could be interfaced with the *UBiCI* framework by connecting it to electrodes of a second Emotiv EPOC™ device. The upside of this setup is that it requires no wired connection, while offering a very good resolution. We were unable to retrieve meaningful values when

[1] see section 4.3 for details

[2] see `http://www.ni.techfak.uni-bielefeld.de`

[3] g.tec *g.GSRsensor* see `http://www.gtec.at`

using both devices together. The reasons for this incompatibility are uncertain and may range from wrong electrode wiring to hardware defects in the antenna of the second device. These compatibility considerations led to a revision of the setup. The system now contains a self-designed GSR sensor with two measurement electrodes. The resolution of the system is lower (approx. 100Hz) and the connection is wired (USB). These specifications are sufficient for the project's use-case as only a slow measurement of relative conductivity changes is needed.

The device is based on an *Arduino Uno* board with a 16MHz ATmega328 processor. The circuitry consist of a 3.5mm stereo audio jack and some resistors. With a human friendly output voltage of 3.3V or 5V, one digital output pin is directly connected to the output electrode. Two analog pins are connected to the measurement electrodes and corresponding resistors to prevent short circuits.

After assembling the casing, two LED indicators were added (using two digital output pins of the board) to show the connection state of the electrodes without running a diagnostic tool. The complete device in operation can be seen in figure 2.

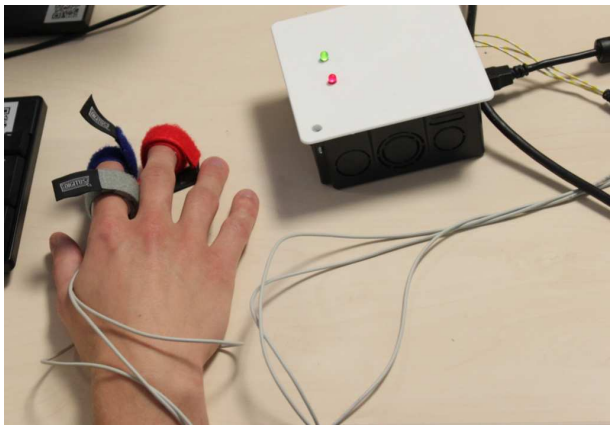


Figure 2: Hand with GSR sensor

The device is connected to the workstation via USB and provides measurements for both electrodes over a serial protocol at 9600 baud.

The electrode connectors are plugged into the audio jack and attached to the participant's free hand. As early tests have shown that the *GSREval* component is sensitive enough to react to heavy finger movements, participants were asked to keep the second hand in a steady position. To prevent conductivity changes when GSR electrodes move, 45mm gold-coated electrocardiogram (ECG) electrode pads with a conductive gel are attached to the fingers and the electrode connectors. The setup of the GSR hardware only takes a few seconds to attach the electrode pads to the fingers before data can be read from the device.

4.3. Hardware: Setup

A workstation (see fig. 3) is set up in the intelligent systems laboratory at Bielefeld University where the necessary sensors can be equipped in a comfortable, yet controlled, environment. The GSR sensor is wired to the workstation PC via USB, while the EEG headset provides a USB dongle that receives data via a wireless protocol.

As explained in the previous section, no special preparation is needed for GSR measurements. 45mm electrode pads are attached to the index and middle finger of the participant and connected to the GSR device using electrode cables.

The EEG headset requires a more complex setup. The electrodes have to be prepared by adding sponges soaked in a $NaCl$ solution for conductivity. The headset has to be positioned properly on the participant's head. Afterwards photos of the participant's head are taken in frontal and sideways perspectives to document the headset placement and reproduce it, if necessary. Attempts to re-use training data from previous sessions by replicating the exact headset placement using these images did not produce satisfying results.

A connection check using the official Emotiv software development kit (SDK) has proven to be necessary after attaching the EEG hardware. The connection of the USB dongle to the workstation, as well as the connection quality of each sensor has to be checked before each experiment. Though our first semester project already contains a visualization of raw sensor data that we modified for the current project, the official SDK provides a more convenient interface for this step.

5. SOFTWARE COMPONENTS

The system contains a number of software components that are essential for the multimodal processing of measured data. This section outlines the overall architecture, as well as interesting details for processing the individual modalities in context of their usage for this project.

The overall system operation is controlled by the **PhaseController**. A software component that sequentially executes multiple executables or shell scripts. For debugging purposes the output of individual processes can either be redirected to files or shown in separate terminal windows.

The subcomponent **ProcessManager** spawns processes and keeps track of their state and output. These components, together with a *runtime configuration* (specifying question set and various other settings), control a single experiment. Each experiment session consists of five successive phases:

1. **Initialization** Working directories are cleaned to provide a starting point for the new session. Participants are prepared before or during this phase.

2. **Baseline** A quiz is shown to the user for a configurable number of questions. While questions are answered, the EEG subsystem gathers data for training a classifier and the *GSREval* component starts to classify stress level changes.

3. **Baseline Post-processing** Scripts automatically perform a cross-validation on the data that was gathered in the previous phase. The best-performing parameter configuration (scored by $\frac{\text{TruePositives}+\text{TrueNegatives}}{\text{FalsePositives}+\text{FalseNegatives}}$) is then used to train the EEG classifier. No user interaction is required during this phase.

4. **Learning** Similar to the baseline phase, the participant interacts with the quiz application. Questions are randomly drawn from a question set. The EEG and GSR subsystems gather data and provide classifications that update the probabilities for a question that is answered correctly or incorrectly.

5. **Reporting** Relevant data is exported as a hypertext page (HTML) to provide feedback for further offline-learning to the participant. The participant can use these results as an indicator for which questions to focus on if he/she decides to review the incorrect answers after a session.

All components can exchange text-based messages through network communication. While this would, in theory, allow a distributed setup, all components are executed on the same workstation during this project.

The GSR processing and classification is done through the Python component **GSREval**. The script is caching each GSR sensor value as it comes in. After a second all new values in this cache are averaged and stored in a larger cache c. This averaging is possible as the overall conductivity only changes very slowly. Whenever a question is answered, the average of the last five seconds is compared to the average of the 50 seconds before that time interval. The average of this long reference interval is multiplied by a static factor $k = 1.2$ (determined manually by evaluating the data) to introduce a threshold. If the average of the recent short interval is higher than the weighted average of the reference interval, the participant is considered more stressed than before:

$$\frac{1}{5}\sum_{t}^{t-5} c(t) > \frac{1}{50}\sum_{t-5}^{t-55} c(t)k$$

$$\rightarrow C_{GSR} = \begin{cases} \text{if } true : 1 \text{ (stressed)} \\ \text{else } : 0 \text{ (not stressed)} \end{cases}$$

The component outputs raw values to the console to allow the participant or some instructor to make sure the serial-via-USB connection was established successfully until the cache is completely filled once. The project also contains a helper tool to visually monitor data gathered from the GSR device for other debugging purposes.

The **EEG** processing components are executed using the, previously mentioned, *UBiCI* framework for brain-computer interfaces. For both semester projects we produced reusable components to interface the Emotiv EPOC™ hardware to the framework. During the first semester *libubici_emokit* was implemented based on libemokit[4], an open-source library in the public domain. This component was extended during the second semester to allow configuration for using multiple EEG devices at the same time. With two Emotiv devices available, one was planned to be connected to a commercial GSR sensor.

The second version of the library *libubici_emotiv* is based on SmartPhoneBrainScanner2[16]. The project aims for compatibility with multiple consumer-grade devices and portability for other platforms (like Android smartphones)[16]. When interfacing an Emotiv device, parts of the official SDK are used to decrypt the datastream coming from the USB dongle. This allows *UBiCI* setups that, limited to bigger hardware changes, work with any current revision of the EEG hardware that was initially used.

Python scripts are used in a modified processing pipeline based on the first semester project in order to process incoming EEG data and associate it to user-generated stimuli of the quiz application.

The PhaseController will start different *UBiCI* pipelines, depending on the phase to enable training or classification of incoming EEG data. In classification mode, the *UBiCI* processing pipeline invokes a script that sends the classification result to the controller:

$$C_{EEG} = \begin{cases} 1 \text{ (User aware of an error)} \\ 0 \text{ (User not aware of an error)} \end{cases}$$

When starting up, the quiz application initializes question scores uniformly at 1.0. The controller gathers data on answered questions, as well as the classification results C_{EEG} and C_{GSR}. Whenever a response can be associated with EEG and GSR data based on their registered timestamps, the question score $\omega_i \; in \, [0.5, 2]$ is updated. These scores are then converted to the range of $\delta_i \; in \, [0, 1]$ and used as probabilities when a new question is sampled from the question set. For correct answers, the score is reduced by a static factor of 0.2 to make it appear less often. Updating question scores for incorrect answers is calculated by:

$$\omega_{new} = max(min(\omega_{old} - C_{GSR} * 0.3 \\ -C_{EEG} * 0.5 + 1.0, 2), 0.5)$$

[4] see `https://github.com/qdot/emokit`

This way, the question score will increase by 1.0 after an incorrect answer if the user was not stressed when answering and did not register his/her own error. If the user was stressed when answering and did not register his/her error, the score will only increase by 0.7. If the participant was stressed and registered his/her error, the score will only be adjusted by 0.2, to account for both modalities.

When randomly sampling a new question from the full set, the quiz application considers the individual question probabilities δ_i that are calculated by converting the question scores. Higher valued questions that have been answered incorrectly will be drawn more likely than those that have been answered correctly in the past.

6. INTERACTION

To start a learning session participants are first placed at the workstation and connected to both sensors. The GSR sensors are attached to the fingers of the right hand. After soaking the electrodes of the EEG headset in $NaCl$ solution, it is placed on the participant's head (figure 3). The exact placement of the headset is photographed for later sessions.

Participants are instructed to give the correct answer within the time limit shown on screen and limit body movement to a minimum. After placing the left hand comfortably on the keyboard keys "1" to "4", the participant can start the baseline phase by him-/herself.

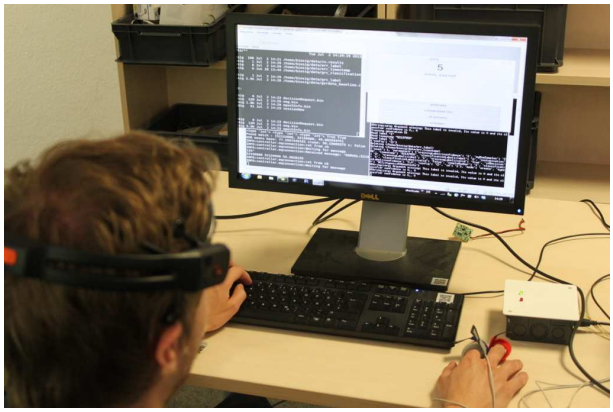


Figure 3: Participant equipped with GSR and EEG sensors

The baseline phase shows 75 questions to the participant, though this value depends on configuration. The participant can decide to take breaks at any time by not answering a question, though the EEG hardware should not be taken off to keep the electrode placement unchanged for the whole session. The GSR sensors are only affected by the last minute of data so they can be taken off or detached from the electrode connectors if absolutely necessary.

After post-processing the baseline data - which means about a 4-10 minutes break, depending on the number of responses that have to be evaluated - the quiz automatically restarts in classification mode. The participant can continue to answer a configurable amount of questions while the scores are updated as the data comes in.

When all questions are answered the quiz stops and the reporting phase generates print-friendly HTML output of all incorrect responses and the corresponding correct answer (see figure 4).

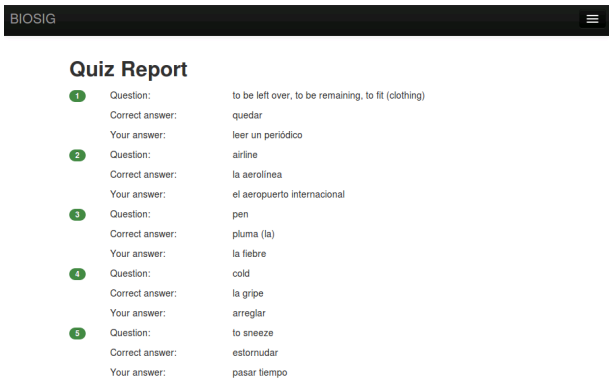


Figure 4: Reporting output

7. EVALUATION

We performed multiple sessions with two participants, during which we gathered GSR and EEG data for baseline and learning phases.

The cross-validation of EEG data gathered for 75 question baseline phases is performed on balanced training-/testbins. In practice this means that, when classifying EEG data occuring after correct responses and data after incorrect responses, some correct answers will be discarded. Results show a maximum of 65-75% accuracy for these EEG classifications, depending on session and participant (see figure 6). Classifying the complete data set, including data in the learning phase showed similar results. These results show that, though the approach is merely an adoption of techniques of the first semester project, it is feasible for a simple error potential classification.

To evaluate the performance of the *GSREval* component, the GSR values at the time of each response were exported and plotted (see fig. 5) after conducting a session. The participants reported that, subjectively, the resulting curve matched the level of stress felt during the experiment. While fig. 5 plotted the GSR value against the times of individual question answers, the curve remained clear and steady even when the full data set was plotted. This indicates that the hardware build is indeed effective enough

for this classification task. Data for participant two (fig. 5) shows one caveat that could potentially affect the quality of GSR classifications. The data shows a clear downward trend, possibly due to initial stress or high outside temperatures.

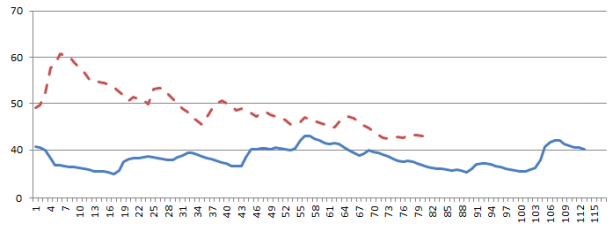


Figure 5: GSR data for two sessions (Participant 1: Solid blue line, participant 2: dashed red line)

Participants did not specifically notice the impact of question scores, though most experiments contained correct answers when an incorrect question was repeated.

Equipping the EEG hardware is still cumbersome and prone to errors. Users of an intelligent room that resembles a regular living environment could hardly be bothered to go through all the necessary preparations for using a wet-electrode EEG headset. Additionally, the classification does not produce consistent results if the EEG electrodes move during longer breaks or due to body movement. The whole baseline and learning process itself takes quite a long time, during which the $NaCl$ solution in the sponges between EEG electrodes and the head tend to dry out, making data gathered in the end of those sessions very unreliable. Even with photos of the exact position on each participant's skull, we had to abort two experiments because the classification did not produce satisfying results after re-equipping the headset.

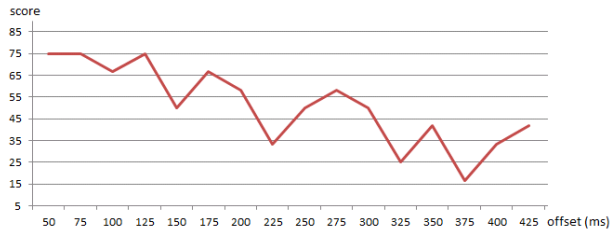


Figure 6: Cross-validation results for session 2 (participant 2)

Though the number of incorrect answers was at stable 5-10%, some of the sessions were only able to identify parameter configurations for 65% accurate classifications. This might be subject of improvements for follow-up projects by introducing more complex classifiers. For the current project, these results are promising enough, especially as they are only one part of a larger system. Other approaches than using multiple cross-validation runs to determine the offsets for meaningful error potentials should also be considered in future projects.

8. DISCUSSION

Improvements can definitely be made regarding the EEG setup and classification. We aimed to improve the classification quality by introducing more complex pre-processing techniques (e.g. fast Fourier transform and deleting channels in normally unrelated brain areas). None of those changes did significantly change the overall performance. Other EEG headsets or hardware setups that introduce more EEG electrodes in the relevant areas of the participant's cortex might also improve the data that is used for classification.

The project setup clearly involves more preparation time than controlling a regular quiz application, as sensors have to be prepared and equipped to the participant. The results, shown in the previous section, indicate that parts of the system can still improve the learning experience of a participant and the obtrusive hardware might be subject of improvement as technology continues to develop.

As discussed in the previous section, the *GSREval* component showed promising and stable results.

Conceptually, a baseline phase for stress levels could be added to handle situations where a participant remains on a high stress level for a longer period of time. Another improvement could be automatic generation of baseline question sets depending on wrong answers. By providing question sets with multiple difficulty levels, the application itself could select easier questions after a question was answered incorrectly. This would avoid the manual selection of a question set prior to a session.

Concerning the technical setup, future projects should setup a dedicated machine for data processing that remains connected even when not conducting an experiment. USB device changes and connection loss turned out to be an unhandy obstacle when setting up for a new session.

9. CONCLUSION

As part of the project, multiple ways to interface the brain-computer interface framework *UBiCI* with the EEG hardware have been developed. Those components can be reused in future projects.

The application of GSR and EEG error-potential modalities to a learning scenario shows satisfying results that can improve learning experience for a participant given a large set of multiple-choice questions. The impact of these improvements would have to be verified in sessions

with more participants. Especially the GSR classification showed promising results while still being relatively unobstrusive to the participant. The system setup in general is accessible enough to be used without requiring a special instructor, though some preparation for each session and limited technical knowledge is required to ensure proper operation. Follow-up projects can use the system as a basis to perform more complex EEG and GSR classifications or apply the learning scenario to other areas than vocabulary quizzes.

10. ACKNOWLEDGEMENT

We would like to thank our supervisors at the neuroinformatics group for the helpful advice and technical help with the hardware during the project.

11. REFERENCES

[1] A. Pauzie, "A method to assess the driver mental workload: The driving activity load index (dali)," *Intelligent Transport Systems, IET*, vol. 2, no. 4, pp. 315–322, 2008.

[2] B. Cinaz, R. La Marca, B. Arnrich, and G. Tröster, "Towards continuous monitoring of mental workload," in *5th International Workshop on Ubiquitous Health and Wellness (UbiHealth 2010)*, 2010.

[3] J. Zhai, A. Barreto, C. Chin, and C. Li, "Realization of stress detection using psychophysiological signals for improvement of human-computer interactions," in *SoutheastCon, 2005. Proceedings. IEEE*, 2005, pp. 415–420.

[4] K. Khalfallah, H. Ayoub, J. H. Calvet, X. Neveu, P. Brunswick, S. Griveau, V. Lair, M. Cassir, and F. Bedioui, "Noninvasive galvanic skin sensor for early diagnosis of sudomotor dysfunction: Application to diabetes," *Sensors Journal, IEEE*, vol. 12, no. 3, pp. 456–463, 2012.

[5] M. V. Villarejo, B. G. Zapirain, and A. M. Zorrilla, "A stress sensor based on galvanic skin response (gsr) controlled by zigbee," *Sensors*, vol. 12, no. 5, pp. 6075–6101, 2012. [Online]. Available: http://www.mdpi.com/1424-8220/12/5/6075

[6] R. Chavarriaga, P. W. Ferrez, and J. d. R. Millán, "To err is human: Learning from error potentials in brain-computer interfaces," in *1st International Conference on Cognitive Neurodynamics (ICCN 2007)*, 0 2007, iDIAP-RR 07-37.

[7] P. L. Davies, S. J. Segalowitz, J. Dywan, and P. E. Pailing, "Error-negativity and positivity as they relate to other {ERP} indices of attentional control and stimulus processing," *Biological Psychology*, vol. 56, no. 3, pp. 191 – 206, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0301051101000801

[8] M. J. Frank, B. S. Woroch, and T. Curran, "Error-related negativity predicts reinforcement learning and conflict biases." *Neuron*, vol. 47, pp. 495–501, 2005.

[9] C. Vi and S. Subramanian, "Detecting error-related negativity for interaction design," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 493–502. [Online]. Available: http://doi.acm.org/10.1145/2207676.2207744

[10] C. T. Vi and S. Subramanian, "Online single trial ern detection as an interaction aid in hci applications," in *CHI 2011 Workshop on Brain and Body Interfaces: Designing for Meaningful Interaction*, May 2011.

[11] N. Ille, P. Berg, and M. Scherg, "Artifact correction of the ongoing eeg using spatial filters based on artifact and brain signal topographies," *Journal of clinical neurophysiology*, vol. 19, no. 2, pp. 113–124, 2002.

[12] J. R. Wessel, C. Danielmeier, and M. Ullsperger, "Error awareness revisited: accumulation of multimodal evidence from central and autonomic nervous systems," *Journal of cognitive neuroscience*, vol. 23, no. 10, pp. 3021–3036, 2011.

[13] "Bilingual Wordlists," Cambridge University Press http://www.cambridge.org/gb/elt/catalogue/subject/project/custom/item6892877/English-Unlimited-Bilingual-Wordlists/, 2009, [Online; accessed 13-May-2013].

[14] G. Hajcak, N. McDonald, and R. F. Simons, "To err is autonomic: Error-related brain potentials, ans activity, and post-error compensatory behavior," *Psychophysiology*, vol. 40, no. 6, pp. 895–903, 2003.

[15] "EEG Features," http://www.emotiv.com/eeg/, [Online; accessed 11-May-2013].

[16] A. Stopczynski, J. Larsen, C. Stahlhut, M. Petersen, and L. Hansen, "A smartphone interface for a wireless eeg headset with real-time 3d reconstruction," *Affective Computing and Intelligent Interaction*, pp. 317–318, 2011.

INTELLIGENT SYSTEMS PROJECT: ARTIFICIAL NEURAL NETWORKS ON THE LOW POWER COREVA PROCESSOR

Christian Ascheberg, Markus Lux, Sebastian Meyer zu Borgsen

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: Marten Vohrmann, Thorsten Jungeblut

ABSTRACT

It is well known that embedded systems play a more and more important role in mobile data processing. We present an energy-efficient implementation of feed-forward artificial neural networks using specialized embedded hardware, namely the *CoreVA* processor which was developed aiming to provide a low-power consumption VLIW architecture. Therefore the system could potentially be used e.g. in autonomous systems with limited resources to perform various machine learning tasks. As an application we demonstrate the systems ability to use a neural network to recognize handwritten digits. We compared energy consumption to a commercially available processor and found that our system consumes less than a hundredth of the energy.

1. INTRODUCTION

These days embedded systems can be found everywhere in daily life. Small devices in cars, smart phones or other portable devices mostly are characterized by their autonomy, space, cost, and resource efficiency. Often the design of such systems involves finding a trade-off between those factors.

There is a rise of mobile intelligent systems like for example robots used in industrial and domestic environments or virtual agents on mobile devices which act as personal assistants. This results in an ever-growing need for more powerful embedded systems. These systems need to process all kinds of data mostly using sophisticated algorithms such as in image processing. Therefore the requirements for fast computational speed increase while energy-efficiency in a world of today plays a crucial role.

A common example of the aforementioned algorithms are artificial neural networks (ANN). They try to resemble inter-connectivity of biological, e.g. human neurons meaning that neurons which have the same activity at a given time will wire more strongly together than neurons which are not active at the same time. In this work we show the use of feed-forward neural networks with one input, one output and one or more hidden layers consisting of artificial neurons and wires between them. We chose neural networks because they are capable of solving a broad variety of problems[1] and are not bound to a specific application scenario.

We implemented a framework for neural networks in an energy-efficient manner whilst having a good computational speed even for larger networks. To achieve this we utilized the Configurable Resource Efficient VLIW Architecture (CoreVA)[12] which was developed at the Cognitronics and Sensorics group at Bielefeld University[11]. This processor uses a special architecture to maintain high energy-efficiency while offering a high clock speed.

Artificial neural networks are able to solve many machine learning tasks we demonstrate a network for a specific application scenario, namely handwritten digit recognition using the described processor.

In the next section we will look at related work followed by a description of our system in section 3 where we introduce the CoreVA as an integral part of our setup and review hardware, software and communication components. In section 4 the handwritten digits example is presented. In section 5 numbers on energy-efficiency are pointed and we conclude the paper with a discussion of problems and possible improvements.

2. RELATED WORK

Other research work using this approach on mobile devices and with power saving hardware was already done. Roppel et al.[4] presented an implementation of a neural network on an embedded system for chemical sensor data processing. As this sensor was targeted for portable use, the data processing had to happen resource efficiently. They suggested breath alcohol detection as a specific use case and reached correct classification rates above 0.9 in their evaluation but did not state how much energy was saved with their optimizations. Bashyal et al.[5] used a neural network on an embedded system for fire classification with gas sensors. An AT89C55 Microcontroller was used to process the

input of multiple gas sensors. This resulted in a low priced, power saving ANN implementation with the downside of being restricted to a low dimensional input space. With this little computation power only 7 input dimensions were handled. Our implementation aims for more flexibility and performance while still being energy-efficient.

3. SYSTEM DESIGN

This project was developed to be used as part of an embedded system. The most important constraints of embedded systems are size and power consumption. Therefore we used the CoreVA ULP, an ultra low power processor which was built to fulfill these requirements. For communication purposes we decided to use an ethernet connection because the standard is supported by a variety of devices and offers a high bandwidth. It is therefore well suitable for connecting distributed components of larger systems.

We will introduce the CoreVA processor with its design and features in the next section while a more general overview over the whole setup is given afterwards. This includes the software components running on the CoreVA, namely the neural network and ethernet stack, as well as the training and the graphical user interface running on an external computer. A general overview of the setup is given in Fig. 1.

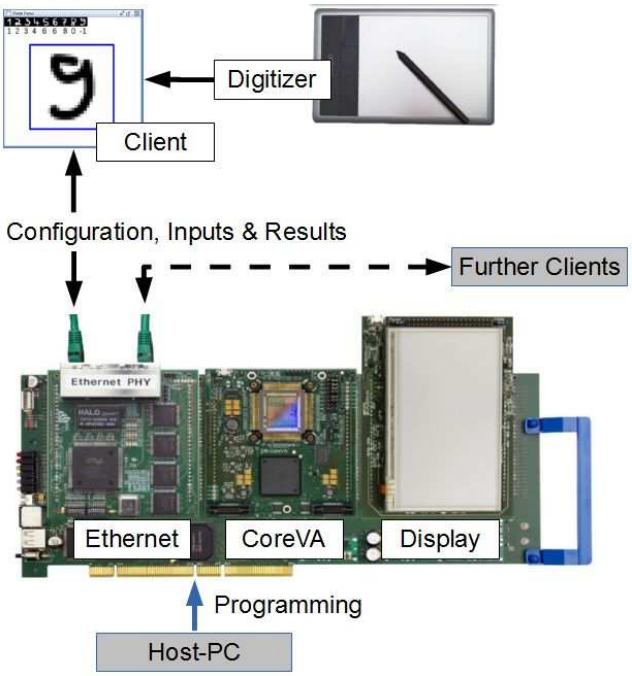


Figure 1: Overview of the setup including CoreVA- , Ethernet- and TFT-daughterboards mounted on a RAPTOR2000 baseboard

3.1. The CoreVA-Processor

The CoreVA-Processor is a Very-Long-Instruction-Word (VLIW)-processor designed by the Cognitronics and Sensor Systems group. This processor follows the Harvard architecture and uses Reduced Instruction Set Computing (RISC). Its modular architecture allows configuration of various parameters at design time. For example the number of VLIW-Slots, function units and arithmetic logic units can be adjusted for the desired use-case. The CoreVA can also compute single instructions on multiple data (SIMD).

It was especially designed to fulfill the requirements of minimal power consumption and small size for mobile use. With $32\,\mathrm{kB}$ on-chip-cache the CoreVA needs only $2.7\,\mathrm{mm}^2$ space. An important feature of the ULP is that it can dynamically adjust clock frequency, supply voltage and thus limit power consumption to a minimum during idle time. At best the clock speed ranges from $10\,\mathrm{kHz}$ to $94\,\mathrm{MHz}$ needing only $9.94\,\mathrm{pJ}$ at low load.

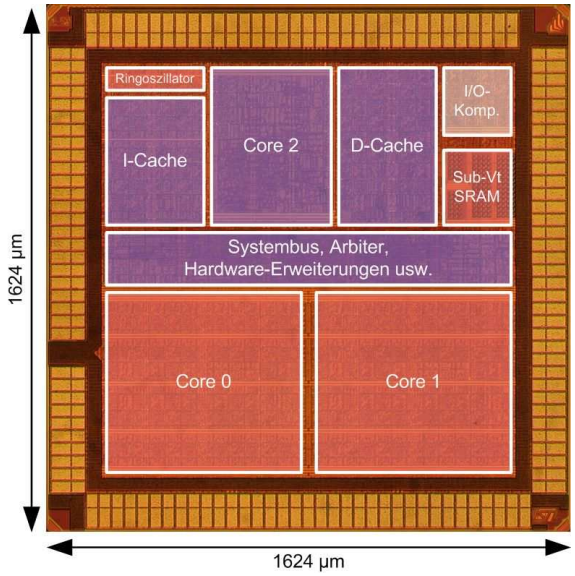


Figure 2: Layout of the CoreVA ULP processor. The chip area is $2.7\,\mathrm{mm}^2$

3.2. Hardware Setup

The main part of the hardware setup is the CoreVA processor. As stated above, in an actual case of application it would be part of a dedicated embedded system. For development purposes though, the processor is mounted on the

DB-CoreVA eval board, to be used within the the rapid prototyping platform RAPTOR [9] as seen in Fig. 1. The board furthermore hosts an ethernet module (EthMAC) and a display module. It is also equipped with a Field Programmable Gate Array (FPGA) chip. The FPGA acts as programmable controller for memory and for external hardware extensions like the EthMAC, see Fig. 3. This enables the CoreVA to communicate with the modules via memory-mapped-IO. It also allows quick prototyping of connections between components on the board.

For power supply the board itself needs to be connected to a host computer via PCI bus. The ethernet module is the only interconnection used for sending input to the CoreVA and receiving the results. The communication protocol developed for this purpose is described in subsection 3.4. An off the shelf digitizer was connected to that computer and used as an input source for handwritten digits.

On the host of the RAPTOR board a program sets up the runtime behavior of the CoreVA onetime. This means it is responsible for configuring the on-board FPGA, initializing the processor, transmitting the compiled program and finally starting it.

3.3. Computation Software

The most important software-related aspect of this project consists of the neural network that is computed on the CoreVA. When it comes to the implementation of neural networks the *Fast Artificial Neural Network Library* (FANN) [13] is a popular choice. Since there exists no suitable port of well-known compilers for the CoreVA processor at this time, we used a custom compiler implementation [10] which only supports ANSI-C out of the box. The code of the FANN library had to be modified in order to match this constraint [8]. In addition to that we improved the port to handle larger network sizes by introducing a more dynamic memory management and therefore optimized memory usage, too.

The training of the network can be performed by using the FANN library on a regular computer. Running the training on the CoreVA processor is not in the interest of the application scenario as the training can be done beforehand. The FANN library can save the result in a fixed-point number format which is very helpful as the CoreVA has no floating point unit.

3.4. Communication Software

The second library that we developed for the CoreVA allows us to transfer data between host computer and CoreVA via the ethernet-based UDP network protocol. On top of this we implemented a slim protocol that allows us to send configurations of neural networks as well as network input data to the CoreVA. The payload being sent consists of a vector of integer values representing either the neural network structure in FANN format or the input vector. This is preceded by integers indicating the data type and the length of the data. This allows us to reconfigure the neural network on the CoreVA at runtime and thus to improve the results or modify its purpose. Once the CoreVA has received and initialized a neural network it is ready to run it on new input data sent via ethernet. As soon as the results are computed they are published via ethernet for further processing. Note that the sender of the input data does not necessarily need to be the receiver of the results. The setup can as well be configured to send the results to a different computer or component. For the ethernet connection we measured a net data rate of 5 Mbit/s. On the CoreVA side this includes correctly merging the received bytes to integer values that can then be used in further data processings.

The counterpart of the CoreVA software is a Java program running on a computer that is connected to the CoreVA via ethernet. The program is able to transfer any trained neural network to the CoreVA and also to send series of input data to it. It is also capable of receiving the classification results from the CoreVA and displaying them, see subsection 4.2.

4. USAGE EXAMPLE

We evaluated the neural network implementation on detection of handwritten digits. This use case is a large-scale real-world example for applied neural networks. The large input vectors allow accurate and stable performance measurements. In conjunction with the MNIST[14]-Database we have a big database of training and test data ready for our network. To evaluate the generalization and communication abilities of our framework in detail, we designed a graphical user interface (GUI) that allows direct input of handwritten digits into the network.

4.1. Data and Network Training

The MNIST data set consists in total of seventy-thousand annotated handwritten digits ranging from zero to nine represented as gray-scale images with size 28x28 pixels. Ten-thousand of these digits are distributed in an extra testing set. These were written by people who were not in the set of those who wrote digits in the training set. So test accuracy values should give a good information on how well the network was trained. As stated earlier the training was performed on another computer using the FANN library. The input and output layers have fixed dimensions of 784 (image dimension) and 10 (number of classes) respectively. We evaluated different hidden layer sizes all of which delivered good test results. A network with 300 hidden neurons was

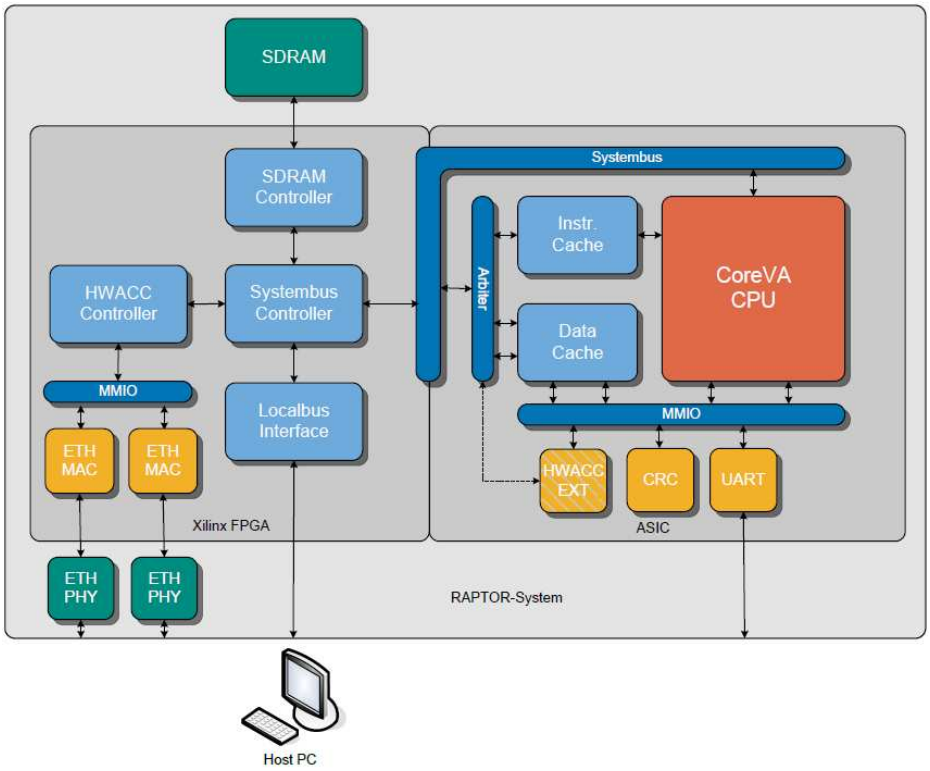


Figure 3: Setup within the RAPTOR system showing the links between the FPGA and the CoreVA

trained using the QuickProp [6] algorithm. By quadratic approximation of the networks error gradient this method usually converges faster than other known training algorithms. In the hidden layer a sigmoid activation function was chosen and a linear activation function in the output layer because this minimizes the cross entropy error [7] and the output values can therefore be interpreted as class probabilities. To aid the problem of converging into local minima we trained multiple networks of this kind and selected the one with the best performance on the test set. The best one achieved an accuracy of about 92 percent.

4.2. User Interface

We implemented a graphical user interface in Java to test the system live and to visualize the results. Figure 4 shows a screen shot of the GUI. This Interface consists mainly of a large free-hand drawing area. Digits can be drawn on the white area by dragging the mouse or for a more intuitive way of writing we implemented the use of a digitizer pen. On Top of the GUI is a visualization of of the last drawn digits and the classification results. After the user has stopped drawing for about a second, the GUI recognizes the last input as an image and renders the result. The trace of the users movements is rendered by connecting the movements with lines. Some image improvement like anti-aliasing and smoothing is done to generate a more natural handwriting. After rendering, rasterization takes place to read the matching number of pixels for the ANN. To ensure that only relevant pixels are read, a bounding box around the drawing is calculated. This input vector is transferred via network to the CoreVA for classification with the neural network. The CoreVA returns the classification result back over network and the GUI can visualize it in the history. In our tests classification of digits drawn in the GUI did not quite reach the scores of the MNIST data. Digits like one or zero are classified correctly almost every time. Here accuracy rates are above 90 percent. Other numbers with more similarities like eight and three, five and six do get mixed up sometimes. The eight, nine and seven seem to be even harder to classify correctly. This might be caused due to significant differences in writing on paper and writing with a digitizer or by the rendering. Variation of line widths might influence the results as well, since the net is trained with handwritten data from MNIST. It is also noteworthy that the ANN was trained on handwriting from Americans. As handwriting differs from region to region, the results may be worse when we evaluate the ANN with European handwriting.

Another training on data generated by the GUI will greatly improve the classification rates.

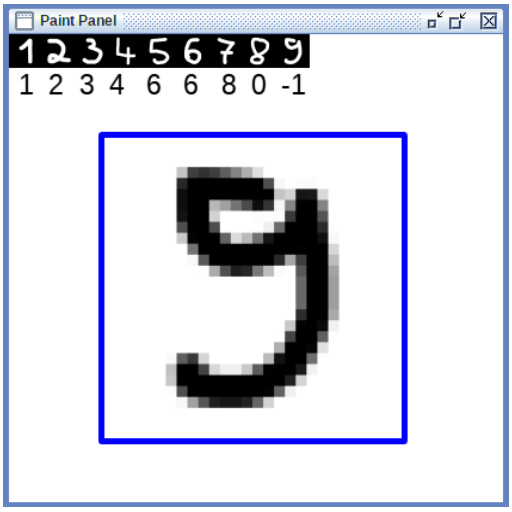


Figure 4: Graphical User Interface for on-the-fly digit classification with a neural network on the CoreVA using a digitizer [15].

5. EVALUATION

In terms of power consumption the CoreVA processor itself is already a very energy-efficient processor. We evaluated energy-efficiency of our artificial neural network implementation by calculating the energy which was consumed by the processor when classifying one single input digit. For this we used measurements from [3] which state how much energy the processor consumes at a given clock frequency and voltage level and combined this information as follows.

When processing a neural network input we clocked the processor to operate at full speed. Since we only used one CPU core it ran at a clock frequency of $80\,\mathrm{MHz}$ and because of adaptive voltage control at this frequency the voltage level was at $1144\,\mathrm{mV}$. The per-cycle-energy (the energy consumed within one CPU cycle) at this voltage level lies at about $100\,\mathrm{pJ}$. Using a clock-counter hardware extension we also measured the number of cycles needed to process one input digit. Using a network with one hidden layer with 500 neurons about $34 \cdot 10^6$ cycles were needed.

$$100\frac{\mathrm{pJ}}{\mathrm{cycle}} \cdot 34 \cdot 10^6\,\mathrm{cycles} = 3.4\,\mathrm{mJ} \qquad (1)$$

As outlined in Equation 1 the consumed energy for the classification of one digit is at about $3.4\,\mathrm{mJ}$. Note that this is the energy which is solely used by the processor and not by other components on the system like for example memory. The number of cycles did not include network transfer. By further dividing the number of cycles by the clock frequency we see that the computation time was around $0.43\,\mathrm{s}$ and therefore the power consumption is $3.4\,\mathrm{mWs}/0.43\,\mathrm{s} \approx 7.9\,\mathrm{mW}$. When idle the CoreVA runs at $0.1\,\mathrm{MHz}$ with a voltage level of $320\,\mathrm{mV}$. This results in a power consumption as follows:

$$15\frac{\mathrm{pJ}}{\mathrm{cycle}} \cdot 0.1 \cdot 10^6\,\frac{\mathrm{cycles}}{\mathrm{s}} = 1.5\,\mu\mathrm{W} \qquad (2)$$

For comparison we also tested the same code and the same neural network on a commercially available mobile processor[1], which has a thermal design power consumption (TDP) of $35\,\mathrm{W}$ and measured the computation time using the *time* command on a Linux system. In average the program finished in $0.02\,\mathrm{s}$ from which we again can calculate the consumed energy $35\,\mathrm{W} \cdot 0.02\,\mathrm{s} \approx 700\,\mathrm{mJ}$. Beware that the actual energy consumption of this CPU may be lower than the obtained value because these calculations are merely based on specifications and not on real world measurements. But even if a power consumption of e.g. $15\,\mathrm{W}$ is assumed, the consumed energy is still about a factor 100 higher than the energy consumed by the CoreVA.

These values should give a good picture how both systems – the CoreVA and a modern mobile processor which is designed to be efficient – compare. For a comparison, see also Table 1.

	energy consumption	power consumption
CoreVA	$3.4\,\mathrm{mJ}$	$7.9\,\mathrm{mW}$
Mobile CPU[1]	$700\,\mathrm{mJ}$	$35\,\mathrm{W}$

Table 1: Comparison of energy and power consumption of the CoreVA and a commercially available mobile processor when classifying one digit one a neural network with one hidden layer with 500 neurons.

6. DISCUSSION

As pointed out in the previous section the neural network implementation on the CoreVA consumes much less energy than the same implementation running on a commercially available mobile processor.

However there is still potential to optimize the setup even more:

- The CoreVA can be designed to have more computation units. Because of its VLIW architecture a sophisticated compiler would be able to massively parallelize the program in order to utilize all available cores and therefore speedup the computations without hurting energy consumption.
- Our program already dynamically adapts the clock frequency of the processor. However the method of setting the frequency to the highest available value is not

[1] Intel ®Core ™i3-2310M

always the best solution. It would be more desirable to implement a different approach where an algorithm would detect the amount of work in a specific past time frame and then adjust the frequency according to the workload. Another approach is to adjust the frequency so that computations take just less than the maximal allowed time for the desired use case.

Another topic which needs to be addressed in the comparison between the CoreVA and another regular processor is the different application scenario. The CoreVA is targeted to be deployed in embedded systems for solving very special purpose problems while a general purpose CPU is way more versatile by the means of possible application scenarios. So a head-to-head comparison may seem unfair but at least it gives an impression of what the benefits would be when using the CoreVA with our implementation. Unfortunately we did not find any other resources to compare our results with (see section 2).

7. CONCLUSION

On the CoreVA processor we developed a framework for feed-forward neural networks which is able to handle even high-dimensional inputs in an energy-efficient manner. The usage example of handwritten digit recognition demonstrates those abilities well. We also compared energy consumption for recognizing one digit with a modern mobile processor and observed that with values around $3.4\,\mathrm{mJ}$ our setup typically consumes only about a hundredth of the energy. However it has also been pointed out that there is still room for improvements both in terms of energy consumption and when looking at generalization in the specific case of our usage example.

8. ACKNOWLEDGMENT

Our project supervisors did a great job of introducing us to a quite non-familiar topic. Not only they but also other members of the Cognitronics and Sensorics group were always cooperative when it came to problems we could not solve on our own. Without their help we sometimes would have struggled to get things done. Thanks a lot!

9. REFERENCES

[1] Csji, Balzs Csand, "Approximation with artificial neural networks.", Faculty of Sciences, Etvs Lornd University, Hungary (2001)

[2] S. Lütkemeier, "Ressourceneffiziente Digitalschaltungen für den Subschwellbetrieb", Dissertation, Universität Paderborn, 2013

[3] Lütkemeier, Sven and Jungeblut, Thorsten and Kristian, Hans and Berge, Otnes and Aunet, Snorre and Porrmann, Mario and Rückert, Ulrich, "A 65 nm 32 b Subthreshold Processor With 9T Multi-Vt SRAM and Adaptive Supply Voltage Control", IEEE Journal of Solid-State Circuits (Volume: 48)

[4] Roppel, T., Wilson, D., Dunman, K., Becanovic, V., "Design of a low-power, portable sensor system using embedded neural networks and hardware preprocessing" , Neural Networks, 1999. IJCNN '99. International Joint Conference on (Volume: 1)

[5] Shishir Bashyal, Ganesh Kumar Venayagamoorthy, Bandana Paudel, "Embedded Neural Network for Fire Classification Using an Array of Gas Sensors", Sensors Applications Symposium, 2008. SAS 2008. IEEE

[6] Fahlman, Scott E, "An empirical study of learning speed in back-propagation networks", 1988

[7] Haschke, Robert, "Vorlesungsskript: Vertiefung Neuronale Netze", http://ni.www.techfak.uni-bielefeld.de/teaching/vertiefung-neuronale-netze

[8] Einhaus, Julian, "Entwurfsumgebung zur ressourceneffizienten Mustererkennung auf dem CoreVA-Prozessor", Masters thesis

[9] Porrmann, Mario and Hagemeyer, Jens and Pohl, Christopher and Romoth, Johannes and Strugholtz, Manuel, "RAPTOR – A Scalable Platform for Rapid Prototyping and FPGA-based Cluster Computing", Parallel Computing: From Multicores and GPU's to Petascale, Advances in Parallel Computing (Volume: 19)

[10] Jungeblut, Thorsten, "Entwurfsraumexploration ressourceneffizienter VLIW-Prozessoren", Dissertation, 2011, Uni Bielefeld

[11] http://www.ks.cit-ec.uni-bielefeld.de/

[12] http://www.ks.cit-ec.uni-bielefeld.de/de/projekte/coreva-vliw-prozessor.html

[13] http://leenissen.dk/fann/wp/

[14] http://yann.lecun.com/exdb/mnist/

[15] Digitizer image by DragonLord (Own work) [CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons, https://commons.wikimedia.org/wiki/File%3AWacom_Bamboo_Capture_tablet_and_pen.jpg

INTELLIGENT SYSTEMS PROJECT: USING EVOLUTIONARY ALGORITHMS TO CONTROL A SEMI-AUTONOMOUS WHEELCHAIR

Daniel Kühn, Nico Lüdike, Matthias Sterz

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisor: Benjamin Inden

ABSTRACT

As part of the seminar "Intelligenter Raum" at the Bielefeld University in summer term 2013, a simulation of a semi-autonomous wheelchair has been developed. This wheelchair is designed for people with any variety of the locked-in syndrome. It should move to any locations in the environment the user wants go to, independently of the starting position. While moving, the users must feel save, so the navigation system of the wheelchair detects any types of obstacles and avoids them to prevent the patients getting injured. The behaviour of the wheelchair is optimised by applying evolutionary algorithms in simulation. This paper discusses a first approach how to design and implement a semi-autonomous wheelchair navigation system.

1. INTRODUCTION

For handicapped people who are reliant on a wheelchair it can be exhausting to manoeuvre their vehicle, even if they are in familiar surroundings. Of course, the difficulty of navigation depends on their kind of disability. The wheelchair designed in this project is intended for people with any variety of the locked-in syndrome[1].

Developing such a semi-autonomous, intelligent wheelchair is not a new field of research in robotic science. Locked-in patients have not the ability neither to control an electric wheelchair nor a "classic" wheelchair. A caregiver is necessary to move around with the wheelchair. In 2011 there was implemented a system that reduce the caregiver's load[2]. The wheelchair navigate alongside the caregiver. But this is not an autonomous way of navigating for handicapped people.

In contrast to Seth Teller's and his group's work[3], the system created in this project does not have to create a map of the environment. There will be no information about walls and doors, only important locations should be stored by the system.

The way how to control electric wheelchair movements is a important issue and depends on the level of the user's physical or mental handicap. The control system of the wheelchair could provide multiple ways of controlling, beginning by using a joystick, speech or head movements to navigate the wheelchair directly, through to a supported control system which avoids collisions with obstacles up to a semi-autonomous navigation system[4][5]. Locked-in patients have not the possibility to use a joystick. So in this project there must be a semi-autonomous system for wheelchair navigation which can be controlled by any other kind of communication. A brain-computer interface is one way to communicate non-verbally, without facial expressions or without body movement. Approximately 40 years of research on brain-computer interfaces results in a good performance on many fields of application. Just thinking of motor execution of right or left hand results in a rotation of the wheelchair to the right or the left[6]. An other study shows, that it is not necessary to think of motor movements. There is a possibility to control the navigation system of the wheelchair by solving an arithmetic calculation, composing a simple letter, counting repeatedly from 1 to 9 in mind or imagine how to move around in a familiar environment[7].

The structure of this paper is as follows. In section 2 the design criteria and constraints which lead to the selected setup will be explained. Section 4 presents the utilised hardware of this project. The developed software modules are described in section 3 followed by section 5 where the working system is shown. An evaluation of this projects is given in section 6. In section 7 the results of the project will be discussed and a brief summary of the entire work is given in section 8.

2. SYSTEM DESIGN

At the beginning of this project, it has to be discussed what kinds of users are the target group for this project. As mentioned before, the intention is to build a wheelchair for patients with the locked-in syndrome. Based on this constraint the goal of this project is to improve a common electric wheelchair by building a semi-autonomous and intelligent navigation system which is easy to control via a comfortable way of communication. Locked-in patients are aware

and awake but they cannot move or communicate verbally due to complete paralysis of nearly all voluntary muscle action in the body except for the eyes. For example, this can be caused by a stroke at the level of the basilar artery denying blood flow to the pons, a traumatic brain injury or Multiple Sclerosis[8].

In this project the communication of the goal point to the system is only theoretical. However, the only communication system which could be used by patients with any variety of locked-in syndrome would be a brain-computer interface. Integrating this feature to this project has been impossible because the manpower was limited. Instead, the goal point where the wheelchair should move to is fixed given at the start of the system.

Furthermore, the system must be safe to use so that the patients cannot be injured while navigating. It would be possible to give the user the power to stop the wheelchair in dangerous situations, but this would be contrary to the goal of building a semi-autonomous wheelchair. The solution presented in this work is to implement a safety box so that the wheelchair stops immediately if something or someone is too close. To scan the environment, only laser sensors around the wheelchair are used. Other technical equipment to measure the surroundings, like cameras, would only inflate the amount of data which must be evaluated, deteriorating the systems performance.

Another goal of this project is to design a resource saving architecture. Under this constraint it should be avoided to generate a map of the environment. For this project it is only necessary for the navigation system to know where the wheelchair is and where it should move to.

3. SOFTWARE COMPONENTS

In this project the Scalable Neuroevolution Project (SNEP) software is used, which is closed source. SNEP is an implementation of the NEATfields method[9], which is an extension of the NEAT neuroevolution method. This extension is used to solve problems with large input and output spaces. There are at least three fields, one input field, one output field, and one internal field, to form a complete NEATfields network. Each field is a 2D array of recurrent neural networks with almost arbitrary topology. So SNEP uses evolutionary algorithms to get better individuals from generation to generation. Each simulated wheelchair in this project is an individual. Via a simple configuration file it is possible to set up the SNEP software, such as population size, number of generations, number of executed evaluations per individual, etc..

Two simulators have been implemented, one 2D and the other one 3D, to calculate the behaviour of a semi-autonomous wheelchair in different environments. Both simulators are interchangeable. The environment is set in a particular mapfile, which is parsed at the beginning of the simulation. Each map contains a goal point, which the wheelchair should navigate to, and an area of starting points. For each individual a start point will be randomly chosen within the starting area. The navigation process finished either the wheelchair reaches the goal point or there is detected a collision or the wheelchair takes to much time and the navigating is aborted by the simulation. In any case of finishing the simulation of an individual, the simulator evaluates the final result. This evaluation process is described in section 6. Due to the interchangeability of the two simulators it is possible to first run the 2D-based one to evolve a well-rated individual, which can be used as a starting point for further physics-based evolution in the 3D simulator. Additionally, this project provides an interface to the Instant Reality[10] framework. This interface provides an opportunity to run the the 3D simulation in the virtual reality laboratory of the Artificial Intelligence group at Bielefeld University.

4. HARDWARE: CONSTRUCTION OR SETUP

In addition to an usual Open-GL based visualisation, an interface to a virtual reality framework is provided. There are used two different setups to make our simulators tangible. The first one consists of three large panel screens which are arranged to form a 90 degrees 'interaction corner'.

Additionally the framework can be connected to the 3-wall-CAVE-Environment[11] at the Artificial Intelligence group at Bielefeld University. In this highly immersive environment allows to augment the visualisation by means of stereovision, sound, wind and a vibrating floor. In this project are not used all available modalities, but nevertheless, the CAVE provides a far better facility to rate the qualities of an evolved wheelchair controller than simple 2D-visualisation does.

Furthermore the interface provides a backchannel so that the simulator not only can be rendered by the VR-Framework, but it is also possible to interact with the system from within the virtual world. In particular a Nintendo Wii Remote can be used to rate the subjective impressions of the quality of a presented individual. Thereby additional non-computable factors can be taken into account when determining the individuals fitness.

5. INTERACTION/OPERATION EXAMPLES

We evaluated many fitness functions, evaluating how good they reach the goal and how they behave until they reach it. Every fitness function was tested with the same constraints as seen in the video (Evolu-Test) .

The random starting positions and directions of the wheelchair where generated using a special random generation function. This function guarantees that every fitness function was tested in the same way. Also debugging gets much easier with the intern random generator because the random generation function also guarantee reproducibility.

The video shows how the wheelchair reacts with our efficient fitness function called ”easy” which calculate the square of the dijkstra distance from the last position of the wheelchair to the goal. It is an easy implementation but as experience had shown us it shows already a good behavior to find its goal where other more complex functions failed. Each individuum of the wheelchair starts on a random starting position and direction within a distance of 15 to 18 meters to a goal. The starting fields can be seen in the video as turquoise fields. The goal of the wheelchair is the only green field seen in the right middle room in. The goals position is fixed. The moving, blue rectangle represents the wheelchair. In the video we show the sensor rays in critical situations. They are always activated so the wheelchair learns to avoid obstacles.

The whole video is subdivided in three tests. The first and second test use the same learning procedure where the wheelchair learns how to drive to the goal without safety box activated. There is only one difference between these two parts. The first test shows the final test with safety box deactivated whereas in the second test they are activated. The third test shows the wheelchairs behavior with the safety box activated in the learning procedure.

The safety box allows the wheelchair to drive freely so the speed of the wheelchair is defined by the output of a neuronal network. The safety box always checks the drive direction of the wheelchair. Then it evaluates the sensor data of the sensors in this direction. If an obstacle lies in its driving-direction the wheelchair rotates until it shows its back to this object. This behavior guarantees a new wheelchair behavior which does not drive against the previous object. This is to prevent the wheelchair from stopping after entering a warning area most of the time. Warning areas are places where the wheelchair is near a critical area where it can collide with walls for example.

As seen in the video, in the first two tests the wheelchair drives most of the time forwards but also a little bit to the left. This is the behavior it learned after 200 generation of learning with an fitness function based on the dijkstra distance [12]. With this behavior it can find the goal from most of his starting positions and directions in the predefined scenario.

In the third test the wheelchair also finds the goal but drives generally backwards. This we find as an interesting behavior change but unfortunately we cannot explain its origin.

6. EVALUATION

To evaluate the behavior of the wheelchair we use SNEP for evolutionary algorithms as described in section 3. After each generation every individuum has a chance - depending on its fitness score - to be added to the next generation. The best fitness score is one, whereas the worst is zero. Every individuum has up to 1200 steps to reach the goal which is sufficient to reach the goal.

Various fitness function have been tested which evaluated the dijkstra distance, rotation, direction, roadmap, speed of the wheelchair in every time step or at the end of each individuum. Every fitness function progressed a learning procedure up to 200 generations long which represents 20000 individuums.

As seen in test one of the video in section 5 our efficient fitness function ”easy” has its pros and cons, like every learning algorithm. In every three tests of the video not all critical situations are covered. For example if the wheelchair starts in a bad position in the upper room it first drives to the corner of this room but then it drives backwards, find the door and finally drives to the goal. Here we have a good example of great learning behavior. It is not perfect but good enough. On the other hand when it starts in the lower room right upper corner it obviously had not learned to react in the same matter.

Another problem of this fitness function - without safety box activated - is, it does not guarantee any collision. The video shows that the wheelchair does avoid the wall most of the time, but not every time. Collision occurs when the wheelchair crashes with an edge of a wall near the doors. As seen in the video the wheelchair avoids the wall if more than one input sensor sense an object before the wheelchair, but if only one sensor sense an object right before the wheelchair, it collides with it.

The next step is to discuss how the same fitness function react with safety box activated, as seen in test two of the video in section 5. Thanks to the safety box system the wheelchair does not crash with objects anymore thus driving further so it may reach the goal more often which it does as we see in the video.

The third test of the video shows how the wheelchair behaves when learning with the same fitness function but also with the safety box activated all the time. As

seen in the video the wheelchair also finds the goal but drives generally backwards. This we find as an interesting behavior change but unfortunately we cannot explain its origin.

In another test not shown in the video, the wheelchair starts in one of three rooms near the room with the goal position as seen in figure 1. Here we wanted to reduce the critical start positions in the upper room. If the wheelchair started on the left side of this room it successfully drives to the goal. But unfortunately, if it starts on the right side of the room it does not reach the goal like our tests seen in the video in section 5.

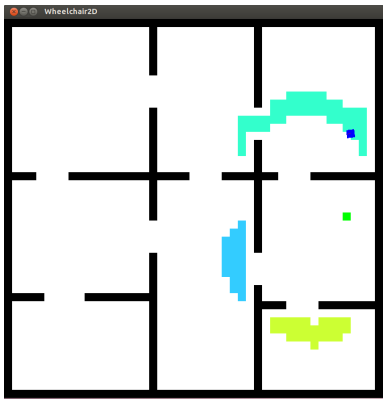


Figure 1: Shows the testing area. The black fields represents walls, the green field in the center-right room the goal, the blue one the wheelchair. The other three colored fields represents three seperate starting areas per individuum.

Another test of ours evaluated the difference of simple and a complex fitness function. For example we created a simple fitness function, which evaluates only the last position of the wheelchair. There we calculate the square of the dijkstra distance of its last position divided by the maximum dijkstra distance of every field the wheelchair can ever drive to. The only exception is when the wheelchair crashes with an object, then the fitness score is set to 0,01. This fitness function we named ”easy” as it shows how good easy fitness functions get to the goal. It’s the same one used in the video seen in section 5.

For an complex fitness function we tested ”current”, a fitness function which evaluates the dijkstra distance as mention earlier, but also the direction, rotation on the spot of the wheelchair. With this fitness function the wheelchair is guaranteed to drive most time forward, does not rotate at a spot.
Figure 2 shows the fitness score progressing up to 200 generation with learning without safety box activated. Here the simple fitness function ”easy” reached its best score after 27 generations where the complex one ”current” only reached a score of 0,82 which is a bad result. In general only fitness scores above 0,95 cover most of the cases which can occur in the predefined scenario.
As seen in section 5 the simple fitness function surprisingly learns to drive forward all the time except a wall is ahead of it then it drives backwards. Also it finds most of the times the door and drives easily to the goal. The only bad behavior seen is its unusual driving technique as described in section 5.
On the other hand the complex one learns only to drive forward without rotating much, even after reducing the weight of the rotation on the spot score.

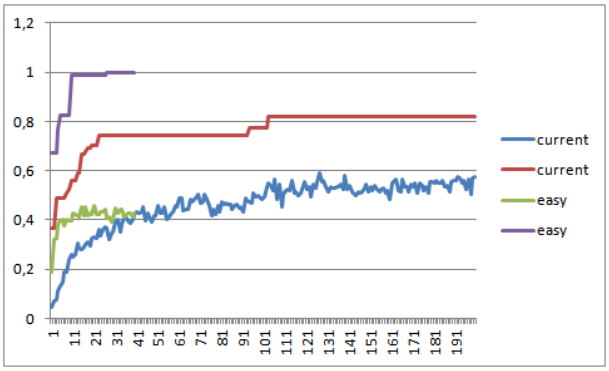


Figure 2: Best and average fitness score of two fitness function in the process. Easy is a simple fitness function whereas current a example of a complex fitness function represents.

After experience of testing many complex fitness functions against the easy one, every complex one got worse wheelchair behavior than the simple ones. For example they learned only to drive forward without getting near the goal.

Next we tested the difference of the mentioned simple fitness function ”easy” learning with and without the safety box-system activated.
As seen in figure 3 the fitness function with safety box activated had always a slightly better average score than the same fitness function without the safety box activated. Without the safety box activated the fitness function reach much faster a top score. So ”easy” without safety box reached a score of one after 27 generation where ”easy-WSB” (”easy” with safety box activated) only reached a score of 0,98 after 50 generations.
This behavior was seen in every tested fitness function which is very interesting because spontaneous most people would think if the average score of one fitness function is always better than the other one it must reach a better score

faster which is not the case.

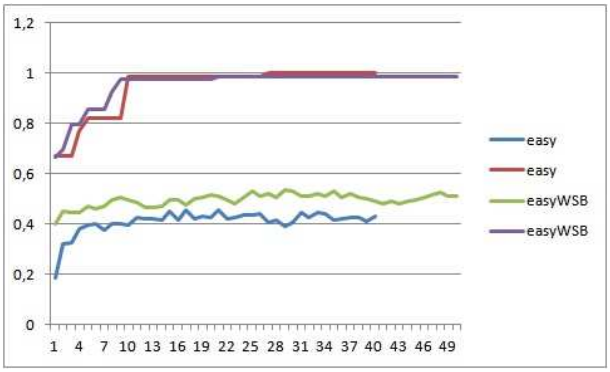


Figure 3: Best and average fitness score of a simple fitness function in the process. EasyWSB represents the same fitness function as easy but with the safetybox-system activated in the learning process.

To avoid the bad behavior shown in section 5, where the wheelchair has learned to move from any starting fields in the upper room but shows not the same behavior in the bottom starting area, we tested the same fitness function with an alternative starting area test where each individuum's three runs covers all three starting rooms. The result of the learning process can be seen in figure 4. This test we named "easySR".
In comparison to "easy", where the wheelchair may not start in each of the three rooms in each individuum, "easySR" has a slightly better behavior in the bottom room. However it does slightly worse than "easy" in the upper room.
As discussed in section 5 every critical starting fields must be covered in the future.

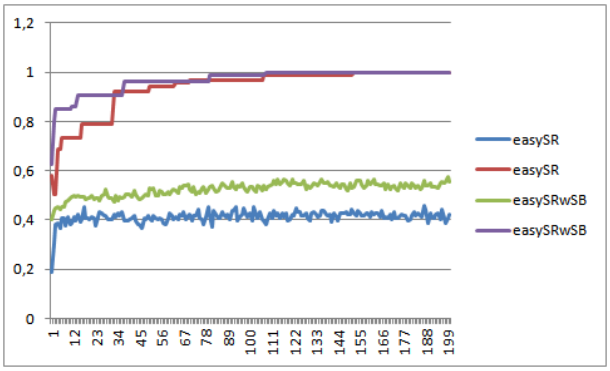


Figure 4: Best and average fitness score of the same fitness function easy but this time with starting position seperated in each of the three rooms.

7. DISCUSSION

As seen in the video in section 5 the fitness function "easy" sometimes collides with a wall. This may be the cause when only one sensor of the wheelchair sense the wall before the wheelchair. In general at least two or three sensors sense an object before the wheelchair so when only one sensor sense it the wheelchair may not learned to avoid it.
To get rid of such misbehavior many hours of evaluation, experience and learning is needed, or simply add a safety box which was our way to solve this problem.

To avoid the problem, where the fitness function does not learn how to drive successfully to the goal at any random position and direction, the random starting position should cover every critical place. But even if doing so, there will be always some places which are not covered. To find them all is an intense process of testing and evaluating. The easy way is to add more instances per individuum but then not all situations may be covered. The best way to deal with it may be a mix of both ideas: to add more instances per individuum which cover most of the critical places but also some completely random places.

To explain the behavior, where the wheelchair which learns with the safety box activated has a higher average fitness than when learning without it, the wheelchair without safety box has much more freedom so it may find quicker a solution of a problem than with the safety box activated in the learning process. Also it does not collide with an object so the fitness score is always higher than 0,01 which represents the fitness score if the wheelchair collides with an object. This explains why the average score with safety boxes is higher at the beginning of the learning process. But after a while even the wheelchair without safety box activated learns to avoid objects most of the times.

One key finding of the project is that the work with evolutionary algorithms requires some kind of instinct with regard to fitness function design. Some changes in this key function may lead to unexpected side effects.

8. CONCLUSION

The implemented system gives an approach to realize the control of a semiautonomous wheelchair using evolutionary algorithms. The evolution can be done step by step using a two dimensional simulator first, before the result is refined in a physics based 3D-Environment. Additionally subjective user impressions can be tested using a virtual re-

ality interface. Thereby an evolved individuals fitness can be influenced by user rating. For evaluation the effects of different fitness functions are presented.

9. ACKNOWLEDGEMENT

10. REFERENCES

[1] G. Bauer, F. Gerstenbrand, and E. Rumpl, "Varieties of the locked-in syndrome," *Journal of Neurology*, vol. 221, no. 2, pp. 77–91, 1979. [Online]. Available: http://dx.doi.org/10.1007/BF00313105

[2] Y. Kobayashi, Y. Kinpara, E. Takano, Y. Kuno, K. Yamazaki, and A. Yamazaki, "A wheelchair which can automatically move alongside a caregiver," in *Human-Robot Interaction (HRI), 2011 6th ACM/IEEE International Conference on*, 2011, pp. 407–407.

[3] S. Teller and N. Roy, "Autonomous wheelchair." [Online]. Available: http://www.csail.mit.edu/videoarchive/research/robo/autonomous-wheelchair

[4] U. Borgolte, H. Hoyer, C. Bhler, H. Heck, and R. Hoelper, "Architectural concepts of a semi-autonomous wheelchair," *Journal of Intelligent and Robotic Systems*, vol. 22, no. 3-4, pp. 233–253, 1998. [Online]. Available: http://dx.doi.org/10.1023/A%3A1007944531532

[5] M. Mazo, "An integral system for assisted mobility [automated wheelchair]," *Robotics & Automation Magazine, IEEE*, vol. 8, no. 1, pp. 46–56, 2001.

[6] D. Huang, K. Qian, D.-Y. Fei, W. Jia, X. Chen, and O. Bai, "Electroencephalography (eeg)-based brain #x2013;computer interface (bci): A 2-d virtual wheelchair control based on event-related desynchronization/synchronization and state control," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 20, no. 3, pp. 379–388, 2012.

[7] R. Chai, S.-H. Ling, G. Hunter, and H. Nguyen, "Mental non-motor imagery tasks classifications of brain computer interface for wheelchair commands using genetic algorithm-based neural network," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 2012, pp. 1–7.

[8] J. R. PATTERSON and M. GRABOIS, "758 locked-in syndrome: A review of 139 cases," *Stroke*, vol. 17, no. 4, 1986.

[9] B. Inden, Y. Jin, R. Haschke, and H. Ritter, "Exploiting inherent regularity in control of multilegged robot locomotion by evolving neural fields," in *Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on*, 2011, pp. 401–408.

[10] [Online]. Available: http://www.techfak.uni-bielefeld.de/ags/wbski/labor.html

[11] [Online]. Available: http://www.techfak.uni-bielefeld.de/ags/wbski/labor.html

[12] [Online]. Available: http://de.wikipedia.org/wiki/Dijkstra-Algorithmus

INTELLIGENT SYSTEMS PROJECT: THE MULTILINGUAL CITEC RECEPTIONIST

K. Buschmeier, H. ter Horst, M. Otto

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: C. Unger, H. van Welbergen, S. Walter, J. Gaspers, S. Kopp, P. Cimiano

ABSTRACT

Navigating through an unfamiliar and thereby complex building can be an exhausting act. Motivated by the new campus at Bielefeld University, we built a virtual multilingual receptionist with focus on two languages: English and German. Our system helps users in finding thier way on campus and an intuitive manner to request information through natural language interaction. In this paper we describe the system architecture and how to achieve language identification. In two experiments we examine our work on the agents multilingual capabilities and on the interpretation of various natural language utterances. As shown in the evaluation both language identification and interpretation have a high accuracy being relevant for the prospected practical application.

1. INTRODUCTION

This year the new campus of Bielefeld University will be inaugurated. In order to provide aid in finding one's way in the new building, we implemented a *Multilingual Receptionist* (MULIREC), a virtual agent which will be placed in the foyer to offer help to German and English speaking visitors by means of natural language dialogue. Our goal is to provide information about where to find people, their telephone numbers and email addresses, as well as information about restrooms, elevators etc., referred to as *special purpose rooms*, in a dialogue setting.

The system implements a common dialogue system pipeline from input recognition to output generation, in such a way that it is easy to later extend the agent with additional dialogue behavior.

To make the agent more intelligent we put focus on considering context information. For example, the user can talk about a specific person and in a new sentence he is able to use personal pronouns to refer to the previously named person. The agent has also the ability to react appropriately to incompletely recognised sentences, e.g. caused by colloquial speech or by the use of unknown words or names. Beyond that the system is able to rephrase uttered sentences in order to clarify misunderstood utterances produced by the agent, which is a natural behavior in spoken human-human interaction.

We evaluate the system's ability to identify the language that is spoken by the dialogue partner (English or German) as well as its performance of understanding utterances and identifying the spoken language.

The paper is structured as follows: First we present the system architecture and describe the system design and the involved components. Then we give a detailed account of the software components, together with a dialogue example. Finally we present our evaluation results and present ideas for future work.

2. RELATED WORK

In the receptionist domain several other dialogue systems exist such es ASKA [4] by Nisimura et al. or the Baysian Receptionist [5] by Horvitz and Paek. In contrast to these systems we incorporate a grammar-based natural language understanding component, similar to the one used in the TALK project [6] by Perera and Ranta.

3. SYSTEM DESIGN

This section describes the architecture of the system, as depicted in Figure 1. It follows a pipeline architecture leading from the user input to the agent's output.

First, the spoken user input is subject to *speech recognition*, based on an operating system integrated tool included in Microsoft Windows. This approach was preferred over other options, such as the open-source speech recogniser ESMERALDA [8], as it does not require extensive training as well as configuration and was successfully applied in a previous project with the same time constraints. Simultaneous speech recognition for German and English was realised by running virtual machines in parallel.

The utterance received from the speech recogniser is then parsed on the basis of a domain-specific grammar, implemented with *Grammatical Framework* [7]. This process reduces the large number of possible interpretations a natural language utterance can have to a small number

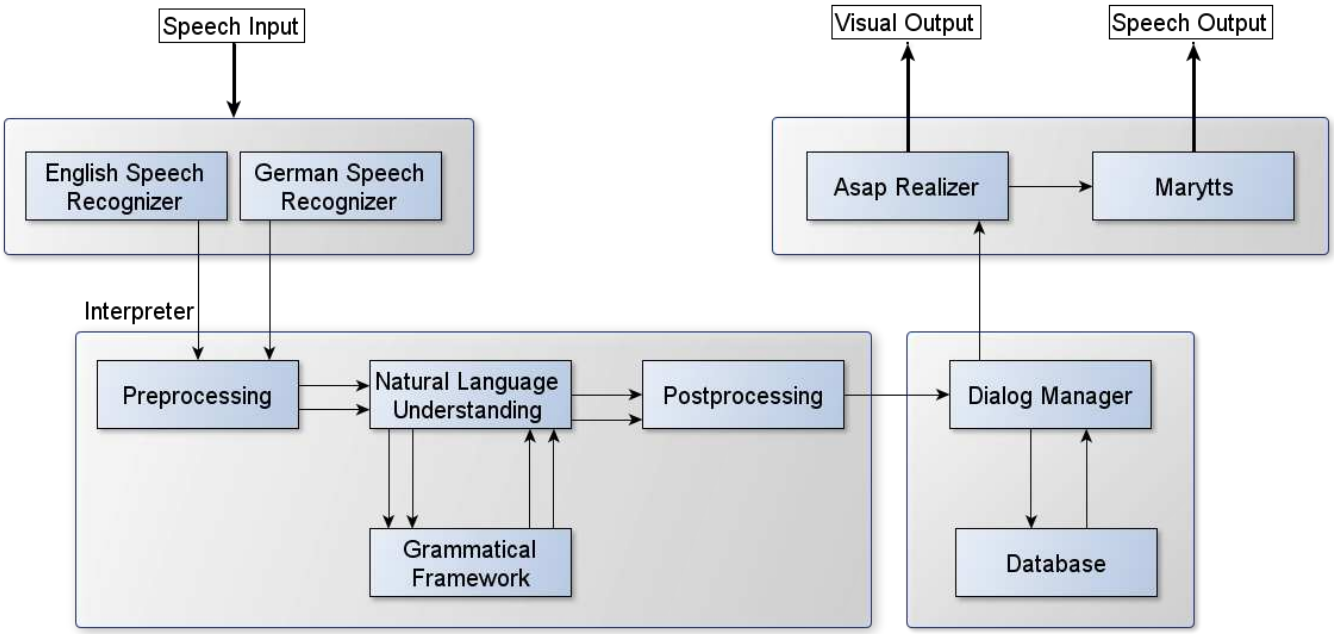


Figure 1: System architecture.

of domain-specific meanings. The resulting parse trees are converted into a specific format described below that specifies semantic information such as *dialogue acts*, for instance *Greet* or *Query* conveyed by utterances like "Good morning" or "Where is the office of Mrs Miller", or *None* in the the case of parse failure. The result of the interpretation process is the basis for language identification, described in Section 4.3.

Once an utterance has been parsed and its language is identified, the virtual agent has to respond to it appropriately. The dialogue manager, described in Section 4.4, manages a finite-state machine of dialogue states and transitions between them, handles requests for the database, saves context information, and decides how the agent is to respond in a particular situation.

Finally, a response utterance is generated, which is synthesised using the *Mary text-to-speech* module [9] (Marytts) and combined with adequate body movements of the agent, see Section 4.5. The agent is visualised by means of the *Articulated Social Agents Platform (Asap) Realizer*[1], (also described in Section 4.5).

For communication between these components we use the *Incremental Processing Architecture for Artificial Conversational Agents* (IPAACA) framework [1], which builds on *RSB* [3]. IPAACA provides an easy way for sending and receiving messages. It allows components to publish messages to a bus-system, and to register on a specific bus in order to receive relevant incoming data. All modules are written in Python. Messages are therefore sent as Python dictionaries that wrap tagged information. The only exception are messages to the Asap realizer module, whose task is to display gestures and movements of the agent. Corresponding to the Asap realizer's system design, information is wrapped in Behavior Markup Language (BML) format[2].

In the following section we describe all components in more detail.

4. SOFTWARE COMPONENTS

4.1. Speech Recognition

Speech recognition was realised using the Microsoft Windows Speech Recognition application (WSR). In order to enable speech recognition in several languages, in our case English and German, two or more virtual machines running the operating system Windows 8 are running on PC1 (host). All virtual machines recieve the raw data from the microphone of the host and have access to the network. Each virtual machine runs the WSR in a different language and sends the results to all computers in the network via IPAACA. If the host is a Linux machine, no second computer is needed. However, this possibility has not been tested as the performance may decrease significantly if all necessary software is executed on one computer. PC2, which is running the agent, receives the speech recognition results, interprets them (see Section 4.2) and collects the

[1] http://asap-project.ewi.utwente.nl/wiki

[2] http://www.mindmakers.org/projects/bml-1-0/

interpretation results for all languages. The most reasonable interpretation is accepted and passed on to the dialogue manager (see Section 4.4). What is considered most reasonable is defined by the language identification component explained in Section 4.3.

4.2. Natural Language Understanding

The natural language understanding component, also referred to as *interpreter*, receives the speech recognition output and returns semantic data in a machine-readable format, which is sufficient for the dialogue manager to determine the intention behind the received utterance. The main modules and steps are as follows:

Preprocessing The received utterances are first normalised, i.e. all letters are changed to lower case, shortened expressions like "I'm" are expanded to "I am", and filler words such as "actually" are removed, as the information that could possibly be conveyed by them is not relevant in the current scenario.

Parsing This is the main part of the natural language understanding module. It tries to parse the received utterances, given a domain-specific grammar in GF format. This grammar consists of an abstract syntax that captures the semantic concepts relevant to the receptionist domain and the dialogue task (such as *Office* and *Greet*), as well as two concrete syntaxes that specify particular verbalisations of these concepts in English and German, respectively. The implementation of the grammar was supplied by the Semantic Computing group.

Postprocessing Since the parse trees reflect the grammatical structure of the utterance, they have to be further abstracted before they are sent to the dialogue manager. The postprocessing module therefore converts the parse trees into a simple construct containing only those information that are essential for the domain-specific communication, e.g. the general dialogue act such as *Greeting* or *Request*, or requested information. These data are organised in fields of a Python dictionary, as shown in Figure 3.

Choosing and publishing a result The number of post-processed outputs varies considerably depending on the complexity of the utterance. This module collects all those outputs and decides, which of them will be passed on to the dialogue manager. This also comprises the identification of the language that was spoken and that thus has to be used for the agent's response (see Section 4.3 for details).

4.3. Language Identification

As the flow chart in Figure 1 indicates, the interpreter modules, which precede the postprocessing are each instantiated twice – once for the German and once for the English interpretation pathway. Notably, the result of the German speech recognition is never tried to be parsed by using an English grammar and vice versa. Another important issue is the asynchrony of this process: The German and English speech recognition can send their results at different points of time, as both processes run independently. Moreover, the parsing can take several seconds, depending on the complexity of the given utterance. The postprocessing synchronises the system again, by waiting a *maximum time* after one interpretation result is registered to receive also the other result. After that time, the other result is assumed to be *None*.

When synchronised, the decision which language was spoken is mostly simple, *i.e.* either only one of the speech recognition results was not parseable or neither of the results was parsable. In the latter case, the agent will ask the user to repeat the utterance in the most recently detected language. When both the English and the German parsing delivers at least a partial interpretation, the language in which more information aspects were detected is chosen. For example, in a German utterance representing a query for the office of a person, the name of this person may be recognised by the English speech recognition, leading to a partial interpretation consisting only of this name. In contrast, the German speech recognition could enable the interpreter to additionally detect the fact that the user has a request and that this request is concerning a path to the person's office. In this example, the German interpretation result is preferred. The same principle is applied for multiple interpretation results in one language: only the most detailed result is kept and compared to the other language's result.

4.4. Dialogue Manager

The dialogue manager is essentially a finite-state machine. It models possible *dialogue states* in the discourse from the agent's perspective, for example *Idle* and *SolutionGiven*, as well as transitions between these states, which constitute actions that the agent will perform. Appendix A lists all implemented dialogue states and Figure 2 shows these states and their possible transitions.

4.4.1. Action Selection

Selecting an appropriate action depends on the current user input (Appendix B shows a list of all input fields) and the context, which consists of the current dialogue state as well as previous user input. More specifically, actions are captured as *restricted actions* that consist of two preconditions, the action itself, and a postcondition. The first precondi-

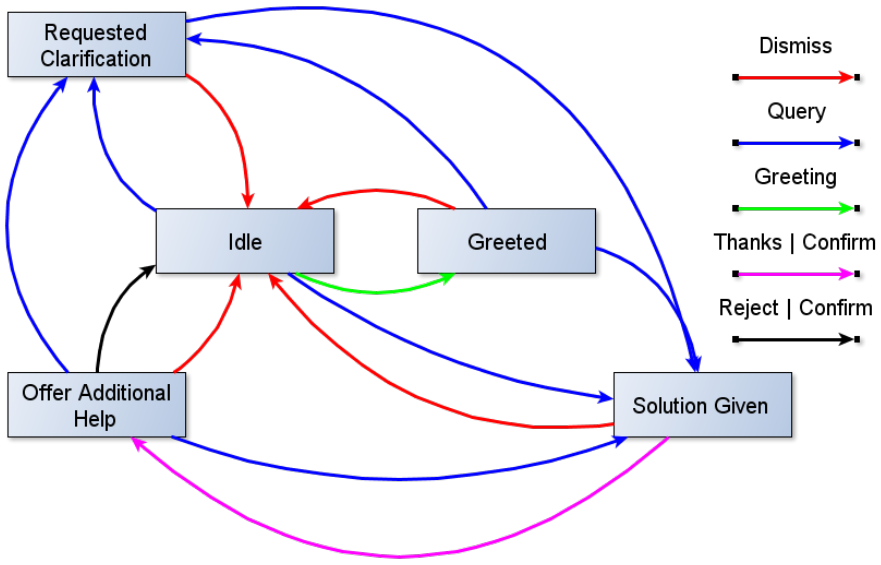


Figure 2: Dialogue states and their possible transition.

tion and the postcondition specifically address the dialogue state, whereas the second precondition handles additional requirements regarding user input and context. So firstly, for an action to get selected, its preconditions have to be met. Is this the case, the action, which often is to give a specific answer, is performed and its results are saved. Secondly, the postcondition has to be met as well for the action to be actually applied. In the example given in Listing 1 the action *Offer Additional Help* is performed if the current state is *Solution Given* and the incoming dialogue act *Confirm*. If after the execution the new state matches *Offered Additional Help* the action is applied.

Listing 1: Example of a restricted action.

```
RestrictedAction(
  match_state('SolutionGiven'),
  lambda i:
    i[INPUT_DIALOGUE_ACT] == 'Confirm',
  offerAdditionalHelp,
  match_state('OfferedAdditionalHelp'))
```

4.4.2. Formulating Answers

Currently all actions our agent is able to perform require an answer to be formulated, which is subsequently uttered. In the easiest case, for example a greeting setting where the agent simply says "Hello" or "Hi", the dialogue manager chooses randomly from a list of possible utterances. We use this random choice solution because it offers a variable and more natural appearance and provides an easy but elegant approach to rephrase misunderstood utterances. More specifically, to repeat itself the agent simply chooses another utterance randomly and adds a phrase like "I said" or "I meant" in 50% of the cases.

More sophisticated cases, for example answering a question that the user asked, require a database lookup, choosing an appropriate answer template and inserting the relevant information.

4.4.3. Information About Persons

User queries can be two-fold. They can either request information about a specific person, such as his or her office, email address or phone number. In this case, an the existing LDAP server with a directory of the staff is queried using the OpenLDAP command line search tool[3]. Afterwards the result is filter for the needed information which are then inserted into the chosen template.

On the other hand, user queries can concern so-called special purpose rooms, like laboratories, restrooms, kitchen facilities, etc. Since this information is not covered by the LDAP database, the dialogue manager obtains the required information about these rooms from a special CSV-file containing information about all rooms in the building.

4.5. Speech Synthesis and Behavior Modeling

The Asap realizer defines and realises gestural behavior and is able to send verbal string messages to the Marytts module. To this end, a behavior message in combination with a verbalised output message is published via IPAACA. This message has the form of a BML specification. Listing 2 shows

[3] http://linux.die.net/man/1/ldapsearch

an example of an easy BML message related to a *Greeting* action. It contains two parts: line two shows the message specifying the gestural behavior of waving, and lines three to seven show the string message "Good Morning" for verbalisations. The Asap realizer is able to interpret the first part of the message and prompt the virtual agent to perform the given gesture. The second part is passed to the Marytts module which extract the verbalised string from the given BML message by looking at the <*speech*> tag.

5. INTERACTION EXAMPLES

Figure 3 outlines a characteristic dialogue between a user and our system.

USER: Where is Hendrik's office?
SYSTEM: Do you mean Hendrik Buschmeier or Hendrik ter Horst?
USER: Hendrik ter Horst.
SYSTEM: Hendrik ter Horst's office is in H1-113.

Figure 3: Common dialogue between a user and the system.

The user asks the system where the office of a person named Hendrik is. First the speech recognition module transforms the spoken input into text, which results in the correct string "Where is Hendrik's office" for English and the meaningless German counterpart "U. r. s. t. Hendriks ob es". Second, the natural language understanding component parses both strings and transforms the resulting parse trees into the specific format that is sent to the dialogue manager. For the German string the result is empty, as it cannot be parsed. For the English string, the output looks as follows:

Listing 3: The output of the interpretation process for the first sentence.

```
output = {
  'language': 'Eng'
  'dialogueAct': 'Query',
  'requested': 'office',
  'ldap_uid': '["hterhors","hbuschme"]'
}
```

That is, the following information is captured: the language is English, the user wants to get an information, hence `'dialogueAct': 'Query'`, about an office, hence `'requested': 'office'`, whose owner's user ID is either `hterhors` or `hbuschme`.

Next, the system decides which interpretation is the best fit and passes it to the dialogue manager. The dialogue manager in turn searches for an appropriate action that suits the context and the newly received information. More specifically, to answer a question regarding an office, it searches for all given user IDs. Because the result contains two such IDs, the dialogue manager lacks sufficient information to decide which person the user is referring to and thus requests the user to clarify the query by directly asking who the user is referring to. After the user specified to whom he or she is referring, the described process starts again for the next user utterance, "Hendrik ter Horst". This time the output of the natural language understanding component looks as follows, i.e. contains only one user ID:

Listing 4: The output of the interpretation process for the second sentence.

```
output = {
  'ldap_uid': '["hterhors"]'
}
```

Again, the system adds information about the language from the previous output and passes it to the dialogue manager. The dialogue manager once more searches for the most suitable action. Since only the user ID but neither a dialogue act nor a specific request is specified, it retrieves this information from the context, i.e. from what was received from the previous user utterance. Now the dialogue manager has all necessary information to choose a corresponding answering action and answer the initial question appropriately.

6. EVALUATION

6.1. Evaluation of Language Identification

In this evaluation we tested whether it is possible to identify the spoken language based on the results of the interpreters for both languages running in parallel. Therefore common utterances which are not only recognised correctly by the speech recognition in the corresponding language but also parsed correctly were used. Since the other speech recognition is not expected to deliver a reasonable result, no interpretation result is expected either. Table 1 shows some example sentence pairs. We used ten German sentences and ten English sentences. As an evaluation measure we counted the number of correctly and incorrectly identified sentences. A sentence is identified correctly if the interpreter's output contains the corresponding input language. As shown in Table 2 the expected accuracy of 100% in language identification was reached. We correctly identified ten out of ten German sentences and ten out of ten English sentences. These results will be discussed in Section 7.1.

Language	Positive	Negative	ERR
Ger	**10** / 10	**0** / 10	**0**%
Eng	**10** / 10	**0** / 10	**0**%

Table 2: Results from the first evaluation.

Listing 2: BMLMessage

```
1 <bml xmlns="http://www.bml-initiative.org/bml/bml-1.0" id="bml1">
2   <faceLexeme id="gesture1" start="1.5" end="3.5"
                lexeme="hello-waving" amount="1"/>
3   <speech xmlns:bmlt="http://hmi.ewi.utwente.nl/bmlt"
            bmlt:voice="cmu-slt-hsmm" id="s1">
4     <text>
5       Good morning.
6     </text>
7   </speech>
8 </bml>
```

x	German Output	English Output
1	"U. r. s. t. ob es oft Kristina"	"Where is the Office of Christina"
2	"Wo ist das Büro von Christina"	"Before this does you'll want to clean up"
3	"Können Sie mir sagen wo sich das Büro von Frau Müller befindet"	"Anything is like the borders does pull on Scott Miller the finance"
4	"Canyon habe ihn u. l. t. Office auf nächstes nenne es"	"Can you tell me where the office of Mrs. Miller is"
5	"U. r. s. drei wird."	"Where is the toilet"
6	"Ich suche die Toilette"	"The salt little metal"
7	"Weißt du die Telefonnummer von Herrn Müller"	"Biased toward each year for us, can lower"
8	"Den AO LV Namen der Taufe Mister r. s. Miller"	"Do you know the phone number of Mr. Miller"
9	"Auf Wiedersehen"	"Often leaders in"
10	"Mit drei"	"Goodbye"

Table 1: Example sentences which were used for language identification. These are the outputs from the speech recogniser.

6.2. Interpreter Robustness

The second evaluation performed focuses on the capability of the interpreter to parse utterances of a realistic variety. In the prospected scenario, the agent will be interacting with various users, each having an individual linguistic style comprising diction, grammatical preferences and error distribution. The test utterances were obtained during one day on the campus of Bielefeld University. Subjects were asked how they would approach a robot or person at the entrance of a building if they had concerns, which are likely to occur in the receptionist scenario.

Subjects were tested either for the robot or the person case. The answers were given directly and were recorded using standard smartphones. The original transcribed utterances were modified as described in Table 3.

In order to quantify the interpreter robustness, a score and five quality classes were defined as follows:

Score The score consists of two values: the denominator denotes how many relevant aspects are to be understood in the given utterance, while the numerator indicates the number of aspects that the interpreter has correctly determined. The *score value* is the result of the division of numerator and denominator.

No interpretation If the interpretation result is *None*, the numerator is always zero; the utterance was *not interpretable*.

False interpretation The numerator can even be negative, *i.e.* if an aspect is detected, which is not contained in the original utterance; the utterance was *interpreted wrong*.

Partial interpretation When the interpretation result contains some, but not all of the aspects contained in the utterance, the score value will be in the interval $]0, 1[$; the utterance was *interpretable partially*.

Correct interpretation When the interpretation result consists of all relevant aspects of the utterance given, the score value is one, the utterance is *interpreted correctly*.

Acceptable interpretation All interpretation results which contain more than the half of the relevant aspects in the given utterance, *i.e.* interpretation results of a score value above 0.5 are considered *acceptable interpretations*.

The results of this quantification are presented in Table 4.

Modification	Original utterance	Modified version
Replace name placeholder	"Wo ist Herr X"	"Wo ist Frau Unger"
Common dialect corrections	"Tschuldigung, ..."	"Entschuldigung, ..."
Only use first answer given	"Wo ist die Toilette? Oder ich würde sagen: Wie finde ich die Toilette?"	"Wo ist die Toilette"

Table 3: Modifications to test utterances for interpreter evaluation.

Scenario	Average score value	Interpretation result				
		False	No	Partial	Correct	Acceptable
person	61.7%	6.4% (3/47)	23.4% (11/47)	6.4% (3/47)	63.8% (30/47)	70.2% (33/47)
robot	84.0%	0% (0/53)	13.0% (7/53)	5.6% (3/53)	81.5% (44/53)	83.3% (45/53)

Table 4: Results of interpreter evaluation.

Of the 100 utterances which consist of 47 utterances from seven subjects and 53 utterances from nine subjects for the person and the robot case respectively, 74% were interpreted correctly. Focussing on the robot case, one can say that **81%** of the utterances, the subjects presume to confront a robot with in the given scenarios, are understood correctly. None of the 53 utterances led to a wrong interpretation, while among the 47 utterances, which subjects presume to confront a person with, three would have been interpreted differently from the user's intention.

7. DISCUSSION

7.1. Language Identification

As can be seen in Table 2 we got an accuracy of 100% in language identification. This is an amazing result and shows that our system does its job pretty well. However, in our evaluation we restricted the input to interpretable sentences, because the identification exploits the interpretation module. By the use of sentences which the interpreter cannot interpret, the system is not able to identify the given language. In this case the previously identified language will be chosen. This also applies for some utterances which we cannot assign a unique language to e.g. utterances like "*Hi*".

This could be a problem interpreting natural language because of the very high complexity. To prevent a high interpretation failure rate we did a second evaluation which evaluates the performance of the module.

7.2. Interpreter Robustness

The interpreter, which is used in the current work, is based on the expectation that the user has several possibilities of expressing the same information. E.g. the order of the words is restricted by grammatical rules, which are therefore implemented in Grammatical Framework, responsible for parsing the utterances. Similarly, the dictionary is limited to the possibilities of expressions which are anticipated to be relevant in the domain. As the developer thereby introduces a personal, *a priori* ontology, the results described in Section 6.2 are crucial to estimate the system's generalising capabilities.

First of all, the evaluation was splitted in subjects imagining a conversation with a person and others imagining a conversation with a robot. The term "robot" was used in order not to drag too much attention to the details of the eventual visualisation, especially considering the variety of systems the interpreter may be applied for. Subjects seem to automatically adapt their linguistic style, when imagining an artificial dialogue partner and interestingly, this adaptation is appropriate to improve the system's interpretation capacity and thereby reducing misunderstandings. Strikingly in this context, in 14 out of 47 utterances (30%) subjects presume to say to a person could not be interpreted or were even falsely interpreted, while among the 53 utterances which subjects imagined to say to a robot, only seven (13%) weren't interpretable and no misunderstandings occured. Although this adaptation of the user is effectively improving the interaction, it has to be focused on enhancing the agent's potential for adaption.

One approach to improve the interaction quality is the ability of the presented system to partially interpret utterances, and to inquire about missing parts. In three cases, the utterance presumed to be said to a person was almost correctly interpreted (a score value above 50%), while among the utterances presumed to be said to a robot only one was almost correctly interpreted. Despite this difference being marginal, it can be expected that the feature of partial interpretation becomes more relevant, when the variety of utterances increases, as it happens, the more naturally the user approaches the virtual agent.

8. CONCLUSION

We implemented a virtual multilingual receptionist which is able to both understand and generate natural multilingual utterances. The applied concepts proved to be promising as the evaluation of the strategy used to identify the uttered language resulted in 100% accuracy. Moreover, the interpreter's capability to parse utterances of a realistic variety has an accuracy of 61.7% for human-human dialogues and 84% for human-robot dialogues.

Towards the project's goal to install the developed system at the new campus at Bielefeld University the next step could be to evaluate the system's performance in a real world context and use the results as a basis for additional dialogue acts as well as for further refinement of the grammar.

9. ACKNOWLEDGEMENT

We would like to take this opportunity to thank our supervisors Christina Unger, Herwin van Welbergen, Sebastian Walter, Judith Gaspers, Stefan Kopp and Philipp Cimiano for their constant guidance and support.

We thank the Sociable Agents group especially Ramin Yaghoubzadeh, Hendrik Buschmeier, Sebastian Ptock and Amir Sadeghipour for providing software and hardware as well as technical support that contributed to the project's success.

10. REFERENCES

[1] D. Schlangen and G. Skantze, "A general, abstract model of incremental dialogue processing", in *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Athens, Greece, 2009, pp. 710–718.

[2] D. Schlangen, T. Baumann, H. Buschmeier, S. Kopp, G. Skantze, and R. Yaghoubzadeh, "Middlewares for incremental processing in conversational agents", in *Proceedings of the 11th Annual SIGdial Meeting on Discourse and Dialogue*, Tokyo, Japan, 2010.

[3] J. Wienke and S. Wrede, "A Middleware for Collaborative Research in Experimental Robotics", *IEEE/SICE International Symposium on System Integration (SII2011)*, Kyoto, Japan: IEEE, pp. 1183–1190, 2011.

[4] R. Nisimura, T. Uchida, A. Lee, H. Saruwatari, K. Shikano, and Y. Matsumoto, "ASKA: receptionist robot with speech dialogue system", in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, pp. 1314–1319.

[5] E. Horvitz and T. Paek, "A computational architecture for conversation", in *Proceedings of the seventh international conference on User modeling (UM '99)*, 1999, pp. 201–210.

[6] N. Perera and A. Ranta, "Dialogue System Localization with the GF Resource Grammar Library", in *Proceedings of the ACL Workshop on Grammar-Based Approaches to Spoken Language Processing (SPEECHGRAM 2007)*, 2007.

[7] A. Ranta, *Grammatical Framework: Programming with Multilingual Grammars*, CSLI, Stanford, 2011.

[8] G.A. Fink and T. Plötz, "Developing Pattern Recognition Systems Based on Markov Models: The ESMERALDA Framework", in *Pattern Recognition and Image Analysis*, vol. 18, no. 2, pp. 207–215, 2008.

[9] M. Schröder and J. Trouvain, "The German Text-to-Speech Synthesis System MARY: A Tool for Research, Development and Teaching", in *International Journal of Speech Technology*, vol. 6, pp. 365–377, 2003.

[10] H. van Welbergen, D. Reidsma, and S. Kopp, "An Incremental Multimodal Realizer for Behavior Co-Articulation and Coordination", in *Intelligent Virtual Agents, 12th International Conference*, LNCS, vol. 7502, Springer, pp 175–188, 2012.

A. APPENDIX – DIALOGUE STATES

The following dialogue states were implemented:

Idle The agent's state before and after a conversation.

Greeted The agent has greeted the user.

SolutionGiven The agent gave a full answer.

RequestedClarification The agents needs more precise information to answer a question.

OfferedAdditionalHelp The agent offers help.

B. APPENDIX – INPUT FIELDS

The following lists describes all fields that can be contained in the dictionary handed from the natural language understanding component to the dialogue manager:

language Identified language.

dialogueAct Identified dialogue act:

- **None** No dialogue act identified.
- **Greet** User greeted the agent.

Bye User said goodbye to the agent.

Thanks User thanked the agent.

Rethanks User responded to a thanks (e.g. "You're welcome")

Confirm (e.g. "Yes")

Reject (e.g. "No")

Repeat User wants the agent to repeat its last utterance.

NotUnderstand User did not understand the agents last utterance.

Query User asks a question. (e.g. "Where is christina's office?")

requested The kind of information the user requested:

office The office of a specified person.

email The email address of a specified person.

telephoneNumber The telephone number of a specified person.

person The owner of an office.

room A special purpose room. (e.g. laboratory, restroom)

ldap_uid A list of IDs of identified persons.

ldap_roomNumber An office room number.

type The type of a special purpose room (e.g. laboratory, restroom).

name Name of a special purpose room (e.g. Media Lab).

INTELLIGENT SYSTEMS PROJECT: SWARM

Sergius Gaulik, Michael Goerlich, Matthias Esau

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: René Griessl, Stefan Herbrechtsmeier, Ulrich Rückert

ABSTRACT

In nature most simple insects are acting in swarms to solve incredible tasks. Japanese Honey Bees are luring attacking Mandarinia Hornet into their hive to kill it by covering it until it dies of heat. In contrast, the simultaneous localisation and mapping problem, known as SLAM in literature, is usually solved with just one robot. In this paper an algorithm is presented that uses several small robots, called BeBots[1], in the Teleworkbench environment[2], seen in Figure 1. The robots are able to localise themselves with optical features in a featureless area and create a map of their environment together.

1. INTRODUCTION

The simultaneous localisation and mapping problem is often addressed by literature in the field of robotics. The common method uses one robot and consists of four steps:

1. Navigate through the area
2. Calculate the new pose
3. Collect information about the environment
4. Update pose with collected data

Navigating through the environment usually causes insecurity about the robots pose, since it is calculated using odometry equations. Due to slippery ground, accidental hitting of objects in the environment, the calculated pose and the real pose may differ. The insecurity caused by driving makes it necessary to gain additional information to keep track of the robots pose. Commonly used sensors are depth-sensors like laser scanners. Some also utilise regular cameras. Additionally the rotations can be observed by a gyroscope as well. The information gained by the sensors need to be merged into the odometry pose estimation.

In this paper an algorithm is described that accomplishes this task with several BeBots in parallel. The BeBots first choose an anchor robot by comparing scores depending on their environment. The anchor robot then serves as orientation point for the other BeBots. In the next step they randomly choose areas to explore near the anchor robot. As distance sensors the BeBots use 12 inexpensive infrared sensors. The BeBots make their pose and the free area around it available to all listening robots. A gyroscope keeps track of the rotation while turning. The final angle is then merged into the pose estimation using a Kalman filter. In case the insecurity about the robots pose got too large the robot will try to find the anchor and remeasure its position and angle by using the Kalman filter. When the area around the anchor has been mapped, the robots choose a new anchor position by individually sampling random positions on the map. The samples are scored with the unknown area around the robot while taking blocked areas into account.

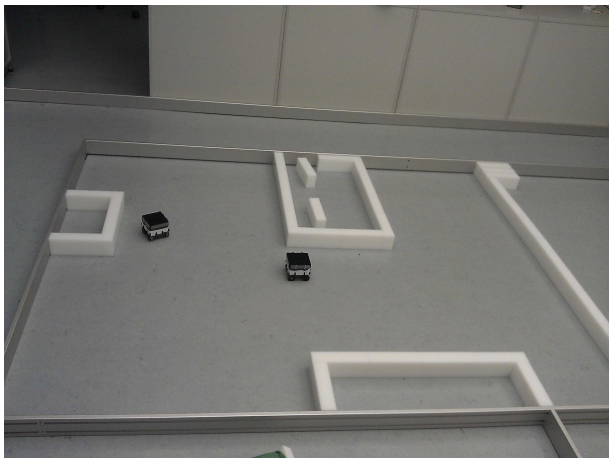


Figure 1: Teleworkbench

2. HARDWARE: CONSTRUCTION AND SETUP

The BeBot is an approximate quadratic small robot as seen in Figure 2. It has two chains and a solid dark-gray chassis. On top there is the black cover with the WiFi antenna. Under it the LED ring is placed. Each side can show a different color. The white stripe around the BeBot increases the ammount of reflected infrared light emitted by the twelve

infrared LEDs below this stripe. Due to the dark color of the BeBot the infrared light is absorbed and the range estimation would be inaccurate without it. In the middle of the white stripe in the BeBots front there is an Omnivision OV9655 camera. It is capable of delivering SVGA RGB images at a rate of 15 fps. At the bottom of the robot there are two chains that drive the robot. In terms of odometry this design is hard to model, but can be approximated quite good with simple differential steering.

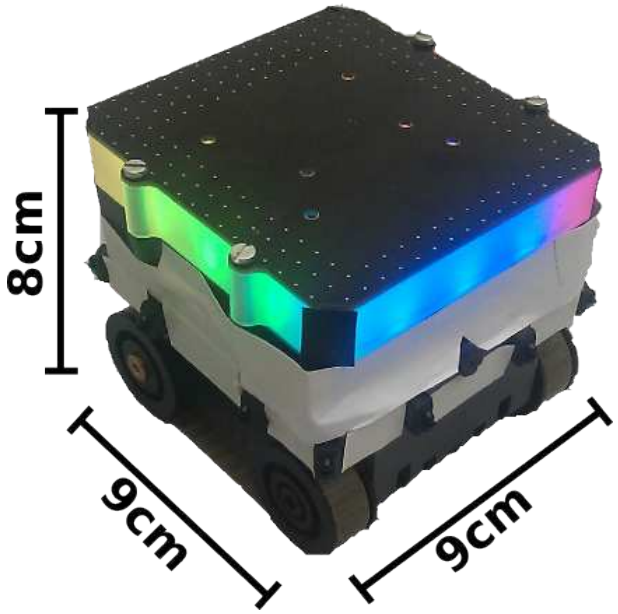


Figure 2: Bebot and its dimensions.

A microcontrollers integrated into the chassis is preprocessing the infrared values since they are prone to noise. Directly under the black cover is the WiFi chipset connected to the CPU board that carries an ARM core 600 MHz processor. It has 512 MB flash memory and 256 MB RAM running a small Linux system. The processor is accompanied by a 430 MHz DSP. Under the computing board there is the base board for controlling the engines.

3. SOFTWARE COMPONENTS

When the software is started it runs the four threads shown in Figure 3. All sensor information and found anchor robots are collected in the controller thread. The increment counters are used to calculate the new pose of the robot. The values of the infrared sensors are used to create the map and to avoid collisions. For observing the robots real pose while navigating through the environment two Kalman filters are used. One keeps track of the robots world angle while turning. It uses the gyroscope as information source. The other one uses the camera image to estimate the relative position

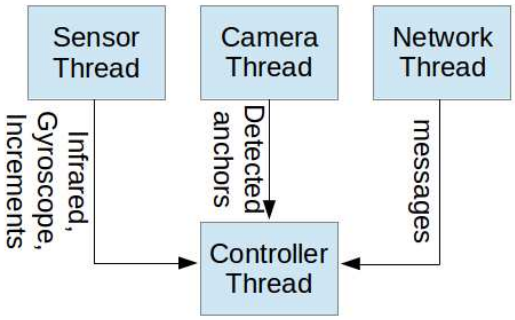


Figure 3: Overview of the systems threads.

and angle to the anchor robot. For initial localisation the camera is used as well but no Kalman filter is utilised.

3.1. Odometry

Due to the architecture of the BeBot it is possible to apply equations for a differential steering robot. These equations are thought to be applied to simulate the behaviour of a robot with two engines at each side. It would be optimal if only the wheels touched the ground. Due to the chains the equations cannot be applied directly when turning on the spot since this causes slippage. This is can be avoided while moving forward.
For applying the equations it is necessary to know the distance driven on each side. The Bebots have an increment counter with a resolution of 128 and a gear that has a ratio of 14. By taking into account the radius of the wheels (1.5 cm) the driven distance is derived by equation 1.

$$distance = \frac{increments}{128 \cdot 14} \cdot 2\Pi \cdot 1.5cm \qquad (1)$$

The distances and the gap between the wheels b can then be applied to the odometry function (2) to get the difference of the robot position.

$$\begin{pmatrix} x_{diff} \\ y_{diff} \\ \theta_{diff} \end{pmatrix} = \begin{pmatrix} \frac{sr+sl}{2} \cdot \cos(\frac{sr-sl}{2b} + \theta) \\ \frac{sr+sl}{2} \cdot \sin(\frac{sr-sl}{2b} + \theta) \\ \frac{sr-sl}{b} \end{pmatrix} \qquad (2)$$

Since a probabilistic approach to localisation is applied, it is needed to get a sense of security. For this reason the pose is seen as the mean of a gaussian distribution. The covariance is governed by using propagation of uncertainty. This enables us to estimate the uncertainty after driving. In equation 3 X is the robot pose and M the movement parameters

(the distance of the left and right wheel).

$$C_i = \frac{F}{\delta X} C_{i-1} \frac{F}{\delta X}^T + \frac{F}{\delta M} C_M \frac{F}{\delta M}^T \quad (3)$$

3.2. Movement Behaviour

The BeBot uses a simple algorithm to reach certain areas and avoid collisions with obstacles. For this purpose it uses the twelve infrared sensors. Given a driving direction the algorithm checks if the nearest four sensors, pointing in the desired direction, hint at a barrier. In the case, where no possible obstacles were found, the BeBot can take this course. Otherwise the checking of the sensors is moved clockwise

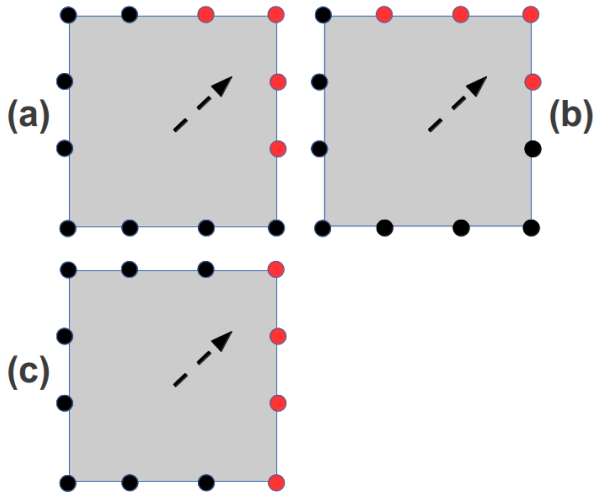


Figure 4: Choosing the next driving direction. The Arrow shows the desired direction. The red sensors in a) are checked first and then ones in b) and c)

and counterclockwise by one until the algorithm finds four sensors, which indicate no barriers nearby (Figure 4). In some situations an obstacle directly in front of the BeBot could cause a collision while turning away from the barrier. The reason for such a collision is the rectangle shape of the BeBot. In this case the BeBot drives backwards for one second to prevent a possible crash with the unexpected obstacle. The behaviour while driving backwards is the same as driving forwards. The speed of the BeBot always depends on the measured obstacle range. The direction for this algorithm can be calculated by using the odometry data and a target point in world or map coordinate system.

3.3. Mapping

Instead of using an expensive laser, the mapping is done using the infrared sensors. On a white surface the range from 4 cm up to 20 cm can be measured with a deviation of about 0.5 cm. The internal map is updated by using the BeBots pose and a polygon, that represents the BeBots surroundings. These information are broadcasted to make sure every listening system can use it to refresh its knowledge about the environment. This polygon (5) is created in three steps:

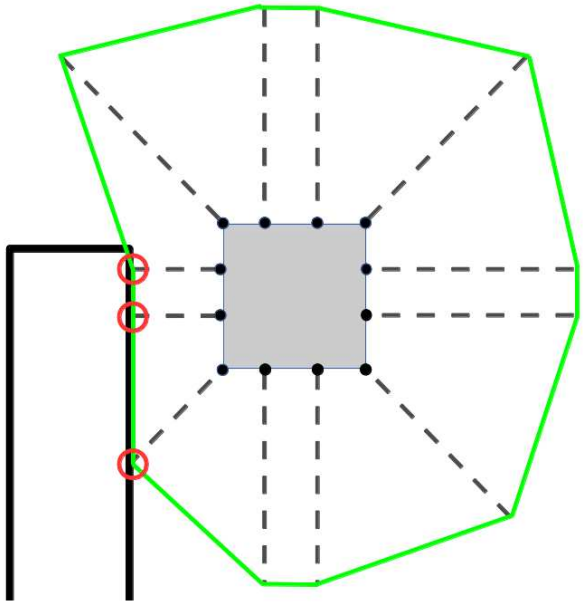


Figure 5: Illustration of the created polygon.

1. The calculated distances given by the sensors are capped at 17 cm to get more accurate data while moving.
2. The points for the polygon are extracted from the measured or capped distance in the direction of the sensors.
3. If a sensor shows a distance less than the cap, the corresponding point of the polygon is marked to indicate an obstacle.

The internal map is a 2D grid with positive or negative values at each point. At the start the map is initialized with zero values, which indicate an unexplored area. Negative values are considered as obstacles and positive values are accessible areas. The map is updated with the polygon and the pose of the BeBot as follows:

1. All zero values are set to one in polygon area to mark this part of the map as explored.
2. At every point, where an obstacle was found, a gaussian mask is subtracted.
3. At the pose of the BeBot a gaussian mask is added.

Because of the addition and subtraction previous recorded values can change their meaning. Furthermore higher values provide a statement about the reliability of the environment. The process of mapping can be visualized by a GUI, which was developed to track the progress of creating a map. The program listens to the broadcasted data and creates a map in real-time.

3.4. Kalman filter

The Kalman filter is a very general approach to sensor fusion. It is described in detail in [3]. It also offers a great framework to calculate how a sensor reading is affecting the robots pose. It only needs a function that describes what sensor values are expected when the robot is at a certain position.

$$h(X, \nu) = f(X) + \nu \tag{4}$$

Applying this function to the robots pose without the noise ν will give the expected value of the sensor. The amount of new information contained in the sensor readings is given by the Kalman gain, which is a matrix containing how the sensor can be used to optimise the position and uncertainty. The Kalman gain is given by equation 5, where R is the error of the sensor, V is $\frac{h}{\delta\nu}$ following the propagation of uncertainty and H is $\frac{h}{\delta X}$.

$$K = CH^T(HCH^T + VRV^T)^{-1} \tag{5}$$

By using the Kalman gain the difference of expectation and measurement, sometimes called innovation, can be converted into a position and an angle to optimise the robots pose and its uncertainty by using the equations 6.

$$\begin{aligned} X_{new} &= X_{old} + K(z - h(X, 0)) \\ C_{new} &= (I - KH)C_{old} \end{aligned} \tag{6}$$

3.5. Gyroscope

Gyroscopes are capable of delivering the rotation applied to them. In general they have a high drift due to the temperature dependence of the chip. This means that they tend to increase their value while no rotation is applied to the sensor. Before the actual start of the program the gyroscope values are accumulated and a mean drift is calculated. This mean drift is later subtracted from the differences of the last and the current value. When turning is detected through differences between speed of the wheels the sensor value is set to the current robot pose. After the speeds are back to very similar the value of the gyroscope is fused into the robot pose by using the Kalman filter.

3.6. Detecting Anchor Bots

Due to the increasing uncertainty of the BeBot a method to reliably readjust position and orientation is needed. To achieve this the camera is used to detect the led strip on top of the anchor bot. Every side has its own color, which allows other bots to know which side of the anchor bot is observed.

3.6.1. Preprocessing

Even though the camera of the BeBot provides white balancing capabilities they seem to be very limited and often produce even worse images than the raw camera. To improve the contrast and color balance of the image a very simple algorithm is used. Low brightness pixels often contain information on the color shift in the image. The average RGB values of these pixels is used to rebalance the image in equation 7.

$$\begin{aligned} sum_{average} &= \sqrt{\frac{R_{average} + G_{average} + B_{average}}{3}} \\ balance &= \left(\frac{sum_{average}}{R_{average}} \frac{sum_{average}}{G_{average}} \frac{sum_{average}}{B_{average}}\right) \\ \begin{pmatrix} R_{new} \\ G_{new} \\ B_{new} \end{pmatrix} &= \begin{pmatrix} R_{old} \\ G_{old} \\ B_{old} \end{pmatrix} \cdot balance \end{aligned} \tag{7}$$

3.6.2. Blob Detection

For simpler extraction of hue values the image is then converted into the HSV color space. To detect the led strip on top of the BeBot a thresholding on the saturation and value of the image is applied. For easier separation of the sides the corners of the strips have been prepared with dark tape. This ensures that blob detection correctly separates the sides from each other. Blob detection can then be used to find the individual sides of the color strips. To determine the color of the sides the average hue of the blob is used. Blobs with low saturation or low value are discarded.

3.6.3. Matching Sides to Bots

The extracted sides are then matched to find possible bots in the image. Every possible pair of sides is scored by the distance of the sides to each other. Pairs that don't match in size or color order are discarded. The pair with the highest score is added to the list of possible bots. All pairs containing sides of the new bot are removed from the list of possible pairs. This process is repeated until there are no pairs left. Sides with no matching partner are added as single sided bots.

3.6.4. Estimating the Angle

The angle at which a one sided bot is seen is given by the offset of its color. For two sided bots the angle is given by

equation 8.

$$\theta = \arctan(\frac{width_{left}}{width_{right}}) + offset(color_{left}) \quad (8)$$

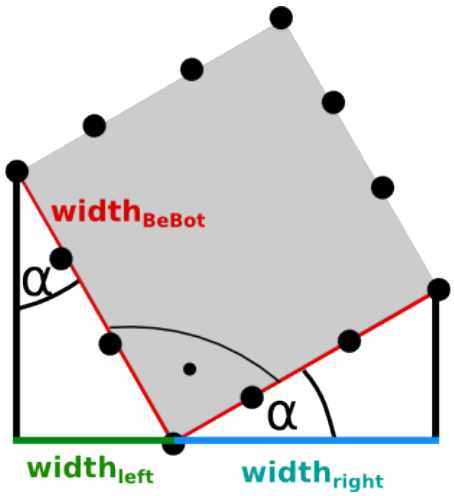


Figure 6: $width_{left}$ and $width_{right}$ are $\sin(\alpha) \cdot width_{BeBot}$ and $\cos(\alpha) \cdot width_{bot}$

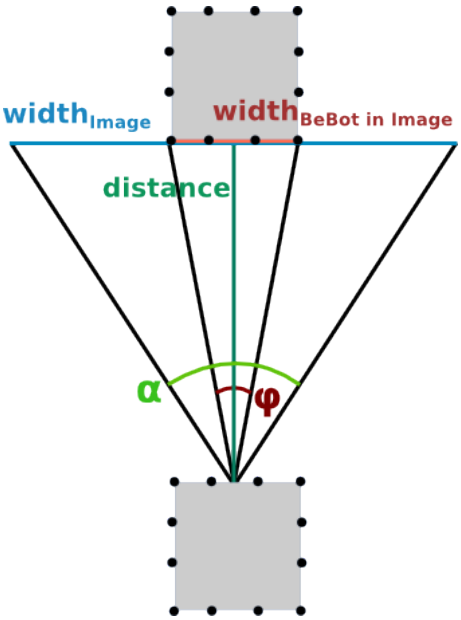


Figure 7: BeBot measuring the anchor bot

3.6.5. *Estimating Position and Orientation*

Given the aperture angle α, the width of the BeBot, the width of the BeBot in the image, and the size in the camera image the equation 9 can estimate the distance of the BeBot (Figure 7).

$$\begin{aligned} \phi &= \alpha \cdot \frac{width_{BeBotinImage}}{width_{Image}} \\ distance &= \frac{width_{BeBot}}{2 \cdot tan(\frac{\phi}{2})} \end{aligned} \quad (9)$$

With the position and orientation of the anchor bot the BeBot can adjust its odometry by fusing them with the Kalman filter.

3.7. Mapping Behaviour

To explore the environment the BeBots follow a behaviour that consists of the three states “Mapping the Outer Edges”, “Mapping the Holes” and “Finding a New Area”.

3.7.1. *Mapping the Outer Edges*

When a new anchor has been chosen the other BeBots will try to explore a rectangular area around it. All bots select a random position on the outer edge of the exploration area and try to approach it. The best path to reach this point is found using the a-star algorithm described in [4]. The search area of the a-star algorithm is limited to the exploration area around the anchor. When the area is completely unexplored this will result in a very direct path to the edge. As soon as the BeBot encounters an obstacle it will be added to the map and the a-star algorithm will find a way around it. When the point is not reachable from inside the exploration area a-star ensures that the BeBot tries to find any possible way that lead to the target position. This way the bot will explore the outer edges of the area. If the target position is reachable the BeBot will select a new unexplored target position on the edge of the exploration area. If all points on the edge have been discovered or found to be unreachable the bots will try to fill the holes left inside the map.

3.7.2. *Mapping the Holes*

To find holes inside the current exploration area a blob detection is applied to find all blobs of undiscovered area not connected to the edge. Next the BeBot will randomly select one of these areas and try to approach its center using the a-star algorithm. Should the center of this area be to close to a previously found unapproachable area the BeBot selects another area. If the center of the target area is found to be unapproachable it will be added to the unapproachable list and the process starts over. If the BeBot was able to approach the center the process starts over. When no new holes in the map can be found the BeBot signals the other bots to start looking for a new area to explore.

3.7.3. Finding a New Area

When the current Area has been explored the BeBots will try to find a new area to explore. Every bot will randomly select multiple points adjacent to the current exploration area and score it. The score is determined by how much unexplored area is around it and the distance that has to be traveled to reach it. All bots broadcast their best candidate and collectively select a new area. After selecting the area they select a new candidate for the next anchor. This candidate will approach the center of the new area. If the approach failed the process of finding a new area starts over. If the center was approached successfully the BeBot turns into the new anchor, the old anchor is released and all bots start mapping the new area.

4. EVALUATION

SLAM algorithms can be evaluated in different ways. One would be to evaluate the speed of the system (this algorithm is running with mostly 16 fps). Another option would be to compare the map to hand crafted a ground truth map. These two maps can be seen in Figure 8 and Figure 9. The red points in Figure 8 are the robots estimated positions.

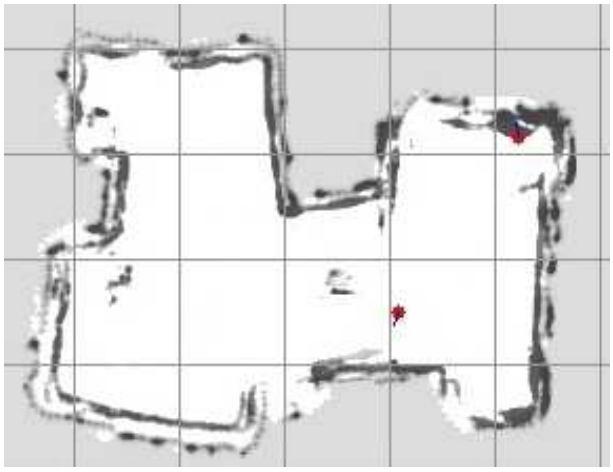


Figure 8: The map created by the BeBots.

Figure 8 and 9 show that the maps look alike, but have certain differences. What seems to be a big problem is rotation while the distances seem to be quiet reasonable. This could be explained with the robots steering. Due to the chains slippage is common, which gyroscope does not seem to counter completely.

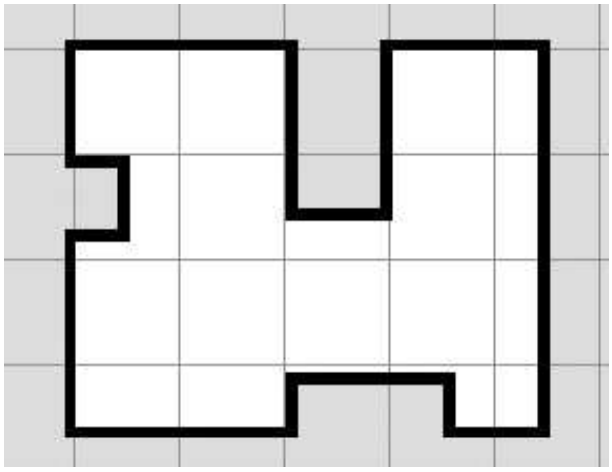


Figure 9: The ground truth.

5. DISCUSSION

During the project it became apparent that complex behaviours and the expectation of high precision seldomly lead to good results. Systems have to be developed to be highly error resistant. Before implementing the Kalman filter a more simple approach should have been implemented which would have allowed earlier and more detailed testing. Additionally the project provided good inside into managing resources on a limited system (making sure threads don't starve, etc.). Nonetheless the project shows, that inexpensive sensors and simple behaviour can achieve a complete exploration of the environment. Sadly accuracy suffers due to multiple factors:

- high drift of the gyroscope
- inaccurate odometry due to the approximated model of the chain drive
- slippage of the chains during sharp turns and rotation

Even though accuracy is not perfect the maps are still good enough to be used for pathfinding which is one of the main uses of these map. Further projects could look for a better model describing odometry. Techniques from [5] could be used to improve slippage correction by coupling the gyroscope and odometry. The BeBots could improve localisation by generating expected sensor readings from the pose and the map and comparing them to actual sensor readings.

6. CONCLUSION

This paper shows, that utilizing multiple robots allows mapping of environments by creating visual anchor features. These features can be used to improve pose estimation of the robots. Inexpensive infrared sensors are used to mea-

sure the distance to surrounding walls. The resulting map is not as accurate ones created by laser scanners or ToF cameras, but are good enough for pathfinding.

7. ACKNOWLEDGEMENT

We would like to thank our supervisors for providing us with ample support, effort and time. Additional thanks goes to Robert Abel for additional technical information and support.

8. REFERENCES

[1] S. Herbrechtsmeier, U. Witkowski, and U. Rückert, "Bebot: A modular mobile miniature robot platform supporting hardware reconfiguration and multi-standard communication," in *Progress in Robotics*. Springer, 2009, pp. 346–356.

[2] A. Tanoto, U. Witkowski, and U. Rückert, "Teleworkbench: A teleoperated platform for multi-robot experiments," in *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*. Springer, 2006, pp. 49–54.

[3] R. E. Kalman *et al.*, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[4] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100–107, 1968.

[5] J. Borenstein and L. Feng, "Gyrodometry: A new method for combining data from gyros and odometry in mobile robots," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 1. IEEE, 1996, pp. 423–428.

INTELLIGENT SYSTEMS PROJECT: VITAL, REAL-TIME ACTIVITY CLASSIFICATION

N. Dehio, C. Menßen, C. Wall

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisor: P. Christ

ABSTRACT

Elderly people often get insufficient exercise in their daily life. The system presented in this paper could solve this problem by monitoring the user equipped with vital sensors to determine the actual activity. The live classification is based on real-time acceleration and heart rate measurements of three sensor nodes including a smart phone. We will present an approach to detect the physical activity even in difficult situations achieving high accuracy rates up to 94%. These results can be achieved of using personalized classifiers. Furthermore we train multiple classifiers to aware an accurate recognition during sensor node failure.

1. INTRODUCTION

Regular physical activity is an important factor to influence the quality of our life and ensure the maintenance of health and wellness [15, 16]. It reduces the risk of dementia significantly [12] and improves cognitive skills [13]. In many cases a lack of physical activity can trigger a disease [10]. In the past many studies used questionnaires and self-autobiographical tests to determine the physical activity. Unfortunately, these approaches are often inaccurate and constitute the reality not sufficiently accurate. So it is difficult to find out relationships between movement and health [8]. Therefore, a system that automatically captures the physical activity could be very attractive for applications in the field of healthcare monitoring and in developing advanced human-machine interfaces.

Currently a number of methods are available to monitor and classify the physical human activity [1, 2, 3, 4, 5, 6, 7, 8, 10]. One of the simplest and cheapest method for this is the pedometer [10]. It is usually attached to the hip or foot to count the number of steps. Furthermore a smart phone could be used as an alternative for movement detection. This shows a study were datasets with 10 subjects were recorded by a smart phone sensor [1]. The activities “walking”, “posture transition”, “gentle motion”, “standing”, “sitting” and “lying” could be detected by a simple single classifier with an accuracy of 63.8%. It has been shown that the classification accuracy becomes better with a two multiclass SVM (support vector machines) which distinguish between motion and motionless activities. The improvement of this classification method compared to a single classifier amounts to 19% (82.8% vs. 63.8%).
Better results can be achieved by using multiple acceleration sensors. With the help of five bi-axial accelerometers and a transmission rate of 76.25 Hz an accuracy of 99.1% was reached [4]. The investigated activities were: “sitting”, “lying”, “standing”, “walking”, “stair climbing”, “running” and “cycling”.

Figure 1: Our system hardware

Another study distinguishes between these seven activities and adds “rowing”, “calisthenics” and “move weight” [6]. Five multi-axis accelerometers and an additional heart rate monitor for the movement intensity were used. The addition of heart rate data improves the result by no more than 2.1%. In table 7 a summary with different activity systems and their accuracy is given.

In this paper we present a real-time activity recognition system that is able to detect eight activities: “sitting”, “standing”, “lying”, “going”, “jumping jacks”, “pushups”, “squats” and “sit-ups”. The system consists of a chest strap sensor that measures the user’s acceleration and heart rate. Additionally an acceleration sensor attached to the wrist and an android smart phone carried in the subject’s pocket is used. The activity is estimated by a generalized standard classifier. For better classification results the classifier can be updated with personal data by our vital-datarecorder. Also we investigate the influence of sensor data blackouts. We test classifiers which are trained with simulated sensor blackouts and compare the results.

This paper is structured as follows. First we describe our system architecture including the hardware construction and the activity classification. Section 3 introduces our study for data collection and specifies the evaluation. The results are presented in the following section and analyzed in section 4. Finally we summarize the main contributions and present our future work.

2. SYSTEM DESIGN

Our system architecture is shown in figure 2. The signal receiver collects the transmitted data from the two accelerometers (as well as the heart rate) and publishes it via the Robotics Service Bus (RSB). The data from the smart phone is published by RSB using a wifi connection. The VitalClassifier (written in Java) listens to the RSB and classifies the incoming data. The results were stored in a MySQL database. The current activity generates and cumulates daily and monthly statistics of the executed activities which are presented on a website.

2.1. Hardware Construction

For our system we use a body sensor system that is designed by the research group Cognitronics and Sensor Systems[1]. The sensor system consists of two sensor modules which are attached to the user’s body. Both sensors send the acceleration data and heart rate to the receiver which can be connected to a standard PC via USB. For this an ANT[2]-compatible transceiver is used, which operates in the ISM band at 2.4 GHz. The sensor modules can capture the heart rate and the acceleration up to $\pm$ 24 g. Further acceleration data is received by smart phone. For this, we use the Samsung Galaxy S2, because it can provide a stable sufficient frequency rate of 15 Hz.

[1] see `http://www.ks.cit-ec.uni-bielefeld.de`
[2] see `http://www.thisisant.com`

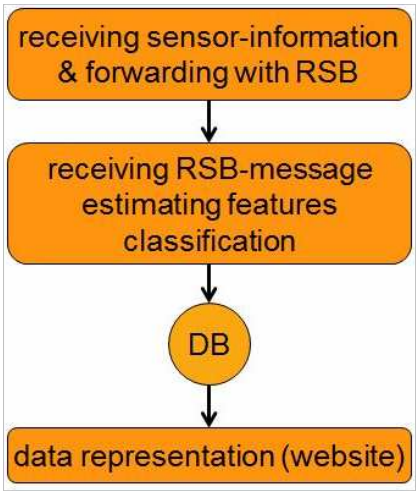


Figure 2: Illustration of the system architecture

2.2. Activity Classification

We used the WEKA-toolkit to classify the received sensor data. WEKA (Waikato Environment for Knowledge Analysis) is a free Java suite of machine learning software [18]. The following classifiers were used: NaiveBayesUpdateable (NBU), MultilayerPerceptron (MLP), sequential minimal optimization algorithm (SMO), K-nearest neighbours classifier (IBk), Locally-weighted learning (LWL), RacedIncrementalLogitBoost(RIB), Nearest Neighbor With Generalization (NNge) and Decision Tree (J48). All classifiers are updateable, except MLP, SMO and J48. The advantage of updateable classifiers is that they can be personalized with new training data later on. To connect our system components we used the Robotics Service Bus, a lightweight and flexible middleware.

For classification we calculate features [9] cumulating a specific period of time Δt. These can be divided into three parts. The acceleration of the body sensors is represented by 5 feature sets:

- Average acceleration for each axis
- Variance for each acceleration axis
- Average absolute difference between the mean and the acceleration for each axis
- Average normalized length of the acceleration vectors
- Acceleration histogram for each axis

Due to the flexible smart phone orientation we calculated adjusted features:

- Average length of the acceleration vectors
- Variance of the acceleration vector length

- Average absolute difference between the mean acceleration vector length and the acceleration vector length

Supplemental to this the heart rate hr measured by the chest sensor is normalized:

$$\hat{hr} = \frac{hr - hr_{min}}{hr_{max} - hr_{min}}$$

whereas hr_{max} and hr_{min} is the personalized heart rate range. These heart rate values are aggregated over $\Delta_{hr}t > \Delta t$:

- Maximum
- Minimum
- Gradient
- Average
- Variance

We divided the classification into two classification methods (one-step / two-step) and determined which method generates the best results. The one-step classifier distinguishes immediately between all activities. A better alternative could be a two-step classifier which is subdivided into three classifiers [1]. The first distinguishes only between sporty and non-sporty activities (first step). The second classifier classified within the sporty and the third differs between the non-sporty activities (second phase). Depending on the result of the first phase, one of the two classifiers is chosen.

The sensor signals can be absorbed by the human body. Due to short recognition times, sensor packet loss can be a great problem. To classify accurately in these situations, we tested several variants to prepare the classifiers for sensor blackouts. One approach is to train the classifiers for each combination of blackouts for a whole sensor node. For three sensor nodes this will lead to seven different classifiers (seven-classifier-variant). During classification we detect inactive sensors and choose the associated classifier for each time interval.

Another approach is to train the classifier several times with the same training data, by blocking one or two sensors in each iteration.

3. EVALUATION

To evaluate our activity classification we have carried out a study. On the collected data we did two evaluations. First we tested different classification methods by varying several classification parameters. Then we determined an appropriate handling for sensor blackouts.

3.1. Data Collection

To collect training data we have carried out a study with ten participants (3 female, 7 male, $\varnothing 25 \pm 3$ years old). The participants took part in three training sessions. Each session included the following activities in the named order. First the non-sporty part including “sitting”, “standing”, “lying”, “going”, and then the sporty part with “jumping jacks”, “pushups”, “squats” and “sit-ups”. Each activity measurement takes 30 second, to a summing up to 12 minutes for each subject. After the second training session the sensors were removed and attached again for the next phase to provide data variance. To simplify data collection we developed a java tool which saves the received sensor data annotated with the current physical activity in several csv files.

3.2. Evaluation of Classification Methods

To obtain the best classification we tested eight classifiers provided by the WEKA-toolkit. We used leave-one-subject-out cross-validation to evaluate them: Each classifier was trained with annotated data from all users, except one. The data of the remaining user was used to measure the classification results. Multiple rounds were performed to calculate the average classification accuracy and in each round a different user was left out for measuring the classification results.

To determine the benefit of a personalized classifier we compared the standard training method with an updated classifier. After training we updated the classifier with two of the three sessions from the test user to personalize the classifier. Then we tested the remaining training session with the personalized classifier. This was done three times to average over all sessions. We examined how well the results have been improved by the updateable classifiers and compared the results with the non-updated classifiers.

For classification we calculated features cumulating a specific period of time. The length of this time interval influences the classification result. To investigate this parameter we used four different ranges: 500 ms, 1500 ms, 2500 ms and 3750 ms. This time period also influences the recognition time of a new activity. For live classification larger ranges are not useful. The heart features are averaged over a longer period to deal with slow heart rate changes.

3.3. Evaluation of Sensor Blackout Handling

The sensor signals can be absorbed by the human body. We described two variants of sensor blackout handling. One approach is to train multiple classifiers which is compared to the standard variant with only one classifier. To reduce calculation time we used only the four best performing classification methods from section 3.2 for evaluation.

The second approach trains the classifier several times with the same training data, by blocking one or two sensors in each iteration. To investigate this technique we evaluate two variants. The one-block-variant consists of four iterations. One iteration without blocking any sensor added with three iterations by blocking each of the sensors alone (one-block-variant); the two-block-variant extends the one-block-variant by adding three iterations blocking two of the sensors at the same time.

To test the different approaches we simulated sensor blackouts in the training data during the whole time. To point out the individual importance of our three sensors we blocked each sensor separately. These results were compared to the normal test procedure without sensor blocking.

3.4. Results

In this section we will present the results of our activity classification. First the selection of the best classification method is shown. Then the effects of sensor blackout handling are listed.

3.4.1. Classification Methods

Table 1 shows the best classification results of the eight classifiers differing between no-update and update. All classifiers reach high accuracy rates of over 84%. As expected, an update with the personal data improves the classification result for all classifiers about 4.2% $\pm$ 2.1%. Therefore a small update is sufficient to personalize the classifier (108 min general training data, 8 min personalized data).

It is remarkable that the non-updateable classifier J48 achieves a very good result in comparison to the best personalized classifiers. Table 2 compares the best results of the one-step classification and the two-step classification variant. In five of eight cases the fragmentation into two steps caused a decrease in the classification accuracy. The best results for different time periods of feature calculation are presented in table 3. No optimal time interval could be determined in our evaluation of different time periods (as seen on table3. A range of 2500 ms could be a good compromise between performance an classification accuracy. The above findings belong all to the classification result of the eight described classes. For the two class problem (sport or no-sport) we achieved an accuracy of 98% (table 2).

3.4.2. Sensor Blackout Handling

Only the four best classifiers were taken into account for the evaluation of sensor blackout handling. These are the non-updateable J48 and the updateable NBU, RIB and NNge using their best time period parameter. Considering the results of section 3.4.1 this was done with the one-step classification variant.

The results of our analysis can be seen in table 4 (appendix). Without simulated packet loss the seven-classifier variant is almost identical to the standard variant with minimal variation ($\pm$ 1%). When sensors are blocked the seven-classifier variant performs considerably better. It is noticeable that blocking the smart phone sensor does not result into any depreciation. For both variants blocking the chest sensor leads to a significant decrease of recognition accuracy.

The evaluation of the second approach is presented in table 5 and table 6 (appendix). Different training and testing variants are permutated. The duplication of training data leads in most cases to a decrease of the recognition accuracy. In comparison to the other sensors the chest sensor again provides the most significant information.

Classifier	without update	with update
MLP	89.9	-
SMO	91.1	-
J48	91.9	-
NBU	90.0	93.2
IBk	84.0	90.7
LWL	87.5	91.0
RIB	**92.6**	**94.3**
NNge	87.5	93.4

Table 1: Standard vs. personalized classifier accuracy [%]

Classifier	one-step	two-step
NBU	93.2	88.9
MLP	87.1	89.9
SMO	91.1	89.9
J48	91.9	90.9
NBU	93.2	88.9
IBk	90.2	90.7
LWL	67.0	91.0
RIB	**94.3**	91.2
NNge	93.4	**91.4**

Table 2: One-step vs. two-step classification accuracy [%]

4. DISCUSSION

We presented a system to recognize the current activity of a person with high precision.

Using the update mechanism it is possible to adapt the classifier to any person without re-training the whole classifier. It is noticeable that classifiers with low accuracy benefit more from an update than classifiers which already performed well. Due to this approach the user can also cre-

Classifier	500 ms	1500 ms	2500 ms	3750 ms
MLP	86.0	87.5	88.2	**89.9**
SMO	88.5	90.9	90.9	**91.1**
J48	90.5	**91.9**	90.9	91.4
NBU	88.2	92.1	92.6	**93.2**
IBk	86.4	89.7	**90.7**	90.2
LWL	79.0	**91.0**	89.6	89.2
RIB	91.8	92.6	**94.3**	80.0
NNge	86.7	91.9	**93.4**	91.5

Table 3: Accuracy by time period [%]

ate new activity classes at runtime. Unfortunately it is well known that adding new classes hampers the classification task. This negative correlation between the number of activities and the accuracy should be considered in practical usage.

Also we examined an approach proposed by [1]. There a classification process is splitted into two steps where in the first step one classifier only differs between sporty and non-sporty activities, performed 19% better. Our results do not support this thesis. No advantage could be observed.

To avoid misclassification during sensor packet loss, we duplicated training data with simulated sensor blackouts. However, the analysis of the blackout is ambiguous. Some of our considered classification algorithms benefited, others were handicapped. For this variant no general conclusion can be drawn and further evaluation is required. This approach also leads to a high computational cost.

Better handling is possible by training several classifiers. After detecting all active sensors within the current time period we use classifiers trained especially for these active sensor nodes. This approach handles sensor blackouts and improves the system significantly.

One slight disadvantage is the higher computing time for training all classifiers. There is no impact to the real-time classification because the training process has to be executed once only. In our case 3 sensor nodes result in seven training phases. Generally this led to $2^n - 1$ training tasks for n sensor nodes.

For systems with many sensor nodes this approach is not viable. Developers will have to decide which permutations should be considered. Expecting only one sensor blackout at a time could be a good strategy, one for all sensor nodes and one for each possible single sensor failure.

As shown in [6] examining heart rate frequency in addition to accelerometers can enhance classification results. Combining a heart rate detector with an accelerometer into one sensor node as used in this study is easier to handle. Considering sensor blackouts also reduces the number of classifiers to train.

In our application a smart phone acceleration sensor does not improve the classification accuracy. One possible reason is the flexible position in the pocket which can not represent the body movement accurately. Furthermore for activity discrimination one acceleration sensor close to the center of mass seems to be sufficient. Our chest sensor provides the acceleration data with a higher frequency and offers a heart rate sensor so the smart phone seems to be negligible.

5. CONCLUSION

In this work we described a method for activity recognition with only three sensors, providing three axial accelerations and the users heart rate. Different approaches for recognizing four sporty and four non-sporty activities were investigated to create an advanced classifier. Activity recognition with accuracy rates of 94% were possible. Analysing different classification methods, we observed a significant improvement by updating a general classifier - trained with data from other persons - with personal data.

In most classification systems sensor blackouts lead to classification failures. Our approach is able to handle sensor blackouts by using multiple classifiers specialized for one or more sensor nodes. Without specific preparations chest sensor failure for example can decrease the classification rate down to 15 - 20%. The blackout handling variant reaches a significantly higher accuracy around 80%.

In our future research we plan to evaluate the importance of all used features. This can point out unimportant features which could be omitted to improve classification performance. To get a more discriminative feature set we will relate signal data from different sensors in combined features. Also we want to use wavelets instead of histograms to approximate the distribution for each axis acceleration. Another idea is to use classification results which are detected several times. This simple method could improve the stability of the system by ignoring single classification faults which can occur during activity changes.

During the evaluation we measured the classification accuracy depending on different training parameters. Another parameter which could be evaluated is the heart rate history length. This would help to choose this value optimal.

An modified hardware setting could also improve the system, for example an accelerometer attached to the foot.

6. ACKNOWLEDGEMENT

This paper would not have been possible without the support of many people. The authors wishes to express their gratitude to their lecturer Dipl.-Inform. Med. Peter Christ.

7. REFERENCES

[1] Shumei Zhang, Paul McCullagh, Chris Nugent, Huiru Zheng, "Activity Monitoring Using a Smart Phones Accelerometer with Hierarchical Classification", 2010 Sixth International Conference on Intelligent Environments

[2] Wolf-Joachim Fischer, Marian Steinert und Andreas Heinig, "Langzeit-Bewegungsmonitoring und Energieumsatzbestimmung mittels eines portablen Diagnosegerätes", 28.05.09, Fraunhofer-Institut für Photonische Mikrosysteme (IPMS)

[3] Andrea Mannini and Angelo Maria Sabatini, "Machine Learning Methods for Classifying Human Physical Activity from On-Body Accelerometers" Published: 1 February 2010

[4] Kuan Zhang, Patricia Werner, Ming Sun, F. Xavier Pi-Sunyer, and Carol N. Boozer, "measurement of human daily physical activity",OBESITY RESEARCH Vol. 11 No. 1 January 2003

[5] K. Aminian, Ph. Robert, D. Hayoz, E.E. Buchser, M. Depairon, B. Rutschmann, " Physical activity monitoring based on accelerrometry, Med. Biol. Eng. Comput., 1999, 37, 304-308

[6] Emmanül Munguia Tapia1, Stephen S. Intille, "Real-Time Recognition of Physical Activities and Their Intensities Using Wireless Accelerometers and a Heart Rate Monitor", 2007 IEEE, House, Massachusetts Institute of Technology, Cambridge, MA, USA

[7] Tal Shany, Stephen J. Redmond, Member, "Sensors-Based Wearable Systems for Monitoring of Human Movement and Falls", IEEE SENSORS JOURNAL, VOL. 12, NO. 3, MARCH 2012

[8] Jasmin Schröckenfuchs, "Validierung zweier Kurzmethoden zur Ermittlung der körperlichen Aktivität mittels Akzelerometrie", Wien, im November 2009, Diplomarbeit

[9] J. Kwapisz, G. Weiss, S. Moore, "Activity Recognition using Cell Phone Accelerometers", Department of Computer and Information Science, Fordham University, 2010, Washington, DC, USA.

[10] de Blok, B.M.J., de Greef, M.H.G., ten Hacken, N.H.T., Sprenger, S.R., Postema, K. & Wempe, J.B. 2006, "The effects of a lifestyle physical activity counseling program with feedback of a pedometer during pulmonary rehabilitation in patients with COPD: a pilot study", Patient education and counseling, vol. 61, no. 1, pp. 48-55.

[11] A. Bee, C.D. Player, and X. Lastname, "A correct citation," in *Proc. of the 1st Int. Conf. (IC)*, Helsinki, Finland, June 2001, pp. 1119-1134.

[12] R.D.Abbot et.al., "Walking and Dementia in Physically Capable Elderly Men", JAMA. 2004; 292(12),1447-1453

[13] N.T.Lautenschlager et.al.,"Effect of Physical Activity on Cognitive Function in Older Adults at Risk for Alzheimer Disease", JAMA. 2008; 300(9),1027-1037

[14] Melissa Luise Jehn, "Physiologische Indikatoren der körperlichen Leistungsfähigkeit bei Herzinsuffizienz" 28.01.2010, Technische Universität München

[15] Craft, L. L. & Landers, D. M. "The effects of exercise on clinical depression and depression resulting from mental illness". Journal of Sport & Exercise Psychology (1998), 20, 339357.

[16] Andreas Schwerdtfeger,"Gibt es einen Zusammenhang zwischen Bewegungsaktivität und psychischem Befinden im Alltag?", Psychologisches Department Institut, Johannes Gutenberg-Universität Mainz

[17] J. Wienke, S. Wrede "A Middleware for Collaborative Research in Experimental Robotics", Institute for Cognition and Robotics (CoR-Lab), Bielefeld University, 33615 Bielefeld, Germany

[18] M. Hall, et.al., "The WEKA Data Mining Software: An Update", SIGKDD Explorations, Volume 11, Issue 1.

Classifier	normal training				blackout handling			
tested sensor blackout	no	wrist	cell	chest	no	wrist	cell	chest
J48	91.9	69.9	73.3	19.9	93.0	92.4	93.7	78.4
NBU	93.2	79.5	78.1	40.8	93.2	84.7	93.3	59.4
RIB	94.3	72.1	88.6	15.9	94.5	93.2	92.2	79.7
NNge	93.4	57.2	55.2	18.5	93.6	89.5	92.8	82.3

Table 4: blackout handling accuracy [%]

Classifier	normal training				normal + single blackout training			
tested sensor blackout	no	wrist	cell	chest	no	wrist	cell	chest
J48	91.9	69.9	73.3	19.9	93.2	23.7	73.3	20.4
NBU	93.2	79.5	78.1	40.8	93.2	78.9	78.1	12.6
RIB	94.3	72.1	88.6	15.9	94.3	61.7	81.7	15.9
NNge	93.4	57.2	55.2	18.5	89.8	57.2	55.2	19.9

Table 5: single blackout training accuracy [%]

Classifier	normal training				normal + all blackout training			
tested sensor blackout	no	wrist	cell	chest	no	wrist	cell	chest
J48	91.9	69.9	73.3	19.9	93.2	23.7	74.7	19.9
NBU	93.2	79.5	78.1	40.8	57.7	78.9	81.9	40.8
RIB	94.3	72.1	88.6	15.9	94.0	72.1	88.6	77.0
NNge	93.4	57.2	55.2	18.5	89.8	57.2	93.0	82.2

Table 6: all blackout training accuracy [%]

Reference	Sensor-types	Algorithm	Activities	Accuracy
[1]	Accelerometer from Smart phone	Multiclass SVM algorithm	6	82.8%
[3]	5 bi-axial accelerometers	Hidden Markov Models	7	99.1%
[4]	5 Accelerometers	*unknown*	6	98.5%
[6]	5 Accelerometer, heart rate monitor	Naive Bayes	10	98.7%
our results	2 accelerometer, heart rate monitor smart phone acceleration	RacedIncrementalLogitBoost	8	94.3%

Table 7: Activity detection system comparison

INTELLIGENT SYSTEMS PROJECT: WEARABLE SONIFICATION

K. Banasiak, F. Hofmann, E. Wall

Faculty of Technology, Bielefeld University
Bielefeld, Germany

Supervisors: T. Hermann, S. Zehe, H. Lex

ABSTRACT

In this paper we present WESON, a system that provides sonification feedback from live tracking of body movements. This feedback is meant to improve learning of said movements. Custom hardware was build to facilitate this goal and was paired with existing systems where appropriate. Additional analysis, visualization and sonification software was implemented.

The putting stroke from golfing was chosen as an exemplary movement and a specialized sonification was created to improve training of this exercise. Finally a small test series was conducted to analyse the performance of the developed system. We conclude that sonifications as a teaching instrument is able to improve the adoption of new movements, but should be supplemented with other kinds of teaching.

1. INTRODUCTION

1.1. Wearable Sonification

Coordinated body movements are difficult to learn for most persons. Even with aids like visual displays or proprioception reproducing such body movements seems to be hard considering their limited capabilities. Visual displays for example have disadvantages in mobility because the user is constrained to look at the display while proprioception is difficult to assess.

Sonification offers a different interface for human computer interaction. Non-speech audio is used to represent information. For wearable sonification, sensors record the body movements and the computer generates sounds based on the received data. Thus, a user gets feedback from his body motions and is able to manipulate the sound in real-time.

1.2. Previous Project

In our former project, we developed a system for physiotherapeutic exercises. Therefor we chose an exercise with three different postures. With sensors attached to arms, legs and the back, the system was able to recognize the current posture. The system determined the body part with the greatest error compared to the target posture and provided a real-time auditory feedback. This way the user could adjust his posture until it matched the target.

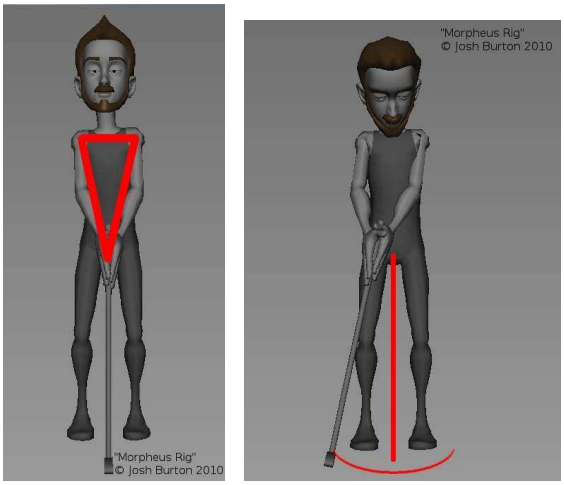
1.3. Current Project

For the current project we aimed at a more dynamic exercise to target a different usage scenario. We searched for a movement that can be captured with the existing hardware. Moreover, a user should be able to listen to the sonified movement data while performing the exercise. The gained information from the sound can then be used to improve following executions. We call this a ”One-Shot-Sonification”.

Our decision was for a golf stroke. The stroke that met our requirements most was the putt. It is performed on the green, the finely-cut grassed area at the end of a golf fairway, to precisely strike the ball towards the target. Controlling the direction and the distance of the stroke is the key to master putting. In our project we focus on the distance.

To perform a putt, the feet should be placed shoulder-wide and the knees are bent slightly. The stretched arms together with the shoulders form a triangle such as shown in Fig. 1a. Movement only occurs in arms and shoulders while the formed triangle remains unchanged. To execute the stroke a pendular movement should be sought. Backswing and downswing have the same distance where the velocity is constant. Such a pendular movement is sketched in Fig. 1b.

Our WESON system is developed to help the user at learning the correct execution of a putt stroke by comparing both, the sound of a correct performed putt and the sound of his own performance. The way body and putter movements are recorded and how the data of the sensors are transformed into sound is explained in following chapters.



(a) Triangle formed by arms and shoulders (b) Pendular movement of the club

Figure 1: Anatomy of a putting stroke

1.4. State-of-the-art

An already existing system for putt training is the Puttronome[1]. It is a smartphone application for helping golf players to practise on controlling the distance of putt strokes. With continuous audio feedback and a visual guide of a perfect execution, the user shall be able to work on his putting tempo. On backswig, the pitch of the hearable tone rises while the audio sweeps from left to right. During downswing the pitch of the tone falls and the audio sweeps from right to left. Thus, the user gets a feeling for the pendular movement[**?**]. The Puttronome supports several different tempos of putt strokes.

2. SYSTEM DESIGN

The overall design of the WESON system is motivated by two goals. First the result should be a wearable setup. Second, the whole setup should improve the execution of the putting stroke in new golf players.

To accomplish this, human attributes coming from anatomy and cognition are taken into account. These give us insight into movements and postures that are possible and therefore classes of errors that we should address in our system. Additionally it imposes constrains onto the system, for example the human hearing system is quite able to notice small latencies in the feedback loop, so care must be taken to keep these latencies low.

The subsystems are described in the following sections (see overview in Figure 2): Input from human movement is captured using the BRIX platform (Section 3), the data is processed on a nearby computer (Section 4), an audible signal is created based on the data (Section 5) and finally played back to the user.

This closes the feedback loop and should result in an improved execution of putt strokes using the WESON system.

[1] http://www.haroldswashputting.co.uk/putting-aids/the-puttronome.html

3. HARDWARE: CONSTRUCTION AND SETUP

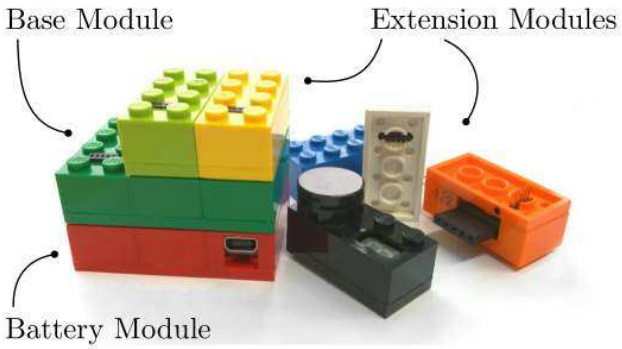


Figure 3: A BRIX stack example and extension modules[1].

3.1. The BRIX Platform

WESON is build around the BRIX platform ([1], see Figure 3). It is used because it provides a lightweight system to connect arbitrary sensors to a computer through a wireless link. The original BRIX system was centered around a base module hosting an ATmega168 8-bit microcontroller, several IO ports and a Bluetooth connection for communication.

During the course of the project, a prototype of the next generation, named BRIX2 got released. BRIX2 is compatible with the Arduino Plattform[2]. Additional it is equipped with an *MPU-9150 9-axis tracking device* that is able to measure 16bits of gyroscope, magnetometer and accelerometer data and calculate euler angles from the measured data.

To facilitate future development with the BRIX platform in the context of ISY and CITEC projects an interface between BRIX and RSB[3] was developed (see 4.1).

3.2. Sensor Assembly

As the task at hand required precise tracking of body movements, care was taken to build a sensor assembly that allowed the gathering of the necessary data. Building upon our former work we continued to use our arm sensor mounting (see Figure 5). Slight adjustments were made to be able to detect erroneous wrist posture.

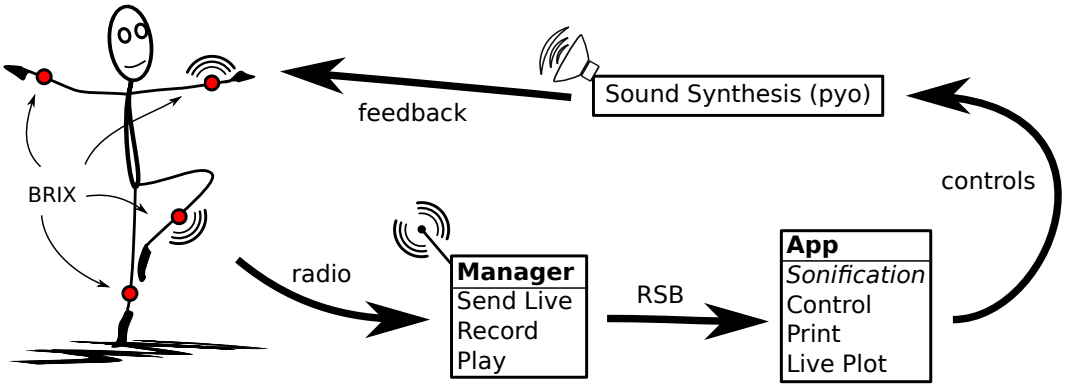


Figure 2: Overview of the WESON System

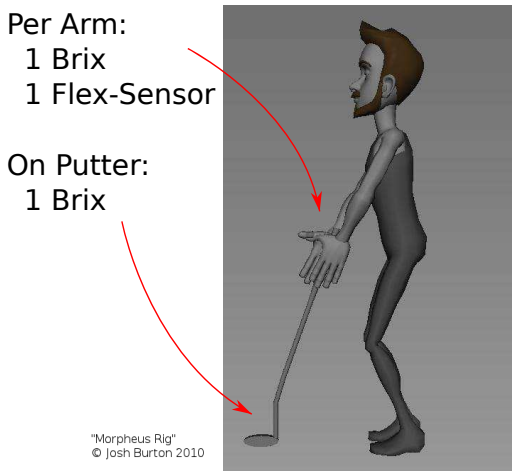


Figure 4: Locations of tracking hardware on person.

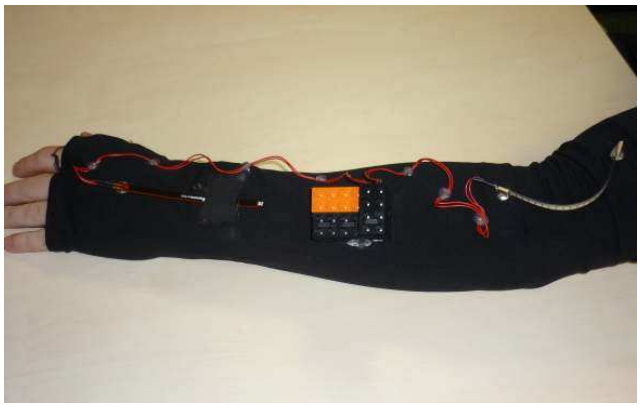


Figure 5: Arm Sensor Assembly

BRIX1 sensors were used in order to track the wrist and elbow posture during the putting actions.

3.3. Putter

In golfing, choosing the right club for the job is most crucial for success. Our putter is a modified *PING Men's Scottsdale Half Pipe Putter*. It was equipped with a BRIX2 module (see Figure 6).

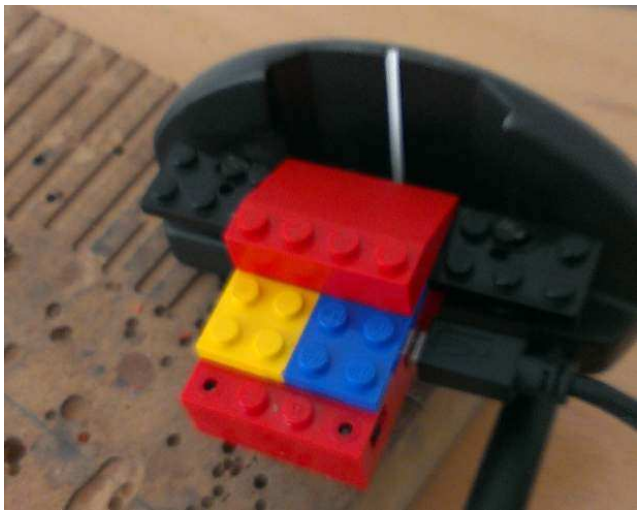
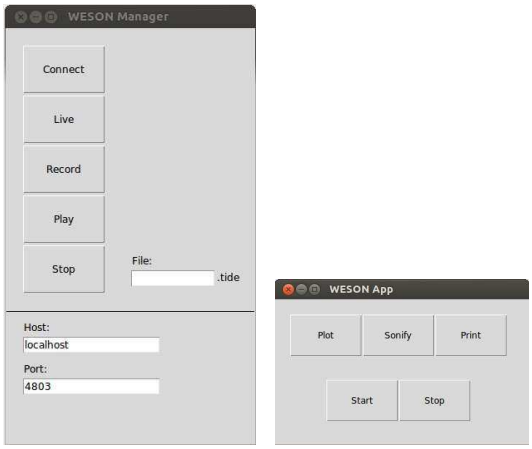


Figure 6: Sensors mounted on Putter

4. SOFTWARE COMPONENTS

4.1. Brix2RSB

To connect BRIX technology to the existing RSB middleware, a gateway component bridge between both was implemented in *Python*. It was written using non-blocking select-based IO to read data from the serial interface representing the BRIX module. Then the *Python RSB Bindings*



(a) Weson Manager (b) Weson App

Figure 7: Screenshots of the WESON Applications

where used to broadcast the received data to RSB.

4.2. WESON App and WESON Manager

The software side of the WESON system is handled by two applications written in Python. The hardware side is handled by the *WESON Manager* while sonification and visualization is done by the *WESON App* (see Figures 7a and 7b respectively).

This provided a separation of concerns between recording the data and processing it. Both applications depend on the RSB middleware for communication.

4.2.1. WESON Manager

The WESON Manager handles the BRIX modules using pyserial and pybluetooth for communication and provides recording and playback facilities by building upon RSBag[2].

4.2.2. WESON App

The WESON App provides visualization of the data using NumPy[3] and SciPy[4] and sonification (see Section 5) using the pyo toolkit[5].

5. SONIFICATION APPROACH

For sonification we need a set of sensor values for processing them live into sound. Our approach gathers three-axis acceleration values $a = (a_x, a_y, a_z)$ from the BRIX2 platform fixed on the golf club for a characteristic push sound as well as angle values of both wrists and elbows for correct posture detection. During pushing the a-vector will be processed with an algorithm that continuously calculates an angle θ as follows:

$$\theta = arctan2(||a_z - a_x||, a_y)$$

We map angle θ directly onto the pitch of a sinus-tone. So each kind of putt creates a specific sound, the user can use for comparison. To make the sound extra-specific for a correct putt, we adjusted the angle-to-pitch-mapping to have a full octave between the minimum and maximum value of an idle putt, which we estimated by some examples of an experienced golf player. Human ear naturally has an intuition for full-octave intervals, which makes it easier to memorize such musical patterns.

5.1. Three stage training system

With that possibility for comparison we set up our sonification training system which we divided into three successive stages: The first stage checks the correct start posture. In the second stage the user hears a sonification of a correct putt, which he has to reproduce in stage III. The exact stage algorithms are described in the following paragraphs.

Stage I

On initial system startup, there is silence in the first stage, while calculating the angle θ as described above. After it reaches a value that is specific to the starting idle posture, a low-pitch tone appears with a rising pitch by resting into that posture. If we leave that posture before it reaches a defined maximum, the pitch will be restored completely. Reaching that maximum by resting in the idle posture for short time initiates stage II accompanied by a signal tone.

Stage II

In the second stage the system plays a sonification example of an ideal putt we recorded from an experienced golf player. Directly after hearing it and keeping the sound in mind, it changes to stage III with a second signal tone, in which the user is supposed to to reproduce this sound by his movements.

Stage III

The users moves get sonified according to the algorithm, described above. The difference between the playback and the self-created putt sound provides feedback as a performance criteria.

[2] https://code.cor-lab.org/projects/rsbag
[3] http://www.numpy.org/
[4] http://www.scipy.org/
[5] https://code.google.com/p/pyo/

During stage III our software also check if there are illegal elbow or wrist movements, which are typical beginner's mistakes. Therefor we defined static maxima of how much two successive incoming angle values are allowed to differ. If the movement exceeds this threshold the user hears a warning tone.
Stage III has to be about as long as stage II to enforce a rhythmical execution. When the given time for stage III is expired, the system restores to stage I automatically, which closes our training cycle.

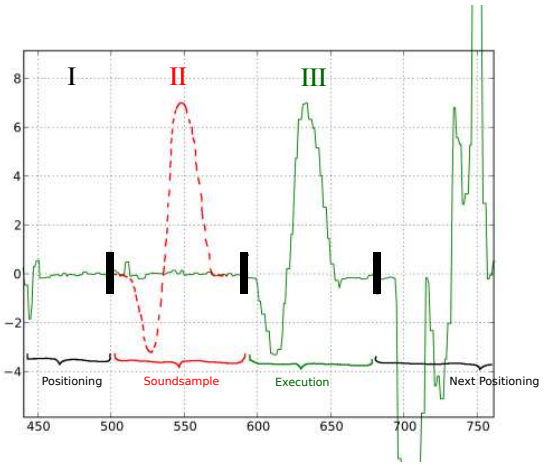


Figure 8: Three Stage Training System

6. INTERACTION/OPERATION EXAMPLES

The interaction video[6] shows a test user practicing putt strokes with our WESON system. Before training both arms are equipped with sensors. After that, the user is preparing for the stroke by going into the starting position. The system starts in stage I, where the putter has to remain in the specific posture to load a tone until a signal occurs and the WESON system changes to stage II. In this stage, the user hears a sound example of an ideal putt which acts as a reference for his own performance. The sound example is followed by another signal that initiates stage III of the system. Finally it is time for the user to perform a putt stroke by himself. While putting the clubs motion is sonified. Example sound and the user generated sound can be compared to analyse the user's execution. In the next try, gained information of the analysis is used to improve the performance. This interaction can be seen in the demonstration video accompanying this paper.

[6] https://www.youtube.com/watch?v=7yXHfgsSp2g

7. EVALUATION

Three participants attended in a small evaluation of the WESON system and were interviewed subsequently. All persons were novice golf players and had neither experience nor knowledge about this sport.

At first, every person was shown the basics of putting like how to hold the putter, stance and stroke execution. Then, the WESON system was presented and explained to them. Afterwards, we equipped the subjects with the sensors so they could practice. Meanwhile, the observations were noted. After the testing phase, the subjects were interviewed about their opinion on our system.

All three test persons had almost no problems using our system. No one found the attached hardware or the sonification disturbing and every user felt comfortable while performing the putt strokes. It took the subjects some time to understand the three different phases and their sequence. During the first few strokes, no improvement could be noticed. But over time, the subjects learned to identify the mistakes they made by generated sonification. They knew how to adjust their performance but the execution was difficult. So overall the improvement was moderate which can be correlated to the missing knowledge and experience in golf.

All three subjects agreed that the WESON system is an interesting new interface. They could imagine to use it for putt training and found the system equal to recording via camera, but they would rather have a personal trainer who can help them improve their performance and give hints.

In summary, the WESON system helps users to identify mistakes in executing putt strokes. Still it remains difficult to make use of this knowledge to improve the performance by correcting these mistakes.

8. DISCUSSION

From our work with the WESON system and the conducted experiment, it remains unclear whether sonification leads to improved performance in precision related tasks. While users noticed their errors due to the audio feedback, they had difficulties to adjust their actions accordingly. This resembles our insights from the last semester (compare Section 1.2), where we experienced similar complications.

In consequence, audible signals alone might not be enough to improve tasks, especially if the users have difficulties to relate the audio signals to improvement hints.

9. CONCLUSION

Finally, we are left to say that working with real-time sound generation is a great and creative field of research. While our project did not conclude in a useful product, other

projects can build upon and continue our work and will hopefully result in useful tools in multiple fields. Currently, we use only sine waves and variation in the stereo field to generate signals. Further projects should be conducted, including more and different ways of audio signal generation. Also, multi-modal feedback could be provided. To keep with our goal of providing a wearable system, head-mounted displays that provide a visual component to the feedback could be added.

10. ACKNOWLEDGEMENT

We would like to thank Thomas Hermann and Sebastian Zehe for the mentoring of this project and support regarding the BRIX System. Additionally, we thank Sebastian Wrede and Florian Lier who provided support for the RSB toolkit and organized events giving background informations. Finally, we want to thank Heiko Lex for an introduction to golf, for helping to choose the right exercise for our project and together with Cornelia Frank for providing the golf equipment.

11. REFERENCES

[1] S. Zehe, T. Grosshauser, and T. Hermann, "Brixan easy-to-use modular sensor and actuator prototyping toolkit," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*. IEEE, 2012, pp. 817–822.

[2] "Arduino homepage," July 2013. [Online]. Available: http://arduino.cc

[3] J. Wienke and S. Wrede, "A middleware for collaborative research in experimental robotics," in *System Integration (SII), 2011 IEEE/SICE International Symposium on*. IEEE, 2011, pp. 1183–1190.